Facility Testing of E4S via E4S Testsuite, Spack Test, and buildtest

Shahzeb Siddiqui (Lawrence Berkeley National Laboratory)

Sep 14th 2021

ECP Event: https://www.exascaleproject.org/event/buildtest-21-09/

https://buildtest.readthedocs.io/

https://github.com/buildtesters/buildtest

http://hpcbuildtest.slack.com/
About Me

• I am an HPC Consultant at NERSC in the **User Engagement Group** that is responsible for user support including support tickets, user documentation, training, and managing software stack for NERSC.

• I am the L4 for Software Integration Group (WBS: 2.4.4.01) in the ECP Project. In this group we are responsible for deploying the Extreme Scale Scientific Software Stack (E4S) at the DOE Facilities (NERSC, OLCF, ALCF)

• Previously held multiple roles throughout my career including Dassault Systems, Pfizer, Penn State, IBM, NASA, and Northrop Grumman

• Creator of **buildtest**: HPC Testing Framework

• Certified Red Hat Certified System Administrator (RHCSA): **200-019-677**

• Masters in Computer Science from KAUST

https://www.linkedin.com/in/shahzebmsiddiqui/

https://github.com/shahzebsiddiqui/
• **Extreme-scale Scientific Software Stack (E4S)** is a curated set of software packages for developing and running scientific application on HPC platforms.

• E4S is a subset of Spack Packages

• E4S is deployed as spack manifest, containers, and buildcache.

• Contains up to 80+ software products including: compilers, data and viz tools, I/O tools, profilers, xSDK and may more
E4S: Extreme-scale Scientific Software Stack

• Curated, Spack based software distribution
• Spack binary build caches for bare-metal installs
  – x86_64, ppc64le (IBM Power 9), and aarch64 (ARM64)
• Container images on DockerHub and E4S website of pre-built binaries of ECP ST products
• Base images and full featured containers (with GPU support)
• GitHub recipes for creating custom images from base images
• GitLab integration for building E4S images
• E4S validation test suite on GitHub
• E4S-cl container launcher tool for MPI substitution in applications using MPICH ABI
• E4S VirtualBox image with support for container runtimes
  • Docker
  • Singularity
  • Shifter
  • Charliecloud
• AWS and GCP images to deploy E4S

https://e4s.io
Extreme-scale Scientific Software Stack (E4S)

- E4S: HPC Software Ecosystem – a curated software portfolio

- A **Spack-based** distribution of software tested for interoperability and portability to multiple architectures

- Available from **source, containers, cloud, binary caches**

- Leverages and enhances SDK interoperability thrust

- Not a commercial product – an open resource for all

- Oct 2018: E4S 0.1 - 24 full, 24 partial release products
- Jan 2019: E4S 0.2 - 37 full, 10 partial release products
- Nov 2019: E4S 1.0 - 50 full, 5 partial release products
- Feb 2020: E4S 1.1 - 61 full release products
- Nov 2020: E4S 1.2 (aka, 20.10) - 67 full release products
- Feb 2021: E4S 21.02 - 67 full release, 4 partial release
- May 2021: E4S 21.05 - 76 full release products

https://e4s.io

Lead: Sameer Shende (U Oregon)

Also include other products .e.g.,
AI: PyTorch, TensorFlow, Horovod
Co-Design: AMReX, Cabana, MFEM

https://e4s.io/talks/E4S_IAW21.pptx (Slide 44)
## 1.2. E4S Facility Deployment Dashboard

### Corl

<table>
<thead>
<tr>
<th>E4S Version</th>
<th>Project URL (Github, Gitlab)</th>
<th>Installed Specs</th>
<th>Compiler</th>
<th>Spack commit</th>
<th>Spack.yml</th>
<th>Installed Specs</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.10</td>
<td><a href="https://software.nersc.gov/nersc/e4s-corl">https://software.nersc.gov/nersc/e4s-corl</a></td>
<td>135</td>
<td>intel@10.1.2.254</td>
<td>e47b6d2b1e4e11a345c02d75504666f1a8607f5e653b7</td>
<td><a href="https://github.com/spack/spack-configs/blob/master/NERSC/corl/e4s-20.10/spack.yml">https://github.com/spack/spack-configs/blob/master/NERSC/corl/e4s-20.10/spack.yml</a></td>
<td><a href="https://github.com/spack/spack-configs/blob/master/NERSC/corl/e4s-20.10/e4s-20.10.txt">https://github.com/spack/spack-configs/blob/master/NERSC/corl/e4s-20.10/e4s-20.10.txt</a></td>
</tr>
<tr>
<td>21.02</td>
<td><a href="https://software.nersc.gov/nersc/e4s-corl">https://software.nersc.gov/nersc/e4s-corl</a></td>
<td>149</td>
<td>intel@10.1.2.254 and gcc@10.1.0</td>
<td>b56d65c6d5743a23339f10cde0d6e8b525f3d3</td>
<td><a href="https://github.com/spack/spack-configs/blob/master/NERSC/corl/e4s-21.02/spack.yml">https://github.com/spack/spack-configs/blob/master/NERSC/corl/e4s-21.02/spack.yml</a></td>
<td><a href="https://github.com/spack/spack-configs/blob/master/NERSC/corl/e4s-21.02/e4s-21.02.txt">https://github.com/spack/spack-configs/blob/master/NERSC/corl/e4s-21.02/e4s-21.02.txt</a></td>
</tr>
</tbody>
</table>

### Perlmutter

<table>
<thead>
<tr>
<th>E4S Version</th>
<th>Project URL (Github, Gitlab)</th>
<th>Installed Specs</th>
<th>Compiler</th>
<th>Spack commit</th>
<th>Spack.yml</th>
<th>Installed Specs</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>E4S Version</th>
<th>Project URL (Github, Gitlab)</th>
<th>Installed Specs</th>
<th>Compiler</th>
<th>Spack commit</th>
<th>Spack.yml</th>
<th>Installed Specs</th>
</tr>
</thead>
</table>

### Summit

<table>
<thead>
<tr>
<th>E4S Version</th>
<th>Project URL (Github, Gitlab)</th>
<th>Installed Specs</th>
<th>Compiler</th>
<th>Spack commit</th>
<th>Spack.yml</th>
<th>Installed Specs</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>E4S Version</th>
<th>Project URL (Github, Gitlab)</th>
<th>Installed Specs</th>
<th>Compiler</th>
<th>Spack commit</th>
<th>Spack.yml</th>
<th>Installed Specs</th>
</tr>
</thead>
</table>

### Spack

<table>
<thead>
<tr>
<th>E4S Version</th>
<th>Project URL (Github, Gitlab)</th>
<th>Installed Specs</th>
<th>Compiler</th>
<th>Spack commit</th>
<th>Spack.yml</th>
<th>Installed Specs</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>E4S Version</th>
<th>Project URL (Github, Gitlab)</th>
<th>Installed Specs</th>
<th>Compiler</th>
<th>Spack commit</th>
<th>Spack.yml</th>
<th>Installed Specs</th>
</tr>
</thead>
</table>

### Articus

<table>
<thead>
<tr>
<th>E4S Version</th>
<th>Project URL (Github, Gitlab)</th>
<th>Installed Specs</th>
<th>Compiler</th>
<th>Spack commit</th>
<th>Spack.yml</th>
<th>Installed Specs</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>E4S Version</th>
<th>Project URL (Github, Gitlab)</th>
<th>Installed Specs</th>
<th>Compiler</th>
<th>Spack commit</th>
<th>Spack.yml</th>
<th>Installed Specs</th>
</tr>
</thead>
</table>
E4S Test Suite

- The **E4S Test Suite** is a collection of tests to validate E4S stack and increase test coverage for deployed stack.

- The main script **test-all.sh** can be run as standalone program which will test everything or you can specify an argument to a directory of tests to run.
E4S Testsuite on Cori

```
siddiq90@cori04 > cat testsuite.sh
#!/bin/bash
 git clone https://github.com/E4S-Project/testsuite.git
cd testsuite
 module load e4s/21.05
 source ./setup.sh
 ./test-all.sh validation_tests/gasnet --settings settings.cori.sh

siddiq90@cori04 > sh testsuite.sh
Cloning into 'testsuite'...
remote: Enumerating objects: 18242, done.
remote: Counting objects: 100% (843/843), done.
remote: Compressing objects: 100% (523/523), done.
remote: Total 18242 (delta 494), reused 576 (delta 307), pack-reused 17399
Receiving objects: 100% (18242/18242), 35.07 MiB | 20.45 MiB/s, done.
Resolving deltas: 100% (8858/8858), done.
Updating files: 100% (14090/14090), done.
===
validation_tests/gasnet
sbw7ukx
Cleaning /global/homes/s/siddiq90/testsuite/validation_tests/gasnet
Compiling /global/homes/s/siddiq90/testsuite/validation_tests/gasnet
Running /global/homes/s/siddiq90/testsuite/validation_tests/gasnet
 Success

siddiq90@cori04 > spack find --format "{hash} {name}@{version}%{compiler}" gasnet
sbw7ukx5aixonvp60162axorzdve3lid gasnet@2021.3.0%intel@19.1.3.304
```
**spack test:** write tests directly in Spack packages, so that they can evolve with the software

```python
class Libsigsegv(AutotoolsPackage, GNUMirrorPackage):
    """GNU libsigsegv is a library for handling page faults in user mode."""

    extra_install_tests = 'tests/.libs'

def test(self):
    data_dir = self.test_suite.current_test_data_dir
    smoke_test_c = data_dir.join('smoke_test.c')

    self.run_test(
        'cc', [
            '-I%s' % self.prefix.include,
            '-L%s' % self.prefix.lib, '-lsigsegv',
            smoke_test_c,
            '-o', 'smoke_test'
        ],
        purpose='check linking')

    self.run_test(
        'smoke_test', [], data_dir.join('smoke_test.out'),
        purpose='run built smoke test')

    self.run_test('sigsegv1': ['Test passed'], purpose='check sigsegv1 output')
    self.run_test('sigsegv2': ['Test passed'], purpose='check sigsegv2 output')
```

Tests are part of a regular Spack recipe class

Easily save source code from the package

User just defines a `test()` method

Retrieve saved source. Link a simple executable.

Spack ensures that `cc` is a compatible compiler

Run the built smoke test and verify output

Run programs installed with package
# Spack Test Command Line Overview

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spack test list</td>
<td>List tests for installed packages</td>
</tr>
<tr>
<td>spack test list --all</td>
<td>List all tests for all spack packages</td>
</tr>
<tr>
<td>spack test run</td>
<td>Run test for all installed specs in environment or installed packages</td>
</tr>
<tr>
<td>spack test run hdf5</td>
<td>Run test for spack package hdf5</td>
</tr>
<tr>
<td>spack test run --alias hdf5 hdf5@1.10.7</td>
<td>Run test for hdf5@1.10.7 and assign alias for suite name hdf5</td>
</tr>
<tr>
<td>spack test results</td>
<td>Show results for all test suites</td>
</tr>
<tr>
<td>spack test results – hdf5@1.10.7</td>
<td>Show test results for spec hdf5@1.10.7</td>
</tr>
<tr>
<td>spack test results &lt;suite-name&gt;</td>
<td>Show test results for suite name</td>
</tr>
<tr>
<td>spack test remove -y</td>
<td>Remove all test results and assume ‘yes’ for each confirmation</td>
</tr>
</tbody>
</table>

```bash
isiddig@004-cori04> spack test list
-- cray-cnl17-haswell / intel@19.1.3.304
---------------------------------------
ud22bud arborx@1.0   j707wmp hypre@0.20.0   wuanopr parallel-netcdf@1.12.2
vfd2jhd binutils@33.1  z5okxptz kokkos@3.4.00  cz61xgn py-libensemble@0.7.2
opjamiu cmake@0.20.2  h126wxt kokkos@3.4.00  fxlw6ij py-mpi4py@3.0.3
sbw7ukx gasnet@2021.3.0 x3kwocp libsigsegv@0.12 mbedwvv mpee bkm libxml2@2.9.10
p4ltqk3 ginkgo@1.3.0 mpee bkm libxml2@2.9.10 fiou2u
36pueem hdf5@1.10.7  oguh3so m4@1.4.10  lqa4mv6
tv6mktz hdf5@1.10.7  21qgyya mfem@4.2.0  vkh4hzt
```

---

- **spack test list**: Lists tests for installed packages.
- **spack test list --all**: Lists all tests for all spack packages.
- **spack test run**: Runs test for all installed specs in environment or installed packages.
- **spack test run hdf5**: Runs test for spack package hdf5.
- **spack test run --alias hdf5 hdf5@1.10.7**: Runs test for hdf5@1.10.7 and assigns an alias for the suite name hdf5.
- **spack test results**: Shows results for all test suites.
- **spack test results – hdf5@1.10.7**: Shows test results for spec hdf5@1.10.7.
- **spack test results <suite-name>**: Shows test results for suite name.
- **spack test remove -y**: Removes all test results and assumes 'yes' for each confirmation.
Running Tests via spack test run

```
siddiq90@cori04> spack test run hdf5
  ==> Spack test st5uy2i326u2y7syw4wwmrpwk2233wta
  ==> Testing package hdf5-1.8.22-36pueen
  ==> Testing package hdf5-1.10.7-v5mkktz

siddiq90@cori04> spack test results
  ==> Results for test suite 'st5uy2i326u2y7syw4wwmrpwk2233wta':
  ==> hdf5-1.8.22-36pueen PASSED
  ==> hdf5-1.10.7-v5mkktz PASSED
```

```
siddiq90@cori04> spack test run --alias py-warpx py-warpx
  ==> Spack test py-warpx
  ==> Testing package py-warpx-21.05-yohkwul
  ==> Testing package py-warpx-21.05-2dopy2n
  ==> Testing package py-warpx-21.05-ownfza6

siddiq90@cori04> spack test results
  ==> Results for test suite 'py-warpx':
  ==> py-warpx-21.05-yohkwul PASSED
  ==> py-warpx-21.05-2dopy2n PASSED
  ==> py-warpx-21.05-ownfza6 PASSED
```
Retrieve Test Results via spack test results

```bash
siddiq90@cori04> spack test results

>>> Results for test suite 'st5uy2i326u2y7syw4wmmrpwk2233wta':
   hdf5-1.8.22-36pueen PASSED
   hdf5-1.18.7-v5mkktz PASSED

>>> Results for test suite 'zevm73tlterufszagrkesihnjox7kr':
   m4-1.4.18-oguh3so PASSED

siddiq90@cori04> spack test results -- m4

>>> Results for test suite 'zevm73tlterufszagrkesihnjox7kr', spec matching 'm4':
   m4-1.4.18-oguh3so PASSED

siddiq90@cori04> spack test results -- m4

>>> Results for test suite 'zevm73tlterufszagrkesihnjox7kr', spec matching 'm4':
   m4-1.4.18-oguh3so PASSED

>>> Testing package m4-1.4.18-oguh3so

>>> [2021-08-13-10:15:32.753276] '/global/common/software/spackecp/e4s-21.06/software/cray-cnl7-haswell/intel-19.1.3.304/m4-1.4.18-oguh3soitsu3ym3fb4pdy7p7jffthl7am/bin/m4' '--version'
m4 (GNU M4) 1.4.18
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Written by Rene' Seindal.
PASSED

>>> [2021-08-13-10:15:32.847466] test: ensuring m4 example succeeds

```
Facility Testing Use Case

- We need comprehensive system and software level testing
- We need to test facility deployment of e4s stacks that are usually tied to fix version of spack.
- Test system layer – configuration, filesystem, job scheduler, drivers, modules
- Run benchmark
- Negative tests to address known bugs in system
- User provided tests as part of User Support Tickets
What is buildtest

- Buildtest is a testing framework that builds and execute tests on your HPC systems
- Buildtest is intended for HPC staff, developers to build a **Facility Testsuite** for their HPC systems
- Tests are written in YAML called **buildspecs** which buildtest process to generate shell scripts.
- Buildspecs are validated with JSON schema.
- Support test execution on local machine or via batch schedulers. Currently we support Slurm, LSF, PBS and Cobalt.
- The framework is implemented in python
- Available on GitHub at [https://github.com/buildtesters/buildtest](https://github.com/buildtesters/buildtest)

**Installation**

```
$ git clone https://github.com/buildtesters/buildtest.git
$ cd buildtest
$ source setup.sh
```
Project Summary

• Total of 37 releases
• Added spack support in v0.10.0
• Added support for Slurm and LSF in v0.8.0, Cobalt in v0.9.1 and PBS in v0.9.5
• Initially a bash program which was converted to python 2 and eventually migrated to python 3.
• Release updates are documented in CHANGELOG.rst
• Distributed as MIT License
• Documentation is built using sphinx and hosted via readthedocs platform
Design Goals

• Perform component level testing for system and software stack
• Provide a standard template for writing tests
• Abstract low-level system configuration
• Framework should automate build and execution of test
• Framework must support local and batch submission test
Schemas

- The schema development is implemented independent to buildtest. The schemas and docs are hosted at https://buildtesters.github.io/buildtest/
- We run regression test against example YAML files for each schema to ensure schemas are written in accordance to desired YAML construct.
- We automate JSON Schema documentation using adobe/jsonschema2md into Markdown pages and publish schema and documentation to GitHub pages.
- Schemas are versioned to allow development to schemas and its YAML structure.
## Preview of buildtest

### Building Buildspecs

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buildtest build -b &lt;file&gt;</td>
<td>Build a single buildspec file</td>
</tr>
<tr>
<td>buildtest build -b &lt;dir&gt;</td>
<td>Build all buildspecs recursively in a given directory</td>
</tr>
<tr>
<td>buildtest build --exclude &lt;file&gt;</td>
<td>Build buildspecs by file and directory</td>
</tr>
<tr>
<td>buildtest build --exclude &lt;dir&gt;</td>
<td>Exclude files and directory when building buildspecs</td>
</tr>
<tr>
<td>buildtest build -t pass -t python</td>
<td>Build buildspecs by tagname 'pass' and 'python'</td>
</tr>
<tr>
<td>buildtest build -e &lt;executor&gt;</td>
<td>Building buildspecs by executor</td>
</tr>
<tr>
<td>buildtest build -b &lt;file&gt; -e &lt;executor&gt;</td>
<td>Building buildspecs with file, directory, tags, and executors</td>
</tr>
<tr>
<td>buildtest build -b tutorials --filter type=script</td>
<td>Build all tests in directory 'tutorials' and filter tests by type='script'</td>
</tr>
<tr>
<td>buildtest build -b tutorials --filter tags=pass</td>
<td>Build all tests in directory 'tutorials' and filter tests by tags='pass'</td>
</tr>
<tr>
<td>buildtest build -b tutorials --filter maintainers=bob</td>
<td>Build all tests in directory 'tutorials' and filter tests by maintainers='bob'</td>
</tr>
<tr>
<td>buildtest build --help</td>
<td>Show list of filter fields used with --filter option</td>
</tr>
<tr>
<td>buildtest -c config.yml build -b &lt;file&gt;</td>
<td>Use buildtest configuration file 'config.yml'</td>
</tr>
<tr>
<td>buildtest build -b &lt;file&gt; --rebuild 5</td>
<td>Rebuild a test 5 times</td>
</tr>
<tr>
<td>buildtest build -b &lt;file&gt; --testdir /tmp</td>
<td>Write tests in /tmp</td>
</tr>
</tbody>
</table>
General Pipeline

- **Discover**: Find buildspecs based on search criteria (file, directory, tags, executor)
- **Parse**: Validates buildspec with JSON Schema
- **Build**: Generates testscript from YAML
- **Run**: Executes tests via local or batch executor and retrieve return code and output/error file.
- **Update Report**: Update report file with test results including any metadata
Every buildspec is validated by global schema and a subschema defined by type field.

Buildtest will skip any buildspecs that fails validation.
Demo – Buildtest Tutorial
# Buildspec Script Schema

<table>
<thead>
<tr>
<th>Name of Test</th>
<th>Schema Type</th>
<th>Tag Name</th>
<th>Description of Test</th>
<th>Script</th>
</tr>
</thead>
</table>
| systemd_default_target: | script | [system] | check if default target is multi-user.target | ```
version: "1.0"
buildspecs:
systemd_default_target:
  executor: generic.local.bash
type: script
tags: [system]
description: check if default target is multi-user.target
run: |
  if [ "multi-user.target" == `systemctl get-default` ]; then
    echo "multi-user is the default target";
    exit 0
  fi
  echo "multi-user is not the default target";
  exit 1
``` |

"$id": "script-v1.0.schema.json",
"$schema": "http://json-schema.org/draft-07/schema#",
"title": "script schema version 1.0",
"description": "The script schema is of `\"type: script\"` in sub-schema which is used for running shell scripts",
"type": "object",
"required": ["type", "run", "executor"],
"additionalProperties": false,
Status Check – Regular Expression

- Buildtest supports status check of test based on regular expression, returncode and runtime. This can be configured via **status** property
Status Check – Return Code

- The return code field can be used to customize how test is passed, by default a return code 0 is a **PASS**. The return code can be a single number or a list of return codes to match.
Status Check - Runtime

- Buildtest can determine PASS/FAIL based on test runtime. This can be specified using `runtime` property with options for specifying `min` or `max` or both if one wants to set a range.

### Table

<table>
<thead>
<tr>
<th>name</th>
<th>id</th>
<th>executor</th>
<th>status</th>
<th>returncode_match</th>
<th>regex_match</th>
<th>runtime_match</th>
<th>returncode</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>timelimit_min</td>
<td>dca84980</td>
<td>generic.local.sh</td>
<td>PASS</td>
<td>False</td>
<td>True</td>
<td>0</td>
<td>2.67962</td>
<td></td>
</tr>
<tr>
<td>timelimit_min</td>
<td>0a0b1087</td>
<td>generic.local.sh</td>
<td>PASS</td>
<td>False</td>
<td>True</td>
<td>0</td>
<td>2.83340</td>
<td></td>
</tr>
<tr>
<td>timelimit_min</td>
<td>3a0b18f6</td>
<td>generic.local.sh</td>
<td>PASS</td>
<td>False</td>
<td>True</td>
<td>0</td>
<td>2.83436</td>
<td></td>
</tr>
<tr>
<td>timelimit_min</td>
<td>839e4576</td>
<td>generic.local.sh</td>
<td>FAIL</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>2.39084</td>
<td></td>
</tr>
<tr>
<td>timelimit_min</td>
<td>d8aad7ca</td>
<td>generic.local.sh</td>
<td>FAIL</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>3.25581</td>
<td></td>
</tr>
</tbody>
</table>

### Code Snippets

```yaml
version: "1.0"
buildspecs:
timelimit_min:
type: script
executor: generic.local.sh
description: "Run a sleep job for 2 seconds and test pass if its within 1.0-3.0sec"
tags: ["tutorial"]
runtime: sleep 2
status:
  runtime:
    min: 1.0
    max: 3.0

timelimit_max:
type: script
executor: generic.local.sh
description: "Run a sleep job for 2 seconds and test pass if it’s exceeds min time of 1.0 sec"
tags: ["tutorial"]
runtime: sleep 2
status:
  runtime:
    max: 5.0

timelimit_min.fail:
type: script
executor: generic.local.sh
description: "This test fails because it runs less than min time of 10 second"
tags: ["tutorial"]
runtime: sleep 2
status:
  runtime:
    min: 10.0

timelimit_max.fail:
type: script
executor: generic.local.sh
description: "This test fails because it exceeds maxtime of 1.0 second"
tags: ["tutorial"]
runtime: sleep 3
status:
  runtime:
    max: 1.0
```
Multi Executors

- Every test must be assigned to an executor that is responsible for running test. This is specified via `executor` property however one can specify a regular expression to run across multiple executors.

- The `executors` property can be used to define configuration based on executor that are specific to each test run.

- The `vars` and `env` are used for declaring variables and environment variables which expects a list of Key/Value pair.

- Executors are defined in your configuration file which can be retrieved via `buildtest config executors`.

- In this example we run a single test with executor `generic.local.sh` and `generic.local.bash`.
Buildspec Compiler Schema

- The compiler schema is used for compiling single source code with compilers
- You must use **type: compiler** to define tests using this schema
- This test will be built with all gcc compilers
- Compilers are defined in buildtest configuration, one can retrieve compilers using **buildtest config compilers**

```json
"type" : "compiler",
"description" : "Vector Addition example with GNU compiler",
"tags" : [ "tutorials", "compile" ],
"executor" : "generic.local.bash",
"source" : "src/vecAdd.c",

"compilers" : {
  "name" : "\{(builtin_gcc|gcc)\}"
}
```

$ buildtest config compilers -l
builtin_gcc
gcc/9.3.0-n7p74fd
gcc/10.2.0-37fmsw7
Override Compiler Default

- Compiler defaults can be overridden in `config` section which expects compiler names defined in buildtest setting.
- Buildtest will ignore compiler in `config` if it’s not picked up in regular expression.

### Compiler Names

```bash
version: "1.0"
builddir:...

hello_c:
  type: compiler
description: "Hello World C Compilation"
executor: generic.local.bash
tags: [tutorials, compile]
source: "src/hello.c"

compilers:
  name: ["^\b(gcc|gfortran)\b"]
    default:
      gcc:
        cflags: -O1
    config:
      gcc/9.3.0-n7p74fd:
        cflags: -O2
      gcc/10.2.0-37fmsw7:
        cflags: -O3
```
Scheduler Support

**Slurm**

```bash
version: "1.0"
bUILDSPECs:
slurm_metadata:
  description: Get metadata from compute node when submitting job
type: script
  executor: cori.slurm.debug
bash:
  - "-t 00:05"
  - "-C haswell"
  - "-N 1"
run:
  export SLURM_JOB_NAME="firstjob"
  echo "jobname": $SLURM_JOB_NAME
  echo "slurmdb_host": $SLURM_NODE_NAME
  echo "pid": $SLURM_TASK_PTD
  echo "submit host": $SLURM_SUBMIT_HOST
  echo "nodeid": $SLURM_NODEID
  echo "partition": $SLURM_JOB_PARTITION
```

**LSF**

```bash
version: "1.0"
bUILDSPECs:
hostname:
  type: script
  executor: ascte.lsf.batch
bsub: "-W 10", "-n nodes 1"
run: jrun hostname
```

**Cobalt**

```bash
version: "1.0"
bUILDSPECs:
yarrow_hostname:
  executor: jlse.cobalt.yarrow
type: script
  cobalt: ["-n 1", "-proccount 1", "-t 10"]
run: hostname
```
Max Pend Time and Poll Interval

• Buildtest will poll batch jobs at set interval to get updated job state for all jobs in queue, once job is complete buildtest will gather job results and metadata of job.

• The **pollinterval** property configures number of seconds to sleep until we poll jobs for updated job state. This value can be overridden on command line via `buildtest build --poll-interval`

• Buildtest will cancel pending or suspended jobs after pending time exceeds **max_pend_time**. This value can be overridden via `buildtest build --max-pend-time`

```bash
| Stage: Running Test |

Launching test: pbs_hold_job
Test ID: b6138507-6f36-4401-b88c-2f688c328dbb
Executor Name: generic.pbs_workq
Running Test: /tmp/GithuADesktop/buildtest/var/tests/generic.pbs_workq/hold/pbs_hold_job/b6138507/pbs_hold_job_build.sh
(pbs_hold_job) JobID: 391.pbs dispatched to scheduler
Polling Jobs in 8 seconds

Current Jobs
---------------------
<table>
<thead>
<tr>
<th>job</th>
<th>id</th>
<th>executor</th>
<th>jobID</th>
<th>jobstate</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>pbs_hold_job</td>
<td>b6138507</td>
<td>generic.pbs_workq</td>
<td>391.pbs</td>
<td>M</td>
<td>0.158</td>
</tr>
</tbody>
</table>

Polling Jobs in 8 seconds
pbs_hold_job/b6138507: Cancelling Job: 391.pbs because job exceeds max pend time: 18 sec with current pend time of 16.208
Cancelled Jobs: [pbs_hold_job/b6138507]
Unable to run any tests
```
Demo – Buildspecs Tutorial
Filter and Format buildspec cache

- We can filter and format buildspec cache using `--filter` and `--format` option.
- The filter option expects a list of `key=value` pair separated by comma.
- To see list of all filter and format fields we can use `--helpfilter` and `--helpformat` option.

```
$ buildtest buildspec find --filter tags=fail

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>executor</th>
<th>tags</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exit_fall</td>
<td>script</td>
<td>generic.local.sh</td>
<td>['tutorials', 'fail']</td>
<td>exit 1 by default is FAIL</td>
</tr>
<tr>
<td>returncode_list_mismatch</td>
<td>script</td>
<td>generic.local.sh</td>
<td>['tutorials', 'fail']</td>
<td>exit 2 failed since it failed to match returncode 1</td>
</tr>
</tbody>
</table>
```

```
$ buildtest buildspec find --filter tags=fail --format name,tags

<table>
<thead>
<tr>
<th>name</th>
<th>tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>exit_fall</td>
<td>['tutorials', 'fail']</td>
</tr>
<tr>
<td>returncode_list_mismatch</td>
<td>['tutorials', 'fail']</td>
</tr>
</tbody>
</table>
```

```
$ buildtest buildspec find --filter tags=tutorials,executor=generic.local.sh,type=script

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>executor</th>
<th>tags</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_bin_sh_shell</td>
<td>script</td>
<td>generic.local.sh</td>
<td>['tutorials']</td>
<td>/bin/sh shell example</td>
</tr>
<tr>
<td>sh_shell</td>
<td>script</td>
<td>generic.local.sh</td>
<td>['tutorials']</td>
<td>sh shell example</td>
</tr>
<tr>
<td>shell_options</td>
<td>script</td>
<td>generic.local.sh</td>
<td>['tutorials']</td>
<td>shell options</td>
</tr>
<tr>
<td>exit1_fail</td>
<td>script</td>
<td>generic.local.sh</td>
<td>['tutorials', 'fail']</td>
<td>exit 1 by default is FAIL</td>
</tr>
<tr>
<td>exit1_pass</td>
<td>script</td>
<td>generic.local.sh</td>
<td>['tutorials', 'pass']</td>
<td>report exit 1 as PASS</td>
</tr>
<tr>
<td>returncode_list_mismatch</td>
<td>script</td>
<td>generic.local.sh</td>
<td>['tutorials', 'fail']</td>
<td>exit 2 failed since it failed to match returncode 1</td>
</tr>
<tr>
<td>returncode_int_match</td>
<td>script</td>
<td>generic.local.sh</td>
<td>['tutorials', 'pass']</td>
<td>exit 128 matches returncode 128</td>
</tr>
</tbody>
</table>
```

Multi key filter is evaluated as logical AND.
Show content of buildspec

- The `buildtest buildspec show` command can show content of buildspec based on a given test name. In this example we show content of test `python_hello`

```
(buildtest) bash-3.2$ buildtest buildspec show python_hello
version: "1.0"
buildspecs:
    python_hello:
        type: script
        description: Hello World python
        executor: generic.local.bash
        tags: python
        run: python hello.py

buildspec: /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/python-hello.yml
```
Validate Buildspecs

- The `buildtest buildspec validate` command can be used to validate buildspecs with JSON schema and command options mimic similar to `buildtest build`.

```
(buildtest) bash-3.2$ buildtest buildspec validate -t python
Processing buildspec: /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/python-shell.yml
Processing buildspec: /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/python-hello.yml
All buildspecs passed validation!!!
```

```
(buildtest) bash-3.2$ buildtest buildspec validate -b tutorials/invalid_tags.yml
file: /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/invalid_tags.yml

['network', 'network'] is not valid under any of the given schemas

Failed validating 'oneOf' in schema['properties'][tags]:
  oneOf: [type: string],
    ref: '#/definitions/list_of_strings']

On instance[tags]:
  [network, network]
```

```
Processing buildspec: /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/invalid_tags.yml
There were 1 buildspecs that failed validation
```
Show all invalid buildspecs

• Buildtest will keep record of all invalid buildspecs in the cache upon running `buildtest buildspec find`, you can retrieve a list of all invalid buildspecs via `buildtest buildspec find invalid` command.

• The `-e` option will print all error messages for every invalid buildspecs

• If you want to load all buildspecs in cache and fix invalid buildspecs then `buildtest buildspec find invalid` would be appropriate, however if you want to validate any buildspec without loading in cache you can use `buildtest buildspec validate`

```
(buildtest) bash-3.2$ buildtest buildspec find invalid

```

```
Invalid value for type field

/Users/siddiq98/Documents/GitHubDesktop/buildtest/tutorials/invalid_buildspecs_section.yml

Can't have duplicate tag names for tags property

/Users/siddiq98/Documents/GitHubDesktop/buildtest/tutorials/invalid_tags.yml
```

```
(buildtest) bash-3.2$ buildtest buildspec find invalid -e
/Users/siddiq98/Documents/GitHubDesktop/buildtest/tutorials/invalid_buildspecs_section.yml

''[/Users/siddiq98/Documents/GitHubDesktop/buildtest/tutorials/invalid_buildspecs_section.yml]: type bash_script is not known to buildtest.'

```

```
/Users/siddiq98/Documents/GitHubDesktop/buildtest/tutorials/invalid_tags.yml

''['network', 'network'] is not valid under any of the given schemas
Failed validating 'oneOf' in schema['properties']['tags']:
  {'oneOf': [{'type': 'string'},
    {'$ref': '#/definitions/list_of_strings'}]}
On instance['tags']:
  ['network', 'network']
```

```
Report Summary

(buildtest) bash-3.2$ buildtest report summary
Total Tests: 6
Total Tests by Names: 6
Number of buildspecs in report: 3

### Breakdown by Test

<table>
<thead>
<tr>
<th>name</th>
<th>runs</th>
<th>pass</th>
<th>fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>variables_bash</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>sleep</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>exit1_fail</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>exit1_pass</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>returncode_list_mismatch</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>returncode_int_match</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### FAIL test

<table>
<thead>
<tr>
<th>name</th>
<th>id</th>
<th>executor</th>
<th>state</th>
<th>returncode</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>exit1_fail</td>
<td>996d9773</td>
<td>generic.local.sh</td>
<td>FAIL</td>
<td>1</td>
<td>0.098565</td>
</tr>
<tr>
<td>returncode_list_mismatch</td>
<td>6049eea8</td>
<td>generic.local.sh</td>
<td>FAIL</td>
<td>2</td>
<td>0.107125</td>
</tr>
</tbody>
</table>
We provide access to test reports through CLI. The reports are stored in JSON file for post-processing.

The buildtest report will display all test results which can be queried with filter and format options.

The --filter option are passed as key=value pair

Multiple filter arguments can be delimited by comma separator and buildtest will treat multiple filter argument as a logical AND operation

The --format option alter the columns in the report tables.
Format And Filter fields for buildtest report

- The **buildtest report** command provides a description of format and filter fields using **–helpformat** and **–helpfilter**

- These fields are lookup keys found in report file, we only expose a subset of these fields suitable for printing purposes

```
$ buildtest report --helpformat
Fields | Description
-------|--------------------------------------
buildspec | Buildspec file
command | Command executed
compiler | Retrieve compiler used for test (applicable for compiler schema)
endtime | End Time for Test in date format
errfile | Error File
executor | Executor name
hostname | Retrieve hostname of machine where job was submitted from
full_id | Full qualified unique build identifier
id | Unique Build Identifier (abbreviated)
name | Name of test defined in buildspec
outfile | Output file
returncode | Return Code from Test Execution
runtime | Total runtime in seconds
schemafile | Schema file used for validation
starttime | Start Time of test in date format
state | Test State reported by buildtest (PASS/FAIL)
tags | Tag name
testroot | Root of test directory
testpath | Path to test
user | Get user who submitted job

$ buildtest report --helpfilter
Filter Fields | Description | Expected Value
--------------|-------------|----------------
buildspec | Filter by buildspec file | FILE
name | Filter by test name | STRING
executor | Filter by executor name | STRING
state | Filter by test state | PASS/FAIL
tags | Filter tests by tag name | STRING
returncode | Filter tests by returncode | INT
```
Inspect a Test

- Buildtest stores all test results in JSON file ($BUILDTEST_ROOT/var/report.json) for retrieval

- The **buildtest inspect** command can retrieve test records from this file.

- We can retrieve all test names and corresponding test IDs using **buildtest inspect list**

- The **buildtest inspect name** can retrieve test records based on test names including all previous runs

- You can pass multiple test names to **buildtest inspect name** command to query multiple records
Inspect a Test Record

```
{(buildtest) bash-3.2$ buildtest inspect name sleep

"sleep": {
  "id": "b0d850d0",
  "full_id": "b0d850d0-04b7-43d1-bd1a-cce0c480a214",
  "description": "sleep 2 seconds",
  "schemafile": "script-v1.0.schema.json",
  "executor": "generic.local.bash",
  "compiler": null,
  "hostname": "DOE-7086392.local",
  "user": "siddiq90",
  "testroot": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/sleep/sleep/b0d850d0",
  "testpath": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/sleep/sleep/b0d850d0/sleep.sh",
  "stagedir": "Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/sleep/sleep/b0d850d0/stage",
  "command": "sh sleep_build.sh",
  "outfile": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/sleep/sleep/b0d850d0/sleep.out",
  "errfile": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/sleep/sleep/b0d850d0/sleep.err",
  "buildspec_content": "version: "1.0"

buildspec:
  type: script
  executor: generic.local.bash
  description: sleep 2 seconds
  vars:
    $SLEEP_TIME: 2
  run: sleep $SLEEP_TIME

test_content: 
"#!/bin/bash

declare shell variables

SLEEP_TIME=2

Content of run section

sleep $SLEEP_TIME

buildscript_content: 
"#/bin/bash

source

Run gene

rate

script

source

Run

next

return
code

0"}

logpath: "/var/folders/1m/_jyw9917k37mkktwmbkmj0002t_q/T/buildtest_04rwo4dk.log",
metrics: {},
"tags": [],
"starttime": "2021/08/16 16:37:35",
"endtime": "2021/08/16 16:37:38",
"runtime": "2.0054989",
"state": "PASS",
"returncode": "0",
"output": "",
"error": "",
"job":{}

build_script": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/sleep/sleep/b0d850d0/sleep_build.sh"
}
```
The **buildtest inspect query** command can be used to query individual test records.

By default, it will retrieve the latest run for given test, however one can use `-d` option to retrieve all records or first or last record.
buildtest inspect query

bash-3.2$ buildtest inspect query --output python_hello
---
executor: generic.local.bash
description: Hello World python
state: PASS
runtime: 0.27114
starttime: 2021/08/25 20:44:49
datetime: 2021/08/25 20:44:50
Start of Output File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.out
End of Output File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.out

bash-3.2$ buildtest inspect query --test --output --error python_hello
---
executor: generic.local.bash
description: Hello World python
state: PASS
runtime: 0.27114
starttime: 2021/08/25 20:44:49
datetime: 2021/08/25 20:44:50
Start of Output File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.out
End of Output File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.out
Start of Error File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.err
End of Error File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.err
Start of Test Path: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.sh
End of Test Path: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.sh

# Content of run section
python_hello.py
---
End of Output File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.out
End of Error File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.err
Start of Test Path: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.sh
End of Test Path: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.sh

bash-3.2$ buildtest inspect query --output python_hello
---
executor: generic.local.bash
description: Hello World python
state: PASS
runtime: 0.27114
starttime: 2021/08/25 20:44:49
datetime: 2021/08/25 20:44:50
Start of Output File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.out
End of Output File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.out

bash-3.2$ buildtest inspect query --test --output --error python_hello
---
executor: generic.local.bash
description: Hello World python
state: PASS
runtime: 0.27114
starttime: 2021/08/25 20:44:49
datetime: 2021/08/25 20:44:50
Start of Output File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.out
End of Output File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.out
Start of Error File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.err
End of Error File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.err
Start of Test Path: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.sh
End of Test Path: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.sh

# Content of run section
python_hello.py
---
End of Output File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.out
End of Error File: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.err
Start of Test Path: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.sh
End of Test Path: /Users/siddiq98/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/python-hello/python_hello/5b5a4a26/python_hello.sh
Demo – Buildspec Interface & Query Test Report
Get Path to tests

- The **buildtest path** command can retrieve path to test given a test name. If no options are specified we retrieve the root where test is available.

- You can specify a test ID by specifying name followed by `backslash (/)` and name of test ID if its not specified buildtest will fetch the latest run.

```bash
(buildtest) bash-3.2$ buildtest path shell_options
/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.sh/shell_examples/shell_options/f91f29d6

(buildtest) bash-3.2$ buildtest path shell_options/b89
/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.sh/shell_examples/shell_options/b890cd6d

(buildtest) bash-3.2$ cat $(buildtest path -t shell_options)
#!/bin/sh -x
# Content of run section
echo $SHELL
```
Query Previous Builds

- The **buildtest history query** command can be used to query previous builds based on build identifier. Every **buildtest build** command will be stored as a new build identifier.

```
(buildtest) bash-3.2# buildtest history query 0
{
  "command": "/Users/siddiq89/Documents/GitHubDesktop/bin/buildtest build -0 /Users/siddiq89/Documents/GitHubDesktop/buildtest/tutorials/vars.yml",
  "user": "siddiq89",
  "hostname": "DOE-7886392.local.dhcp.lbl.gov",
  "platform": "Darwin",
  "date": "2021/08/25 20:43:45",
  "buildtest": "/Users/siddiq89/Documents/GitHubDesktop/bin/buildtest",
  "python": "/Users/siddiq89/Local/shares/virtualenvs/buildtest-KLdJw9W/bin/python",
  "python_version": "3.7.3",
  "testdir": "/Users/siddiq89/Documents/GitHubDesktop/buildtest/var/tests",
  "configuration": "/Users/siddiq89/Documents/GitHubDesktop/buildtest/buildtest/settings/config.yml",
  "system": "generic",
  "logpath": "/Users/siddiq89/Documents/GitHubDesktop/buildtest/var/history/0/buildtest_64v_jp2b.leg",
  "invalid_buildspecs": []
}

```

```
"buildspecs": {
  "detected": [
    "/Users/siddiq89/Documents/GitHubDesktop/buildtest/tutorials/vars.yml"
  ],
  "included": [
    "/Users/siddiq89/Documents/GitHubDesktop/buildtest/tutorials/vars.yml"
  ],
  "excluded": []
},

"test_summary": {
  "pass": "1",
  "fail": "0",
  "total": "1",
  "pass_rate": "100.000%",
  "fail_rate": "0.000%"
}

"builders": {
  "d1ae7f7d1-38b4-4b07-b142-aa44574eabc": {
    "name": "variables_bash",
    "buildspec": "/Users/siddiq89/Documents/GitHubDesktop/buildtest/tutorials/vars.yml",
    "tags": []
  },
  "executors": "generic.local.bash",
  "state": "PASS",
  "returncode": 0,
  "runTime": 0.549762,
  "testpath": "/Users/siddiq89/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/vars/variables_bash/d1ae7f7d1/variables_bash.sh",
  "errfile": "/Users/siddiq89/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/vars/variables_bash/d1ae7f7d1/variables_bash.err",
  "outfile": "/Users/siddiq89/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/vars/variables_bash/d1ae7f7d1/variables_bash.out"
}
```
Spack support in buildtest

- Spack support was added recently in buildtest v0.10.0 to write buildspecs using the spack schema.
- Current support includes
  - Installing specs
  - Managing environments (create, activate, remove)
  - `spack test` support
  - Specify scheduler options.
- For more details on spack support see https://buildtest.readthedocs.io/en/devel/buildspecs/spack.html

```bash
#!/bin/bash
source /Users/siddiq90/spack/share/spack/setup-env.sh
spack install zlib
```

```
version: "1.0"
buildspecs:
  install zlib:
    type: spack
    executor: generic.local.sh
    description: "Install zlib"
    tags: [spack]
    spack:
      root: $HOME/spack
      install:
        specs: ['zlib']
```

Use Spack schema

Root of spack

List of specs to install
Activate Spack environment

• The `env` property is used for managing spack environment and maps to `spack env` command

• The `activate` property maps to `spack env activate` used for activating named environment

• `concretize` is a boolean type that will determine if `spack concretize -f` will be injecting in the test.
Create spack environment

- The **create** is a property under **env** that is used for creating spack environment.

- User is responsible for activate spack environment upon creation.

- The **install** property maps to **spack install** and one can pass options via option property.

```bash
#!/bin/bash
source /Users/siddiq90/spack/share/spack/setup-env.sh
spack compiler find
spack env create m4_zlib
spack env activate m4_zlib
spack add m4
spack add zlib
spack concretize -f
spack install --keep-prefix
```

**Find spack compilers**

**Create spack environment**

Find spack compilers

Create spack environment

Pass options to spack install
Creating spack environment via spack.yaml

- We can create spack environment based on `spack.yaml` which can be specified via `manifest` property which expects path to spack.yaml file.

```yaml
version: "1.0"
buildspecs:
  spack_env_create_from_manifest:
    type: spack
    executor: generic.local.sh
    description: "Create spack environment from spack.yaml"
    tags: [spack]
    spack:
      root: $HOME/spack
    env:
      create:
        name: 'manifest_example'
        manifest: "$BUILDTEST_ROOT/tutorials/spack/example/spack.yaml"
      activate:
        name: 'manifest_example'
        concretize: true
```
Remove spack environment

• Buildtest provides two methods for removing spack environment, one is via `rm` property which gives user control over how to remove spack environment. The alternative is let builtest automatically remove environment which can be specified via `remove_environment` which expects a boolean.

• The `remove_environment` is a property under `create` while `rm` is property under `env` which maps to `spack env rm`.

• The `remove_environment` will remove environment based on `name` property

```bash
#!/bin/bash
source /Users/siddiqi98/spack/share/spack/setup-env.sh
spack env rm -y remove_environment
spack env create remove_environment
spack env activate remove_environment
spack add bzip2
spack concretize -f
```

```bash
#!/bin/bash
source /Users/siddiqi98/spack/share/spack/setup-env.sh
spack env rm -y dummy
spack env create dummy
spack env activate dummy
spack add bzip2
spack concretize -f
```
Running test via `spack test`

- The `test` property maps to `spack test` command and `run` expects a list of specs to run that is defined via `specs`.

- Buildtest will write one line per `spack test run` and create an alias for each spec so one can retrieve the result via suite name.

- `pre_cmds` are list of commands run before sourcing `spack`

- `post_cmds` are list of commands run after `spack`

```
Fetch: 1.32s. Build: 52.35s. Total: 53.67s.

[+] /private/tmp/spack/opt/spack/darwin-big-sky-ral/apache-clang-11.0.3/diffutils-3.7-3dfrh6
    == Installing bzip2-1.0.8-avjwsvsoavulfugopw4ap7rffhejxzu
    == No binary for bzip2-1.0.8-avjwsvsoavulfugopw4ap7rffhejxzu found: installing from source
    == Fetching https://mirror.spack.io/_source-cache/archive/ab/ab5a03176ee106d3f0fa90e381da478d
    == Ran patch() for bzip2
    == bzip2: Executing phase: 'install'
    == bzip2: Successfully installed bzip2-1.0.8-avjwsvsoavulfugopw4ap7rffhejxzu

Fetch: 1.42s. Build: 1.84s. Total: 3.26s.

[+] /private/tmp/spack/opt/spack/darwin-big-sky-ral/apache-clang-11.0.3/bzip2-1.0.8-avjwsvsoa:
    == Spack test bzip2
    == Testing package bzip2-1.0.8-avjwsvso
    == Results for test suite 'bzip2':
    == bzip2-1.0.8-avjwsvso PASSED
    -- darwin-big-sky-ral / apache-clang@11.0.3
    bzip2@1.0.8
diffutils@3.7
libiconv@1.16
```

```
# /bin/bash

####### START OF PRE COMMANDS #######

  cd /tmp
  git clone https://github.com/spack/spack

####### END OF PRE COMMANDS       #######

source /private/tmp/spack-test-no-env/share/spack/setup-env.sh
spack install bzip2
spack test run --alias bzip2 bzip2
spack test results bzip2

####### START OF POST COMMANDS #######

  spack find
  rm -rf $SPACK_ROOT

####### END OF POST COMMANDS       #######
```
Cori Testsuite
Cori Testsuite

• The Cori Testsuite ([https://github.com/buildtesters/buildtest-cori](https://github.com/buildtesters/buildtest-cori)) is buildtest tests that run for Cori and Perlmutter system.

• There is a GitHub-GitLab CI/CD workflow to trigger pipeline at NERSC GitLab server: [https://software.nersc.gov](https://software.nersc.gov)

• Test results are pushed to CDASH at [https://my.cdash.org/index.php?project=buildtest-cori](https://my.cdash.org/index.php?project=buildtest-cori)

• Test are run with a single user name e4s
Scheduled Pipelines

- We have a mirror setup at https://software.nersc.gov/siddiq90/buildtest-cori

- Currently, we have two scheduled pipelines for daily system check and E4S tests
## Stage: Test Summary

<table>
<thead>
<tr>
<th>name</th>
<th>id</th>
<th>executor</th>
<th>status</th>
<th>returncode</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>filesystem_benchmarkCreates</td>
<td>2d2be9f1</td>
<td>cori.local.bash</td>
<td>PASS</td>
<td>0</td>
<td>11.3265</td>
</tr>
<tr>
<td>stream_uniprocess_c</td>
<td>4cc74d4f</td>
<td>cori.local.bash</td>
<td>PASS</td>
<td>1</td>
<td>1.63912</td>
</tr>
<tr>
<td>stream_openmp_c</td>
<td>c927ebc</td>
<td>cori.local.bash</td>
<td>PASS</td>
<td>0</td>
<td>4.45247</td>
</tr>
<tr>
<td>filesystem_benchmark_last</td>
<td>78f706c2</td>
<td>cori.local.bash</td>
<td>PASS</td>
<td>0</td>
<td>58.2977</td>
</tr>
<tr>
<td>mkl_intel_threaded_dgemm</td>
<td>85189a2e</td>
<td>cori.slurm.haswell_debug</td>
<td>PASS</td>
<td>0</td>
<td>283.427</td>
</tr>
<tr>
<td>mkl_gnu_threaded</td>
<td>47b31f7f</td>
<td>cori.slurm.haswell_debug</td>
<td>PASS</td>
<td>0</td>
<td>139.536</td>
</tr>
<tr>
<td>libsci_gnu_dgemm</td>
<td>6578df9f</td>
<td>cori.slurm.haswell_debug</td>
<td>PASS</td>
<td>0</td>
<td>333.814</td>
</tr>
<tr>
<td>mkl_intel_sequential_dgemm</td>
<td>74184c6d</td>
<td>cori.slurm.haswell_debug</td>
<td>PASS</td>
<td>0</td>
<td>582.243</td>
</tr>
</tbody>
</table>

Passed Tests: 8/8 Percentage: 100.0000%
Failed Tests: 0/8 Percentage: 0.0000%

Writing Logfile to: /tmp/buildtest_cmwuruk.log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /global/home/e/e4s/builds/5s9RF_SV/0/siddiq00/buildtest-cori/buildtest/buildtest.log
$ buildtest report --filter state=FAIL -r $BUILDTEST_ROOT/report.json
Reading report file: /global/u/e/e4s/builds/5s9RF_SV/0/siddiq00/buildtest-cori/buildtest/report.json

$ mkdir -p $CI_PROJECT_DIR/.artifacts
$ cp $BUILDTEST_ROOT/(buildtest.log,report.json) $CI_PROJECT_DIR/.artifacts
$ buildtest --CI_PROJECT_DIR/config.yml cdash upload benchmark -r $BUILDTEST_ROOT/report.json
Reading configuration file: /global/u/e/e4s/builds/5s9RF_SV/0/siddiq00/buildtest-cori/config.yml
Reading report file: /global/u/e/e4s/builds/5s9RF_SV/0/siddiq00/buildtest-cori/buildtest/report.json
build name: benchmark
site: cori
status: 20210401-1704-Experimental
MD5SUM: 1f691c23beac9f21c1371c59df2e8c3
PUT STATUS: 200
You can view the results at: https://my.cdash.org/viewTest.php?buildId=2963659
CDASH Results

- We push test results to public CDASH server: https://my.cdash.org/index.php?project=buildtest-cori for both scheduled pipelines.

- The build names correspond to GitLab job

- The `buildtest cdash upload` command can push results to CDASH given a report file. The report file can be passed via `-r` option

```
$ buildtest -c SCI_PROJECT_DIR/config.yml cdash upload -r $BUILDTEST_ROOT/report.json
```

![CDASH screenshot](image)

You can view the results at: https://my.cdash.org/viewTest.php?buildid=7056685
# CDASH Results

Testing started on 2021-09-09 17:26:58

Site Name: cori
Build Name: 4.4
Total time: 5m 59s 300ms

29 passed, 1 failed, 0 not run, 0 missing.

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Time</th>
<th>Details</th>
<th>Labels</th>
<th>History</th>
<th>Summary</th>
<th>Description</th>
<th>Hostname</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>e4a_hdfs</td>
<td>Failed</td>
<td>5m 59s 300ms</td>
<td></td>
<td>e4s</td>
<td>Broken</td>
<td>Broken</td>
<td>Run hdfs test from E45 Testsuite</td>
<td>cori2 4.4</td>
<td>e4s</td>
</tr>
<tr>
<td>dashan_cpi_example</td>
<td>Passed</td>
<td>1m 27s 370ms</td>
<td></td>
<td>e4s</td>
<td>Stable</td>
<td>Stable</td>
<td>MPI test to calculate PI and use dashan parser to view log file</td>
<td>cori2 4.4</td>
<td>e4s</td>
</tr>
<tr>
<td>default_spark_sanity_check</td>
<td>Passed</td>
<td>6s 860ms</td>
<td></td>
<td>spark</td>
<td>Stable</td>
<td>Stable</td>
<td>Test default spark module to see if some basic commands work</td>
<td>cori2 4.4</td>
<td>e4s</td>
</tr>
<tr>
<td>default_spark_version</td>
<td>Passed</td>
<td>1s 460ms</td>
<td></td>
<td>spark</td>
<td>Stable</td>
<td>Stable</td>
<td>Default spark version should be 5.14.2</td>
<td>cori2 4.4</td>
<td>e4s</td>
</tr>
<tr>
<td>e4s_20.10_spark</td>
<td>Passed</td>
<td>2s 690ms</td>
<td></td>
<td>e4s</td>
<td>Stable</td>
<td>Stable</td>
<td>Check e4s20.10 spark instance</td>
<td>cori2 4.4</td>
<td>e4s</td>
</tr>
<tr>
<td>e4s_21.02_spark</td>
<td>Passed</td>
<td>2s 670ms</td>
<td></td>
<td>e4s</td>
<td>Stable</td>
<td>Stable</td>
<td>Check e4s21.02 spark instance</td>
<td>cori2 4.4</td>
<td>e4s</td>
</tr>
<tr>
<td>e4s_21.05_spark</td>
<td>Passed</td>
<td>2s 740ms</td>
<td></td>
<td>e4s</td>
<td>Stable</td>
<td>Stable</td>
<td>Check e4s21.05 spark instance</td>
<td>cori2 4.4</td>
<td>e4s</td>
</tr>
<tr>
<td>e4s_adros2</td>
<td>Passed</td>
<td>5m 50s 950ms</td>
<td></td>
<td>e4s</td>
<td>Stable</td>
<td>Stable</td>
<td>Run adros2 test from E45 Testsuite</td>
<td>cori2 4.4</td>
<td>e4s</td>
</tr>
<tr>
<td>e4s_bolt</td>
<td>Passed</td>
<td>5m 55s 200ms</td>
<td></td>
<td>e4s</td>
<td>Stable</td>
<td>Stable</td>
<td>Run bolt test from E45 Testsuite</td>
<td>cori2 4.4</td>
<td>e4s</td>
</tr>
<tr>
<td>e4s_hydra</td>
<td>Passed</td>
<td>5m 57s 310ms</td>
<td></td>
<td>e4s</td>
<td>Stable</td>
<td>Stable</td>
<td>Run hydra test from E45 Testsuite</td>
<td>cori2 4.4</td>
<td>e4s</td>
</tr>
<tr>
<td>e4s_tau.pdt.papi</td>
<td>Passed</td>
<td>5m 56s 400ms</td>
<td></td>
<td>e4s</td>
<td>Stable</td>
<td>Stable</td>
<td>Run NPB3.1 test from E45 Testsuite to test TAU, POT and PAPI</td>
<td>cori2 4.4</td>
<td>e4s</td>
</tr>
<tr>
<td>moduletest_e4s_20.10</td>
<td>Passed</td>
<td>1m 1s 450ms</td>
<td></td>
<td>e4s</td>
<td>Stable</td>
<td>Stable</td>
<td>Run module load test for e4s_20.10 module tree</td>
<td>cori2 4.4</td>
<td>e4s</td>
</tr>
<tr>
<td>openmpsid_ts_version_e4s_21.05</td>
<td>Passed</td>
<td>6s 620ms</td>
<td></td>
<td>e4s</td>
<td>Stable</td>
<td>Stable</td>
<td>Run openmpsid-ts version check for e4s_21.05 Test</td>
<td>cori2 4.4</td>
<td>e4s</td>
</tr>
</tbody>
</table>
E4S Tests on Cori
Cori E4S Testing Strategy

- We are testing the facility deployed e4s stacks (e4s/21.05, e4s/21.02, e4s/20.10). Typically one has to load one of the e4s modules `module load e4s/21.05` and run `module load` or `spack load` to load the software before running the test.

- Please see [https://docs.nersc.gov/applications/e4s/](https://docs.nersc.gov/applications/e4s/) for more details regarding our facility deployment of e4s.

- We leverage `spack test` and [E4S testsuite](https://github.com/buildtesters/buildtest-cori/tree/devel/buildspecs/e4s) to test the e4s stack and sometimes we develop tests that are site specific.

- E4S tests are available at [https://github.com/buildtesters/buildtest-cori/tree/devel/buildspecs/e4s](https://github.com/buildtesters/buildtest-cori/tree/devel/buildspecs/e4s)

- We run all e4s tests using the `e4s` tags: `buildtest build --tags e4s`
Spack Test Example - Gasnet

version: "1.0"
buildspecs:
  spack_test_gasnet_e4s_21.05:
    type: spack
    executor: cori.local.sh
description: "Test gasnet@2021.3.0%intel with e4s/21.05 via spack test"
tags: e4s
pre_cmds: module load e4s/21.05
spack:
  root: /global/common/software/spackcp/e4s-21.05/spack/
  verify_spack: false
test:
  run:
    specs: ['gasnet@2021.3.0%intel']
    results:
      option: '-l'
      specs: ['gasnet@2021.3.0%intel']
maintainers:
  - "shahzebsiddiqui"
  - "PHargrove"
  - "bonachea"

Test Content

#!/bin/bash

########## START OF PRE COMMANDS ##########
module load e4s/21.05
########## END OF PRE COMMANDS ##########

source /global/common/software/spackcp/e4s-21.05/spack/share/spack/setup-env.sh
spack test run --alias gasnet@2021.3.0%intel gasnet@2021.3.0%intel
spack test results -l -- gasnet@2021.3.0%intel

Test output

### Spack test gasnet@2021.3.0%intel
### Testing package gasnet-2021.3.0-sbw7ukx
### Results for test suite 'gasnet@2021.3.0%intel'
tags: e4s
### pre_cmds: module load e4s/21.05
spack:
  root: /global/common/software/spackcp/e4s-21.05/spack/
  verify_spack: false
test:
  run:
    specs: ['gasnet@2021.3.0%intel']
    results:
      option: '-l'
      specs: ['gasnet@2021.3.0%intel']
maintainers:
  - "shahzebsiddiqui"
  - "PHargrove"
  - "bonachea"

https://my.cdash.org/test/40848138
OpenPMD Test

<table>
<thead>
<tr>
<th>Test ID</th>
<th>e603dcf1aabb-4b2e-ba91-fadv5da47e11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Code</td>
<td>0</td>
</tr>
<tr>
<td>user</td>
<td>e4s</td>
</tr>
<tr>
<td>hostname</td>
<td>cori02</td>
</tr>
<tr>
<td>description</td>
<td>Run openpmd-ls version check for e4s/21.05</td>
</tr>
<tr>
<td>command</td>
<td>sh openpmd-ls_version_e4s_21.05_build.sh</td>
</tr>
<tr>
<td>executor</td>
<td>cori.local.bash</td>
</tr>
<tr>
<td>tags</td>
<td>e4s</td>
</tr>
<tr>
<td>testroot</td>
<td>/global/cfs/cdirs/ws055/buildtests/runs/e4s_tests/2021-09-09/cori.local.bash/openpmd-ls/openpmd-ls_version_e4s_21.05/e665df0f staging</td>
</tr>
<tr>
<td>stagedir</td>
<td>/global/cfs/cdirs/ws055/buildtests/runs/e4s_tests/2021-09-09/cori.local.bash/openpmd-ls/openpmd-ls_version_e4s_21.05/e665df0f/staging</td>
</tr>
<tr>
<td>build_script</td>
<td>/global/cfs/cdirs/ws055/buildtests/runs/e4s_tests/2021-09-09/cori.local.bash/openpmd-ls/openpmd-ls_version_e4s_21.05/e665df0f/openpmd-ls_version_e4s_21.05_build.sh</td>
</tr>
<tr>
<td>testpath</td>
<td>/global/cfs/cdirs/ws055/buildtests/runs/e4s_tests/2021-09-09/cori.local.bash/openpmd-ls/openpmd-ls_version_e4s_21.05/e665df0f/openpmd-ls_version_e4s_21.05 build.sh</td>
</tr>
<tr>
<td>outfile</td>
<td>/global/cfs/cdirs/ws055/buildtests/runs/e4s_tests/2021-09-09/cori.local.bash/openpmd-ls/openpmd-ls_version_e4s_21.05/e665df0f/openpmd-ls_version_e4s_21.05.out</td>
</tr>
<tr>
<td>errfile</td>
<td>/global/cfs/cdirs/ws055/buildtests/runs/e4s_tests/2021-09-09/cori.local.bash/openpmd-ls/openpmd-ls_version_e4s_21.05/e665df0f/openpmd-ls_version_e4s_21.05.err</td>
</tr>
<tr>
<td>starttime</td>
<td>2021/09/09 10:20:54</td>
</tr>
<tr>
<td>endtime</td>
<td>2021/09/09 10:20:54</td>
</tr>
<tr>
<td>logpath</td>
<td>/tmp/buildtest_D1/prg9484.log</td>
</tr>
<tr>
<td>compiler</td>
<td>gcc</td>
</tr>
<tr>
<td>schemafile</td>
<td>script.v1.0.schema.json</td>
</tr>
</tbody>
</table>

View GitLab CI results

---

**Test output**

`openpmd-ls (openPMD-api) 0.13.4`

Copyright 2017-2020 openPMD contributors

Authors: Axel Rubli et al.

License: LGPLv3

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

---

[https://my.cdash.org/test/40848126](https://my.cdash.org/test/40848126)
UPC Test

version: "1.0"
buildspecs:
  upc_hello_04:
    type: compiler
    executor: cori.slurm.haswell_debug
    description: Hello world in upc for e6s/20.10
    source: src/hello_upcxx.cpp
    tags: [compile, n4s]
    compilers:
      name: ["builtin_gcc9"]
      default:
        gcc:
          sbatch: ["-t 10m", "-N 1"]
          cxx: "upcxx"
          cxxflags: ["-L/opt/cray/pe/pmi/default/lib64 -L/opt/cray/ucxmi/default/lib64 -L/opt/cray/xdrlg/default/lib64 -L/opt/cray/xpmem/default/lib64 -g --network=aries"
          pre_build:
            module swap intel intel/19.1.2.254
            module load e6s/20.10
            module load upcxx@0.28.3.0
          run: "upcxx-run -N 1 -n 8 s_EXEC"
      maintainers:
        - PMWangrove
        - boonschea

Test Content

```bash
#!/bin/bash
#### START OF SCHEDULER DIRECTIVES ####
#SBATCH --ntasks=1
#SBATCH --nodes=1
#SBATCH --job-name=upc_hello_04
#SBATCH --output=upc_hello_04.out
#SBATCH --error=upc_hello_04.err
#### END OF SCHEDULER DIRECTIVES ####

# name of executable
 upcxx_exec=src/hello_upcxx.cpp
## START OF PRE BUILD SECTION ##
module swap intel intel/19.1.2.254
module load e6s/20.10
module load upcxx@0.28.3.0
## END OF PRE BUILD SECTION ##

## Compilation Line
 upcxx -L/opt/cray/pe/pmi/default/lib64 -L/opt/cray/ucxmi/default/lib64 -L/opt/cray/xdrlg/default/lib64 -L/opt/cray/xpmem/default/lib64 -g --network=aries $ EXEC /global/u1/e6s/builds/5504F_NV/6/sidestag/f/buildtest-cori/buildspecs/apps/upc/hello_upcxx.cmake

## Run executable
 upcxx-run -N 1 -n 8 $ EXEC
```

https://my.cdash.org/test/40848145
E4S Testsuite Example – ADIOS2

Test: e4s_adios2 (Passed)
Build: e4s (cori) on 2021-09-09 17:19:52
Labels: e4s

--- CLAMP LOG ---
rm -f hello-world
Compiling /global/cfs/cdirs/m3503/buildtest/runs/e4s_tests/2021-09-09/cori.slamwell_premium/e4s_21.05/e4s_adios2/84225b0c/stage/testsuite/validation_tests/adios2
--- COMPILF LOG ---
Skipping load:  Environment already setup
+ export ADIOS2_LIB_DIR=/global/common/software/spack/cpe4s-21.05/software/cray-cnl7-hastwell/intel-19.1.3.384/adios2-2.7.1-16vlr1f407tkwynhgdq5gjwv0lde7/1ib
+ ADIOS2_LIB_DIR=/global/common/software/spack/cpe4s-21.05/software/cray-cnl7-hastwell/intel-19.1.3.384/adios2-2.7.1-16vlr1f407tkwynhgdq5gjwv0lde7/1lib
+ [[ ! -d /global/common/software/spack/cpe4s-21.05/software/cray-cnl7-hastwell/intel-19.1.3.384/adios2-2.7.1-16vlr1f407tkwynhgdq5gjwv0lde7/1lib ]] + export ADIOS2_LIB_DIR=/global/common/software/spack/cpe4s-21.05/software/cray-cnl7-hastwell/intel-19.1.3.384/adios2-2.7.1-16vlr1f407tkwynhgdq5gjwv0lde7/1lib
+ ADIOS2_LIB_DIR=/global/common/software/spack/cpe4s-21.05/software/cray-cnl7-hastwell/intel-19.1.3.384/adios2-2.7.1-16vlr1f407tkwynhgdq5gjwv0lde7/1lib64
+ make
CC -- -lgLOBAL/common/software/spack/cpe4s-21.05/software/cray-cnl7-hastwell/intel-19.1.3.384/adios2-2.7.1-16vlr1f407tkwynhgdq5gjwv0lde7/include -c hello-HelloWorld hello-world.o -lgLOBAL/common/software/spack/cpe4s-21.05/software/cray-cnl7-hastwell/intel-19.1.3.384/adios2-2.7.1-16vlr1f407tkwynhgdq5gjwv0lde7/lib
Running /global/cfs/cdirs/m3503/buildtest/runs/e4s_tests/2021-09-09/cori.slamwell_premium/e4s_21.05/e4s_adios2/84225b0c/stage/testsuite/validation_tests/adios2
Skipping load: Environment already setup
Hello World from ADIOS2
Hello World from ADIOS2
Hello World from ADIOS2
Hello World from ADIOS2
Hello World from ADIOS2
Hello World from ADIOS2
Hello World from ADIOS2
Hello World from ADIOS2
Success

https://my.cdash.org/test/40848140
Current Issues

- superlu@5.2.1 ([https://github.com/buildtesters/buildtest-cori/issues/70](https://github.com/buildtesters/buildtest-cori/issues/70)) – Permission Error writing make.inc file in install directory

- hypre@2.20 ([https://github.com/buildtesters/buildtest-cori/issues/69](https://github.com/buildtesters/buildtest-cori/issues/69)) – Can’t find mpicc


```bash
160
161  # Write configuration options to make.inc file
162  make_file.inc = joinpath(self.install_root, self.make_hdr_file)
>> 163  with open(make_file_inc,'w') as inc:
164       for option in config_args:
165      inc.write('{0}\n'.format(option))
166
>>> Error: TestFailure: 7 tests failed.
Command exited with status 127:
  ./ex5_line-of-sight_solution
  ./ex5_line-of-sight_solution: error while loading shared libraries: libRAJA.so: cannot open shared object file: No such file or directory
```
Available E4S Tests in buildtest Cori

<table>
<thead>
<tr>
<th>name</th>
<th>tags</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e4s_adios2</td>
<td>e4s</td>
<td>Run adios2 test from E4S Testsuite</td>
</tr>
<tr>
<td>e4s_bolt</td>
<td>e4s</td>
<td>Run bolt test from E4S Testsuite</td>
</tr>
<tr>
<td>e4s_hdf5</td>
<td>e4s</td>
<td>Run hdf5 test from E4S Testsuite</td>
</tr>
<tr>
<td>e4s_hypre</td>
<td>e4s</td>
<td>Run hypre test from E4S Testsuite</td>
</tr>
<tr>
<td>e4s_MP3.1</td>
<td>e4s</td>
<td>Run MP3.1 test from E4S Testsuite to test TAU, PDT and PAPI</td>
</tr>
<tr>
<td>e4s_adios2</td>
<td>e4s</td>
<td>Run adios2 test from E4S Testsuite</td>
</tr>
<tr>
<td>e4s_bolt</td>
<td>e4s</td>
<td>Run bolt test from E4S Testsuite</td>
</tr>
<tr>
<td>e4s_hdf5</td>
<td>e4s</td>
<td>Run hdf5 test from E4S Testsuite</td>
</tr>
<tr>
<td>e4s_hypre</td>
<td>e4s</td>
<td>Run hypre test from E4S Testsuite</td>
</tr>
<tr>
<td>e4s_tau.pdt.papi</td>
<td>e4s</td>
<td>Run TAU test from E4S Testsuite to test TAU, PDT and PAPI</td>
</tr>
<tr>
<td>spack_test_hdf5_e4s_21.02</td>
<td>e4s</td>
<td>Test hdf5/1.8.7gcc and hdf5/1.0.7intel from e4s/21.02 using spack test</td>
</tr>
<tr>
<td>spack_test_hdf5_e4s_21.05</td>
<td>e4s</td>
<td>Test hdf5/1.8.22intel and hdf5/1.0.7intel from e4s/21.05 using spack test</td>
</tr>
<tr>
<td>spack_test_umpire_e4s_21.02</td>
<td>e4s</td>
<td>Test umpire/0.4.1.2 E4S 21.02 test via spack test</td>
</tr>
<tr>
<td>spack_test_gasanet_e4s_21.05</td>
<td>e4s</td>
<td>Test gasanet/0.2021.3.0intel with e4s/21.05 via spack test</td>
</tr>
<tr>
<td>spack_test_hypre_e4s_21.02</td>
<td>e4s</td>
<td>Test hypre/0.20.0 for e4s/21.02 test via spack test</td>
</tr>
<tr>
<td>spack_test_hypre_e4s_21.05</td>
<td>e4s</td>
<td>Test hypre/0.20.0 for e4s/21.05 test via spack test</td>
</tr>
<tr>
<td>spack_test_upcxx_e4s_21.05</td>
<td>e4s</td>
<td>Test upcxx/0.2021.03.0 for e4s/21.05 test via spack test</td>
</tr>
<tr>
<td>spack_test_scf_e4s_21.02</td>
<td>e4s</td>
<td>Test scf/0.13.0 e4s/21.02 test via spack test</td>
</tr>
<tr>
<td>spack_test_scf_e4s_21.05</td>
<td>e4s</td>
<td>Test scf/0.13.0 for e4s/21.05 test via spack test</td>
</tr>
<tr>
<td>spack_test_strumpack_e4s_21.02</td>
<td>e4s</td>
<td>Test strumpack/0.1.1 E4S 21.02 test via spack test</td>
</tr>
<tr>
<td>spack_test_strumpack_e4s_21.05</td>
<td>e4s</td>
<td>Test strumpack/0.1.1 for e4s/21.05 test via spack test</td>
</tr>
<tr>
<td>spack_test_qthreads_e4s_21.02</td>
<td>e4s</td>
<td>qthreads E4S 21.02 test via spack test</td>
</tr>
<tr>
<td>spack_test_qthreads_e4s_21.05</td>
<td>e4s</td>
<td>Test qthreads E4S 21.02 test via spack test</td>
</tr>
<tr>
<td>spack_test_qthreads_e4s_21.02</td>
<td>e4s</td>
<td>Test qthreads E4S 21.02 test via spack test</td>
</tr>
<tr>
<td>spack_test_qthreads_e4s_21.05</td>
<td>e4s</td>
<td>Test qthreads E4S 21.02 test via spack test</td>
</tr>
<tr>
<td>spack_test_superlu_e4s_21.02</td>
<td>e4s</td>
<td>Test superlu/0.1.1 E4S 21.02 test via spack test</td>
</tr>
<tr>
<td>spack_test_superlu_e4s_21.05</td>
<td>e4s</td>
<td>Test superlu/0.1.1 and superlu-dist/0.4.0 for e4s/21.05 test via spack test</td>
</tr>
<tr>
<td>moduletest_e4s_20.10</td>
<td>e4s</td>
<td>Run module load test for e4s 20.10 module tree</td>
</tr>
</tbody>
</table>
Closing Remarks

- The facility deployment of E4S impacts how tests are written. We need a spack instance for deployment in order to test the user-facing environment. At Cori we can load e4s via `module load e4s` which activates a spack environment pre-installed with e4s packages.

- `spack test` and E4S Testsuite requires a spack instance to run tests which is focused on testing spack stack whereas buildtest is focused on writing facility tests.

- Issues with `spack test` at Facility will be addressed in future spack release, though facility deployment will be fixed to version. In those case we need to develop tests at facility when appropriate or periodically rebuild software with new version.

- Buildtest leverages `spack test` or `E4S Testsuite` to run the E4S tests targeting our e4s deployment at NERSC, when test fails we would write a facility flavored test.

- Tests may require need for batch submission for different schedulers and buildtest can support job submission for Slurm, LSF, PBS and Cobalt.

- Test needs to be run on recurrent basis and automation can be done through the use of Gitlab. Finally test results needs to be published somewhere to analyze results.

- There are different ways to pass test including: return code, regular expression, runtime. Every test would need some criteria for success or failure.

- We need a human to analyze test result and report issues for facility tests. We need help from Developers to help contribute test and analyze test results for facility results.
References

- Schema Docs: [https://buildtesters.github.io/buildtest/](https://buildtesters.github.io/buildtest/)
- Slack: [http://hpcbuildtest.slack.com/](http://hpcbuildtest.slack.com/)
- E4S Testsuite: [https://github.com/E4S-Project/testsuite](https://github.com/E4S-Project/testsuite)
Acknowledgement

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, in support of the nation’s exascale computing imperative.

This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231.