
buildtest Documentation

Release 0.10.0

Shahzeb Siddiqui

Jul 30, 2021

BACKGROUND

1	Status	3
2	Useful Links	5
3	Description	7
3.1	Summary of buildtest	7
3.2	Terminology	14
3.3	Installing buildtest	15
3.4	Getting Started	17
3.5	Configuring buildtest	110
3.6	Writing buildsspecs	147
3.7	Build and Test Process	317
3.8	Using buildtest at HPC sites	319
3.9	Conference and Publications	320
3.10	Contributing Guide	320
3.11	API Reference	340
3.12	Buildtest Command Reference	392
4	License	403
5	Indices and tables	405
	Python Module Index	407
	Index	409

This documentation was last rebuild on Jul 30, 2021 and is intended for version 0.10.0.

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.

CHAPTER
ONE

STATUS

USEFUL LINKS

1. Source Code: <https://github.com/buildtesters/buildtest>
2. Documentation: <http://buildtest.rtf.d.io/>
3. Schema Docs: <https://buildtesters.github.io/buildtest/>
4. ReadTheDocs: <https://readthedocs.org/projects/buildtest/>
5. CodeCov: <https://codecov.io/gh/buildtesters/buildtest>
6. Slack Channel: <http://hpcbuildtest.slack.com>
7. Slack Invite: <https://hpcbuildtest.herokuapp.com>
8. CodeFactor: <https://www.codefactor.io/repository/github/buildtesters/buildtest>
9. Snyk: <https://app.snyk.io/org/buildtesters/>
10. Cori Test Repository: <https://github.com/buildtesters/buildtest-cori>

DESCRIPTION

`buildtest` is a testing framework to help HPC sites write test for their system as part of their routine acceptance & regression test. `buildtest` provides a YAML interface to write tests which `buildtest` can validate and generate shell scripts that can run on your HPC system. The test template (YAML) is called **buildspec** which can define one or more test instance that is validated by a `json schema`. `buildtest` supports the following batch schedulers: `IBM Spectrum LSF`, `Slurm`, `PBS` and `Cobalt`. We publish the schema documentation, json schemas, and schema examples at <https://buildtesters.github.io/buildtest/> which is useful when you are *writing buildspecs*.

To get started with `buildtest`, please see *installing buildtest* and *getting started guide*.

A spin-off project called `lmodule` is a Python API for `Lmod`. The `buildtest` module features were deprecated and moved to `lmodule` with the main objective is to automate module load testing. For more details on `lmodule` see <https://github.com/buildtesters/lmodule>

3.1 Summary of buildtest

Contents

- *Summary of buildtest*
 - *Background*
 - *Motivation*
 - *Inception of buildtest*
 - *Preview of buildtest*
 - *Target Audience & Use Case*
 - *Timeline*
 - *Related Projects and community efforts*

3.1.1 Background

HPC System and Software Stack are tightly integrated with underlying architecture which makes them highly sensitive to changes in system such as OS, kernel, driver, or vendor updates. We need a testing framework to automate acceptance testing of an HPC system so that HPC Support Teams can increase **confidence** of their HPC system throughout the system lifecycle.

3.1.2 Motivation

There are many build automations tools for compiling source code into binary code, the most used tool is the **make** utility found in most Linux systems. Build scripts like **configure**, **cmake** and **autoconf** can generate files used by make for installing the software. Makefile is a file used by make program that shows how to compile and link a program which is the basis for building a software package. One can invoke **make test** which will run the target named **test** in Makefile that dictates how tests are compiled and run. Makefile is hard to interpret and requires in-depth experience with shell-scripting and strong understanding of how package is built and tested. Note that package maintainers must provide the source files, headers, and additional libraries to test the software and make test simply the test compilation and execution. Tools like *configure*, *cmake* and *autoconf* are insufficient for testing because HPC software stack consist of applications packaged in many formats and some are make-incompatible.

We wanted a framework that hides the complexity for compiling source code and provide an easy markup language to define test configuration to create the test. This leads to buildtest, which is a testing framework that generates test-scripts using YAML that is validated with JSON Schemas. YAML was picked given its ease-of-use and it lowers the barrier for writing tests.

3.1.3 Inception of buildtest

buildtest was founded by [Shahzeb Siddiqui](#) in 2017 when he was at [Pfizer](#) tasked for testing software stack for a data center migration.

Shahzeb was tasked with testing the software ecosystem by focusing on the most important application due to time constraints. During this period, several dozen test scripts were developed in shell-script that targeted core HPC tools such as compilers, **MPI**, **R**, **Python**, etc. A single master script was used to run all the tests which led to *buildtest*.

3.1.4 Preview of buildtest

You can run `buildtest help` which will provide an overview of buildtest commands.

```
$ buildtest help

Building Buildspects
-----

Build a single buildspect file
$ buildtest build -b <file>

Search all buildspects recursively in a given directory for all '.yaml' files and
↳ build all buildspects
$ buildtest build -b <dir>

Build buildspects by file and directory
$ buildtest build -b <file> -b <dir>
```

(continues on next page)

(continued from previous page)

Exclude files when building buildsspecs

```
$ buildtest build -b <file> -b <dir> -x <file> -x <dir>
```

Building buildsspecs by tag

```
$ buildtest build -t <tagname1> -t <tagname2>
```

Building buildsspecs by executor

```
$ buildtest build -e <executor1> -e <executor2>
```

building buildsspecs with file, directory, tags, and executors

```
$ buildtest build -b <file> -b <dir> -t <tagname1> -e <executor1>
```

Filter tests by tagname for all buildsspecs discovered

```
$ buildtest build -b <file> -t <tagname1> -ft <filter-tagname1> -ft <filter-tagname2>
```

Specify alternate buildtest configuration file when building test.

```
$ buildtest -c <config> build -b <file>
```

Specify alternate location of test directory (/tmp)

```
$ buildtest build -b <file> --testdir /tmp
```

Set max-pend-time to 30s which overrides 'max_pend_time' field in configuration file.

↪ buildtest will cancel job

job if its pending in queue beyond max-pend-time value

```
$ buildtest build -b <file> --max-pend-time 30
```

Set pollinterval to 60s which overrides 'pollinterval' field in configuration file.

↪ buildtest will poll jobs based on 'pollinterval' value.

```
$ buildtest build -b <file> --poll-interval 60
```

View report

Display all tests results

```
$ buildtest report
```

Filter test results by returncode=0

```
$ buildtest report --filter returncode=0
```

Filter test by multiple filter fields. In this example, buildtest will report all

↪ tests that 'PASS' for tagname 'python'

```
$ buildtest report --filter state=PASS,tags=python
```

Format report table by --format fields, in this example we only report columns 'name

↪ ', 'state', 'buildspec'

```
$ buildtest report --format name,state,buildspec
```

List all filter fields

```
$ buildtest report --helpfilter
```

List all format fields

(continues on next page)

(continued from previous page)

```
$ buidtest report --helpformat
```

Retrieve oldest record for all tests based on timestamp

```
$ buidtest report --oldest
```

Retrieve latest record for all tests based on timestamp

```
$ buidtest report --latest
```

Specify alternate report file to display test results

```
$ buidtest report -r <report-file>
```

Inspect Tests

Display record of all runs for test name 'hello'.

```
$ buidtest inspect name hello
```

Display record of test name 'foo' and 'bar'

```
$ buidtest inspect name foo bar
```

Display all test names and ids

```
$ buidtest inspect list
```

Display record of test by unique identifier

```
$ buidtest inspect id <ID>
```

Finding Buildsspecs

Discover and validate all buildsspecs and load all validated buildsspecs in cache.↵

↵ Finally buidtest will report all tests for all validated buildsspecs

```
$ buidtest buildspec find
```

Rebuild cache

```
$ buidtest buildspec find --rebuild
```

List all available buildsspecs from cache

```
$ buidtest buildspec find --buildspec
```

List all unique tags from cache

```
$ buidtest buildspec find --tags
```

List all unique executors from cache

```
$ buidtest buildspec find --executors
```

List all maintainers from cache

```
$ buidtest buildspec find --maintainers
```

Show breakdown of all buildsspecs by maintainer names.

```
$ buidtest buildspec find --maintainers-by-buildspecs
```

(continues on next page)

(continued from previous page)

Validate buildsspecs

Validate buildsspecs by file and directory

```
$ buildtest buildspec validate -b <file> -b <dir>
```

Validate buildsspecs by tagname 'python' and 'mac'

```
$ buildtest buildspec validate -t python -t mac
```

Validate buildsspecs by executor name

```
$ buildtest buildspec validate -e <executor>
```

Build History

The 'buildtest history' command can report history of all builds when you run
 ↪ 'buildtest build' command
 along with log files.

To display a list of all builds

```
$ buildtest history list
```

Query information for a given build by unique identifier.

```
$ buildtest history query <ID>
```

Get logfile for given build

```
$ buildtest history query <ID> --log
```

Buildtest Configuration

View content of configuration file

```
$ buildtest config view
```

Validate configuration file with JSON schema

```
$ buildtest config validate
```

List all executors from configuration file

```
$ buildtest config executors
```

List all available system entries in configuration file

```
$ buildtest config systems
```

Specify alternate configuration file

```
$ buildtest -c <config> config validate
```

Upload Tests to CDASH

Upload all tests to cdash with build name 'DEMO'

```
$ buildtest cdash upload DEMO
```

(continues on next page)

(continued from previous page)

```
Upload all tests from report file /tmp/result.json with build name 'DAILY_CHECK'
$ buildtest cdash upload 'DAILY_CHECK' --report /tmp/result.json
```

```
Upload tests to CDASH with site named called 'laptop'
$ buildtest cdash upload --site laptop DEMO
```

```
Open CDASH project in web-browser
$ buildtest cdash view
```

```
Open CDASH project with url name
$ buildtest cdash view --url https://my.cdash.org/index.php?project=buildtest
```

3.1.5 Target Audience & Use Case

buildtest target audience is *HPC Staff* that wants to perform acceptance & regression testing of their HPC system.

buildtest is not

- replacement for *make*, *cmake*, *autoconf*, *ctest*
- a software build framework ([easybuild](#), [spack](#), [nix](#) , [guix](#))
- a replacement for benchmark tools or test suite from upstream package
- a replacement for writing tests, you will need to write your tests defined by buildtest schemas, however you can copy/paste & adapt tests from other sites that are applicable to you.

Typical use-case:

- Run your test suite during system maintenance
- Perform daily tests for testing various system components. These tests should be short
- Run weekly/biweekly test on medium/large workload including micro-benchmark
- Run tests for newly installed software package typically requested by user.

If you are interested trying out buildtest check out [Getting Started](#) and [Join Slack Channel](#).

3.1.6 Timeline

Date	Version	Description
Jul 13th 2021	v0.10.0	In this release we added spack support in buildtest by creating a new schema to write buildsspecs that will generate spack commands. For more details see Spack Schema . We added bash completion for buildtest commands which is enabled when installing buildtest. We added a new command <code>buildtest buildspec validate</code> that can be used for validating buildsspecs with JSON Schema.
Jun 11th 2021	v0.9.6	Added buildtest CDASH integration using <code>buildtest cdash</code> to upload test results. In this release we added <code>buildtest history</code> command to retrieve build history and query logfiles. Add global option <code>-c</code> in buildtest to specify alternate configuration file.
Mar 31th 2021	v0.9.5	Add support for PBS scheduler and reimplement <code>buildtest inspect</code> command
Mar 14th 2021	v0.9.4	Introduced major change in buildtest configuration file (<code>settings.schema.json</code>) to define multiple HPC systems in configuration file. This lead to change in how <code>executors</code> are referenced in buildspec file.
Feb 22nd 2021	v0.9.3	Change Copyright details for project to include LBNL . We added dependabot for managing dependencies, added OLCF facility pipelines for running regression test.
Jan 12th 2021	v0.9.2	Contains major refactor to <code>compiler-v1.0-schema.json</code> for writing compiler test using regular expression to search for compilers that are defined in configuration file.
Nov 24st 2020	v0.9.1	Added support for Cobalt Scheduler .
Sep 3rd 2020	v0.8.0	Introduced JSON Schema for validating buildspec. Add support for Slurm and LSF scheduler for job submission. Add support for building buildsspecs by file, directory and tagname and command line interface to schema.
Sep 11th 2018	v0.4.0	buildtest was ported from Python 2 to 3.
Aug 20th 2017	v0.1.5	buildtest was converted from bash to Python and project was moved into github https://github.com/HPC-buildtest/buildtest .
Feb 18th 2017	N/A	Start of project

3.1.7 Related Projects and community efforts

Project	Description	State
ReFrame	is a high level regression framework for writing regression test for HPC systems. Tests are written in Python class and it has support for cray programming environment, job scheduler, module integration, parameter tests, test dependency, and sanity check. The project is led by CSCS.	Active
Pavilion2	is a framework for running and analyzing tests targeting HPC systems. Tests are written in YAML and majority of pavilion commands are implemented through python plugins using yapsy. Pavilion2 is developed by LANL.	Active
Automatic Testing of Installed Software (ATIS)	This project was presented by Xavier Besseron in FOSDEM14 that targets MPI testing using ctest and cdash. This project is no longer in development.	Obsolete
hpcswtest	is a HPC Software Stack Testing Framework developed by Idaho National Lab. The framework is built using C++11 and JSON file to define test configuration.	Obsolete
PVCS	is a validation engine to run large tests for HPC systems, the framework is written in Perl and recipe known as Test Expression (TE) are written in YAML. This project is developed by CEA.	Obsolete

The System Test Working Group hosted a BOF [HPC System Testing: Procedures, Acceptance, Regression Testing, and Automation](#) in SuperComputing '19. This working group is aimed at discussing acceptance and regression testing procedure and lessons learned from other HPC centers.

3.2 Terminology

Name	Description
Buildspec	is a YAML file that buildtest interprets when generating the test. A Buildspec may contain one or more test that is validated with a global schema and sub schema .
Schema	is a JSON Schema file (.schema.json) that defines structure of a buildspec file and it is used for validating a buildspec
Global Schema	is a JSON schema that validates buildspec file. buildtest will validate all buildspecs with global schema
Sub Schema	Each test section in a buildspec file is validated with one sub-schema defined by type field. The buildspec test section can only be validated with one sub-schema
Test Script	is a generated shell script by buildtest as a result of processing one of the Buildspec.
Settings	is a buildtest configuration file in YAML that configures buildtest at your site. The Settings file must be compatible with the Settings Schema.
Settings Schema	is a special schema file that defines structure of buildtest settings.
Executor	is responsible for running a TestScript . An executor can be of several types such as local, slurm, lsf which defines if test is run locally or via a scheduler. The executors are defined in the Settings file.

3.3 Installing buildtest

3.3.1 Requirements

You need the following packages to install buildtest.

- `git`
- `Python` ≥ 3.6

3.3.2 Cloning buildtest

To get started, clone the buildtest repository in your local machine as follows:

```
# HTTPS
$ git clone https://github.com/buildtesters/buildtest.git

# SSH
$ git clone git@github.com:buildtesters/buildtest.git
```

If you prefer the latest release use the **master** branch:

```
$ git clone -b master git@github.com:buildtesters/buildtest.git
```

3.3.3 Installing buildtest

To install buildtest, navigate to buildtest repo and source the setup script as follows:

```
# BASH users
$ source setup.sh

# CSH users
$ source setup.csh
```

This will add `buildtest` command in your `$PATH` and set environment variable `$BUILDTEST_ROOT` which points to root of buildtest repo.

You may want to create an isolated python environment of choice depending on your preference you can use any of the following:

- `virtualenv`
- `conda`
- `pipenv`

buildtest will provide tab completion for bash shell, this is managed by script `bash_completion.sh`, if you encounter any issues with tab completion please raise an issue at <https://github.com/buildtesters/buildtest/issues/new/choose>.

3.3.4 Development Dependencies (Optional)

If you plan to contribute back to buildtest, you will need to install additional dependencies found in the requirements file in docs/requirements.txt as follows:

```
$ pip install -r docs/requirements.txt
```

3.3.5 Usage (buildtest --help)

Once you are setup, you can run `buildtest --help` for more details on how to use buildtest. Shown below is the output

```
$ buildtest --help
usage: buildtest [options] [COMMANDS]

buildtest is a HPC testing framework for building and running tests.

optional arguments:
  -h, --help            show this help message and exit
  -V, --version          show program's version number and exit
  -c CONFIGFILE, --config CONFIGFILE
                        Specify Path to Configuration File
  -d, --debug           Print debug messages to screen

COMMANDS:

  build                Build and Run test
  history              Query build history
  buildspect           Options for querying buildspect cache
  config              Query buildtest configuration
  report              Query test report
  inspect              Inspect a test based on NAME or ID
  schema              List schema contents and examples
  cdash               Upload test to CDASH server
  docs                Open buildtest docs in browser
  schemadocs          Open buildtest schema docs in browser
  help                buildtest command guide
```

References

GitHub:	https://github.com/buildtesters/buildtest
Documentation:	https://buildtest.readthedocs.io/en/latest/index.html
Schema Documentation:	https://buildtesters.github.io/buildtest/
Slack:	http://hpcbuildtest.slack.com/

Please report issues at <https://github.com/buildtesters/buildtest/issues>

Copyright (c) 2021, The Regents of the University of California, through Lawrence_↪
Berkeley National Laboratory (subject to receipt of any required approvals from the U.
↪S. Dept. of Energy), Shahzeb Siddiqui, and Vanessa Sochat. All rights reserved.

If you have got this far, please go to the next section on *Getting Started*

3.4 Getting Started

3.4.1 Building Test via buildtest

This guide will get you familiar with buildtest command line interface. Once you complete this section, you can proceed to *writing buildspects* section where we will cover how to write buildspects.

Once you install buildtest, you should find the *buildtest* command in your **\$PATH**. You can check the path to buildtest command by running:

```
$ which buildtest
```

If you don't see buildtest go back and *install buildtest*.

When you clone buildtest, you also get a set of buildspects that you can run on your system. The buildtest build command is used for building and running tests. Buildtest will read one or more buildspects file that adheres to one of the buildtest schemas. For a complete list of build options, run buildtest build --help

Build Usage

```
$ buildtest build --help
usage: buildtest [options] [COMMANDS] build [-h] [-b BUILDSPEC] [-x EXCLUDE] [-t TAGS] [-
↳ft FILTER_TAGS] [-e EXECUTOR]
                                     [-s {parse,build}] [--testdir TESTDIR] [--
↳rebuild REBUILD] [-r REPORT]
                                     [--max-pend-time MAX_PEND_TIME] [--poll-
↳interval POLL_INTERVAL] [-k]

optional arguments:
  -h, --help                show this help message and exit
  -b BUILDSPEC, --buildspec BUILDSPEC
                             Specify a buildspect (file or directory) to build. A buildspect
↳must end in '.yaml' extension.
  -x EXCLUDE, --exclude EXCLUDE
                             Exclude one or more buildspects (file or directory) from
↳processing. A buildspect must end in
                             '.yaml' extension.
  -t TAGS, --tags TAGS       Discover buildspects by tags found in buildspect cache
  -ft FILTER_TAGS, --filter-tags FILTER_TAGS
                             Filter buildspects by tags when building tests.
  -e EXECUTOR, --executor EXECUTOR
                             Discover buildspects by executor name found in buildspect cache
  -s {parse,build}, --stage {parse,build}
                             control behavior of buildtest build
  --testdir TESTDIR          Specify a custom test directory where to write tests. This
↳overrides configuration file and
                             default location.
  --rebuild REBUILD          Rebuild test X number of times. Must be a positive number
↳between [1-50]
  -r REPORT, --report REPORT
                             Specify a report file where tests will be written.
  --max-pend-time MAX_PEND_TIME
```

(continues on next page)

(continued from previous page)

```

Specify Maximum Pending Time (sec) for job before cancelling job.
↪ This only applies for batch
    job submission.
--poll-interval POLL_INTERVAL
Specify Poll Interval (sec) for polling batch jobs
-k, --keep-stage-dir Keep stage directory after job completion.

```

Building a Test

To build a test, we use the `--buildspec` or short option `-b` to specify the path to buildspec file. Let's see some examples, first we specify a full path to buildspec file. In this example, buildtest will *discover buildspecs* followed by parsing the test with appropriate schema and generate a shell script that is run by buildtest. You can learn more about *build and test process*.

```

$ buildtest build -b $BUILDTEST_ROOT/tutorials/vars.yml

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:30
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↪ 10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↪ python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↪ 10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↪ checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↪ bin/buildtest build -b /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↪ checkouts/v0.10.0/tutorials/vars.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
↪ -----+
| Discovered Buildsspecs
↪ |
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↪ vars.yml |
+-----+
↪ -----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

```

(continues on next page)

(continued from previous page)

```

+-----+
| Stage: Parsing Buildsspecs |
+-----+

  schemafile          | validstate  | buildspec
+-----+-----+-----+
↪ script-v1.0.schema.json | True        | /home/docs/checkouts/readthedocs.org/user_
↪ builds/buildtest/checkouts/v0.10.0/tutorials/vars.yml

name          description
+-----+-----+
variables_bash  Declare shell variables in bash

+-----+
| Stage: Building Test |
+-----+

  name          | id          | type    | executor          | tags          | testpath
+-----+-----+-----+-----+-----+-----+
↪
↪
variables_bash | 4d34c3aa | script | generic.local.bash | ['tutorials'] | /home/docs/
↪ checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/generic.
↪ local.bash/vars/variables_bash/0/variables_bash_build.sh

+-----+
| Stage: Running Test |
+-----+

  name          | id          | executor          | status  | returncode
+-----+-----+-----+-----+-----+
variables_bash | 4d34c3aa | generic.local.bash | PASS    | 0

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 1/1 Percentage: 100.000%
Failed Tests: 0/1 Percentage: 0.000%

Writing Logfile to: /tmp/buildtest_dyv8b4j0.log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /home/docs/checkouts/
↪ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log

```

Note: buildtest will only read buildsspecs with .yaml extension, if you specify a .yaml it will be ignored by buildtest.

The `--buildspec` option can be used to specify a file or directory path. If you want to build multiple buildsspecs in a directory you can specify the directory path and buildtest will recursively search for all .yaml files. In the next example, we build all tests in directory **general_tests/configuration**.

```
$ buildtest build -b general_tests/configuration/

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:51
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳ python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/buildtest/settings/config.yaml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ bin/buildtest build -b general_tests/configuration/

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
↳ -----+
| Discovered Buildsspecs |
↳ |
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
↳ tests/configuration/disk_usage.yaml |
+-----+
↳ -----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
↳ tests/configuration/ssh_localhost.yaml |
+-----+
↳ -----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
↳ tests/configuration/ulimits.yaml |
+-----+
↳ -----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
↳ tests/configuration/kernel_state.yaml |
+-----+
↳ -----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
↳ tests/configuration/systemd-default-target.yaml |
```

(continues on next page)

(continued from previous page)

```
+-----+
↪-----+
Discovered Buildsspecs: 5
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 5

+-----+
| Stage: Parsing Buildsspecs |
+-----+

schemafile          | validstate | buildspec
+-----+-----+-----+
↪-----+
script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↪builds/buildtest/checkouts/v0.10.0/general_tests/configuration/disk_usage.yml
script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↪builds/buildtest/checkouts/v0.10.0/general_tests/configuration/ssh_localhost.yml
script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↪builds/buildtest/checkouts/v0.10.0/general_tests/configuration/ulimits.yml
script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↪builds/buildtest/checkouts/v0.10.0/general_tests/configuration/kernel_state.yml
script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↪builds/buildtest/checkouts/v0.10.0/general_tests/configuration/systemd-default-target.
↪yml

name                  description
+-----+-----+
↪-----+
root_disk_usage       Check root disk usage and report if it exceeds threshold
ssh_localhost_remotecommand Test if ssh on localhost works and if we can run remote_
↪command.
ulimit_filelock_unlimited Check if file lock is set to unlimited in ulimits
ulimit_cputime_unlimited  Check if cputime is set to unlimited in ulimits
ulimit_stacksize_unlimited Check if stack size is set to unlimited in ulimits
ulimit_vmsize_unlimited   Check virtual memory size and check if its set to unlimited
ulimit_filedescriptor_4096 Check if open file descriptors limit is set to 4096
ulimit_max_user_process_2048 Check max number of user process limit is set to 2048
kernel_swapusage       Retrieve Kernel Swap Usage
systemd_default_target  check if default target is multi-user.target

+-----+
| Stage: Building Test |
+-----+

name                  | id      | type  | executor      | tags
↪                  | testpath
+-----+-----+-----+-----+-----+
↪-----+
↪-----+
↪-----+
↪-----+
```

(continues on next page)

(continued from previous page)

```

root_disk_usage          | 3dd8135a | script | generic.local.bash | ['filesystem',
↳ 'storage'] | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/var/tests/generic.local.bash/disk_usage/root_disk_usage/0/root_disk_usage_build.sh
ssh_localhost_remotecommand | c2386a01 | script | generic.local.bash | ['ssh']
↳ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↳ 0/var/tests/generic.local.bash/ssh_localhost/ssh_localhost_remotecommand/0/ssh_
↳ localhost_remotecommand_build.sh
ulimit_filelock_unlimited  | 3bb18d20 | script | generic.local.bash | ['system']
↳ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↳ 0/var/tests/generic.local.bash/ulimits/ulimit_filelock_unlimited/0/ulimit_filelock_
↳ unlimited_build.sh
ulimit_cputime_unlimited   | cd4bfa0d | script | generic.local.bash | ['system']
↳ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↳ 0/var/tests/generic.local.bash/ulimits/ulimit_cputime_unlimited/0/ulimit_cputime_
↳ unlimited_build.sh
ulimit_stacksize_unlimited | 155261e0 | script | generic.local.bash | ['system']
↳ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↳ 0/var/tests/generic.local.bash/ulimits/ulimit_stacksize_unlimited/0/ulimit_stacksize_
↳ unlimited_build.sh
ulimit_vmsize_unlimited    | 4c268496 | script | generic.local.bash | ['system']
↳ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↳ 0/var/tests/generic.local.bash/ulimits/ulimit_vmsize_unlimited/0/ulimit_vmsize_
↳ unlimited_build.sh
ulimit_filedescriptor_4096 | c22ab4c1 | script | generic.local.bash | ['system']
↳ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↳ 0/var/tests/generic.local.bash/ulimits/ulimit_filedescriptor_4096/0/ulimit_
↳ filedescriptor_4096_build.sh
ulimit_max_user_process_2048 | 5e687dd4 | script | generic.local.bash | ['system']
↳ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↳ 0/var/tests/generic.local.bash/ulimits/ulimit_max_user_process_2048/0/ulimit_max_user_
↳ process_2048_build.sh
kernel_swapusage         | 2a6ab5c2 | script | generic.local.bash | ['configuration
↳ '] | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/var/tests/generic.local.bash/kernel_state/kernel_swapusage/0/kernel_swapusage_
↳ build.sh
systemd_default_target    | 2185cf71 | script | generic.local.bash | ['system']
↳ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↳ 0/var/tests/generic.local.bash/systemd-default-target/systemd_default_target/0/systemd_
↳ default_target_build.sh

+-----+
| Stage: Running Test |
+-----+

name          | id          | executor          | status  | returncode
+-----+-----+-----+-----+-----+
root_disk_usage | 3dd8135a | generic.local.bash | PASS    | 0
ssh_localhost_remotecommand | c2386a01 | generic.local.bash | FAIL    | 255

```

(continues on next page)

(continued from previous page)

ulimit_filelock_unlimited	3bb18d20	generic.local.bash	PASS		0
ulimit_cputime_unlimited	cd4bfa0d	generic.local.bash	PASS		0
ulimit_stacksize_unlimited	155261e0	generic.local.bash	FAIL		0
ulimit_vmsize_unlimited	4c268496	generic.local.bash	PASS		0
ulimit_filedescriptor_4096	c22ab4c1	generic.local.bash	FAIL		0
ulimit_max_user_process_2048	5e687dd4	generic.local.bash	FAIL		0
kernel_swapusage	2a6ab5c2	generic.local.bash	FAIL		127
systemd_default_target	2185cf71	generic.local.bash	FAIL		1

```

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 4/10 Percentage: 40.000%
Failed Tests: 6/10 Percentage: 60.000%

Writing Logfile to: /tmp/buildtest_3ujxy0fh.log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /home/docs/checkouts/
↳ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log

```

Building Multiple Buildsspecs

You can append `-b` option to build multiple buildsspecs in the same command. Buildtest will discover buildsspecs for every argument (`-b`) and accumulate a list of buildsspecs to run. In this example, we instruct buildtest to build a buildspec file and all buildsspecs in a directory path.

```

$ buildtest build -b general_tests/configuration/ -b tutorials/vars.yml

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:52
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳ python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ bin/buildtest build -b general_tests/configuration/ -b tutorials/vars.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

```

(continues on next page)

(continued from previous page)

```

+-----+
| Discovered Buildsspecs |
+-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
vars.yml |
+-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
tests/configuration/kernel_state.yml |
+-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
tests/configuration/systemd-default-target.yml |
+-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
tests/configuration/ulimits.yml |
+-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
tests/configuration/disk_usage.yml |
+-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
tests/configuration/ssh_localhost.yml |
+-----+
Discovered Buildsspecs: 6
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 6

+-----+
| Stage: Parsing Buildsspecs |
+-----+

schemafile | validstate | buildspec
+-----+
script-v1.0.schema.json | True | /home/docs/checkouts/readthedocs.org/user_
builds/buildtest/checkouts/v0.10.0/tutorials/vars.yml
script-v1.0.schema.json | True | /home/docs/checkouts/readthedocs.org/user_
builds/buildtest/checkouts/v0.10.0/general_tests/configuration/kernel_state.yml
script-v1.0.schema.json | True | /home/docs/checkouts/readthedocs.org/user_
builds/buildtest/checkouts/v0.10.0/general_tests/configuration/systemd-default-target.
yml
script-v1.0.schema.json | True | /home/docs/checkouts/readthedocs.org/user_
builds/buildtest/checkouts/v0.10.0/general_tests/configuration/ulimits.yml
script-v1.0.schema.json | True | /home/docs/checkouts/readthedocs.org/user_
builds/buildtest/checkouts/v0.10.0/general_tests/configuration/disk_usage.yml
script-v1.0.schema.json | True | /home/docs/checkouts/readthedocs.org/user_
builds/buildtest/checkouts/v0.10.0/general_tests/configuration/ssh_localhost.yml

```

(continues on next page)

(continued from previous page)

name	description
variables_bash	Declare shell variables in bash
kernel_swapusage	Retrieve Kernel Swap Usage
systemd_default_target	check if default target is multi-user.target
ulimit_filelock_unlimited	Check if file lock is set to unlimited in ulimits
ulimit_cputime_unlimited	Check if cputime is set to unlimited in ulimits
ulimit_stacksize_unlimited	Check if stack size is set to unlimited in ulimits
ulimit_vm_size_unlimited	Check virtual memory size and check if its set to unlimited
ulimit_filedescriptor_4096	Check if open file descriptors limit is set to 4096
ulimit_max_user_process_2048	Check max number of user process limit is set to 2048
root_disk_usage	Check root disk usage and report if it exceeds threshold
ssh_localhost_remotecommand	Test if ssh on localhost works and if we can run remote_
command.	

name	id	type	executor	tags
variables_bash	7597f889	script	generic.local.bash	['tutorials']
kernel_swapusage	808c8828	script	generic.local.bash	['configuration']
systemd_default_target	40a017a5	script	generic.local.bash	['system']
ulimit_filelock_unlimited	de34bf13	script	generic.local.bash	['system']
ulimit_cputime_unlimited	e40139a3	script	generic.local.bash	['system']
ulimit_stacksize_unlimited	6bd977c0	script	generic.local.bash	['system']

(continues on next page)

(continued from previous page)

```

ulimit_vmsize_unlimited      | d8212946 | script | generic.local.bash | ['system']
→ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
→0/var/tests/generic.local.bash/ulimits/ulimit_vmsize_unlimited/1/ulimit_vmsize_
→unlimited_build.sh
ulimit_filedescriptor_4096  | 55160d1a | script | generic.local.bash | ['system']
→ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
→0/var/tests/generic.local.bash/ulimits/ulimit_filedescriptor_4096/1/ulimit_
→filedescriptor_4096_build.sh
ulimit_max_user_process_2048 | 3f4aa574 | script | generic.local.bash | ['system']
→ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
→0/var/tests/generic.local.bash/ulimits/ulimit_max_user_process_2048/1/ulimit_max_user_
→process_2048_build.sh
root_disk_usage             | a4e7fc98 | script | generic.local.bash | ['filesystem',
→ 'storage'] | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
→10.0/var/tests/generic.local.bash/disk_usage/root_disk_usage/1/root_disk_usage_build.sh
ssh_localhost_remotecommand | 9e0d34b7 | script | generic.local.bash | ['ssh']
→ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
→0/var/tests/generic.local.bash/ssh_localhost/ssh_localhost_remotecommand/1/ssh_
→localhost_remotecommand_build.sh

```

```

+-----+
| Stage: Running Test |
+-----+

```

name	id	executor	status	returncode
variables_bash	7597f889	generic.local.bash	PASS	0
kernel_swapusage	808c8828	generic.local.bash	FAIL	127
systemd_default_target	40a017a5	generic.local.bash	FAIL	1
ulimit_filelock_unlimited	de34bf13	generic.local.bash	PASS	0
ulimit_cputime_unlimited	e40139a3	generic.local.bash	PASS	0
ulimit_stacksize_unlimited	6bd977c0	generic.local.bash	FAIL	0
ulimit_vmsize_unlimited	d8212946	generic.local.bash	PASS	0
ulimit_filedescriptor_4096	55160d1a	generic.local.bash	FAIL	0
ulimit_max_user_process_2048	3f4aa574	generic.local.bash	FAIL	0
root_disk_usage	a4e7fc98	generic.local.bash	PASS	0
ssh_localhost_remotecommand	9e0d34b7	generic.local.bash	FAIL	255

```

+-----+
| Stage: Test Summary |
+-----+

```

Passed Tests: 5/11 Percentage: 45.455%
Failed Tests: 6/11 Percentage: 54.545%

Writing Logfile to: /tmp/buildtest_s8xzvrxp.log

A copy of logfile can be found at \$BUILDTEST_ROOT/buildtest.log - /home/docs/checkouts/
→readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log (continues on next page)

(continued from previous page)

Excluding Buildspects

So far we learned how to build buildspects by file and directory path using the `-b` option. Next, we will discuss how one may exclude buildspects which behaves similar to `-b` option. You can exclude buildspects via `--exclude` or short option `-x` which can be useful when you want to exclude certain files or sub directory.

For example we can build all buildspects in `tutorials` but exclude file `tutorials/vars.yml` by running:

```
$ buildtest build -b tutorials -x tutorials/vars.yml
```

buildtest will discover all buildspects and then exclude any buildspects specified by `-x` option. You can specify `-x` multiple times just like `-b` option.

For example, we can undo discovery by passing same option to `-b` and `-x` as follows

```
$ buildtest build -b tutorials/ -x tutorials/

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:52
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳bin/buildtest build -b tutorials/ -x tutorials/
There are no Buildspect files to process.
```

Buildtest will stop immediately if there are no Buildspects to process, this is true if you were to specify files instead of directory.

In this example, we build all buildspects in a directory but exclude a file. Buildtest will report the excluded buildspects in the output and `-x` option can be appended multiple times. The `-x` can be a file or a directory and behaves similar to `-b` option.

```
$ buildtest build -b general_tests/configuration/ -x general_tests/configuration/ulimits.
↳yml

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:53
```

(continues on next page)

(continued from previous page)

```
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳bin/buildtest build -b general_tests/configuration/ -x general_tests/configuration/
↳ulimits.yml
```

```
+-----+
| Stage: Discovering Buildsspecs |
+-----+
```

```
+-----+
↳-----+
| Discovered Buildsspecs |
```

```
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
↳tests/configuration/systemd-default-target.yml |
```

```
+-----+
↳-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
↳tests/configuration/ulimits.yml |
```

```
+-----+
↳-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
↳tests/configuration/kernel_state.yml |
```

```
+-----+
↳-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
↳tests/configuration/ssh_localhost.yml |
```

```
+-----+
↳-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
↳tests/configuration/disk_usage.yml |
```

```
+-----+
↳-----+
| Excluded Buildsspecs |
```

```
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
↳tests/configuration/ulimits.yml |
```

```
+-----+
↳-----+
```

(continues on next page)

(continued from previous page)

```

Discovered Buildsspecs: 5
Excluded Buildsspecs: 1
Detected Buildsspecs after exclusion: 4

+-----+
| Stage: Parsing Buildsspecs |
+-----+

schemafile          | validstate | buildspec
+-----+-----+-----+
↪ script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↪ builds/buildtest/checkouts/v0.10.0/general_tests/configuration/systemd-default-target.
↪ yml
↪ script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↪ builds/buildtest/checkouts/v0.10.0/general_tests/configuration/kernel_state.yml
↪ script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↪ builds/buildtest/checkouts/v0.10.0/general_tests/configuration/ssh_localhost.yml
↪ script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↪ builds/buildtest/checkouts/v0.10.0/general_tests/configuration/disk_usage.yml

name                  description
+-----+-----+
↪ ----
systemd_default_target  check if default target is multi-user.target
kernel_swapusage       Retrieve Kernel Swap Usage
ssh_localhost_remotecommand  Test if ssh on localhost works and if we can run remote_
↪ command.
root_disk_usage         Check root disk usage and report if it exceeds threshold

+-----+-----+
| Stage: Building Test |
+-----+-----+

name                  | id          | type   | executor          | tags
↪                  | testpath
+-----+-----+-----+-----+-----+
↪
↪
↪
↪
systemd_default_target | 46bdeee3 | script | generic.local.bash | ['system']
↪ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↪ 0/var/tests/generic.local.bash/systemd-default-target/systemd_default_target/2/systemd_
↪ default_target_build.sh
kernel_swapusage       | 6b8d64b1 | script | generic.local.bash | ['configuration
↪ '] | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↪ 10.0/var/tests/generic.local.bash/kernel_state/kernel_swapusage/2/kernel_swapusage_
↪ build.sh
ssh_localhost_remotecommand | 734cb524 | script | generic.local.bash | ['ssh']
↪ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↪ 0/var/tests/generic.local.bash/ssh_localhost/ssh_localhost_remotecommand/2/ssh_
↪ localhost_remotecommand_build.sh

```

(continues on next page)

(continued from previous page)

```

root_disk_usage      | af270f01 | script | generic.local.bash | ['filesystem',
↪ 'storage'] | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↪ 10.0/var/tests/generic.local.bash/disk_usage/root_disk_usage/2/root_disk_usage_build.sh

+-----+
| Stage: Running Test |
+-----+

name                  | id          | executor          | status  | returncode
+-----+-----+-----+-----+-----+
systemd_default_target | 46bdeee3   | generic.local.bash | FAIL    | 1
kernel_swapusage      | 6b8d64b1   | generic.local.bash | FAIL    | 127
ssh_localhost_remotecommand | 734cb524 | generic.local.bash | FAIL    | 255
root_disk_usage       | af270f01   | generic.local.bash | PASS    | 0

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 1/4 Percentage: 25.000%
Failed Tests: 3/4 Percentage: 75.000%

Writing Logfile to: /tmp/buildtest_w12bth6s.log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /home/docs/checkouts/
↪ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log

```

Building By Tags

buildtest can perform builds by tags by using `--tags` or short option `(-t)`. In order to use this feature, buildtest must load buildsspecs in [cache](#) which can be run via `buildtest buildspect find`. If you are unsure of the available tags you can run `buildtest buildspect find --tags` or let buildtest tab-complete the available tags. For more details see [Querying buildspect tags](#).

Let's assume you want to build by tag name `network`, buildtest will attempt to find all tests that contain tags: `['network']` in the buildspect which is loaded in the buildcache cache. If a test matches the tag name, the test will be picked up during the discover process.

```

$ buildtest build -t network

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:53
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↪ 10.0/bin/buildtest

```

(continues on next page)

(continued from previous page)

```

buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳bin/buildtest build -t network

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
↳-----+
| Discovered Buildsspecs |
↳
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳tags_example.yml |
+-----+
↳-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

BREAKDOWN OF BUILDSPECS BY TAGS
-----
Detected Tag Names: ['network']
+-----+
↳-----+
| network |
↳
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳tags_example.yml |
+-----+
↳-----+

+-----+
| Stage: Parsing Buildsspecs |
+-----+

schemafile          | validstate | buildspec
-----+-----+-----
↳
script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↳builds/buildtest/checkouts/v0.10.0/tutorials/tags_example.yml

```

(continues on next page)

(continued from previous page)

```

name                description
-----
string_tag          tags can be a string
list_of_strings_tags tags can be a list of strings

+-----+
| Stage: Building Test |
+-----+

name                | id          | type   | executor          | tags          |
--testpath
-----+-----+-----+-----+-----+
↳ -----
↳ -----
string_tag          | 405b133c | script | generic.local.bash | network      | /
↳ home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/
↳ generic.local.bash/tags_example/string_tag/0/string_tag_build.sh
list_of_strings_tags | 4ae61a72 | script | generic.local.bash | ['network', 'ping'] | /
↳ home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/
↳ generic.local.bash/tags_example/list_of_strings_tags/0/list_of_strings_tags_build.sh

+-----+
| Stage: Running Test |
+-----+

name                | id          | executor          | status  | returncode
-----+-----+-----+-----+-----+
string_tag          | 405b133c | generic.local.bash | PASS    | 0
list_of_strings_tags | 4ae61a72 | generic.local.bash | FAIL    | 127

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 1/2 Percentage: 50.000%
Failed Tests: 1/2 Percentage: 50.000%

Writing Logfile to: /tmp/buildtest_9r9m7gya.log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /home/docs/checkouts/
↳ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log

```

You can build by multiple tags by specifying `--tags` multiple times. In next example we build all tests with tag name `pass` and `python`.

```

$ buildtest build -t python -t pass

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:53
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳bin/buildtest build -t python -t pass

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
↳-----+
| Discovered Buildsspecs |
↳-----+
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳python-shell.yml |
+-----+
↳-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳python-hello.yml |
+-----+
↳-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳pass_returncode.yml |
+-----+
↳-----+
Discovered Buildsspecs: 3
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 3

BREAKDOWN OF BUILDSPECS BY TAGS
-----
Detected Tag Names: ['python', 'pass']
+-----+
↳-----+
| python |
↳-----+
+=====+

```

(continues on next page)

(continued from previous page)

```
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳python-shell.yml |
+-----+
↳-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳python-hello.yml |
+-----+
↳-----+

+-----+
↳-----+
| pass
↳
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳pass_returncode.yml |
+-----+
↳-----+

+-----+
| Stage: Parsing Buildsspecs |
+-----+

schemafile          | validstate  | buildspec
+-----+-----+-----+
↳-----+
script-v1.0.schema.json | True        | /home/docs/checkouts/readthedocs.org/user_
↳builds/buildtest/checkouts/v0.10.0/tutorials/python-shell.yml
script-v1.0.schema.json | True        | /home/docs/checkouts/readthedocs.org/user_
↳builds/buildtest/checkouts/v0.10.0/tutorials/python-hello.yml
script-v1.0.schema.json | True        | /home/docs/checkouts/readthedocs.org/user_
↳builds/buildtest/checkouts/v0.10.0/tutorials/pass_returncode.yml

name                description
+-----+-----+
circle_area         Calculate circle of area given a radius
python_hello        Hello World python
exit1_fail           exit 1 by default is FAIL
exit1_pass           report exit 1 as PASS
returncode_list_mismatch exit 2 failed since it failed to match returncode 1
returncode_int_match exit 128 matches returncode 128

+-----+
| Stage: Building Test |
+-----+

name                | id          | type   | executor          | tags
↳ | testpath
```

(continues on next page)

(continued from previous page)

```

-----+-----+-----+-----+-----+
↪ -----+-----+-----+-----+-----+
↪ -----+-----+-----+-----+-----+
↪ -----+-----+-----+-----+-----+
circle_area          | 178ef5ce | script | generic.local.python | ['tutorials',
↪ 'python'] | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↪ 0/var/tests/generic.local.python/python-shell/circle_area/1/circle_area_build.sh
python_hello        | 04a87317 | script | generic.local.bash   | python
↪          | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↪ var/tests/generic.local.bash/python-hello/python_hello/1/python_hello_build.sh
exit1_fail          | 84c02c94 | script | generic.local.sh     | ['tutorials',
↪ 'fail']   | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↪ 0/var/tests/generic.local.sh/pass_returncode/exit1_fail/1/exit1_fail_build.sh
exit1_pass          | 3e06bf49 | script | generic.local.sh     | ['tutorials',
↪ 'pass']   | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↪ 0/var/tests/generic.local.sh/pass_returncode/exit1_pass/1/exit1_pass_build.sh
returncode_list_mismatch | ed0a8187 | script | generic.local.sh     | ['tutorials',
↪ 'fail']   | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↪ 0/var/tests/generic.local.sh/pass_returncode/returncode_list_mismatch/1/returncode_
↪ list_mismatch_build.sh
returncode_int_match  | 7f0c8ed0 | script | generic.local.sh     | ['tutorials',
↪ 'pass']   | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↪ 0/var/tests/generic.local.sh/pass_returncode/returncode_int_match/1/returncode_int_
↪ match_build.sh

+-----+
| Stage: Running Test |
+-----+

name          | id          | executor          | status  | returncode
-----+-----+-----+-----+-----+
circle_area   | 178ef5ce   | generic.local.python | PASS    | 0
python_hello  | 04a87317   | generic.local.bash  | PASS    | 0
exit1_fail    | 84c02c94   | generic.local.sh    | FAIL    | 1
exit1_pass    | 3e06bf49   | generic.local.sh    | PASS    | 1
returncode_list_mismatch | ed0a8187 | generic.local.sh    | FAIL    | 2
returncode_int_match  | 7f0c8ed0   | generic.local.sh    | PASS    | 128

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 4/6 Percentage: 66.667%
Failed Tests: 2/6 Percentage: 33.333%

Writing Logfile to: /tmp/buildtest_u8hjwt1_.log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /home/docs/checkouts/
↪ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log (continues on next page)

```

(continued from previous page)

When multiple tags are specified, we search each tag independently and if it's found in the builds spec cache we retrieve the builds spec file and add file to queue. This queue is a list of builds specs that buildtest will process (i.e parse, build, run). You can *query tags* from builds specs cache to see all available tags by running `buildtest builds spec find --tags`.

Note: The `--tags` is used for discovering builds spec file and not filtering tests by tag. If you want to filter tests by tags use `--filter-tags`.

The `--filter-tags` or short option `-ft` is used for filtering tests by tag name. The `--filter-tags` is used in conjunction with other options like `--buildspec`, `--tags`, or `--executor` for discovering builds specs. Let's rerun the previous example and filter tests by pass. Now we only see tests built with tagname pass and all remaining tests were ignored.

```
$ buildtest build --tags pass --filter-tags pass -b tutorials/python-hello.yml

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:54
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳bin/buildtest build --tags pass --filter-tags pass -b tutorials/python-hello.yml

+-----+
| Stage: Discovering Builds Specs |
+-----+

+-----+
↳-----+
| Discovered Builds Specs |
↳-----+
+-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳python-hello.yml |
+-----+
+-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳pass_returncode.yml |
+-----+
↳-----+
```

(continues on next page)

(continued from previous page)

```

Discovered Buildsspecs: 2
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 2

BREAKDOWN OF BUILDSPECS BY TAGS
-----
Detected Tag Names: ['pass']
+-----+
↪-----+
| pass                                     ↪
↪
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↪pass_returncode.yml |
+-----+
↪-----+

[python_hello][/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↪10.0/tutorials/python-hello.yml]: test is skipped because it is not in tag filter↪
↪list: ['pass']
[exit1_fail][/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↪0/tutorials/pass_returncode.yml]: test is skipped because it is not in tag filter↪
↪list: ['pass']
[returncode_list_mismatch][/home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↪checkouts/v0.10.0/tutorials/pass_returncode.yml]: test is skipped because it is not in↪
↪tag filter list: ['pass']

+-----+
| Stage: Parsing Buildsspecs |
+-----+

schemafilename | validstate | buildspectestpath
+-----+
↪-----+
script-v1.0.schema.json | True | /home/docs/checkouts/readthedocs.org/user_
↪builds/buildtest/checkouts/v0.10.0/tutorials/python-hello.yml
script-v1.0.schema.json | True | /home/docs/checkouts/readthedocs.org/user_
↪builds/buildtest/checkouts/v0.10.0/tutorials/pass_returncode.yml

name | description
-----
exit1_pass | report exit 1 as PASS
returncode_int_match | exit 128 matches returncode 128

+-----+
| Stage: Building Test |
+-----+

name | id | type | executor | tags |
↪testpath

```

(continues on next page)

(continued from previous page)

```

-----+-----+-----+-----+-----+-----+-----+-----+
↪ -----+-----+-----+-----+-----+-----+-----+-----+
↪ -----+-----+-----+-----+-----+-----+-----+-----+
exit1_pass          | dec97567 | script | generic.local.sh | ['tutorials', 'pass'] | /
↪ home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/
↪ generic.local.sh/pass_returncode/exit1_pass/2/exit1_pass_build.sh
returncode_int_match | 56e33ad6 | script | generic.local.sh | ['tutorials', 'pass'] | /
↪ home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/
↪ generic.local.sh/pass_returncode/returncode_int_match/2/returncode_int_match_build.sh

+-----+
| Stage: Running Test |
+-----+

name          | id          | executor          | status  | returncode
-----+-----+-----+-----+-----+
exit1_pass     | dec97567   | generic.local.sh  | PASS    | 1
returncode_int_match | 56e33ad6 | generic.local.sh  | PASS    | 128

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 2/2 Percentage: 100.000%
Failed Tests: 0/2 Percentage: 0.000%

Writing Logfile to: /tmp/buildtest_j9xotkv4.log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /home/docs/checkouts/
↪ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log

```

The `--filter-tags` option can be appended multiple times to filter tests by multiple tags. If buildtest detects no tests were found when filtering tests by tag name then buildtest will terminate with a message.

You can combine `--tags` with `--buildspec` to discover buildspecs in a single command. buildtest will query tags and buildspecs independently and combine all discovered buildspecs together.

```

$ buildtest build --tags pass --buildspec tutorials/python-hello.yml

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:54
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↪ 10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↪ python

```

(continues on next page)

(continued from previous page)

```

python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳bin/buildtest build --tags pass --buildspec tutorials/python-hello.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
↳-----+
| Discovered Buildsspecs |
↳
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳pass_returncode.yml |
+-----+
↳-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳python-hello.yml |
+-----+
↳-----+
Discovered Buildsspecs: 2
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 2

BREAKDOWN OF BUILDSPECS BY TAGS
-----
Detected Tag Names: ['pass']
+-----+
↳-----+
| pass |
↳
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳pass_returncode.yml |
+-----+
↳-----+

+-----+
| Stage: Parsing Buildsspecs |
+-----+

schemafile          | validstate | buildspec
-----+-----+-----
↳-----+
script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↳builds/buildtest/checkouts/v0.10.0/tutorials/pass_returncode.yml

```

(continues on next page)

(continued from previous page)

```

script-v1.0.schema.json | True          | /home/docs/checkouts/readthedocs.org/user_
↳ builds/buildtest/checkouts/v0.10.0/tutorials/python-hello.yml

name                description
-----
exit1_fail          exit 1 by default is FAIL
exit1_pass          report exit 1 as PASS
returncode_list_mismatch exit 2 failed since it failed to match returncode 1
returncode_int_match exit 128 matches returncode 128
python_hello        Hello World python

+-----+
| Stage: Building Test |
+-----+

name                | id          | type   | executor          | tags
↳ | testpath
-----+-----+-----+-----+-----
↳ +-----+-----+-----+-----+-----
↳ -----
exit1_fail          | 93ad2bb3 | script | generic.local.sh  | ['tutorials', 'fail
↳ '] | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↳ tests/generic.local.sh/pass_returncode/exit1_fail/2/exit1_fail_build.sh
exit1_pass          | 7c74495b | script | generic.local.sh  | ['tutorials', 'pass
↳ '] | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↳ tests/generic.local.sh/pass_returncode/exit1_pass/3/exit1_pass_build.sh
returncode_list_mismatch | 8a13233b | script | generic.local.sh  | ['tutorials', 'fail
↳ '] | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↳ tests/generic.local.sh/pass_returncode/returncode_list_mismatch/2/returncode_list_
↳ mismatch_build.sh
returncode_int_match   | 478bf901 | script | generic.local.sh  | ['tutorials', 'pass
↳ '] | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↳ tests/generic.local.sh/pass_returncode/returncode_int_match/3/returncode_int_match_
↳ build.sh
python_hello         | de751b51 | script | generic.local.bash | python
↳ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↳ tests/generic.local.bash/python-hello/python_hello/2/python_hello_build.sh

+-----+
| Stage: Running Test |
+-----+

name                | id          | executor          | status  | returncode
-----+-----+-----+-----+-----
exit1_fail          | 93ad2bb3 | generic.local.sh  | FAIL    | 1

```

(continues on next page)

(continued from previous page)

exit1_pass	7c74495b	generic.local.sh	PASS		1
returncode_list_mismatch	8a13233b	generic.local.sh	FAIL		2
returncode_int_match	478bf901	generic.local.sh	PASS		128
python_hello	de751b51	generic.local.bash	PASS		0
+-----+					
Stage: Test Summary					
+-----+					
Passed Tests: 3/5 Percentage: 60.000%					
Failed Tests: 2/5 Percentage: 40.000%					
Writing Logfile to: /tmp/buildtest_5ra6r44a.log					
A copy of logfile can be found at \$BUILDTEST_ROOT/buildtest.log - /home/docs/checkouts/					
readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log					

As you may see, there are several ways to build buildsspecs with buildtest. Tags is great way to build a whole collection of tests if you don't know path to all the files. You can specify multiple tags per buildsspecs to classify how test can be run.

Building by Executors

Every buildspec is associated to an executor which is responsible for running the test. You can instruct buildtest to run all tests by given executor via `--executor` option or short option `-e`. For instance, if you want to build all test associated to executor `generic.local.python` you can run:

```
$ buildtest build --executor generic.local.python
```

buildtest will query buildspec cache for the executor name and retrieve a list of buildsspecs with matching executor name. To see a list of available executors in buildspec cache see [querying buildspec executor](#).

Note: By default all tests are run in buildspec file. The `buildtest build --executor` option discovers buildsspecs if one of the test matches the executor name. The `--executor` option is **not filtering on test level** like `--filter-tags` option.

In this example we run all tests that are associated to `generic.local.python` executor.

```
$ buildtest build --executor generic.local.python

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:54
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
               ↪10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
               ↪python
python version: 3.6.12
```

(continues on next page)

(continued from previous page)

```

Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ bin/buildtest build --executor generic.local.python

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
↳ -----+
| Discovered Buildsspecs                                     |
↳ |
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳ run_only_platform.yml |
+-----+
↳ -----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳ python-shell.yml      |
+-----+
↳ -----+
Discovered Buildsspecs: 2
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 2

BREAKDOWN OF BUILDSPECS BY EXECUTORS
-----
Detected Executor Names: ['generic.local.python']
+-----+
↳ -----+
| generic.local.python                                     |
↳ |
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳ run_only_platform.yml |
+-----+
↳ -----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳ python-shell.yml      |
+-----+
↳ -----+

[run_only_platform_darwin][/home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/tutorials/run_only_platform.yml]: test is skipped because this test
↳ is expected to run on platform: Darwin but detected platform: Linux.

+-----+
| Stage: Parsing Buildsspecs |

```

(continues on next page)

(continued from previous page)

```

+-----+
schemafile          | validstate  | buildspect
+-----+-----+-----+
↳ -----
script-v1.0.schema.json | True          | /home/docs/checkouts/readthedocs.org/user_
↳ builds/buildtest/checkouts/v0.10.0/tutorials/run_only_platform.yml
script-v1.0.schema.json | True          | /home/docs/checkouts/readthedocs.org/user_
↳ builds/buildtest/checkouts/v0.10.0/tutorials/python-shell.yml

name                description
+-----+-----+
run_only_platform_linux This test will only run if target platform is Linux
circle_area          Calculate circle of area given a radius

+-----+
| Stage: Building Test |
+-----+

name                | id          | type   | executor          | tags
↳ | testpath
+-----+-----+-----+-----+-----+
↳ -----
↳ -----
↳ -----
run_only_platform_linux | d018a512 | script | generic.local.python | ['tutorials']
↳ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ var/tests/generic.local.python/run_only_platform/run_only_platform_linux/0/run_only_
↳ platform_linux_build.sh
circle_area          | 3475676d | script | generic.local.python | ['tutorials',
↳ 'python'] | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↳ 0/var/tests/generic.local.python/python-shell/circle_area/2/circle_area_build.sh

+-----+
| Stage: Running Test |
+-----+

name                | id          | executor          | status | returncode
+-----+-----+-----+-----+-----+
run_only_platform_linux | d018a512 | generic.local.python | PASS   | 0
circle_area          | 3475676d | generic.local.python | PASS   | 0

+-----+
| Stage: Test Summary |
+-----+

```

(continues on next page)

(continued from previous page)

```
Passed Tests: 2/2 Percentage: 100.000%
```

```
Failed Tests: 0/2 Percentage: 0.000%
```

```
Writing Logfile to: /tmp/buildtest_3p596gw_.log
```

```
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /home/docs/checkouts/  
↪readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log
```

Note: The `--executor` option can be appended to discover tests by multiple executors.

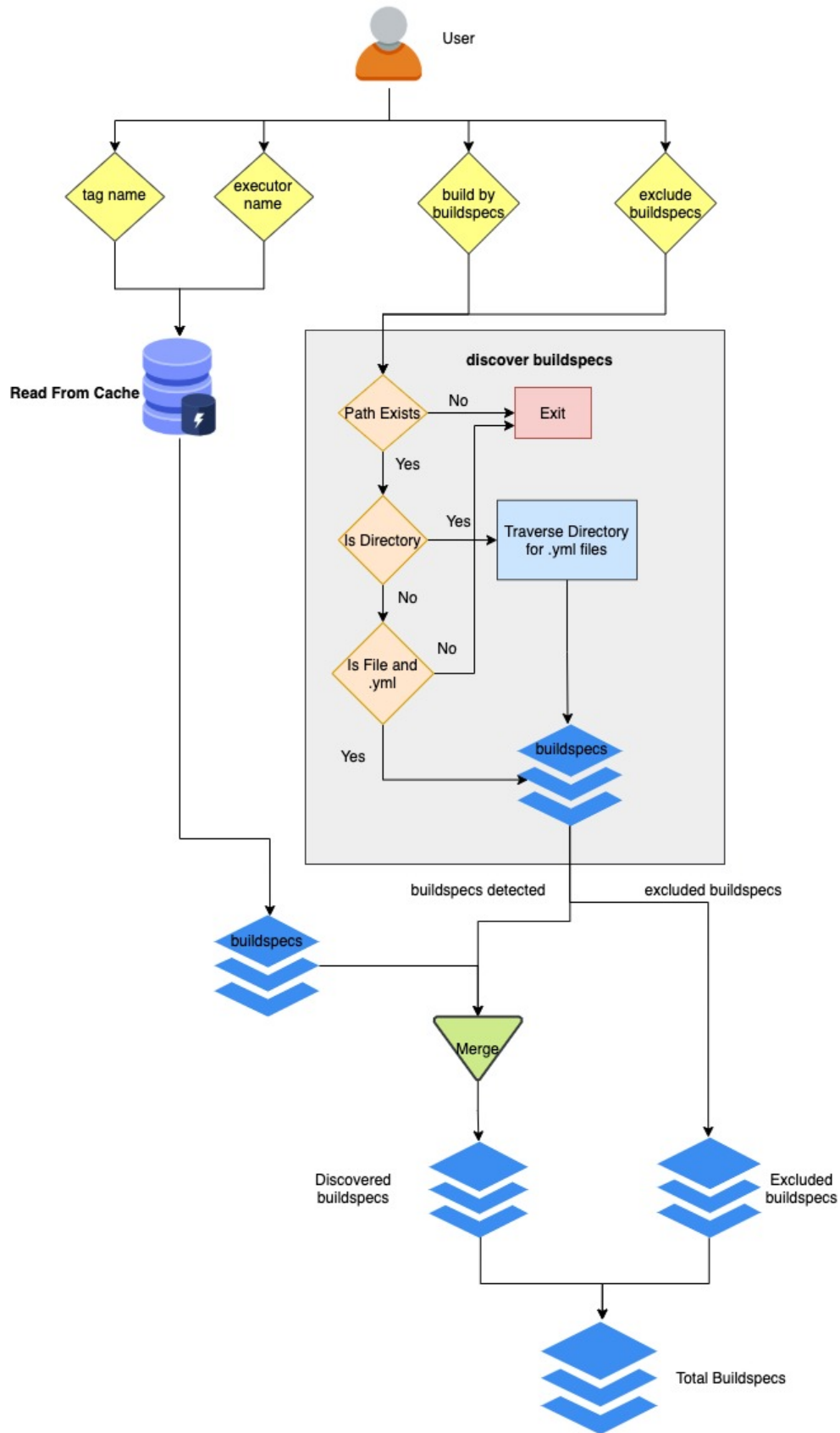
Discover Buildsspecs

Now, let's discuss how buildtest discovers buildsspecs since there are several ways to build buildsspecs.

The buildspec search resolution is described as follows:

- If file or directory specified by `-b` option doesn't exist we exit immediately.
- If buildspec path is a directory, traverse directory recursively to find all `.yaml` extensions
- If buildspec path is a file, check if file extension is not `.yaml`, exit immediately
- If user specifies `--tags` or `--executor` we search in buildspec cache to discover buildsspecs.

Shown below is a diagram on how buildtest discovers buildsspecs. The user can build buildsspecs by `--buildspec`, `-tags`, or `-executor` which will discover the buildsspecs. You can *exclude buildsspecs* using `--exclude` option which is processed after discovering buildsspecs. The excluded buildsspecs are removed from list if found and final list of buildsspecs is processed.



Control builds by Stages

We can control behavior of buildtest build command to stop at certain phase using `--stage` option. The `--stage` option accepts `parse` or `build`, which will instruct buildtest to stop at `parse` or `build` phase of the pipeline.

Buildtest will validate all the buildsspecs in the `parse` stage, so you can instruct buildtest to stop at `parse` stage via `--stage=parse`. This can be useful when debugging buildsspecs that are invalid. In this example below, we instruct buildtest to stop after `parse` stage.

```
$ buildtest build -b tutorials/vars.yml --stage=parse

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:55
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳ python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ bin/buildtest build -b tutorials/vars.yml --stage=parse

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
↳ -----+
| Discovered Buildsspecs |
↳ |
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳ vars.yml |
+-----+
↳ -----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

+-----+
| Stage: Parsing Buildsspecs |
+-----+

schemafile          | validstate | buildspec
+-----+
↳ -----+
script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↳ builds/buildtest/checkouts/v0.10.0/tutorials/vars.yml
```

(continues on next page)

(continued from previous page)

name	description
variables_bash	Declare shell variables in bash

Likewise, if you want to troubleshoot your test script without running them you can use `--stage=build` which will stop after build phase. This can be used when you are writing builds spec to troubleshoot how test is generated. In this next example, we inform buildtest to stop after build stage.

```
$ buildtest build -b tutorials/vars.yml --stage=build

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:55
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳bin/buildtest build -b tutorials/vars.yml --stage=build

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
↳-----+
| Discovered Buildsspecs |
↳      |
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳vars.yml |
+-----+
↳-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

+-----+
| Stage: Parsing Buildsspecs |
+-----+
```

(continues on next page)

(continued from previous page)

schemafile	validstate	buildspec
script-v1.0.schema.json	True	/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/vars.yml

name	description
variables_bash	Declare shell variables in bash


```
+-----+
| Stage: Building Test |
+-----+
```


name	id	type	executor	tags	testpath
variables_bash	521f895a	script	generic.local.bash	['tutorials']	/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/generic.local.bash/vars/variables_bash/2/variables_bash_build.sh

Invalid Buildsspecs

buildtest will skip any buildsspecs that fail to validate, in that case the test script will not be generated. Here is an example where we have an invalid buildspec.

```
$ buildtest build -b tutorials/invalid_buildspec_section.yml

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:55
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/bin/buildtest build -b tutorials/invalid_buildspec_section.yml

+-----+
| Stage: Discovering Buildsspecs |
```

(continues on next page)

(continued from previous page)

```
| Discovered Buildsspecs
|
+-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
| invalid_executor.yml |
+-----+
+-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

Error Messages from Stage: Parse
+-----+
/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
| invalid_executor.yml
No buildsspecs to process because there are no valid buildsspecs
```

Rebuild Tests

buildtest can rebuild tests using the `--rebuild` option which can be useful if you want to test a particular test multiple times. The rebuild option works across all discovered buildsspecs and create a new test instance (unique id) and test directory path. To demonstrate we will build `tutorials/python-shell.yml` three times using `--rebuild=3`.

```
$ buildtest build -b tutorials/python-shell.yml --rebuild=3

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:55
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
| 10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
| python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
| 10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
| checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
| bin/buildtest build -b tutorials/python-shell.yml --rebuild=3

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
| Discovered Buildsspecs
|
```

(continues on next page)

(continued from previous page)

```

+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳python-shell.yml |
+-----+
↳-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

+-----+
| Stage: Parsing Buildsspecs |
+-----+

schemafile          | validstate | buildspec
+-----+-----+-----+
↳-----+
script-v1.0.schema.json | True       | /home/docs/checkouts/readthedocs.org/user_
↳builds/buildtest/checkouts/v0.10.0/tutorials/python-shell.yml

name                description
+-----+-----+
circle_area         Calculate circle of area given a radius
circle_area         Calculate circle of area given a radius
circle_area         Calculate circle of area given a radius

+-----+
| Stage: Building Test |
+-----+

name      | id      | type  | executor          | tags          |
↳testpath
+-----+-----+-----+-----+-----+
↳-----+
↳-----+
circle_area | a1f47ba1 | script | generic.local.python | ['tutorials', 'python'] | /
↳home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/
↳generic.local.python/python-shell/circle_area/3/circle_area_build.sh
circle_area | 66d02fbd | script | generic.local.python | ['tutorials', 'python'] | /
↳home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/
↳generic.local.python/python-shell/circle_area/4/circle_area_build.sh
circle_area | 020608ed | script | generic.local.python | ['tutorials', 'python'] | /
↳home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/
↳generic.local.python/python-shell/circle_area/5/circle_area_build.sh

+-----+
| Stage: Running Test |

```

(continues on next page)

(continued from previous page)

+-----+				
name	id	executor	status	returncode
+-----+				
circle_area	a1f47ba1	generic.local.python	PASS	0
circle_area	66d02fbd	generic.local.python	PASS	0
circle_area	020608ed	generic.local.python	PASS	0
+-----+				
Stage: Test Summary				
+-----+				
Passed Tests: 3/3 Percentage: 100.000%				
Failed Tests: 0/3 Percentage: 0.000%				
Writing Logfile to: /tmp/buildtest_krchtpxi.log				
A copy of logfile can be found at \$BUILDTTEST_ROOT/buildtest.log - /home/docs/checkouts/				
↪readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log				

The rebuild works with all options including: --buildspec, --exclude, --tags and --executors. buildtest will perform rebuild for all discovered tests, for instance in this next example we will discover all tests by tag name **fail** and each test is rebuild twoce.

```
$ buildtest build -t fail --rebuild 2
```

User: docs
 Hostname: build-14361642-project-280831-buildtest
 Platform: Linux
 Current Time: 2021/07/30 15:08:56
 buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/bin/buildtest
 ↪10.0/bin/buildtest
 buildtest version: 0.10.0
 python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/↪python
 python version: 3.6.12
 Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests
 ↪10.0/var/tests
 Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/↪checkouts/v0.10.0/buildtest/settings/config.yml
 Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/↪bin/buildtest build -t fail --rebuild 2

+-----+				
Stage: Discovering Buildsspecs				
+-----+				
+-----+				
↪				
Discovered Buildsspecs				
↪				

(continues on next page)

(continued from previous page)

```

=====
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳ pass_returncode.yml |
=====
↳ -----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

BREAKDOWN OF BUILDSPECS BY TAGS
-----
Detected Tag Names: ['fail']
=====
↳ -----+
| fail
↳
=====
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳ pass_returncode.yml |
=====
↳ -----+

+-----+
| Stage: Parsing Buildsspecs |
+-----+

schemafilename | validstate | buildspectype
-----
↳ -----+
script-v1.0.schema.json | True | /home/docs/checkouts/readthedocs.org/user_
↳ builds/buildtest/checkouts/v0.10.0/tutorials/pass_returncode.yml

name | description
-----
exit1_fail | exit 1 by default is FAIL
exit1_pass | report exit 1 as PASS
returncode_list_mismatch | exit 2 failed since it failed to match returncode 1
returncode_int_match | exit 128 matches returncode 128
exit1_fail | exit 1 by default is FAIL
exit1_pass | report exit 1 as PASS
returncode_list_mismatch | exit 2 failed since it failed to match returncode 1
returncode_int_match | exit 128 matches returncode 128

+-----+
| Stage: Building Test |
+-----+

name | id | type | executor | tags
↳ | testpath

```

(continues on next page)

(continued from previous page)

```

-----+-----+-----+-----+-----+-----+-----+-----+-----+
↪ +-----+-----+-----+-----+-----+-----+-----+-----+-----+
↪ -----
exit1_fail          | e1052582 | script | generic.local.sh | ['tutorials', 'fail']_
↪ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↪ tests/generic.local.sh/pass_returncode/exit1_fail/3/exit1_fail_build.sh
exit1_pass          | 74ba735f | script | generic.local.sh | ['tutorials', 'pass']_
↪ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↪ tests/generic.local.sh/pass_returncode/exit1_pass/4/exit1_pass_build.sh
returncode_list_mismatch | 21d2f5cb | script | generic.local.sh | ['tutorials', 'fail']_
↪ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↪ tests/generic.local.sh/pass_returncode/returncode_list_mismatch/3/returncode_list_
↪ mismatch_build.sh
returncode_int_match | e52db458 | script | generic.local.sh | ['tutorials', 'pass']_
↪ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↪ tests/generic.local.sh/pass_returncode/returncode_int_match/4/returncode_int_match_
↪ build.sh
exit1_fail          | 65444282 | script | generic.local.sh | ['tutorials', 'fail']_
↪ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↪ tests/generic.local.sh/pass_returncode/exit1_fail/4/exit1_fail_build.sh
exit1_pass          | 68a9fdf0 | script | generic.local.sh | ['tutorials', 'pass']_
↪ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↪ tests/generic.local.sh/pass_returncode/exit1_pass/5/exit1_pass_build.sh
returncode_list_mismatch | c13bc1ff | script | generic.local.sh | ['tutorials', 'fail']_
↪ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↪ tests/generic.local.sh/pass_returncode/returncode_list_mismatch/4/returncode_list_
↪ mismatch_build.sh
returncode_int_match | 6d86d63b | script | generic.local.sh | ['tutorials', 'pass']_
↪ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↪ tests/generic.local.sh/pass_returncode/returncode_int_match/5/returncode_int_match_
↪ build.sh

+-----+
| Stage: Running Test |
+-----+

name          | id          | executor          | status  | returncode
-----+-----+-----+-----+-----+
exit1_fail    | e1052582   | generic.local.sh | FAIL    | 1
exit1_pass    | 74ba735f   | generic.local.sh | PASS    | 1
returncode_list_mismatch | 21d2f5cb   | generic.local.sh | FAIL    | 2
returncode_int_match | e52db458   | generic.local.sh | PASS    | 128
exit1_fail    | 65444282   | generic.local.sh | FAIL    | 1
exit1_pass    | 68a9fdf0   | generic.local.sh | PASS    | 1
returncode_list_mismatch | c13bc1ff   | generic.local.sh | FAIL    | 2
returncode_int_match | 6d86d63b   | generic.local.sh | PASS    | 128

```

(continues on next page)

(continued from previous page)

```
+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 4/8 Percentage: 50.000%
Failed Tests: 4/8 Percentage: 50.000%

Writing Logfile to: /tmp/buildtest_a0lx323f.log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /home/docs/checkouts/
↪readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log
```

The rebuild option expects a range between **1-50**, the `--rebuild=1` is equivalent to running without `--rebuild` option. We set a max limit for rebuild option to avoid system degradation due to high workload.

If you try to exceed this bound you will get an error such as:

```
.. command-output:: buildtest build -b tutorials/pass_returncode.yml --rebuild 51
```

Use Alternate Configuration file

If you want to use an alternate configuration file when building test you can use `buildtest -c <config> build`. buildtest will prefer configuration file on command line over the user configuration (`$HOME/.buildtest/config.yml`). For more details see *Which configuration file does buildtest read?*.

Keeping Stage Directory

buildtest will create setup the test environment in the *stage* directory where test will be executed. Once test is complete, buildtest will remove the *stage* directory. If you want to preserve the stage directory you can use `buildtest build --keep-stage-dir`, this is only useful if you want to run the test manually

3.4.2 Buildsspecs Interface

Now that we learned how to build tests, in this section we will discuss how one can query a builds spec cache. In buildtest, one can load all builds specs which is equivalent to validating all builds specs with the appropriate schema. Buildtest will ignore all invalid builds specs and store them in a separate file.

The `buildtest builds spec find` command is used for finding builds specs from builds spec cache. This command is also used for generating the builds spec cache. Shown below is a list of options for `buildtest builds spec find`.

```
$ buildtest builds spec find --help
usage: buildtest [options] [COMMANDS] builds spec find [-h] [--root ROOT] [-r] [-t] [-b] [-
↪e] [-p] [--group-by-tags]
                                     [--group-by-executor] [-m] [-mb] [--
↪filter FILTER]
                                     [--format FORMAT] [--helpfilter] [--
↪helpformat] [--terse]

optional arguments:
  -h, --help                show this help message and exit
  --root ROOT                Specify root builds specs (directory) path to load builds specs into.
↪builds spec cache.
```

(continues on next page)

(continued from previous page)

<code>-r, --rebuild</code>	Rebuild buildspec cache and find all buildspecs again
<code>-t, --tags</code>	List all available tags
<code>-b, --buildspec</code>	Get all buildspec files from cache
<code>-e, --executors</code>	get all unique executors from buildspecs
<code>-p, --paths</code>	print all root buildspec paths
<code>--group-by-tags</code>	Group tests by tag name
<code>--group-by-executor</code>	Group tests by executor name
<code>-m, --maintainers</code>	Get all maintainers for all buildspecs
<code>-mb, --maintainers-by-buildspecs</code>	Show maintainers breakdown by buildspecs
<code>--filter FILTER</code>	Filter buildspec cache with filter fields in format <code>--filter key1=val1,key2=val2</code>
<code>--format FORMAT</code>	Format buildspec cache with format fields in format <code>--format field1,field2,...</code>
<code>--helpfilter</code>	Show Filter fields for <code>--filter</code> option for filtering buildspec
<code>--helpformat</code>	Show Format fields for <code>--format</code> option for formatting buildspec
<code>--terse</code>	Print output in machine readable format

Finding Buildspecs

To find all buildspecs you can run `buildtest buildspec find` which will discover all buildspecs by recursively searching all `.yaml` extensions. `buildtest` will validate each buildspec file with the json schema and `buildtest` will display all valid buildspecs in the output, all invalid buildspecs will be stored in a file for post-processing.

```
$ buildtest buildspec find
```

name	description	type	executor	tags
skip	This test is skipped	script	generic.local.bash	['tutorials']
unskipped	This test is not skipped	script	generic.local.bash	['tutorials']
executor_regex_script_schema	regular expression test with executor using script schema	script	generic.local.(bash sh)	['tutorials']

(continues on next page)

(continued from previous page)

add_numbers		script	generic.local.bash	['tutorials']	↵
↪	Add X+Y				↵
↪					
+-----+		+-----+	+-----+	+-----+	
↪					
↪	+-----+				
python_hello		script	generic.local.bash	python	↵
↪	Hello World python				↵
↪					
+-----+		+-----+	+-----+	+-----+	
↪					
↪	+-----+				
_bin_sh_shell		script	generic.local.sh	['tutorials']	↵
↪	/bin/sh shell example				↵
↪					
+-----+		+-----+	+-----+	+-----+	
↪					
↪	+-----+				
_bin_bash_shell		script	generic.local.bash	['tutorials']	↵
↪	/bin/bash shell example				↵
↪					
+-----+		+-----+	+-----+	+-----+	
↪					
↪	+-----+				
bash_shell		script	generic.local.bash	['tutorials']	↵
↪	bash shell example				↵
↪					
+-----+		+-----+	+-----+	+-----+	
↪					
↪	+-----+				
sh_shell		script	generic.local.sh	['tutorials']	↵
↪	sh shell example				↵
↪					
+-----+		+-----+	+-----+	+-----+	
↪					
↪	+-----+				
shell_options		script	generic.local.sh	['tutorials']	↵
↪	shell options				↵
↪					
+-----+		+-----+	+-----+	+-----+	
↪					
↪	+-----+				
bash_login_shebang		script	generic.local.bash	tutorials	↵
↪	customize shebang line with bash login shell				↵
↪					
+-----+		+-----+	+-----+	+-----+	
↪					
↪	+-----+				
bash_nonlogin_shebang		script	generic.local.bash	tutorials	↵
↪	customize shebang line with default bash (nonlogin) shell				↵
↪					
+-----+		+-----+	+-----+	+-----+	
↪					
↪	+-----+				

(continues on next page)

(continued from previous page)

run_only_macos_distro	script	generic.local.bash	['mac']	┌
↳ Run test only if distro is darwin.				└
↳				
+-----+-----+-----+-----+				
↳ +-----+				
↳ +-----+				
run_only_linux_distro	script	generic.local.bash	['mac']	┌
↳ Run test only if distro is CentOS.				└
↳				
+-----+-----+-----+-----+				
↳ +-----+				
↳ +-----+				
selinux_disable	script	generic.local.bash	['tutorials']	┌
↳ Check if SELinux is Disabled				└
↳				
+-----+-----+-----+-----+				
↳ +-----+				
↳ +-----+				
exit1_fail	script	generic.local.sh	['tutorials',	
↳ 'fail']	exit 1 by default is FAIL			└
↳				
+-----+-----+-----+-----+				
↳ +-----+				
↳ +-----+				
exit1_pass	script	generic.local.sh	['tutorials',	
↳ 'pass']	report exit 1 as PASS			└
↳				
+-----+-----+-----+-----+				
↳ +-----+				
↳ +-----+				
returncode_list_mismatch	script	generic.local.sh	['tutorials',	
↳ 'fail']	exit 2 failed since it failed to match returncode 1			└
↳				
+-----+-----+-----+-----+				
↳ +-----+				
↳ +-----+				
returncode_int_match	script	generic.local.sh	['tutorials',	
↳ 'pass']	exit 128 matches returncode 128			└
↳				
+-----+-----+-----+-----+				
↳ +-----+				
↳ +-----+				
sleep	script	generic.local.bash	['tutorials']	┌
↳ sleep 2 seconds				└
↳				
+-----+-----+-----+-----+				
↳ +-----+				
↳ +-----+				
csh_shell	script	generic.local.csh	['tutorials']	┌
↳ csh shell example				└
↳				
+-----+-----+-----+-----+				
↳ +-----+				
↳ +-----+				

(continues on next page)

(continued from previous page)

string_tag	script	generic.local.bash	network	↳
↳ tags can be a string				↳
↳				
+-----+	+-----+	+-----+	+-----+	
↳ +-----+				
↳ +-----+				
list_of_strings_tags	script	generic.local.bash	['network',	
↳ 'ping']	tags can be a list of strings			↳
↳				
+-----+	+-----+	+-----+	+-----+	
↳ +-----+				
↳ +-----+				
bash_env_variables	script	generic.local.bash	['tutorials']	↳
↳ Declare environment variables in default shell (bash)				↳
↳				
+-----+	+-----+	+-----+	+-----+	
↳ +-----+				
↳ +-----+				
csh_env_declaration	script	generic.local.csh	['tutorials']	↳
↳ csh shell example to declare environment variables				↳
↳				
+-----+	+-----+	+-----+	+-----+	
↳ +-----+				
↳ +-----+				
tcsh_env_declaration	script	generic.local.csh	['tutorials']	↳
↳ tcsh shell example to declare environment variables				↳
↳				
+-----+	+-----+	+-----+	+-----+	
↳ +-----+				
↳ +-----+				
hello_world	script	generic.local.bash	tutorials	↳
↳ hello world example				↳
↳				
+-----+	+-----+	+-----+	+-----+	
↳ +-----+				
↳ +-----+				
variables_bash	script	generic.local.bash	['tutorials']	↳
↳ Declare shell variables in bash				↳
↳				
+-----+	+-----+	+-----+	+-----+	
↳ +-----+				
↳ +-----+				
run_only_platform_darwin	script	generic.local.python	['tutorials']	↳
↳ This test will only run if target platform is Darwin				↳
↳				
+-----+	+-----+	+-----+	+-----+	
↳ +-----+				
↳ +-----+				
run_only_platform_linux	script	generic.local.python	['tutorials']	↳
↳ This test will only run if target platform is Linux				↳
↳				
+-----+	+-----+	+-----+	+-----+	
↳ +-----+				
↳ +-----+				

(continues on next page)

(continued from previous page)

```

| circle_area          | script  | generic.local.python  | ['tutorials',
↳ 'python']          | Calculate circle of area given a radius
↳
+-----+-----+-----+-----+
↳-----+
↳-----+
| timelimit_min_max    | script  | generic.local.sh      | ['tutorials']
↳              | Run a sleep job for 2 seconds and test pass if its within 1.0-3.0sec
↳              |
+-----+-----+-----+-----+
↳-----+
↳-----+
| timelimit_min        | script  | generic.local.sh      | ['tutorials']
↳              | Run a sleep job for 2 seconds and test pass if its exceeds min time
↳ of 1.0 sec |
+-----+-----+-----+-----+
↳-----+
↳-----+
| timelimit_max        | script  | generic.local.sh      | ['tutorials']
↳              | Run a sleep job for 2 seconds and test pass if it's within max time:
↳ 5.0 sec    |
+-----+-----+-----+-----+
↳-----+
↳-----+
| timelimit_min_fail   | script  | generic.local.sh      | ['tutorials']
↳              | This test fails because it runs less than mintime of 10 second
↳              |
+-----+-----+-----+-----+
↳-----+
↳-----+
| timelimit_max_fail   | script  | generic.local.sh      | ['tutorials']
↳              | This test fails because it exceeds maxtime of 1.0 second
↳              |
+-----+-----+-----+-----+
↳-----+
↳-----+
| run_only_as_root     | script  | generic.local.bash    | ['tutorials']
↳              | This test will only run if current user is root
↳              |
+-----+-----+-----+-----+
↳-----+
↳-----+
| spack_test           | spack   | generic.local.sh      | ['spack']
↳              | Install bzip2 and run spack test and report results
↳              |
+-----+-----+-----+-----+
↳-----+
↳-----+
| spack_env_directory  | spack   | generic.local.sh      | ['spack']
↳              | Concretize m4 in a spack environment named m4
↳              |
+-----+-----+-----+-----+
↳-----+
↳-----+

```

(continues on next page)

(continued from previous page)

run_pre_post_commands	spack	generic.local.sh	['spack']	┌
└─ Install zlib				└
-----+	-----+	-----+	-----+	
remove_environment_automatically	spack	generic.local.sh	['spack']	┌
└─ remove spack environment automatically before creating a new				└
environment				
-----+	-----+	-----+	-----+	
remove_environment_explicit	spack	generic.local.sh	['spack']	┌
└─ remove spack environment explicitly using the 'rm' property				└
-----+	-----+	-----+	-----+	
spack_test_results_specs_format	spack	generic.local.sh	['spack']	┌
└─ Run spack test results with spec format				└
-----+	-----+	-----+	-----+	
concretize_m4_in_spack_env	spack	generic.local.sh	['spack']	┌
└─ Concretize m4 in a spack environment named m4				└
-----+	-----+	-----+	-----+	
spack_env_create_from_manifest	spack	generic.local.sh	['spack']	┌
└─ Create spack environment from spack.yaml				└
-----+	-----+	-----+	-----+	
spack_sbatch_example	spack	generic.local.sh	['spack']	┌
└─ sbatch directives can be defined in spack schema				└
-----+	-----+	-----+	-----+	
add_mirror	spack	generic.local.sh	['spack']	┌
└─ Declare spack mirror				└
-----+	-----+	-----+	-----+	
add_mirror_in_spack_env	spack	generic.local.sh	['spack']	┌
└─ Declare spack mirror in spack environment				└
-----+	-----+	-----+	-----+	

(continues on next page)

(continued from previous page)

install_zlib	spack	generic.local.sh	['spack']	┌
└─ Install zlib				└
-----	-----	-----	-----	
└─	└─	└─	└─	
openmp_hello_c_example	compiler	generic.local.bash	['tutorials',	
└─ 'compile']	OpenMP Hello World C example			└
-----	-----	-----	-----	
└─	└─	└─	└─	
default_status_regex	compiler	generic.local.bash	['tutorials',	
└─ 'compile']	Regular expression check in stdout for gcc group			└
-----	-----	-----	-----	
└─	└─	└─	└─	
override_status_regex	compiler	generic.local.bash	['tutorials',	
└─ 'compile']	Override regular expression for compiler gcc/10.2.0-37fmsw7			└
-----	-----	-----	-----	
└─	└─	└─	└─	
override_environmentvars	compiler	generic.local.bash	['tutorials',	
└─ 'compile']	override default environment variables			└
-----	-----	-----	-----	
└─	└─	└─	└─	
custom_run_by_compilers	compiler	generic.local.bash	['tutorials',	
└─ 'compile']	Customize binary launch based on compiler			└
-----	-----	-----	-----	
└─	└─	└─	└─	
vecadd_gnu_exclude	compiler	generic.local.bash	['tutorials',	
└─ 'compile']	Vector Addition example with GNU compilers but exclude gcc@10.2.0			└
-----	-----	-----	-----	
└─	└─	└─	└─	
hello_c	compiler	generic.local.bash	['tutorials',	
└─ 'compile']	Hello World C Compilation			└
-----	-----	-----	-----	
└─	└─	└─	└─	
vecadd_gnu	compiler	generic.local.bash	['tutorials',	
└─ 'compile']	Vector Addition example with GNU compiler			└
-----	-----	-----	-----	
└─	└─	└─	└─	

(continues on next page)

(continued from previous page)

hello_f	compiler generic.local.bash	['tutorials',	
↳ 'compile']	Hello World Fortran Compilation		↳
↳			
+-----+			
↳			
↳			
pre_post_build_run	compiler generic.local.bash	['tutorials',	
↳ 'compile']	example using pre_build, post_build, pre_run, post_run example		↳
↳			
+-----+			
↳			
↳			
show_lsf_user_groups	script generic.local.bash	lsf	
↳	Show information about all LSF user groups		↳
↳			
+-----+			
↳			
↳			
show_host_groups	script generic.local.bash	lsf	
↳	Show information about host groups using bmggroup		↳
↳			
+-----+			
↳			
↳			
show_lsf_queues	script generic.local.bash	lsf	
↳	Show LSF queues		↳
↳			
+-----+			
↳			
↳			
show_lsf_queues_formatted	script generic.local.bash	lsf	
↳	Show LSF queues with formatted columns		↳
↳			
+-----+			
↳			
↳			
show_lsf_queues_current_user	script generic.local.bash	lsf	
↳	Show LSF queues available for current user		↳
↳			
+-----+			
↳			
↳			
show_lsf_configuration	script generic.local.bash	lsf	
↳	Show LSF configuration using lsinfo		↳
↳			
+-----+			
↳			
↳			
show_lsf_models	script generic.local.bash	lsf	
↳	Show information about host models in LSF cluster		↳
↳			
+-----+			
↳			
↳			

(continues on next page)

(continued from previous page)

show_lsf_resources	script	generic.local.bash	lsf	↵
↵	Show information about LSF resources			↵
↵				
+-----+				
↵				
↵				
+-----+				
lsf_version	script	generic.local.bash	lsf	↵
↵	Display lsf version using lsinfo			↵
↵				
+-----+				
↵				
↵				
+-----+				
display_lsf_hosts	script	generic.local.bash	lsf	↵
↵	Show all hosts in LSF cluster			↵
↵				
+-----+				
↵				
↵				
+-----+				
display_hosts_format	script	generic.local.bash	lsf	↵
↵	Show all hosts with column hostname and status			↵
↵				
+-----+				
↵				
↵				
+-----+				
bhosts_version	script	generic.local.bash	lsf	↵
↵	display version from bhosts command			↵
↵				
+-----+				
↵				
↵				
+-----+				
current_user_queue	script	generic.local.bash	['slurm']	↵
↵	show all current pending jobs for current user (squeue -u \$USER)			↵
↵				
+-----+				
↵				
↵				
+-----+				
show_all_jobs	script	generic.local.bash	['slurm']	↵
↵	show all pending + running jobs (squeue -a)			↵
↵				
+-----+				
↵				
↵				
+-----+				
show_accounts	script	generic.local.bash	['slurm']	↵
↵	run sacctmgr list accounts			↵
↵				
+-----+				
↵				
↵				
+-----+				
show_users	script	generic.local.bash	['slurm']	↵
↵	run sacctmgr list users			↵
↵				
+-----+				
↵				
↵				
+-----+				

(continues on next page)

(continued from previous page)

show_qos	script	generic.local.bash	['slurm']	↵
↪ run sacctmgr list qos				↵
↪				
+-----+	+-----+	+-----+	+-----+	
↪ +-----+				
↪ +-----+				
show_tres	script	generic.local.bash	['slurm']	↵
↪ run sacctmgr list tres				↵
↪				
+-----+	+-----+	+-----+	+-----+	
↪ +-----+				
↪ +-----+				
slurm_config	script	generic.local.bash	['slurm']	↵
↪ run scontrol show config				↵
↪				
+-----+	+-----+	+-----+	+-----+	
↪ +-----+				
↪ +-----+				
show_partition	script	generic.local.bash	['slurm']	↵
↪ run scontrol show partition				↵
↪				
+-----+	+-----+	+-----+	+-----+	
↪ +-----+				
↪ +-----+				
nodes_state_down	script	generic.local.bash	['slurm']	↵
↪ Show nodes in DOWN state				↵
↪				
+-----+	+-----+	+-----+	+-----+	
↪ +-----+				
↪ +-----+				
nodes_state_reboot	script	generic.local.bash	['slurm']	↵
↪ Show nodes in REBOOT state				↵
↪				
+-----+	+-----+	+-----+	+-----+	
↪ +-----+				
↪ +-----+				
nodes_state_allocated	script	generic.local.bash	['slurm']	↵
↪ Show nodes in ALLOCATED state				↵
↪				
+-----+	+-----+	+-----+	+-----+	
↪ +-----+				
↪ +-----+				
nodes_state_completing	script	generic.local.bash	['slurm']	↵
↪ Show nodes in COMPLETING state				↵
↪				
+-----+	+-----+	+-----+	+-----+	
↪ +-----+				
↪ +-----+				
nodes_state_idle	script	generic.local.bash	['slurm']	↵
↪ Show nodes in IDLE state				↵
↪				
+-----+	+-----+	+-----+	+-----+	
↪ +-----+				
↪ +-----+				

(continues on next page)

(continued from previous page)

node_down_fail_list_reason	script	generic.local.bash	['slurm']	↵
↵	Show nodes DOWN, DRAINED, FAIL or FAILING and list reason			↵
↵				
+-----+				
↵				
↵				
dead_nodes	script	generic.local.bash	['slurm']	↵
↵	Show non-responding nodes			↵
↵				
+-----+				
↵				
↵				
get_partitions	script	generic.local.bash	['slurm']	↵
↵	Get all slurm partitions			↵
↵				
+-----+				
↵				
↵				
sinfo_version	script	generic.local.bash	['slurm']	↵
↵	get slurm version			↵
↵				
+-----+				
↵				
↵				
qsub_version	script	generic.local.sh	['cobalt']	↵
↵	print version for qsub command			↵
↵				
+-----+				
↵				
↵				
qselect_version	script	generic.local.sh	['cobalt']	↵
↵	print version for qselect			↵
↵				
+-----+				
↵				
↵				
cqsub_version	script	generic.local.sh	['cobalt']	↵
↵	print version for cqsub command			↵
↵				
+-----+				
↵				
↵				
qdel_version	script	generic.local.sh	['cobalt']	↵
↵	print version for qdel command			↵
↵				
+-----+				
↵				
↵				
qmove_version	script	generic.local.sh	['cobalt']	↵
↵	print version for qmove command			↵
↵				
+-----+				
↵				
↵				

(continues on next page)

(continued from previous page)

show_jobs	script	generic.local.sh	['cobalt']	↵
↵	Show all jobs in queue			↵
↵				
+-----+				
↵				
↵				
+-----+				
show_queues	script	generic.local.sh	['cobalt']	↵
↵	Show all queues			↵
↵				
+-----+				
↵				
↵				
+-----+				
root_disk_usage	script	generic.local.bash	['filesystem',	
↵ 'storage']	Check root disk usage and report if it exceeds threshold			↵
↵				
+-----+				
↵				
↵				
+-----+				
systemd_default_target	script	generic.local.bash	['system']	↵
↵	check if default target is multi-user.target			↵
↵				
+-----+				
↵				
↵				
+-----+				
ssh_localhost_remotecommand	script	generic.local.bash	['ssh']	↵
↵	Test if ssh on localhost works and if we can run remote command.			↵
↵				
+-----+				
↵				
↵				
+-----+				
kernel_swapusage	script	generic.local.bash	['configuration	
↵ '']	Retrieve Kernel Swap Usage			↵
↵				
+-----+				
↵				
↵				
+-----+				
ulimit_filelock_unlimited	script	generic.local.bash	['system']	↵
↵	Check if file lock is set to unlimited in ulimits			↵
↵				
+-----+				
↵				
↵				
+-----+				
ulimit_cputime_unlimited	script	generic.local.bash	['system']	↵
↵	Check if cputime is set to unlimited in ulimits			↵
↵				
+-----+				
↵				
↵				
+-----+				
ulimit_stacksize_unlimited	script	generic.local.bash	['system']	↵
↵	Check if stack size is set to unlimited in ulimits			↵
↵				
+-----+				
↵				
↵				
+-----+				

(continues on next page)

(continued from previous page)

ulimit_vmsize_unlimited	script	generic.local.bash	['system']	┌
└─> Check virtual memory size and check if its set to unlimited				└
└─>				
+-----+-----+-----+-----+				
└─> +-----+				
└─> +-----+				
ulimit_filedescriptor_4096	script	generic.local.bash	['system']	┌
└─> Check if open file descriptors limit is set to 4096				└
└─>				
+-----+-----+-----+-----+				
└─> +-----+				
└─> +-----+				
ulimit_max_user_process_2048	script	generic.local.bash	['system']	┌
└─> Check max number of user process limit is set to 2048				└
└─>				
+-----+-----+-----+-----+				
└─> +-----+				
└─> +-----+				
runImage	script	generic.local.bash	['containers',	
└─> 'singularity'] run container docker://godlovedc/lolcow				┌
└─>				
+-----+-----+-----+-----+				
└─> +-----+				
└─> +-----+				
build_sif_from_dockerimage	script	generic.local.bash	['containers',	
└─> 'singularity'] build sif image from docker image docker://godlovedc/lolcow				┌
└─>				
+-----+-----+-----+-----+				
└─> +-----+				
└─> +-----+				
build_sandbox_image	script	generic.local.bash	['containers',	
└─> 'singularity'] build sandbox image from docker image docker://godlovedc/lolcow				┌
└─>				
+-----+-----+-----+-----+				
└─> +-----+				
└─> +-----+				
build_remoteimages	script	generic.local.bash	['containers',	
└─> 'singularity'] build remote hosted image from AWS				┌
└─>				
+-----+-----+-----+-----+				
└─> +-----+				
└─> +-----+				
pullImage_dockerhub	script	generic.local.bash	['containers',	
└─> 'singularity'] Pull image docker://godlovedc/lolcow from DockerHub				┌
└─>				
+-----+-----+-----+-----+				
└─> +-----+				
└─> +-----+				
pullImage_sylabscloud	script	generic.local.bash	['containers',	
└─> 'singularity'] Pull image library://alpine:latest from Sylabs Cloud				┌
└─>				
+-----+-----+-----+-----+				
└─> +-----+				
└─> +-----+				

(continues on next page)

(continued from previous page)

```
| pullImage_shub          | script | generic.local.bash      | ['containers',
↳ 'singularity'] | Pull image shub://vsoch/singularity-images from SingularityHub
↳
+-----+-----+-----+-----+
↳ -----+
↳
| inspect_image          | script | generic.local.bash      | ['containers',
↳ 'singularity'] | Inspect image via 'singularity inspect'
↳
+-----+-----+-----+-----+
↳ -----+
↳
```

buildtest will load all discovered buildsspecs in a cache file (JSON) which is created upon running `buildtest buildspec find`. Any subsequent runs will read from cache and update if any new buildsspecs are added. If you make changes to buildspec you should rebuild the buildspec cache by running:

```
$ buildtest buildspec find --rebuild
```

If you want to find all buildspec files in cache you can run `buildtest buildspec find --buildspec`. Shown below is an example output.

```
$ buildtest buildspec find --buildspec
+-----+
↳ -----+
| buildspecs
↳
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳ skip_tests.yml
|
+-----+
↳ -----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳ executor_regex_script.yml
|
+-----+
↳ -----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳ add_numbers.yml
|
+-----+
↳ -----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳ python-hello.yml
|
+-----+
↳ -----+
...

```

The `buildtest buildspec find --paths` will display a list of root directories buildtest will search for buildsspecs when running `buildtest buildspec find`. One can define these directories in the configuration file or pass them via command line.

```
$ buildtest buildspec find --paths
/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials
/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
↳ tests
```

(continues on next page)

(continued from previous page)

buildtest will search buildsspecs in *buildspecs root* defined in your configuration, which is a list of directory paths to search for buildsspecs. If you want to load buildsspecs from a directory path, you can specify a directory via `--root` option in the format: `buildtest buildspec find --root <path> --rebuild`. buildtest will load all valid buildsspecs into cache and ignore the rest. It's important to add `--rebuild` if you want to regenerate buildspec cache.

Filtering buildspec

Once you have a buildspec cache, we can query the buildspec cache for certain attributes. When you run **buildtest buildspec find** it will report all buildsspecs from cache which can be difficult to process. Therefore, we have a filter option (`--filter`) to restrict our search. Let's take a look at the available filter fields that are acceptable with filter option.

```
$ buildtest buildspec find --helpfilter
```

Field	Description	Type
-----	-----	-----
executor	Filter by executor name	STRING
tags	Filter by tag name	STRING
type	Filter by schema type	STRING

The `--filter` option expects an arguments in **key=value** format as follows:

```
buildtest buildspec find --filter key1=value1,key2=value2,key3=value3
```

We can filter buildspec cache by `tags=fail` which will query all tests with associated tag field in test.

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

name	type	executor	tags	
description				
=====	=====	=====	=====	=====
exit1_fail	script	generic.local.sh	['tutorials', 'fail']	exit 1
by default is FAIL				
-----	-----	-----	-----	-----
returncode_list_mismatch	script	generic.local.sh	['tutorials', 'fail']	exit 2
failed since it failed to match returncode 1				
-----	-----	-----	-----	-----

In addition, we can query buildspecs by schema type using `type` property. In this example we query all tests by **type** property

```
$ buildtest buildspec find --filter type=script
```

name	type	executor	tags
description			

(continues on next page)

```

| skip | script | generic.local.bash | ['tutorials']
↳ | This test is skipped
↳ |
+-----+-----+-----+-----+
↳-----+
↳---+
| unskipped | script | generic.local.bash | ['tutorials']
↳ | This test is not skipped
↳ |
+-----+-----+-----+-----+
↳-----+
↳---+
| executor_regex_script_schema | script | generic.local.(bash|sh) | ['tutorials']
↳ | regular expression test with executor using script schema
↳ |
+-----+-----+-----+-----+
↳-----+
↳---+
| add_numbers | script | generic.local.bash | ['tutorials']
↳ | Add X+Y
↳ |
+-----+-----+-----+-----+
↳-----+
↳---+
| python_hello | script | generic.local.bash | python
↳ | Hello World python
↳ |
+-----+-----+-----+-----+
↳-----+
↳---+
| _bin_sh_shell | script | generic.local.sh | ['tutorials']
↳ | /bin/sh shell example
↳ |
+-----+-----+-----+-----+
↳-----+
↳---+
| _bin_bash_shell | script | generic.local.bash | ['tutorials']
↳ | /bin/bash shell example
↳ |
+-----+-----+-----+-----+
↳-----+
↳---+
| bash_shell | script | generic.local.bash | ['tutorials']
↳ | bash shell example
↳ |
+-----+-----+-----+-----+
↳-----+
↳---+
| sh_shell | script | generic.local.sh | ['tutorials']
↳ | sh shell example
↳ |

```

(continued from previous page)

```
+-----+-----+-----+-----+
↪-----+-----+-----+-----+
↪---+
...

```

Finally, we can combine multiple filter fields separated by comma, in the next example we can query all buildsspecs with tags=tutorials, executor=generic.local.sh, and type=script

```
$ buildtest buildspec find --filter tags=tutorials,executor=generic.local.sh,type=script
+-----+-----+-----+-----+-----+
↪-----+-----+-----+-----+
| name                | type  | executor      | tags                |  |
↪description          |       |               |                    |  |
+=====+=====+=====+=====+=====+
| _bin_sh_shell       | script | generic.local.sh | ['tutorials']       | /bin/sh
↪shell example       |       |               |                    |  |
+-----+-----+-----+-----+-----+
↪-----+-----+-----+-----+
| sh_shell            | script | generic.local.sh | ['tutorials']       | sh
↪shell example       |       |               |                    |  |
+-----+-----+-----+-----+-----+
↪-----+-----+-----+-----+
| shell_options       | script | generic.local.sh | ['tutorials']       | shell
↪options             |       |               |                    |  |
+-----+-----+-----+-----+-----+
↪-----+-----+-----+-----+
| exit1_fail          | script | generic.local.sh | ['tutorials', 'fail'] | exit 1
↪by default is FAIL |       |               |                    |  |
+-----+-----+-----+-----+-----+
↪-----+-----+-----+-----+
| exit1_pass          | script | generic.local.sh | ['tutorials', 'pass'] | report
↪exit 1 as PASS      |       |               |                    |  |
+-----+-----+-----+-----+-----+
↪-----+-----+-----+-----+
| returncode_list_mismatch | script | generic.local.sh | ['tutorials', 'fail'] | exit 2
↪failed since it failed to match returncode 1 |       |               |                    |  |
+-----+-----+-----+-----+-----+
↪-----+-----+-----+-----+
| returncode_int_match  | script | generic.local.sh | ['tutorials', 'pass'] | exit
↪128 matches returncode 128 |       |               |                    |  |
+-----+-----+-----+-----+-----+
↪-----+-----+-----+-----+
| timelimit_min_max    | script | generic.local.sh | ['tutorials']       | Run a
↪sleep job for 2 seconds and test pass if its within 1.0-3.0sec |       |               |                    |  |
+-----+-----+-----+-----+-----+
↪-----+-----+-----+-----+
| timelimit_min        | script | generic.local.sh | ['tutorials']       | Run a
↪sleep job for 2 seconds and test pass if its exceeds min time of 1.0 sec |       |               |                    |  |
+-----+-----+-----+-----+-----+
↪-----+-----+-----+-----+
| timelimit_max        | script | generic.local.sh | ['tutorials']       | Run a
↪sleep job for 2 seconds and test pass if it's within max time: 5.0 sec |       |               |                    |  |

```

(continues on next page)

(continued from previous page)

```

+-----+-----+-----+-----+-----+-----+
↪ | timelimit_min_fail      | script | generic.local.sh | ['tutorials'] | This_
↪ | test fails because it runs less than mintime of 10 second |
+-----+-----+-----+-----+-----+-----+
↪ | timelimit_max_fail      | script | generic.local.sh | ['tutorials'] | This_
↪ | test fails because it exceeds maxtime of 1.0 second |
+-----+-----+-----+-----+-----+-----+
↪ |

```

Format buildspec cache

We have seen how one can filter buildspecs, but we can also configure which columns to display in the output of **buildtest buildspec find**. By default, we show a pre-selected format fields in the output, however there are more format fields available that can be configured at the command line.

The format fields are specified in comma separated format such as **buildtest buildspec find --format <field1>,<field2>,...**. You can see a list of all format fields by **--helpformat** option as shown below

```

$ buildtest buildspec find --helpformat
Field      Description
-----
description Format by description
executor    Format by executor type
file        Format by file
name        Format by test name
tags        Format by tag name
type        Format by schema type

```

In the next example, we utilize **--format** field with **--filter** option to show how format fields affect table columns. **buildtest** will display the table in order of format fields specified in command line.

```

$ buildtest buildspec find --format name,description,file --filter tags=tutorials,
↪ executor=generic.local.sh
+-----+-----+-----+-----+-----+-----+
↪ | name | description |
↪ |      | file       |
↪ |      |           |
+=====+=====+=====+=====+=====+=====+
↪ | _bin_sh_shell | /bin/sh shell example |
↪ |               | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↪ |               | checkouts/v0.10.0/tutorials/shell_examples.yml |
+-----+-----+-----+-----+-----+-----+
↪ | sh_shell | sh shell example |
↪ |          | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↪ |          | checkouts/v0.10.0/tutorials/shell_examples.yml |

```

(continues on next page)

(continued from previous page)

```

+-----+
+-----+
+-----+
| shell_options          | shell options                                     |
+-----+-----+
|                         | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
+-----+-----+
|                         | checkouts/v0.10.0/tutorials/shell_examples.yml      |
+-----+-----+
+-----+
+-----+
+-----+
| exit1_fail             | exit 1 by default is FAIL                         |
+-----+-----+
|                         | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
+-----+-----+
|                         | checkouts/v0.10.0/tutorials/pass_returncode.yml     |
+-----+-----+
+-----+
+-----+
+-----+
| exit1_pass             | report exit 1 as PASS                             |
+-----+-----+
|                         | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
+-----+-----+
|                         | checkouts/v0.10.0/tutorials/pass_returncode.yml     |
+-----+-----+
+-----+
+-----+
+-----+
| returncode_list_mismatch | exit 2 failed since it failed to match returncode 1
+-----+-----+
|                         | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
+-----+-----+
|                         | checkouts/v0.10.0/tutorials/pass_returncode.yml     |
+-----+-----+
+-----+
+-----+
+-----+
| returncode_int_match    | exit 128 matches returncode 128                   |
+-----+-----+
|                         | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
+-----+-----+
|                         | checkouts/v0.10.0/tutorials/pass_returncode.yml     |
+-----+-----+
+-----+
+-----+
+-----+
| timelimit_min_max      | Run a sleep job for 2 seconds and test pass if its within 1.
+-----+-----+
| 0-3.0sec               | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
+-----+-----+
|                         | checkouts/v0.10.0/tutorials/runtime_status_test.yml |
+-----+-----+
+-----+
+-----+
+-----+
| timelimit_min          | Run a sleep job for 2 seconds and test pass if its exceeds
+-----+-----+
| min time of 1.0 sec    | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
+-----+-----+
|                         | checkouts/v0.10.0/tutorials/runtime_status_test.yml |
+-----+-----+
+-----+
+-----+
+-----+
| timelimit_max          | Run a sleep job for 2 seconds and test pass if it's within
+-----+-----+
| max time: 5.0 sec      | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
+-----+-----+
|                         | checkouts/v0.10.0/tutorials/runtime_status_test.yml |
+-----+-----+
+-----+
+-----+
+-----+
| timelimit_min_fail     | This test fails because it runs less than mintime of 10
+-----+-----+
| second                 | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
+-----+-----+
|                         | checkouts/v0.10.0/tutorials/runtime_status_test.yml |
+-----+-----+

```

(continued from previous page)

```
+-----+-----+
| timelimit_max_fail | This test fails because it exceeds maxtime of 1.0 second |
|                     | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/ |
|                     | checkouts/v0.10.0/tutorials/runtime_status_test.yml |
+-----+-----+
```

buildtest makes use of python library named `tabulate` to generate these tables which are found in commands line like `buildtest buildspect find` and `buildtest report`.

Querying buildspect tags

If you want to retrieve all unique tags from all buildspects you can run `buildtest buildspect find --tags`. This can be useful if you want to know available tags in your buildspect cache.

```
$ buildtest buildspect find --tags
+-----+
| Tags   |
+-----+
| singularity |
+-----+
| lsf      |
+-----+
| python   |
+-----+
| compile  |
+-----+
| configuration |
+-----+
| pass     |
+-----+
| fail     |
+-----+
| mac      |
+-----+
| ssh      |
+-----+
| containers |
+-----+
| slurm     |
+-----+
| system    |
+-----+
| spack     |
+-----+
| tutorials |
+-----+
| filesystem |
```

(continues on next page)

(continued from previous page)

```
+-----+
| storage |
+-----+
| cobalt  |
+-----+
| network |
+-----+
| ping    |
+-----+
```

In addition, buildtest can group tests by tags via `buildtest buildspect find --group-by-tags` which can be useful if you want to know which tests get executed when running `buildtest build --tags`. The output is grouped by tag names, followed by name of test and description.

```
$ buildtest buildspect find --group-by-tags
+-----+-----+-----+
| tags      | name                               | description                                     |
+-----+-----+-----+
| tutorials | skip                               | This test is skipped                         |
+-----+-----+-----+
| tutorials | unskipped                         | This test is not skipped                     |
+-----+-----+-----+
| tutorials | executor_regex_script_schema      | regular expression test with_                |
|           | executor using script schema       | |                                              |
+-----+-----+-----+
| tutorials | add_numbers                       | Add X+Y                                     |
+-----+-----+-----+
| tutorials | _bin_sh_shell                     | /bin/sh shell example                       |
+-----+-----+-----+
| tutorials | _bin_bash_shell                   | /bin/bash shell example                     |
+-----+-----+-----+
| tutorials | bash_shell                        | bash shell example                         |
+-----+-----+-----+
| tutorials | sh_shell                          | sh shell example                           |
+-----+-----+-----+
```

(continues on next page)

(continued from previous page)

tutorials	shell_options	shell options	↵
↵			
+-----+	+-----+	+-----+	
↵			
tutorials	bash_login_shebang	customize shebang line with bash	↵
↵login shell			
+-----+	+-----+	+-----+	
↵			
tutorials	bash_nonlogin_shebang	customize shebang line with default	↵
↵bash (nonlogin) shell			
+-----+	+-----+	+-----+	
↵			
tutorials	selinux_disable	Check if SELinux is Disabled	↵
↵			
+-----+	+-----+	+-----+	
↵			
tutorials	exit1_fail	exit 1 by default is FAIL	↵
↵			
+-----+	+-----+	+-----+	
↵			
tutorials	exit1_pass	report exit 1 as PASS	↵
↵			
+-----+	+-----+	+-----+	
↵			
tutorials	returncode_list_mismatch	exit 2 failed since it failed to	↵
↵match returncode 1			
+-----+	+-----+	+-----+	
↵			
tutorials	returncode_int_match	exit 128 matches returncode 128	↵
↵			
+-----+	+-----+	+-----+	
↵			
tutorials	sleep	sleep 2 seconds	↵
↵			
+-----+	+-----+	+-----+	
↵			
tutorials	csh_shell	csh shell example	↵
↵			
+-----+	+-----+	+-----+	
↵			
tutorials	bash_env_variables	Declare environment variables in	↵
↵default shell (bash)			
+-----+	+-----+	+-----+	
↵			
...			

Querying buildspec executor

If you want to know all executors in your buildspec cache use the `buildtest buildspec find --executors` command. This can be useful when you want to build by executors (`buildtest build --executor`).

```
$ buildtest buildspec find --executors
+-----+
| executors |
+=====+
| generic.local.(bash|sh) |
+-----+
| generic.local.bash |
+-----+
| generic.local.python |
+-----+
| generic.local.sh |
+-----+
| generic.local.csh |
+-----+
```

Similar to `--group-by-tags`, `buildtest` has an option to group tests by executor using `--group-by-executor` option. This will show tests grouped by executor, name of test and test description. Shown below is an example output.

```
$ buildtest buildspec find --group-by-executor
+-----+-----+-----+
| executor | name | description |
+-----+-----+-----+
| generic.local.bash | skip | This test is skipped |
+-----+-----+-----+
| generic.local.bash | unskipped | This test is not skipped |
+-----+-----+-----+
| generic.local.bash | add_numbers | Add X+Y |
+-----+-----+-----+
| generic.local.bash | python_hello | Hello World python |
+-----+-----+-----+
| generic.local.bash | _bin_bash_shell | /bin/bash shell example |
+-----+-----+-----+
| generic.local.bash | bash_shell | bash shell example |
+-----+-----+-----+
```

(continues on next page)

(continued from previous page)

generic.local.bash	bash_login_shebang	customize shebang line
↳with bash login shell		
+-----+	+-----+	+-----+
↳		
generic.local.bash	bash_nonlogin_shebang	customize shebang line
↳with default bash (nonlogin) shell		
+-----+	+-----+	+-----+
↳		
generic.local.bash	run_only_macos_distro	Run test only if distro
↳is darwin.		
+-----+	+-----+	+-----+
↳		
generic.local.bash	run_only_linux_distro	Run test only if distro
↳is CentOS.		
+-----+	+-----+	+-----+
↳		
generic.local.bash	selinux_disable	Check if SELinux is
↳Disabled		
+-----+	+-----+	+-----+
↳		
generic.local.bash	sleep	sleep 2 seconds
↳		
+-----+	+-----+	+-----+
↳		
generic.local.bash	string_tag	tags can be a string
↳		
+-----+	+-----+	+-----+
↳		
generic.local.bash	list_of_strings_tags	tags can be a list of
↳strings		
+-----+	+-----+	+-----+
↳		
...		

Query Maintainers in buildspects

When you are writing your buildspects, you can specify the `maintainers` field to assign authors to buildspects. buildtest can query the maintainers from the cache once buildspects are loaded. You can retrieve all maintainers using `--maintainers` option or `-m` short option. In this example, we show all maintainers for buildspects in buildspect cache

```
$ buildtest buildspect find --maintainers
+-----+
| maintainers |
+=====+
| @shahzebsiddiqui |
+-----+
```

If you want to see a breakdown of maintainers by buildspect file you can use `--maintainers-by-buildspecs` or `-mb` short option. This can be useful to get correlation between maintainers and the buildspect file.

```
$ buildtest buildspec find -mb
+-----+-----+
| maintainers      | buildspec
|
+-----+-----+
| @shahzebsiddiqui | ['/home/docs/checkouts/readthedocs.org/user_builds/buildtest/
|                   | checkouts/v0.10.0/tutorials/hello_world.yml', '/home/docs/checkouts/readthedocs.org/
|                   | user_builds/buildtest/checkouts/v0.10.0/general_tests/configuration/ulimits.yml'] |
+-----+-----+
|
+-----+-----+
```

Validate Buildsspecs

buildtest can validate buildsspecs through the `buildtest buildspec validate` command which provides analogous options for `buildtest build` for selecting buildsspecs such as `-b`, `-e`, `-t` and `-x`. This command can be used to validate buildsspecs with the JSON Schema which can be useful if you are writing a buildspec and want to validate the buildspec without running the test.

Shown below are the available command options.

```
$ buildtest buildspec validate --help
usage: buildtest [options] [COMMANDS] buildspec validate [-h] [-b BUILDSPEC] [-x EXCLUDE] [-t TAG] [-e EXECUTOR]

optional arguments:
  -h, --help            show this help message and exit
  -b BUILDSPEC, --buildspec BUILDSPEC
                        Specify path to buildspec (file, or directory) to validate
  -x EXCLUDE, --exclude EXCLUDE
                        Specify path to buildspec to exclude (file or directory) during validation
  -t TAG, --tag TAG      Specify buildsspecs by tag name to validate
  -e EXECUTOR, --executor EXECUTOR
                        Specify buildsspecs by executor name to validate
```

The `-b` option can be used to specify path to buildspec file or directory to validate buildsspecs. If its a directory, buildtest will traverse all directories recursively and find any `.yaml` file extensions and attempt to validate each buildspec. Shown below is an example output of what it may look like

```
$ buildtest buildspec validate -b tutorials/vars.yaml
Processing buildspec: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
| checkouts/v0.10.0/tutorials/vars.yaml
All buildsspecs passed validation!!!
```

If buildtest detects an error during validation, the error message will be displayed to screen as we see in this example

```
$ buildtest buildspec validate -b tutorials/invalid_tags.yaml
```

(continues on next page)

(continued from previous page)

```
file: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ 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 buildspect: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/tutorials/invalid_tags.yml
There were 1 buildspects that failed validation

Similarly we can search buildspects based on tags if you want to validate a group of buildspects using the `-t` option. We can append `-t` option multiple times to search by multiple tag names. In this next example, we will validate all buildspects for **python** and **pass** tags.

```
$ buildtest buildspect validate -t python -t pass
Processing buildspect: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/tutorials/pass_returncode.yml
Processing buildspect: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/tutorials/python-shell.yml
Processing buildspect: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/tutorials/python-hello.yml
All buildspects passed validation!!!
```

Finally we can also search by executors using the `-e` option which can be appended to search by multiple executors.

```
$ buildtest buildspect validate -e generic.local.csh
Processing buildspect: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/tutorials/csh_shell_examples.yml
Processing buildspect: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/tutorials/environment.yml
All buildspects passed validation!!!
```

3.4.3 Query Test Report

buildtest keeps track of all tests and results in a JSON file. This file is read by **buildtest report** command to extract certain fields from JSON file and display them in table format. We use python [tabulate](#) library for pretty print data in tables. Shown below is command usage to query test reports.

```
$ buildtest report --help
usage: buildtest [options] [COMMANDS] report [-h] [--helpformat] [--helpfilter] [--
↳ format FORMAT] [--filter FILTER]

                                [--latest] [--oldest] [-r REPORT]
                                ...
```

(continues on next page)

(continued from previous page)

optional arguments:

```

-h, --help            show this help message and exit
--helpformat          List of available format fields
--helpfilter          List available filter fields to be used with --filter option
--format FORMAT       format field for printing purposes. For more details see --
↳ helpformat for list of available
                        fields. Fields must be separated by comma (usage: --format
↳ <field1>,<field2>,...)
--filter FILTER       Filter report by filter fields. The filter fields must be a
↳ key=value pair and multiple fields
                        can be comma separated in the following format: --filter
↳ key1=val1,key2=val2 . For list of
                        filter fields run: --helpfilter.
--latest              Retrieve latest record of particular test
--oldest              Retrieve oldest record of particular test
-r REPORT, --report REPORT
                        Specify a report file to read

```

subcommands:

Fetch test results from report file and print them in table format

```

clear                delete report file
list                 List all report files

```

You may run `buildtest report` without any option, and `buildtest` will display **all** test results with default format fields. To see a list of all format fields, click [here](#).

```

$ buildtest report
Reading report file: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/var/report.json

+-----+-----+-----+-----+-----+
↳ +-----+-----+-----+-----+-----+
↳ -----+
↳ -----+
| name                | id          | state  | returncode | starttime
↳ | endtime           | runtime    | tags   | buildspec
↳
↳
+=====+=====+=====+=====+=====+
| exit1_fail          | ae3ff33a    | FAIL   | 1          | 2021/07/30 15:08:30
↳ | 2021/07/30 15:08:30 | 0.004566    | tutorials fail | /home/docs/checkouts/
↳ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/pass_returncode.yml
↳
↳
+-----+-----+-----+-----+-----+
↳ +-----+-----+-----+-----+-----+
↳ -----+
↳ -----+
| exit1_fail          | 84c02c94    | FAIL   | 1          | 2021/07/30 15:08:53
↳ | 2021/07/30 15:08:53 | 0.004414    | tutorials fail | /home/docs/checkouts/
↳ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/pass_returncode.yml
↳

```

(continues on next page)

(continued from previous page)

```

+-----+-----+-----+-----+
↳ +-----+-----+-----+-----+
↳ -----+
| exit1_fail          | 93ad2bb3 | FAIL   |          1 | 2021/07/30 15:08:54_
↳ | 2021/07/30 15:08:54 | 0.004397 | tutorials fail | /home/docs/checkouts/
↳ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/pass_returncode.yml _
↳ |
+-----+-----+-----+-----+
↳ +-----+-----+-----+-----+
↳ -----+
| exit1_fail          | e1052582 | FAIL   |          1 | 2021/07/30 15:08:56_
↳ | 2021/07/30 15:08:56 | 0.00506  | tutorials fail | /home/docs/checkouts/
↳ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/pass_returncode.yml _
↳ |
+-----+-----+-----+-----+
↳ +-----+-----+-----+-----+
↳ -----+
| exit1_fail          | 65444282 | FAIL   |          1 | 2021/07/30 15:08:56_
↳ | 2021/07/30 15:08:56 | 0.004771 | tutorials fail | /home/docs/checkouts/
↳ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/pass_returncode.yml _
↳ |
+-----+-----+-----+-----+
↳ +-----+-----+-----+-----+
↳ -----+
| exit1_pass          | f8323253 | PASS   |          1 | 2021/07/30 15:08:30_
↳ | 2021/07/30 15:08:30 | 0.004372 | tutorials pass  | /home/docs/checkouts/
↳ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/pass_returncode.yml _
↳ |
+-----+-----+-----+-----+
↳ +-----+-----+-----+-----+
↳ -----+
| exit1_pass          | 3e06bf49 | PASS   |          1 | 2021/07/30 15:08:53_
↳ | 2021/07/30 15:08:53 | 0.004803 | tutorials pass  | /home/docs/checkouts/
↳ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/pass_returncode.yml _
↳ |
+-----+-----+-----+-----+
↳ +-----+-----+-----+-----+
↳ -----+
| exit1_pass          | dec97567 | PASS   |          1 | 2021/07/30 15:08:54_
↳ | 2021/07/30 15:08:54 | 0.004444 | tutorials pass  | /home/docs/checkouts/
↳ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/pass_returncode.yml _
↳ |
...

```

Format Reports

Available Format Fields

The **buildtest report** command displays a default format fields that can be changed using the `--format` option. The report file (JSON) contains many more fields and we expose some of the fields with the **-format** option. To see a list of available format fields you can run `buildtest report --helpformat`. This option will list all format fields with their description.

```
$ buildtest report --helpformat
Fields      Description
-----
buildspec   Builds spec 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
```

Format Field Usage

The `--format` field are specified in comma separated format (i.e `--format <field1>,<field2>`). In this example we format table by fields `--format id,executor,state,returncode`.

```
$ buildtest report --format name,id,executor,state,returncode
Reading report file: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/var/report.json

+-----+-----+-----+-----+-----+
↳ -+
| name          | id          | executor      | state       |  ↳
↳ returncode |
+=====+=====+=====+=====+=====+
| exit1_fail    | ae3ff33a   | generic.local.sh | FAIL       |  ↳
↳ 1 |
+-----+-----+-----+-----+-----+
↳ -+
| exit1_fail    | 84c02c94   | generic.local.sh | FAIL       |  ↳
↳ 1 |
```

(continues on next page)

(continued from previous page)

↪ -+					
exit1_fail	93ad2bb3	generic.local.sh	FAIL		└
↪ 1					
↪ -+					
exit1_fail	e1052582	generic.local.sh	FAIL		└
↪ 1					
↪ -+					
exit1_fail	65444282	generic.local.sh	FAIL		└
↪ 1					
↪ -+					
exit1_pass	f8323253	generic.local.sh	PASS		└
↪ 1					
↪ -+					
exit1_pass	3e06bf49	generic.local.sh	PASS		└
↪ 1					
↪ -+					
exit1_pass	dec97567	generic.local.sh	PASS		└
↪ 1					
↪ -+					
...					

Filter Reports

The **buildtest report** command will display all tests results, which can be quite long depending on number of tests so therefore we need a mechanism to filter the test results. The **--filter** option can be used to filter out tests in the output based on filter fields. First, lets see the available filter fields by run **buildtest report --helpfilter** which shows a list of filter fields and their description.

\$ 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

The **--filter** option expects arguments in **key=value** format. You can specify multiple filter delimited by comma. buildtest will treat multiple filters as logical **AND** operation. The filter option can be used with **--format** field. Let's see some examples to illustrate the point.

Filter by returncode

If you want to retrieve all tests with a given returncode, we can use the **returncode** property. For instance, let's retrieve all tests with returncode of 2 by setting `--filter returncode=2`.

```
$ buildtest report --filter returncode=2 --format=name,id,returncode
Reading report file: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/var/report.json
```

name	id	returncode
returncode_list_mismatch	a2c8f242	2
returncode_list_mismatch	ed0a8187	2
returncode_list_mismatch	8a13233b	2
returncode_list_mismatch	21d2f5cb	2
returncode_list_mismatch	c13bc1ff	2

Note: buildtest automatically converts returncode to integer when matching returncode, so `--filter returncode="2"` will work too

Filter by test name

If you want to filter by test name, use the **name** attribute in filter option. Let's assume we want to filter all tests by name `exit1_pass`, this can be achieved by setting filter field as follows: `--filter name=exit1_pass`. Shown below is an example using **name** filter field to filter test results.

```
$ buildtest report --filter name=exit1_pass --format=name,id,returncode,state
Reading report file: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/var/report.json
```

name	id	returncode	state
exit1_pass	f8323253	1	PASS
exit1_pass	3e06bf49	1	PASS
exit1_pass	dec97567	1	PASS
exit1_pass	7c74495b	1	PASS
exit1_pass	74ba735f	1	PASS
exit1_pass	68a9fdf0	1	PASS

Filter by buildspec

Likewise, we can filter results by buildspec file using **buildspec** attribute via `--filter buildspec=<file>`. The **buildspec** attribute must resolve to a file path which can be relative or absolute path. buildtest will resolve path (absolute path) and find the appropriate tests that belong to the buildspec file. If file doesn't exist or is not found in cache it will raise an error.

```
$ buildtest report --filter buildspec=tutorials/python-hello.yml --format=name,id,state,
↳ buildspec
Reading report file: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/var/report.json
```

name	id	state	buildspec
python_hello	04a87317	PASS	/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/python-hello.yml
python_hello	de751b51	PASS	/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/python-hello.yml

Filter by test state

If you want to filter results by test state, use the **state** property. This can be useful if you want to know all pass or failed tests. The state property expects value of [PASS|FAIL] since these are the two recorded test states marked by buildtest. We can also pass multiple filter fields for instance if we want to find all **FAIL** tests for executor **generic.local.sh** we can do the following.

```
$ buildtest report --filter state=FAIL,executor=generic.local.sh --format=name,id,state,
↳ executor
Reading report file: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/var/report.json
```

name	id	state	executor
exit1_fail	ae3ff33a	FAIL	generic.local.sh
exit1_fail	84c02c94	FAIL	generic.local.sh
exit1_fail	93ad2bb3	FAIL	generic.local.sh
exit1_fail	e1052582	FAIL	generic.local.sh
exit1_fail	65444282	FAIL	generic.local.sh
returncode_list_mismatch	a2c8f242	FAIL	generic.local.sh

(continues on next page)

(continued from previous page)

returncode_list_mismatch	ed0a8187	FAIL	generic.local.sh
returncode_list_mismatch	8a13233b	FAIL	generic.local.sh
returncode_list_mismatch	21d2f5cb	FAIL	generic.local.sh
returncode_list_mismatch	c13bc1ff	FAIL	generic.local.sh
timelimit_min_fail	d6e452ef	FAIL	generic.local.sh
timelimit_max_fail	9d91d66e	FAIL	generic.local.sh

Filter Exception Cases

The returncode filter field expects an integer value, so if you try a non-integer returncode you will get the following message

```
$ buildtest report --filter returncode=1.5
Traceback (most recent call last):
  File "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/bin/
↳ buildtest", line 17, in <module>
    buildtest.main.main()
  File "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/main.py", line 124, in main
    report_cmd(args)
  File "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/cli/report.py", line 439, in report_cmd
    results = Report(args.filter, args.format, args.latest, args.oldest, args.report)
  File "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/cli/report.py", line 72, in __init__
    self._check_filter_fields()
  File "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/cli/report.py", line 97, in _check_filter_fields
    f"Invalid returncode:{self.filter[key]} must be an integer"
buildtest.exceptions.BuildTestError: 'Invalid returncode:1.5 must be an integer'
```

The state filter field expects value of PASS or FAIL so if you specify an invalid state you will get an error as follows.

```
$ buildtest report --filter state=UNKNOWN
filter argument 'state' must be 'PASS' or 'FAIL' got value UNKNOWN
```

The builds spec field expects a valid file path, it can be an absolute or relative path, buildtest will resolve absolute path and check if file exist and is in the report file. If it's an invalid file we get an error such as

```
$ buildtest report --filter builds spec=/path/to/invalid.yml
Invalid File Path for filter field 'builds spec': /path/to/invalid.yml
```

You may have a valid filepath for builds spec filter field such as \$BUILDTEST_ROOT/tutorials/invalid_executor.yml, but there is no record of a test in the report cache because this test wasn't run. In this case you will get the following message.

```
$ buildtest report --filter buildspec=$BUILDTEST_ROOT/tutorials/invalid_executor.yml
buildspec file: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/tutorials/invalid_executor.yml not found in cache
```

Find Latest or Oldest test

We can search for oldest or latest test for any given test. This can be useful if you want to see first or last test run. If you want to retrieve the oldest test you can use `--oldest` option. buildtest will append tests, therefore last record in dictionary will be latest record, similarly first record is the oldest record.

Let's take a look at this example, we filter by test name `hello_f` which retrieves three entries. Now let's filter by oldest record by specifying `--oldest` option and it will retrieve the first record which is test id **349f3ada**.

```
$ buildtest report --filter name=hello_f --format name,id,starttime
Reading Report File: /Users/siddiq90/.buildtest/report.json

+-----+-----+-----+
| name   | id      | starttime      |
+-----+-----+-----+
| hello_f | 349f3ada | 2021/02/11 18:13:08 |
+-----+-----+-----+
| hello_f | ecd4a3f2 | 2021/02/11 18:13:18 |
+-----+-----+-----+
| hello_f | 5c87978b | 2021/02/11 18:13:33 |
+-----+-----+-----+

$ buildtest report --filter name=hello_f --format name,id,starttime --oldest
Reading Report File: /Users/siddiq90/.buildtest/report.json

+-----+-----+-----+
| name   | id      | starttime      |
+-----+-----+-----+
| hello_f | 349f3ada | 2021/02/11 18:13:08 |
+-----+-----+-----+
```

If you want to retrieve the latest test result you can use `--latest` option which will retrieve the last record, in the same example we will retrieve test id `5c87978b`.

```
$ buildtest report --filter name=hello_f --format name,id,starttime --latest
Reading Report File: /Users/siddiq90/.buildtest/report.json

+-----+-----+-----+
| name   | id      | starttime      |
+-----+-----+-----+
| hello_f | 5c87978b | 2021/02/11 18:13:33 |
+-----+-----+-----+
```

You may combine `--oldest` and `--latest` options in same command, in this case buildtest will retrieve the first and last record of every test.

```
$ buildtest report --format name,id,starttime --oldest --latest | more
Reading Report File: /Users/siddiq90/.buildtest/report.json
```

(continues on next page)

(continued from previous page)

name	id	starttime
variables_bash	750f48bc	2021/02/11 18:13:03
variables_bash	1bdfd403	2021/02/11 18:13:32
ulimit_filelock_unlimited	b7b852e4	2021/02/11 18:13:03
ulimit_filelock_unlimited	56345a43	2021/02/11 18:13:18

Inspect Tests Records

In previous examples we saw how we can retrieve test records using `buildtest report` which is printed in table format. We have limited the output to a limited fields however, if you want to analyze a particular, we have a separate command called `buildtest inspect` that can be used for inspecting a test record based on name or id. Shown below is the command usage for `buildtest inspect` command.

```
$ buildtest inspect --help
usage: buildtest [options] [COMMANDS] inspect [-h] [-r REPORT] ...

optional arguments:
  -h, --help            show this help message and exit
  -r REPORT, --report REPORT
                        Specify a report file to load when inspecting test

subcommands:
  Inspect Test result based on Test ID or Test Name

  name                  Specify name of test
  id                    Specify a Test ID
  list                  List all test ids
```

You can report all test names and corresponding ids using `buildtest inspect list` which will be used for querying tests by name or id.

```
$ buildtest inspect list
+-----+-----+
| name          | id          |
+-----+-----+
| exit1_fail    | ae3ff33a-161a-4ca0-b3a1-eafbd1dedf5e |
+-----+-----+
| exit1_fail    | 84c02c94-9890-4e9f-9e22-393d70ea7436 |
+-----+-----+
| exit1_fail    | 93ad2bb3-b779-47a7-b589-1e10a8c7fdf4 |
+-----+-----+
| exit1_fail    | e1052582-0c75-4bfa-82d0-5d8b55b07f6f |
+-----+-----+
```

(continues on next page)

(continued from previous page)

exit1_fail	65444282-0e03-45d6-a050-0825dcee3d61	
+-----+	+-----+	+-----+
exit1_pass	f8323253-0578-425c-877e-db50756a07de	
+-----+	+-----+	+-----+
exit1_pass	3e06bf49-230a-4b16-9804-cdbeda798ce4	
+-----+	+-----+	+-----+
exit1_pass	dec97567-03b8-4af0-8c76-2c9a4da48e37	
+-----+	+-----+	+-----+
exit1_pass	7c74495b-825b-4300-a702-514ae61bb5d4	
...		

Inspecting Test by Name

The `buildtest inspect name` expects a list of positional argument that correspond to name of test you want to query and `buildtest` will fetch all records for each named test. Let's see an example to illustrate the point. We can see that each test is stored as a JSON format and `buildtest` keeps track of metadata for each test such as *user*, *hostname*, *command*, path to output and error file, content of test, state of test, returncode, etc... In this example, we will retrieve record for test name **circle_area** which will print the raw content of the test in JSON format.

```
$ buildtest inspect name circle_area
{
  "circle_area": [
    {
      "id": "178ef5ce",
      "full_id": "178ef5ce-35b4-40bb-bc40-a619b8b52ac9",
      "description": "Calculate circle of area given a radius",
      "schemafile": "script-v1.0.schema.json",
      "executor": "generic.local.python",
      "compiler": null,
      "hostname": "build-14361642-project-280831-buildtest",
      "user": "docs",
      "testroot": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/1",
      "testpath": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/1/stage/circle_area.sh
↪",
      "stagedir": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/1/stage",
      "command": "sh circle_area_build.sh",
      "outfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/1/circle_area.out",
      "errfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/1/circle_area.err",
      "buildspec_content": "version: \"1.0\"\nbuildspecs:\n  circle_area:\n    executor:
↪generic.local.python\n    type: script\n    shell: python\n    description: \
↪\"Calculate circle of area given a radius\"\n    tags: [tutorials, python]\n    run: |\
↪\n    import math\n    radius = 2\n    area = math.pi * radius * radius\n
↪\n    print(\"Circle Radius \", radius)\n    print(\"Area of circle \", area)\n",
      "test_content": "#!/bin/bash\npython /home/docs/checkouts/readthedocs.org/user_
↪builds/buildtest/checkouts/v0.10.0/var/tests/generic.local.python/python-shell/circle_
↪area/1/stage/circle_area.py",
```

(continues on next page)

(continued from previous page)

```

    "logpath": "/tmp/buildtest_u8hbw81_.log",
    "tags": "tutorials python",
    "starttime": "2021/07/30 15:08:53",
    "endtime": "2021/07/30 15:08:53",
    "runtime": 0.044649,
    "state": "PASS",
    "returncode": 0,
    "output": "Circle Radius 2\nArea of circle 12.566370614359172\n",
    "error": "circle_area_build.sh: 14: circle_area_build.sh: source: not found\n",
    "job": null,
    "build_script": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↪checkouts/v0.10.0/var/tests/generic.local.python/python-shell/circle_area/1/circle_
↪area_build.sh"
  },
  {
    "id": "3475676d",
    "full_id": "3475676d-2ed6-48e0-be38-f53e1ac2d8b7",
    "description": "Calculate circle of area given a radius",
    "schemafilename": "script-v1.0.schema.json",
    "executor": "generic.local.python",
    "compiler": null,
    "hostname": "build-14361642-project-280831-buildtest",
    "user": "docs",
    "testroot": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/2",
    "testpath": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/2/stage/circle_area.sh
↪",
    "stagedir": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/2/stage",
    "command": "sh circle_area_build.sh",
    "outfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/2/circle_area.out",
    "errfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/2/circle_area.err",
    "buildspec_content": "version: \"1.0\"\n\nbuildspecs:\n  circle_area:\n    executor: ↪
↪generic.local.python\n    type: script\n    shell: python\n    description: ↪
↪\"Calculate circle of area given a radius\"\n    tags: [tutorials, python]\n    run: |\n
↪    import math\n    radius = 2\n    area = math.pi * radius * radius\n    ↪
↪print(\"Circle Radius \", radius)\n    print(\"Area of circle \", area)\n",
    "test_content": "#!/bin/bash\npython /home/docs/checkouts/readthedocs.org/user_
↪builds/buildtest/checkouts/v0.10.0/var/tests/generic.local.python/python-shell/circle_
↪area/2/stage/circle_area.py",
    "logpath": "/tmp/buildtest_3p596gw_.log",
    "tags": "tutorials python",
    "starttime": "2021/07/30 15:08:54",
    "endtime": "2021/07/30 15:08:54",
    "runtime": 0.042053,
    "state": "PASS",
    "returncode": 0,
    "output": "Circle Radius 2\nArea of circle 12.566370614359172\n",
    "error": "circle_area_build.sh: 14: circle_area_build.sh: source: not found\n",

```

(continues on next page)

(continued from previous page)

```

    "job": null,
    "build_script": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↪checkouts/v0.10.0/var/tests/generic.local.python/python-shell/circle_area/2/circle_
↪area_build.sh"
  },
  {
    "id": "a1f47ba1",
    "full_id": "a1f47ba1-caa5-4eeb-a1a4-dfed987ef2b5",
    "description": "Calculate circle of area given a radius",
    "schemafile": "script-v1.0.schema.json",
    "executor": "generic.local.python",
    "compiler": null,
    "hostname": "build-14361642-project-280831-buildtest",
    "user": "docs",
    "testroot": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/3",
    "testpath": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/3/stage/circle_area.sh
↪",
    "stagedir": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/3/stage",
    "command": "sh circle_area_build.sh",
    "outfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/3/circle_area.out",
    "errfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/3/circle_area.err",
    "buildspec_content": "version: \"1.0\"\\nbuildspecs:\\n  circle_area:\\n    executor:_
↪generic.local.python\\n    type: script\\n    shell: python\\n    description: \\
↪\"Calculate circle of area given a radius\"\\n    tags: [tutorials, python]\\n    run: |\\
↪n      import math\\n      radius = 2\\n      area = math.pi * radius * radius\\n
↪print(\"Circle Radius \", radius)\\n      print(\"Area of circle \", area)\\n",
    "test_content": "#!/bin/bash\\npython /home/docs/checkouts/readthedocs.org/user_
↪builds/buildtest/checkouts/v0.10.0/var/tests/generic.local.python/python-shell/circle_
↪area/3/stage/circle_area.py",
    "logpath": "/tmp/buildtest_krchtpxi.log",
    "tags": "tutorials python",
    "starttime": "2021/07/30 15:08:55",
    "endtime": "2021/07/30 15:08:56",
    "runtime": 0.04379,
    "state": "PASS",
    "returncode": 0,
    "output": "Circle Radius 2\\nArea of circle 12.566370614359172\\n",
    "error": "circle_area_build.sh: 14: circle_area_build.sh: source: not found\\n",
    "job": null,
    "build_script": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↪checkouts/v0.10.0/var/tests/generic.local.python/python-shell/circle_area/3/circle_
↪area_build.sh"
  },
  {
    "id": "66d02fbd",
    "full_id": "66d02fbd-fd86-4710-95b9-d7c33f252523",
    "description": "Calculate circle of area given a radius",

```

(continues on next page)

(continued from previous page)

```

    "schemafilename": "script-v1.0.schema.json",
    "executor": "generic.local.python",
    "compiler": null,
    "hostname": "build-14361642-project-280831-buildtest",
    "user": "docs",
    "testroot": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↳ v0.10.0/var/tests/generic.local.python/python-shell/circle_area/4",
    "testpath": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↳ v0.10.0/var/tests/generic.local.python/python-shell/circle_area/4/stage/circle_area.sh
↳ ",
    "stagedir": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↳ v0.10.0/var/tests/generic.local.python/python-shell/circle_area/4/stage",
    "command": "sh circle_area_build.sh",
    "outfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↳ v0.10.0/var/tests/generic.local.python/python-shell/circle_area/4/circle_area.out",
    "errfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↳ v0.10.0/var/tests/generic.local.python/python-shell/circle_area/4/circle_area.err",
    "buildspec_content": "version: \"1.0\"\\nbuildspecs:\\n  circle_area:\\n    executor:↳
↳ generic.local.python\\n    type: script\\n    shell: python\\n    description: \\
↳ \"Calculate circle of area given a radius\"\\n    tags: [tutorials, python]\\n    run: |\\
↳ \\n    import math\\n    radius = 2\\n    area = math.pi * radius * radius\\n
↳ \\n    print(\"Circle Radius \", radius)\\n    print(\"Area of circle \", area)\\n\",
    "test_content": "#!/bin/bash\\npython /home/docs/checkouts/readthedocs.org/user_
↳ builds/buildtest/checkouts/v0.10.0/var/tests/generic.local.python/python-shell/circle_
↳ area/4/stage/circle_area.py",
    "logpath": "/tmp/buildtest_krchtpxi.log",
    "tags": "tutorials python",
    "starttime": "2021/07/30 15:08:56",
    "endtime": "2021/07/30 15:08:56",
    "runtime": 0.042957,
    "state": "PASS",
    "returncode": 0,
    "output": "Circle Radius 2\\nArea of circle 12.566370614359172\\n",
    "error": "circle_area_build.sh: 14: circle_area_build.sh: source: not found\\n",
    "job": null,
    "build_script": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/var/tests/generic.local.python/python-shell/circle_area/4/circle_
↳ area_build.sh"
  },
  {
    "id": "020608ed",
    "full_id": "020608ed-c2cd-486f-9ac8-1146edf20c25",
    "description": "Calculate circle of area given a radius",
    "schemafilename": "script-v1.0.schema.json",
    "executor": "generic.local.python",
    "compiler": null,
    "hostname": "build-14361642-project-280831-buildtest",
    "user": "docs",
    "testroot": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↳ v0.10.0/var/tests/generic.local.python/python-shell/circle_area/5",
    "testpath": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↳ v0.10.0/var/tests/generic.local.python/python-shell/circle_area/5/stage/circle_area.sh
↳ ",

```

(continues on next page)

(continued from previous page)

```

    "stagedir": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/5/stage",
    "command": "sh circle_area_build.sh",
    "outfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/5/circle_area.out",
    "errfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.python/python-shell/circle_area/5/circle_area.err",
    "buildspec_content": "version: \"1.0\"\\nbuidspecs:\\n  circle_area:\\n    executor:
↪generic.local.python\\n    type: script\\n    shell: python\\n    description: \\
↪\"Calculate circle of area given a radius\\\"\\n    tags: [tutorials, python]\\n    run: |\\
↪n      import math\\n      radius = 2\\n      area = math.pi * radius * radius\\n
↪print(\"Circle Radius \", radius)\\n      print(\"Area of circle \", area)\\n\",
    "test_content": "#!/bin/bash\\npython /home/docs/checkouts/readthedocs.org/user_
↪builds/buildtest/checkouts/v0.10.0/var/tests/generic.local.python/python-shell/circle_
↪area/5/stage/circle_area.py",
    "logpath": "/tmp/buildtest_krchtpxi.log",
    "tags": "tutorials python",
    "starttime": "2021/07/30 15:08:56",
    "endtime": "2021/07/30 15:08:56",
    "runtime": 0.042338,
    "state": "PASS",
    "returncode": 0,
    "output": "Circle Radius 2\\nArea of circle 12.566370614359172\\n",
    "error": "circle_area_build.sh: 14: circle_area_build.sh: source: not found\\n",
    "job": null,
    "build_script": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↪checkouts/v0.10.0/var/tests/generic.local.python/python-shell/circle_area/5/circle_
↪area_build.sh"
  }
]
}

```

You can query multiple tests as positional arguments in the format: `buildtest inspect name <test1> <test2>`
 In this next example, we will retrieve test records for `bash_shell` and `python_hello`.

```

$ buildtest inspect name bash_shell python_hello
{
  "python_hello": [
    {
      "id": "04a87317",
      "full_id": "04a87317-adf9-419b-8ade-863d58f0fe72",
      "description": "Hello World python",
      "schemafile": "script-v1.0.schema.json",
      "executor": "generic.local.bash",
      "compiler": null,
      "hostname": "build-14361642-project-280831-buildtest",
      "user": "docs",
      "testroot": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.bash/python-hello/python_hello/1",
      "testpath": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.bash/python-hello/python_hello/1/stage/python_hello.sh
↪",
    },
  ]
}

```

(continues on next page)

(continued from previous page)

```

    "stagedir": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.bash/python-hello/python_hello/1/stage",
    "command": "sh python_hello_build.sh",
    "outfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.bash/python-hello/python_hello/1/python_hello.out",
    "errfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.bash/python-hello/python_hello/1/python_hello.err",
    "buildspec_content": "version: \"1.0\"\\nbuildspecs:\\n  python_hello:\\n    type:␣
↪script\\n    description: Hello World python\\n    executor: generic.local.bash\\n    ␣
↪tags: python\\n    run: python hello.py\\n\\n",
    "test_content": "#!/bin/bash \\n# Content of run section\\npython hello.py",
    "logpath": "/tmp/buildtest_u8hbw81_.log",
    "tags": "python",
    "starttime": "2021/07/30 15:08:53",
    "endtime": "2021/07/30 15:08:53",
    "runtime": 0.04198,
    "state": "PASS",
    "returncode": 0,
    "output": "Hello World\\n",
    "error": "python_hello_build.sh: 14: python_hello_build.sh: source: not found\\n",
    "job": null,
    "build_script": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↪checkouts/v0.10.0/var/tests/generic.local.bash/python-hello/python_hello/1/python_
↪hello_build.sh"
  },
  {
    "id": "de751b51",
    "full_id": "de751b51-171e-4ce8-b202-213762071814",
    "description": "Hello World python",
    "schemafilename": "script-v1.0.schema.json",
    "executor": "generic.local.bash",
    "compiler": null,
    "hostname": "build-14361642-project-280831-buildtest",
    "user": "docs",
    "testroot": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.bash/python-hello/python_hello/2",
    "testpath": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.bash/python-hello/python_hello/2/stage/python_hello.sh
↪",
    "stagedir": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.bash/python-hello/python_hello/2/stage",
    "command": "sh python_hello_build.sh",
    "outfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.bash/python-hello/python_hello/2/python_hello.out",
    "errfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↪v0.10.0/var/tests/generic.local.bash/python-hello/python_hello/2/python_hello.err",
    "buildspec_content": "version: \"1.0\"\\nbuildspecs:\\n  python_hello:\\n    type:␣
↪script\\n    description: Hello World python\\n    executor: generic.local.bash\\n    ␣
↪tags: python\\n    run: python hello.py\\n\\n",
    "test_content": "#!/bin/bash \\n# Content of run section\\npython hello.py",
    "logpath": "/tmp/buildtest_5ra6r44a.log",
    "tags": "python",

```

(continues on next page)

(continued from previous page)

```

    "starttime": "2021/07/30 15:08:54",
    "endtime": "2021/07/30 15:08:54",
    "runtime": 0.042309,
    "state": "PASS",
    "returncode": 0,
    "output": "Hello World\n",
    "error": "python_hello_build.sh: 14: python_hello_build.sh: source: not found\n",
    "job": null,
    "build_script": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↪checkouts/v0.10.0/var/tests/generic.local.bash/python-hello/python_hello/2/python_
↪hello_build.sh"
  }
]
}

```

Inspecting Test by ID

The `buildtest inspect id` works similar to `buildtest inspect name` except that it operates on test id. This can be useful if you want to extract a particular test record and not see all test records at once.

You only need to specify a few characters and `buildtest` will resolve full test id if there is a match. The `buildtest inspect id` can operate on single or multiple ids if you want to specify multiple ids in single command you can do `buildtest inspect id <identifier1> <identifier2>`.

Let's see an example where we query a single test record. Notice, that we only specify a few characters **fee** and `buildtest` found a matching record **fee66c67-db4e-4d35-8c6d-28ac5cbbaba0**

```

$ buildtest inspect id fee
Reading Report File: /Users/siddiq90/.buildtest/report.json

{
  "fee66c67-db4e-4d35-8c6d-28ac5cbbaba0": {
    "id": "fee66c67",
    "full_id": "fee66c67-db4e-4d35-8c6d-28ac5cbbaba0",
    "schemafile": "script-v1.0.schema.json",
    "executor": "generic.local.bash",
    "compiler": null,
    "hostname": "DOE-7086392.local",
    "user": "siddiq90",
    "testroot": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.bash/
↪python-hello/python_hello/2",
    "testpath": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.bash/
↪python-hello/python_hello/2/stage/generate.sh",
    "stagedir": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.bash/
↪python-hello/python_hello/2/stage",
    "rundir": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.bash/
↪python-hello/python_hello/2/run",
    "command": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.bash/
↪python-hello/python_hello/2/stage/generate.sh",
    "outfile": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.bash/
↪python-hello/python_hello/2/run/python_hello.out",
    "errfile": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.bash/
↪python-hello/python_hello/2/run/python_hello.err",

```

(continues on next page)

(continued from previous page)

```

    "buildspec_content": "version: \"1.0\"\\nbuidspecs:\\n  python_hello:\\n    type: script\\n    description: Hello World python\\n    executor: generic.local.bash\\n    tags: python\\n    run: python hello.py\\n\\n",
    "test_content": "#!/bin/bash \\nsource /Users/siddiq90/Documents/github/buildtest/var/
executors/generic.local.bash/before_script.sh\\npython hello.py\\nsource /Users/siddiq90/
Documents/github/buildtest/var/executors/generic.local.bash/after_script.sh",
    "tags": "python",
    "starttime": "2021/03/31 11:18:21",
    "endtime": "2021/03/31 11:18:21",
    "runtime": 0.104714,
    "state": "PASS",
    "returncode": 0,
    "output": "Hello World\\n",
    "error": "",
    "job": null
  }
}

```

We can pass multiple IDs to `buildtest inspect id` and `buildtest` will retrieve test record if there is a match. You only need to specify a few characters to ensure we have a unique test ID and `buildtest` will retrieve the record.

```

$ buildtest inspect id 944 a76
Reading Report File: /Users/siddiq90/.buildtest/report.json

{
  "a76799db-f11e-4050-8dcb-8b147092c536": {
    "id": "a76799db",
    "full_id": "a76799db-f11e-4050-8dcb-8b147092c536",
    "schemafile": "script-v1.0.schema.json",
    "executor": "generic.local.bash",
    "compiler": null,
    "hostname": "DOE-7086392.local",
    "user": "siddiq90",
    "testroot": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.
bash/disk_usage/root_disk_usage/0",
    "testpath": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.
bash/disk_usage/root_disk_usage/0/stage/generate.sh",
    "stagedir": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.
bash/disk_usage/root_disk_usage/0/stage",
    "rundir": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.bash/
disk_usage/root_disk_usage/0/run",
    "command": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.bash/
disk_usage/root_disk_usage/0/stage/generate.sh",
    "outfile": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.bash/
disk_usage/root_disk_usage/0/run/root_disk_usage.out",
    "errfile": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.bash/
disk_usage/root_disk_usage/0/run/root_disk_usage.err",
    "buildspec_content": "version: \"1.0\"\\nbuidspecs:\\n  root_disk_usage:\\n    type: script\\n    tags: [filesystem, storage]\\n    description: Check root disk usage and report if it exceeds threshold\\n    env:\\n    threshold: 90\\n    run: |\\n      root_disk_usage=`df -a / | tail -n 1 | awk '{print $5}' | sed 's/[^0-9]*//g'`\\n      # if root exceeds threshold\\n      if [ \"$root_disk_
usage\" -gt \"$threshold\" ]; then\\n        echo \"[WARNING] Root Disk Usage: $root_
disk_usage% exceeded threshold of $threshold%\"\\n        exit 1\\n      fi\\n      echo \"[OK] Root disk is below threshold of $threshold%\"\\n",

```

(continues on next page)

(continued from previous page)

```

    "test_content": "#!/bin/bash \nsource /Users/siddiq90/Documents/github/buildtest/
↪var/executors/generic.local.bash/before_script.sh\nexport threshold=90\nroot_disk_
↪usage=`df -a / | tail -n 1 | awk '{print $5}' | sed 's/[^0-9]*/g'\n# if root_
↪exceeds threshold\nif [ \"$root_disk_usage\" -gt \"$threshold\" ]; then\n echo \
↪\"[WARNING] Root Disk Usage: $root_disk_usage% exceeded threshold of $threshold%\"\n
↪exit 1\nfi\nnecho \"[OK] Root disk is below threshold of $threshold%\"\n\nsource /Users/
↪siddiq90/Documents/github/buildtest/var/executors/generic.local.bash/after_script.sh",
    "tags": "filesystem storage",
    "starttime": "2021/03/31 11:17:50",
    "endtime": "2021/03/31 11:17:50",
    "runtime": 0.114321,
    "state": "PASS",
    "returncode": 0,
    "output": "[OK] Root disk is below threshold of 90%\n",
    "error": "",
    "job": null
},
    "944f6399-b82b-47f9-bb15-8f529dedd4e6": {
        "id": "944f6399",
        "full_id": "944f6399-b82b-47f9-bb15-8f529dedd4e6",
        "schemafile": "script-v1.0.schema.json",
        "executor": "generic.local.python",
        "compiler": null,
        "hostname": "DOE-7086392.local",
        "user": "siddiq90",
        "testroot": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.
↪python/python-shell/circle_area/0",
        "testpath": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.
↪python/python-shell/circle_area/0/stage/generate.sh",
        "stagedir": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.
↪python/python-shell/circle_area/0/stage",
        "rundir": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.
↪python/python-shell/circle_area/0/run",
        "command": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.
↪python/python-shell/circle_area/0/stage/generate.sh",
        "outfile": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.
↪python/python-shell/circle_area/0/run/circle_area.out",
        "errfile": "/Users/siddiq90/Documents/github/buildtest/var/tests/generic.local.
↪python/python-shell/circle_area/0/run/circle_area.err",
        "buildspec_content": "version: \"1.0\"\nbuilder:\n  circle_area:\n    executor:
↪generic.local.python\n    type: script\n    shell: python\n    description: \
↪\"Calculate circle of area given a radius\"\n    tags: [tutorials, python]\n    run: |\
↪\n    import math\n    radius = 2\n    area = math.pi * radius * radius\n
↪print(\"Circle Radius \", radius)\n    print(\"Area of circle \", area)\n",
        "test_content": "#!/bin/bash\nsource /Users/siddiq90/Documents/github/buildtest/var/
↪executors/generic.local.python/before_script.sh\npython /Users/siddiq90/Documents/
↪github/buildtest/var/tests/generic.local.python/python-shell/circle_area/0/stage/
↪circle_area.py\nsource /Users/siddiq90/Documents/github/buildtest/var/executors/
↪generic.local.python/after_script.sh",
        "tags": "tutorials python",
        "starttime": "2021/03/31 11:18:00",
        "endtime": "2021/03/31 11:18:00",

```

(continues on next page)

(continued from previous page)

```

    "runtime": 0.144171,
    "state": "PASS",
    "returncode": 0,
    "output": "Circle Radius  2\nArea of circle  12.566370614359172\n",
    "error": "",
    "job": null
  }
}

```

If you specify an invalid test id using `buildtest inspect id` you will get an error message as follows.

```
$ buildtest inspect id lad
```

```

Unable to find any test records based on id: ['lad'], please run 'buildtest inspect list
↪' to see list of ids.

```

You will see similar message if you specify an invalid test name using `buildtest inspect name` command.

Using Alternate Report File

The `buildtest report` and `buildtest inspect` command will read from the report file tracked by buildtest which is stored in `$BUILDTEST_ROOT/var/report.json`. This single file can become an issue if you are running jobs through CI where you can potentially overwrite same file or if you want separate report files for each set of builds. Luckily we have an option to handle this using the `buildtest build -r /path/to/report` option which can be used to specify an alternate location to report file.

buildtest will write the report file in the desired location, then you can specify the path to report file via `buildtest report -r /path/to/report` and `buildtest inspect -r /path/to/report` to load the report file when reporting tests.

The report file must be valid JSON file that buildtest understands in order to use *buildtest report* and *buildtest inspect* command. Shown below are some examples using the alternate report file using `buildtest report` and `buildtest inspect` command.

```
$ buildtest report -r python.json --format name,id
Reading report file: /Users/siddiq90/Documents/GitHubDesktop/buildtest/docs/python.json
```

```

+-----+-----+
| name      | id      |
+=====+=====+
| circle_area | 6be6c404 |
+-----+-----+
| python_hello | f21ba744 |
+-----+-----+

```

```
$ buildtest inspect -r test.json name variables_bash
Reading Report File: /Users/siddiq90/Documents/GitHubDesktop/buildtest/test.json
```

```

{
  "variables_bash": [
    {
      "id": "cd0511ce",

```

(continues on next page)

(continued from previous page)

```

    "full_id": "cd0511ce-377e-4ed2-95f4-f244e5518732",
    "schemafile": "script-v1.0.schema.json",
    "executor": "generic.local.bash",
    "compiler": null,
    "hostname": "DOE-7086392.local",
    "user": "siddiq90",
    "testroot": "/Users/siddiq90/.buildtest/var/tests/generic.local.bash/vars/
↪variables_bash/1",
    "testpath": "/Users/siddiq90/.buildtest/var/tests/generic.local.bash/vars/
↪variables_bash/1/stage/generate.sh",
    "stagedir": "/Users/siddiq90/.buildtest/var/tests/generic.local.bash/vars/
↪variables_bash/1/stage",
    "rundir": "/Users/siddiq90/.buildtest/var/tests/generic.local.bash/vars/variables_
↪bash/1/run",
    "command": "/Users/siddiq90/.buildtest/var/tests/generic.local.bash/vars/variables_
↪bash/1/stage/generate.sh",
    "outfile": "/Users/siddiq90/.buildtest/var/tests/generic.local.bash/vars/variables_
↪bash/1/run/variables_bash.out",
    "errfile": "/Users/siddiq90/.buildtest/var/tests/generic.local.bash/vars/variables_
↪bash/1/run/variables_bash.err",
    "buildspec_content": "version: \"1.0\"\\nbuildspecs:\\n  variables_bash:\\n    type:
↪script\\n    executor: generic.local.bash\\n    description: Declare shell variables in
↪bash\\n    tags: [tutorials]\\n    vars:\\n      X: 1\\n      Y: 2\\n      literalstring: |\\
↪n      \\\"this is a literal string ':' '\\\"\\n      singlequote: '\\\"singlequote'\\\"\\n
↪doublequote: '\\\"\\\"doublequote\\\"\\\"\\\"\\n      current_user: '\\$(whoami)'\\\"\\n      files_
↪homedir: '\\`find $HOME -type f -maxdepth 1`\\\"\\n      run: |\\n      echo '\\$X+$Y=' '$((
↪$X+$Y))\\n      echo $literalstring\\n      echo $singlequote\\n      echo $doublequote\\n\\
↪n      echo $current_user\\n      echo $files_homedir",
    "test_content": "#!/bin/bash \\nsource /Users/siddiq90/.buildtest/executor/generic.
↪local.bash/before_script.sh\\nX=1\\nY=2\\nliteralstring='\"this is a literal string ':' '\\
↪n\\nsinglequote='singlequote'\\ndoublequote='\"doublequote'\\ncurrent_user=$(whoami)\\
↪nfiles_homedir=`find $HOME -type f -maxdepth 1`\\necho '\\$X+$Y=' '$(($X+$Y))\\necho
↪$literalstring\\necho $singlequote\\necho $doublequote\\n\\necho $current_user\\necho
↪$files_homedir\\nsource /Users/siddiq90/.buildtest/executor/generic.local.bash/after_
↪script.sh",
    "tags": "tutorials",
    "starttime": "2021/04/16 14:29:25",
    "endtime": "2021/04/16 14:29:25",
    "runtime": 0.213196,
    "state": "PASS",
    "returncode": 0,
    "output": "1+2= 3\\nthis is a literal string ':'\\nsinglequote\\ndoublequote\\
↪nsiddiq90\\n/Users/siddiq90/buildtest_e7yxgttm.log /Users/siddiq90/.anyconnect /Users/
↪siddiq90/buildtest_utwighb8w.log /Users/siddiq90/.DS_Store /Users/siddiq90/.serverauth.
↪555 /Users/siddiq90/.CFUserTextEncoding /Users/siddiq90/.wget-hsts /Users/siddiq90/.
↪bashrc /Users/siddiq90/.zshrc /Users/siddiq90/.coverage /Users/siddiq90/.serverauth.
↪87055 /Users/siddiq90/buildtest_r7bck5zh.log /Users/siddiq90/.zsh_history /Users/
↪siddiq90/.lessht /Users/siddiq90/calltracker.py /Users/siddiq90/.git-completion.bash /
↪Users/siddiq90/buildtest_wvjaaztp.log /Users/siddiq90/buildtest.log /Users/siddiq90/
↪darhan.log /Users/siddiq90/ascent.yml /Users/siddiq90/.cshrc /Users/siddiq90/buildtest_
↪nyq22whj.log /Users/siddiq90/github-tokens /Users/siddiq90/buildtest_ozb8b52z.log /
↪Users/siddiq90/.zcompdump /Users/siddiq90/buildtest_nab_ckph.log /Users/siddiq90/.
↪serverauth.543 /Users/siddiq90/.s.PGSQL.15007.lock /Users/siddiq90/.bash_profile (continues on next page)
↪Users/siddiq90/.Xauthority /Users/siddiq90/.python_history /Users/siddiq90/.gitconfig /
↪Users/siddiq90/output.txt /Users/siddiq90/.bash_history /Users/siddiq90/.viminfo\\n",

```

(continued from previous page)

```

    "error": "",
    "job": null
  },
  {
    "id": "e0901505",
    "full_id": "e0901505-a66b-4c91-9b29-d027cb6fabbb6",
    "schemafile": "script-v1.0.schema.json",
    "executor": "generic.local.bash",
    "compiler": null,
    "hostname": "DOE-7086392.local",
    "user": "siddiq90",
    "testroot": "/Users/siddiq90/.buildtest/var/tests/generic.local.bash/vars/
↳ variables_bash/2",
    "testpath": "/Users/siddiq90/.buildtest/var/tests/generic.local.bash/vars/
↳ variables_bash/2/stage/generate.sh",
    "stagedir": "/Users/siddiq90/.buildtest/var/tests/generic.local.bash/vars/
↳ variables_bash/2/stage",
    "rundir": "/Users/siddiq90/.buildtest/var/tests/generic.local.bash/vars/variables_
↳ bash/2/run",
    "command": "/Users/siddiq90/.buildtest/var/tests/generic.local.bash/vars/variables_
↳ bash/2/stage/generate.sh",
    "outfile": "/Users/siddiq90/.buildtest/var/tests/generic.local.bash/vars/variables_
↳ bash/2/run/variables_bash.out",
    "errfile": "/Users/siddiq90/.buildtest/var/tests/generic.local.bash/vars/variables_
↳ bash/2/run/variables_bash.err",
    "buildspec_content": "version: \"1.0\"\\nbuildspecs:\\n  variables_bash:\\n    type:
↳ script\\n    executor: generic.local.bash\\n    description: Declare shell variables in
↳ bash\\n    tags: [tutorials]\\n    vars:\\n      X: 1\\n      Y: 2\\n      literalstring: |\\
↳ \\n      \\\"this is a literal string ':' '\\\"\\n      singlequote: '\\\"singlequote'\\\"\\n
↳ doublequote: '\\\"\\\"doublequote\\\"\\\"\\\"\\n      current_user: '\\$(whoami)'\\\"\\n      files_
↳ homedir: '\\`find $HOME -type f -maxdepth 1`\\\"\\n\\n      run: |\\n      echo '\\$X+$Y=' '\\$((
↳ $X+$Y))\\n      echo $literalstring\\n      echo $singlequote\\n      echo $doublequote\\n\\
↳ \\n      echo $current_user\\n      echo $files_homedir",
    "test_content": "#!/bin/bash \\nsource /Users/siddiq90/.buildtest/executor/generic.
↳ local.bash/before_script.sh\\nX=1\\nY=2\\nliteralstring='\\\"this is a literal string ':' '\\\"\\
↳ \\n\\nsinglequote='singlequote'\\ndoublequote='\\\"doublequote\\\"\\n\\ncurrent_user=$(whoami)\\
↳ \\nfiles_homedir=`find $HOME -type f -maxdepth 1`\\necho '\\$X+$Y=' '\\$((($X+$Y))\\necho
↳ $literalstring\\necho $singlequote\\necho $doublequote\\n\\necho $current_user\\necho
↳ $files_homedir\\nsource /Users/siddiq90/.buildtest/executor/generic.local.bash/after_
↳ script.sh",
    "tags": "tutorials",
    "starttime": "2021/04/16 14:29:58",
    "endtime": "2021/04/16 14:29:58",
    "runtime": 0.075224,
    "state": "PASS",
    "returncode": 0,
    "output": "1+2= 3\\nthis is a literal string ':'\\nsinglequote\\ndoublequote\\
↳ nsiddiq90\\n/Users/siddiq90/buildtest_e7yxgttm.log /Users/siddiq90/.anyconnect /Users/
↳ siddiq90/buildtest_utwigb8w.log /Users/siddiq90/.DS_Store /Users/siddiq90/.serverauth.
↳ 555 /Users/siddiq90/.CFUserTextEncoding /Users/siddiq90/.wget-hsts /Users/siddiq90/.
↳ bashrc /Users/siddiq90/.zshrc /Users/siddiq90/.coverage /Users/siddiq90/.serverauth.
↳ 87055 /Users/siddiq90/buildtest_r7bck5zh.log /Users/siddiq90/.zsh_history /Users/
↳ siddiq90/.lessht /Users/siddiq90/calltracker.py /Users/siddiq90/.git-completion-bash
↳ Users/siddiq90/buildtest_wvjaaztp.log /Users/siddiq90/buildtest.log /Users/siddiq90/
↳ darhan.log /Users/siddiq90/ascent.yml /Users/siddiq90/.cshrc /Users/siddiq90/buildtest_
↳ nyq22whj.log /Users/siddiq90/github-tokens /Users/siddiq90/buildtest_02b8b32z.log /
↳ Users/siddiq90/.zcompdump /Users/siddiq90/buildtest_nab_ckph.log /Users/siddiq90/.
↳ serverauth.543 /Users/siddiq90/.s.PGSQL.15007.lock /Users/siddiq90/.bash_profile /
↳ Users/siddiq90/.Xauthority /Users/siddiq90/.python_history /Users/siddiq90/.gitconfig /

```

(continued from previous page)

```

    "error": "",
    "job": null
  }
]
}

```

3.4.4 Additional Features

Accessing build history

buildtest keeps track of all builds (`buildtest build`) that can be retrieved using `buildtest history` command which can be useful when you want to analyze or troubleshoot past builds. The *buildtest history* command comes with two subcommands `buildtest history list` and `buildtest history query`.

If you want to list all builds you should run **buildtest history list** which will report a table style format of all builds with corresponding build ID to differentiate each build. Shown below is an example output. The build IDs start at **0** and increment as you run **buildtest build** command.

```

$ buildtest history list
+-----+-----+-----+-----+-----+-----+-----+
↪ --+-----+-----+-----+-----+-----+-----+-----+
↪ -----+
↪ -----+
↪ -----+
↪ -----+
|  id | hostname | user | system | date |
↪ | pass_tests | fail_tests | total_tests | pass_rate | fail_rate | command |
↪ |
↪ |
+=====+=====+=====+=====+=====+=====+=====+
|  2 | build-14361642-project-280831-buildtest | docs | generic | 2021/07/30 |
↪ 15:08:30 | 2 | 0 | 2 | 100 | 0 | /
↪ home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/bin/
↪ buildtest build -b tutorials/shebang.yml
↪ |
+-----+-----+-----+-----+-----+-----+-----+
↪ --+-----+-----+-----+-----+-----+-----+
↪ -----+
↪ -----+
|  0 | build-14361642-project-280831-buildtest | docs | generic | 2021/07/30 |
↪ 15:08:30 | 2 | 2 | 4 | 50 | 50 | /
↪ home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/bin/
↪ buildtest build -b tutorials/pass_returncode.yml
↪ |
+-----+-----+-----+-----+-----+-----+-----+
↪ --+-----+-----+-----+-----+-----+-----+
↪ -----+
↪ -----+
|  1 | build-14361642-project-280831-buildtest | docs | generic | 2021/07/30 |
↪ 15:08:30 | 1 | 0 | 1 | 100 | 0 | /
↪ home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/bin/
↪ buildtest build -b /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↪ tutorials/vars.yml |

```

(continues on next page)

(continued from previous page)

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
↪ +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
↪ +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
↪ +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
↪ +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
3 build-14361642-project-280831-buildtest docs generic 2021/07/30 ↪									
↪ 15:08:31 1 0 1 100 0 /									
↪ home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/bin/									
↪ buildtest build -b tutorials/skip_tests.yml ↪									
↪									
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
↪ +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
↪ +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
↪ +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
4 build-14361642-project-280831-buildtest docs generic 2021/07/30 ↪									
↪ 15:08:42 3 2 5 60 40 /									
↪ home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/bin/									
↪ buildtest build -b tutorials/runtime_status_test.yml ↪									
↪									
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
↪ +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
↪ +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
↪ +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									

The `buildtest history query` command is particularly useful when you want to inspect a particular build. This command expects a *Build Identifier* which can be found by inspecting output column *id* in *buildtest history list*.

Shown below is an output of build ID 0 which reports relevant detail for the build such as input command, username, hostname, platform, date, etc...

```
$ buildtest history query 0
{
  "command": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↪ 0/bin/buildtest build -b tutorials/pass_returncode.yml",
  "user": "docs",
  "hostname": "build-14361642-project-280831-buildtest",
  "platform": "Linux",
  "date": "2021/07/30 15:08:30",
  "buildtest": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↪ 10.0/bin/buildtest",
  "python": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↪ python",
  "python-ver": "3.6.12",
  "testdir": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↪ 0/var/tests",
  "configfile": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↪ 10.0/buildtest/settings/config.yml",
  "system": "generic",
  "logpath": "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.
↪ 0/var/.history/0/buildtest_suysjl54.log",
  "invalid_buildspecs": [],
```

(continues on next page)

(continued from previous page)

```

"buildspecs": {
  "detected": [
    "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↪tutorials/pass_returncode.yml"
  ],
  "included": [
    "/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↪tutorials/pass_returncode.yml"
  ],
  "excluded": []
},
"test_summary": {
  "pass": "2",
  "fail": "2",
  "total": "4",
  "pass_rate": "50.000",
  "fail_rate": "50.000"
}
}

```

buildtest schemas

buildtest uses JSON Schema for validating buildspecs and *buildtest configuration file*. You can use `buildtest schema` command to see the list of schemas supported by buildtest. The schema files are denoted by `.schema.json` file extension.

```

$ buildtest schema
global.schema.json
definitions.schema.json
settings.schema.json
compiler-v1.0.schema.json
spack-v1.0.schema.json
script-v1.0.schema.json

```

Shown below is the command usage of `buildtest schema`

```

$ buildtest schema --help
usage: buildtest [options] [COMMANDS] schema [-h] [-n Schema Name] [-e] [-j]

optional arguments:
  -h, --help            show this help message and exit
  -n Schema Name, --name Schema Name
                        show schema by name (e.g., script)
  -e, --example         Show schema examples
  -j, --json            Display json schema file

```

The json schemas are published at <https://buildtesters.github.io/buildtest/> and we provide a command line interface to view schema files and examples. You must use the `--name` option to select a schema, for instance if you want to view the JSON Schema for `script-v1.0.schema.json` you can run the following:

```

buildtest schema --name script-v1.0.schema.json --json

```

Similarly, if you want to view example buildsspecs for a schema use the `--example` option with a schema. For example to view all example schemas for **compiler-v1.0.schema.json** run the following:

```
buildtest schema --name compiler-v1.0.schema.json --example
```

To learn more about schema files and examples click [here](#).

Accessing buildtest documentation

We provide two command line options to access main documentation and schema docs. This will open a browser on your machine.

To access [buildtest docs](#) you can run:

```
buildtest docs
```

To access [schema docs](#) you can run:

```
buildtest schemadocs
```

Color Mode

buildtest will display output in color, if you want to disable color you can set environment variable `BUILDTEST_COLOR` to **False** and buildtest will not display the ANSI codes. This can be useful if you don't want to see ANSI color codes in the text output.

CDASH Integration

The `buildtest cdash` command is responsible for uploading tests to CDASH server. You will need to specify [CDASH Configuration](#) in your configuration file. Shown below is the command usage.

```
$ buildtest cdash --help
usage: buildtest [options] [COMMANDS] cdash [-h] ...

optional arguments:
  -h, --help  show this help message and exit

subcommands:
  buildtest CDASH integration

  view      Open CDASH project in webbrowser
  upload    Upload Test to CDASH server
```

The `buildtest cdash upload` command is responsible for uploading all tests in *report.json* into CDASH. You must specify a buildname when using **buildtest cdash upload** in this example we will specify a buildname called *tutorials*:

```
$ buildtest cdash upload tutorials
Reading configuration file: /Users/siddiq90/Documents/GitHubDesktop/buildtest/buildtest/
↪ settings/config.yml
Reading report file: /Users/siddiq90/.buildtest/report.json
build name: tutorials
```

(continues on next page)

(continued from previous page)

```
site: generic
stamp: 20210428-1512-Experimental
MD5SUM: d7651cb3fbdd19298b0188c441704c3a
PUT STATUS: 200
You can view the results at: https://my.cdash.org//viewTest.php?buildid=2004360
```

We can see the output of these tests in CDASH if we go to url <https://my.cdash.org//viewTest.php?buildid=2004360>

Name	Status	Time	Details	Labels	Summary	description
timelimit_min_max	Passed	2s 220ms		tutorials	Stable	Run a sleep job for 2 seconds and test pass if its within 1.0-4.0sec.
timelimit_min	Passed	2s 60ms		tutorials	Stable	Run a sleep job for 2 seconds and test pass if its exceeds min time of 1.0 sec
timelimit_max	Passed	2s 60ms		tutorials	Stable	Run a sleep job for 2 seconds and test pass if its within max time 5.0 sec.
timelimit_min_fail	Failed	2s 70ms		tutorials	Broken	This test fails because it runs less than minime of 1.0 second
timelimit_max_fail	Failed	2s 90ms		tutorials	Broken	This test fails because it exceeds maxime of 1.0 second

By default buildtest will read the report file in your **\$HOME/.buildtest/report.json**, we can specify an alternate report file. First let's see the available help options for buildtest cdash upload.

```
$ buildtest cdash upload --help
usage: buildtest [options] [COMMANDS] cdash upload [-h] [--site SITE] [-r REPORT]
       buildname

positional arguments:
  buildname              Specify Build Name reported in CDASH

optional arguments:
  -h, --help            show this help message and exit
  --site SITE            Specify site name reported in CDASH
  -r REPORT, --report REPORT
                        Path to report file to upload test results
```

We can pass an alternate report file using **-r** option when uploading tests to CDASH. This can be useful if you want to map test results to different buildnames in CDASH perhaps running a different subset of tests via **buildtest build --tags** and upload the test results with different buildname assuming you have different paths to report file.

Let's say we want to build all python tests using tags and store them in a report file which we want to push to CDASH with buildgroup name python we can do that as follows

```
$ buildtest build --tags python -r $BUILDTEST_ROOT/python.json

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:49
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
              10.0/bin/buildtest
```

(continues on next page)

(continued from previous page)

```

buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳bin/buildtest build --tags python -r /home/docs/checkouts/readthedocs.org/user_builds/
↳buildtest/checkouts/v0.10.0/python.json

```

```

+-----+
| Stage: Discovering Buildsspecs |
+-----+

```

```

+-----+
↳-----+
| Discovered Buildsspecs |
↳

```

```

+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳python-hello.yml |
+-----+

```

```

↳-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳python-shell.yml |
+-----+

```

```

↳-----+
Discovered Buildsspecs: 2
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 2

```

BREAKDOWN OF BUILDSPECS BY TAGS

```

-----
Detected Tag Names: ['python']

```

```

+-----+
↳-----+
| python |
↳

```

```

+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳python-hello.yml |
+-----+

```

```

↳-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳python-shell.yml |
+-----+

```

(continues on next page)

(continued from previous page)

```

+-----+
| Stage: Parsing Buildsspecs |
+-----+

schemafile          | validstate | buildspect
+-----+-----+-----+
↳ -----
script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↳ builds/buildtest/checkouts/v0.10.0/tutorials/python-hello.yml
script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↳ builds/buildtest/checkouts/v0.10.0/tutorials/python-shell.yml

name                description
+-----+-----+
python_hello        Hello World python
circle_area         Calculate circle of area given a radius

+-----+
| Stage: Building Test |
+-----+

name      | id      | type   | executor          | tags          |
↳ testpath
+-----+-----+-----+-----+-----+
↳ -----
python_hello | 29eb4d7b | script | generic.local.bash | python        | /
↳ home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/
↳ generic.local.bash/python-hello/python_hello/0/python_hello_build.sh
circle_area  | 3ca5caef | script | generic.local.python | ['tutorials', 'python'] | /
↳ home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/
↳ generic.local.python/python-shell/circle_area/0/circle_area_build.sh

+-----+
| Stage: Running Test |
+-----+

name      | id      | executor          | status | returncode
+-----+-----+-----+-----+-----+
python_hello | 29eb4d7b | generic.local.bash | PASS   | 0
circle_area  | 3ca5caef | generic.local.python | PASS   | 0

+-----+
| Stage: Test Summary |
+-----+

```

(continues on next page)

(continued from previous page)

```
Passed Tests: 2/2 Percentage: 100.000%  
Failed Tests: 0/2 Percentage: 0.000%
```

```
Writing Logfile to: /tmp/buildtest_it3plduo.log
```

```
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /home/docs/checkouts/  
↳readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log
```

Next we upload the tests using the `-r` option to specify the report file

```
$ buildtest cdash upload -r $BUILDTEST_ROOT/python.json python  
Reading configuration file: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/  
↳checkouts/v0.10.0/buildtest/settings/config.yml  
Reading report file: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/  
↳checkouts/v0.10.0/python.json  
build name: python  
site: generic  
stamp: 20210730-1508-Experimental  
MD5SUM: 979268f2accbe89b18229edfd5d6b2ce  
PUT STATUS: 200  
You can view the results at: https://my.cdash.org//viewTest.php?buildid=2045626
```

The `buildtest cdash view` command can be used to open CDASH project in a web browser using the command line. This feature assumes you have set the CDASH setting in your configuration file.

3.5 Configuring buildtest

3.5.1 Overview

We assume you are familiar with general concepts presented in [getting started](#) and your next step is to configure buildtest to run at your site. This guide will present you the necessary steps to get you started.

When you clone buildtest, we provide a [default configuration](#) that can be used to run on your laptop or workstation that supports Linux or Mac. The buildtest configuration uses a JSON schemafile [settings.schema.json](#). for validating your configuration. We have published the schema guide for settings schema which you can find [here](#).

Which configuration file does buildtest read?

buildtest will read configuration files in the following order:

- Command line `buildtest -c <config>.yaml build`
- User Configuration - `$HOME/.buildtest/config.yaml`
- Default Configuration - `$BUILDTEST_ROOT/buildtest/settings/config.yaml`

Default Configuration

Buildtest comes with a default configuration that can be found at [buildtest/settings/config.yml](#) relative to root of repo. At the start of buildtest execution, buildtest will load the configuration file and validate the configuration with JSON schema `settings.schema.json`. If it fails to validate, buildtest will raise an error.

We recommend you copy the default configuration as a template to configure buildtest for your site. To get started you should copy the file in `$HOME/.buildtest/config.yml`. Please run the following command:

```
$ cp $BUILDTEST_ROOT/buildtest/settings/config.yml $HOME/.buildtest/config.yml
```

Shown below is the default configuration provided by buildtest.

```
$ cat $BUILDTEST_ROOT/buildtest/settings/config.yml
system:
  generic:
    # specify list of hostnames where buildtest can run for given system record
    hostnames: [".*"]

    # system description
    description: Generic System
    # specify module system used at your site (environment-modules, lmod)
    moduletool: N/A
    # boolean to determine if buildsspecs provided in buildtest repo should be loaded in_
    ↪buildspec cache
    load_default_buildspecs: True

  executors:
    # define local executors for running jobs locally
    local:
      bash:
        description: submit jobs on local machine using bash shell
        shell: bash

      sh:
        description: submit jobs on local machine using sh shell
        shell: sh

      csh:
        description: submit jobs on local machine using csh shell
        shell: csh

      zsh:
        description: submit jobs on local machine using zsh shell
        shell: zsh

    python:
      description: submit jobs on local machine using python shell
      shell: python

  # compiler block
  compilers:
    # regular expression to search for compilers based on module pattern. Used with
    ↪'buildtest config compilers find' to generate compiler instance
```

(continues on next page)

(continued from previous page)

```

# find:
# gcc: "^(gcc)"
# intel: "^(intel)"
# cray: "^(craype)"
# pgi: "^(pgi)"
# cuda: "^(cuda)"
# clang: "^(clang)"

# declare compiler instance which can be site-specific. You can let 'buildtest_
↪config compilers find' generate compiler section
compiler:
  gcc:
    builtin_gcc:
      cc: gcc
      fc: gfortran
      cxx: g++

# location of log directory
# logdir: /tmp/

# specify location where buildtest will write tests
# testdir: /tmp

# specify one or more directory where buildtest should load buildsspecs
# buildspect_roots: []

cdash:
  url: https://my.cdash.org/
  project: buildtest
  site: generic
  buildname: tutorials

```

As you can see the layout of configuration starts with keyword `system` which is used to define one or more systems. Your HPC site may contain more than one cluster, so you should define your clusters with meaningful names as this will impact when you reference *executors* in buildsspecs. In this example, we define one cluster called `generic` which is a dummy cluster used for running tutorial examples. The **required** fields in the system scope are the following:

```

"required": ["executors", "moduletool", "load_default_buildspecs", "hostnames", "compilers
↪"]

```

The `hostnames` field is a list of nodes that belong to the cluster where buildtest should be run. Generally, these hosts should be your login nodes in your cluster. buildtest will process **hostnames** field across all system entry using `re.match` until a hostname is found, if none is found we report an error.

In this example we defined two systems *machine*, *machine2* with the following hostnames.

```

system:
  machine1:
    hostnames: ['loca$', '^1DOE']
  machine2:
    hostnames: ['BOB|JOHN']

```

In this example, none of the host entries match with hostname **DOE-7086392.local** so we get an error since buildtest needs to detect a system before proceeding.

```
buildtest.exceptions.BuildTestError: "Based on current system hostname: DOE-7086392.
↳local we cannot find a matching system ['machine1', 'machine2'] based on current
↳hostnames: {'machine1': ['loca$', '^1DOE'], 'machine2': ['BOB|JOHN']}"
```

Let's assume you we have a system named mycluster that should run on nodes login1, login2, and login3. You can specify hostnames as follows.

```
system:
  mycluster:
    hostnames: ["login1", "login2", "login3"]
```

Alternately, you can use regular expression to condense this list

```
system:
  mycluster:
    hostnames: ["login[1-3]"]
```

Configuring Module Tool

You should configure the moduletool property to the module-system installed at your site. Valid options are the following:

```
# environment-modules
moduletool: environment-modules

# for lmod
moduletool: lmod

# specify N/A if you don't have modules
moduletool: N/A
```

buildspec roots

buildtest can discover buildspec using buildspec_roots keyword. This field is a list of directory paths to search for buildspecs. For example we clone the repo <https://github.com/buildtesters/buildtest-cori> at \$HOME/buildtest-cori and assign this to buildspec_roots as follows:

```
buildspec_roots:
  - $HOME/buildtest-cori
```

This field is used with the buildtest buildspec find command. If you rebuild your buildspec cache via --rebuild option, buildtest will search for all buildspecs in directories specified by buildspec_roots property. buildtest will recursively find all .yaml extension and validate each buildspec with appropriate schema.

Load Default Buildspecs

By default buildtest will add the `$BUILDTEST_ROOT/tutorials` and `$BUILDTEST_ROOT/general_tests` to search path when searching for buildsspecs with `buildtest buildspect find` command. This can be configured via `load_default_buildspecs` property which expects a boolean value.

By default we enable this property, however in practice you would want to disable this `load_default_buildspecs: False` if you only care about running your facility tests.

What is an executor?

An executor is responsible for running the test and capture output/error file and return code. An executor can be local executor which runs tests on local machine or batch executor that can be modelled as partition/queue. A batch executor is responsible for **dispatching** job, then **poll** job until its finish, and **gather** job metrics from scheduler.

Executor Declaration

The `executors` is a JSON *object*, that defines one or more executors. The executors are grouped by their type followed by executor name. In this example we define two local executors `bash`, `sh` and one slurm executor called `regular`:

```
system:
  generic:
    executors:
      local:
        bash:
          shell: bash
          description: bash shell
        sh:
          shell: sh
          description: sh shell
      slurm:
        regular:
          queue: regular
```

The **LocalExecutors** are defined in section *local* where each executor must be unique name and they are referenced in buildspect using `executor` field in the following format:

```
executor: <system>.<type>.<name>
```

For instance, if a buildspect wants to reference the LocalExecutor *bash* from the *generic* cluster, you would specify the following in the buildspect:

```
executor: generic.local.bash
```

In our example configuration, we defined a `bash` executor as follows:

```
executors:
  # define local executors for running jobs locally
  local:
    bash:
      description: submit jobs on local machine using bash shell
      shell: bash
```

The local executors requires the `shell` key which takes the pattern `"^(/bin/bash|/bin/sh|/bin/csh|/bin/tcsh|/bin/zsh|sh|bash|csh|tcsh|zsh|python).*"`. Any buildspec that references this executor will submit job using bash shell.

You can pass options to shell which will get passed into each job submission. For instance if you want all bash scripts to run in login shell you can specify `bash --login`:

```
executors:
  local:
    login_bash:
      shell: bash --login
```

Then you can reference this executor as `executor: generic.local.login_bash` and your tests will be submitted via `bash --login /path/to/test.sh`.

Once you define your executors, you can *query the executors* via `buildtest config executors` command.

Configuring test directory

The default location where tests are written is `$BUILDTEST_ROOT/var/tests` where `$BUILDTEST_ROOT` is the root of buildtest repo. You may specify `testdir` in your configuration to instruct where tests can be written. For instance, if you want to write tests in `/tmp` you can set the following:

```
testdir: /tmp
```

Alternately, one can specify test directory via `buildtest build --testdir <path>` which has highest precedence and overrides configuration and default value.

Configuring log path

You can configure where buildtest will write logs using `logdir` property. For example, in example below buildtest will write log files `$HOME/Documents/buildtest/var/logs`. buildtest will resolve variable expansion to get real path on filesystem.

```
# location of log directory
logdir: $HOME/Documents/buildtest/var/logs
```

`logdir` is not required field in configuration, if it's not specified then buildtest will write logs based on `tempfile` library which may vary based on platform (Linux, Mac).

The buildtest logs will start with `buildtest_` followed by random identifier with a `.log` extension.

buildtest will write the same log file in `$BUILDTEST_ROOT/buildtest.log` which can be used to fetch last build log. This can be convenient if you don't remember the directory path to log file.

before_script for executors

Often times, you may want to run a set of commands for a group of tests before running a test. We can do this using this using the `before_script` field which is defined in each executor that is of string type that expects bash commands.

This can be demonstrated with an executor name **local.e4s** responsible for building **E4S Testsuite**

```
local:
  e4s:
    description: "E4S testsuite locally"
    shell: bash
    before_script: |
      cd $SCRATCH
      git clone https://github.com/E4S-Project/testsuite.git
      cd testsuite
      source /global/common/software/spackcp/luke-wyatt-testing/spack/share/spack/setup-
      ↪env.sh
      source setup.sh
```

The *e4s* executor attempts to clone E4S Testsuite in `$SCRATCH` and activate a spack environment and run the initialize script `source setup.sh`. buildtest will write a `before_script.sh` for every executor. This can be found in `var/executors` directory as shown below

```
$ tree var/executors/
var/executors/
|-- local.bash
|   |-- before_script.sh
|-- local.e4s
|   |-- before_script.sh
|-- local.python
|   |-- before_script.sh
|-- local.sh
|   |-- before_script.sh
```

4 directories, 4 files

The `before_script` field is available for all executors and if its not specified the file will be empty. Every test will source these scripts for the appropriate executor.

Cori @ NERSC

Shown below is the configuration file used at Cori.

```
$ wget -q -O - https://raw.githubusercontent.com/buildtesters/buildtest-cori/devel/
↪config.yml 2>&1
system:
  gerty:
    hostnames:
      - gert01.nersc.gov
    load_default_buildspecs: false
    moduletool: environment-modules
    executors:
      local:
```

(continues on next page)

(continued from previous page)

```

bash:
  description: submit jobs on local machine using bash shell
  shell: bash
sh:
  description: submit jobs on local machine using sh shell
  shell: sh
csh:
  description: submit jobs on local machine using csh shell
  shell: csh
python:
  description: submit jobs on local machine using python shell
  shell: python
compilers:
  compiler:
    gcc:
      builtin_gcc:
        cc: /usr/bin/gcc
        cxx: /usr/bin/g++
        fc: /usr/bin/gfortran
cdash:
  url: https://my.cdash.org
  project: buildtest-cori
  site: gerty

perlmutter:
  hostnames:
    - login*
  load_default_buildspecs: false
  moduletool: lmod
  executors:
    defaults:
      pollinterval: 60
      launcher: sbatch
      max_pend_time: 90
  local:
    bash:
      description: submit jobs on local machine using bash shell
      shell: bash
    sh:
      description: submit jobs on local machine using sh shell
      shell: sh
    csh:
      description: submit jobs on local machine using csh shell
      shell: csh
    python:
      description: submit jobs on local machine using python shell
      shell: python
  compilers:
    find:
      gcc: ^(gcc)
    compiler:
      gcc:

```

(continues on next page)

(continued from previous page)

```
builtin_gcc:
  cc: /usr/bin/gcc
  cxx: /usr/bin/g++
  fc: /usr/bin/gfortran

cdash:
  url: https://my.cdash.org
  project: buildtest-cori
  site: perlmutter
cori:
  hostnames:
    - cori*
  load_default_buildspecs: false
  moduletool: environment-modules
  cdash:
    url: https://my.cdash.org
    project: buildtest-cori
    site: cori
  executors:
    defaults:
      pollinterval: 30
      launcher: sbatch
      max_pend_time: 300
    local:
      bash:
        description: submit jobs on local machine using bash shell
        shell: bash
      sh:
        description: submit jobs on local machine using sh shell
        shell: sh
      csh:
        description: submit jobs on local machine using csh shell
        shell: csh
      python:
        description: submit jobs on local machine using python shell
        shell: python
      e4s:
        description: E4S testsuite locally
        shell: bash
        before_script: |
          module load e4s/20.10
          cd $SCRATCH/testsuite
          source setup.sh

    slurm:
      haswell_debug:
        qos: debug
        cluster: cori
        options:
          - -C haswell
        description: debug queue on Haswell partition
      haswell_shared:
```

(continues on next page)

(continued from previous page)

```
qos: shared
cluster: cori
options:
- -C haswell
description: shared queue on Haswell partition
haswell_regular:
qos: normal
cluster: cori
options:
- -C haswell
description: normal queue on Haswell partition
haswell_premium:
qos: premium
cluster: cori
options:
- -C haswell
description: premium queue on Haswell partition
knl_flex:
description: overrun queue on KNL partition
qos: overrun
cluster: cori
options:
- -C knl
bigmem:
description: bigmem jobs
cluster: escori
qos: bigmem
max_pend_time: 300
xfer:
description: xfer qos jobs
qos: xfer
cluster: escori
options:
- -C haswell
compile:
description: compile qos jobs
qos: compile
cluster: escori
options:
- -N 1
knl_debug:
qos: debug
cluster: cori
options:
- -C knl,quad,cache
description: debug queue on KNL partition
knl_regular:
qos: normal
cluster: cori
options:
- -C knl,quad,cache
description: normal queue on KNL partition
```

(continues on next page)

(continued from previous page)

```

knl_premium:
  qos: premium
  cluster: cori
  options:
    - -C knl,quad,cache
  description: premium queue on KNL partition
knl_low:
  qos: low
  cluster: cori
  options:
    - -C knl,quad,cache
  description: low queue on KNL partition
knl_overrun:
  description: overrun queue on KNL partition
  qos: overrun
  cluster: cori
  options:
    - -C knl
    - --time-min=01:00:00
gpu:
  description: submit jobs to GPU partition
  options:
    - -C gpu
  cluster: escori
  max_pend_time: 300
e4s:
  description: E4S runner
  cluster: cori
  max_pend_time: 20000
  options:
    - -q regular
    - -C knl
    - -t 10
    - -n 4
  before_script: |
    module load e4s/20.10
    cd $SCRATCH/testsuite
    source setup.sh

compilers:
  find:
    gcc: ^(gcc|PrgEnv-gnu)
    cray: ^(PrgEnv-cray)
    intel: ^(intel|PrgEnv-intel)
    cuda: ^(cuda/)
    upcxx: ^(upcxx)
  compiler:
    gcc:
      builtin_gcc:
        cc: /usr/bin/gcc
        cxx: /usr/bin/g++
        fc: /usr/bin/gfortran

```

(continues on next page)

(continued from previous page)

```
PrgEnv-gnu/6.0.5:
  cc: gcc
  cxx: g++
  fc: gfortran
  module:
    load:
      - PrgEnv-gnu/6.0.5
    purge: false
PrgEnv-gnu/6.0.7:
  cc: gcc
  cxx: g++
  fc: gfortran
  module:
    load:
      - PrgEnv-gnu/6.0.7
    purge: false
PrgEnv-gnu/6.0.9:
  cc: gcc
  cxx: g++
  fc: gfortran
  module:
    load:
      - PrgEnv-gnu/6.0.9
    purge: false
gcc/6.1.0:
  cc: gcc
  cxx: g++
  fc: gfortran
  module:
    load:
      - gcc/6.1.0
    purge: false
gcc/7.3.0:
  cc: gcc
  cxx: g++
  fc: gfortran
  module:
    load:
      - gcc/7.3.0
    purge: false
gcc/8.1.0:
  cc: gcc
  cxx: g++
  fc: gfortran
  module:
    load:
      - gcc/8.1.0
    purge: false
gcc/8.2.0:
  cc: gcc
  cxx: g++
  fc: gfortran
```

(continues on next page)

(continued from previous page)

```
module:
  load:
    - gcc/8.2.0
  purge: false
gcc/8.3.0:
  cc: gcc
  cxx: g++
  fc: gfortran
  module:
    load:
      - gcc/8.3.0
    purge: false
gcc/9.3.0:
  cc: gcc
  cxx: g++
  fc: gfortran
  module:
    load:
      - gcc/9.3.0
    purge: false
gcc/10.1.0:
  cc: gcc
  cxx: g++
  fc: gfortran
  module:
    load:
      - gcc/10.1.0
    purge: false
gcc/6.3.0:
  cc: gcc
  cxx: g++
  fc: gfortran
  module:
    load:
      - gcc/6.3.0
    purge: false
gcc/8.1.1-openacc-gcc-8-branch-20190215:
  cc: gcc
  cxx: g++
  fc: gfortran
  module:
    load:
      - gcc/8.1.1-openacc-gcc-8-branch-20190215
    purge: false
cray:
  PrgEnv-cray/6.0.5:
    cc: cc
    cxx: CC
    fc: ftn
    module:
      load:
        - PrgEnv-cray/6.0.5
```

(continues on next page)

(continued from previous page)

```
    purge: false
PrgEnv-cray/6.0.7:
  cc: cc
  cxx: CC
  fc: ftn
  module:
    load:
      - PrgEnv-cray/6.0.7
    purge: false
PrgEnv-cray/6.0.9:
  cc: cc
  cxx: CC
  fc: ftn
  module:
    load:
      - PrgEnv-cray/6.0.9
    purge: false
intel:
  PrgEnv-intel/6.0.5:
    cc: icc
    cxx: icpc
    fc: ifort
    module:
      load:
        - PrgEnv-intel/6.0.5
      purge: false
  PrgEnv-intel/6.0.7:
    cc: icc
    cxx: icpc
    fc: ifort
    module:
      load:
        - PrgEnv-intel/6.0.7
      purge: false
  PrgEnv-intel/6.0.9:
    cc: icc
    cxx: icpc
    fc: ifort
    module:
      load:
        - PrgEnv-intel/6.0.9
      purge: false
intel/19.0.3.199:
  cc: icc
  cxx: icpc
  fc: ifort
  module:
    load:
      - intel/19.0.3.199
    purge: false
intel/19.1.2.254:
  cc: icc
```

(continues on next page)

(continued from previous page)

```
cxx: icpc
fc: ifort
module:
  load:
    - intel/19.1.2.254
  purge: false
intel/16.0.3.210:
  cc: icc
  cxx: icpc
  fc: ifort
  module:
    load:
      - intel/16.0.3.210
    purge: false
intel/17.0.1.132:
  cc: icc
  cxx: icpc
  fc: ifort
  module:
    load:
      - intel/17.0.1.132
    purge: false
intel/17.0.2.174:
  cc: icc
  cxx: icpc
  fc: ifort
  module:
    load:
      - intel/17.0.2.174
    purge: false
intel/18.0.1.163:
  cc: icc
  cxx: icpc
  fc: ifort
  module:
    load:
      - intel/18.0.1.163
    purge: false
intel/18.0.3.222:
  cc: icc
  cxx: icpc
  fc: ifort
  module:
    load:
      - intel/18.0.3.222
    purge: false
intel/19.0.0.117:
  cc: icc
  cxx: icpc
  fc: ifort
  module:
    load:
```

(continues on next page)

(continued from previous page)

```

    - intel/19.0.0.117
    purge: false
intel/19.0.8.324:
  cc: icc
  cxx: icpc
  fc: ifort
  module:
    load:
      - intel/19.0.8.324
      purge: false
intel/19.1.0.166:
  cc: icc
  cxx: icpc
  fc: ifort
  module:
    load:
      - intel/19.1.0.166
      purge: false
intel/19.1.1.217:
  cc: icc
  cxx: icpc
  fc: ifort
  module:
    load:
      - intel/19.1.1.217
      purge: false
intel/19.1.2.275:
  cc: icc
  cxx: icpc
  fc: ifort
  module:
    load:
      - intel/19.1.2.275
      purge: false
intel/19.1.3.304:
  cc: icc
  cxx: icpc
  fc: ifort
  module:
    load:
      - intel/19.1.3.304
      purge: false
cuda:
  cuda/9.2.148:
    cc: nvcc
    cxx: nvcc
    fc: None
    module:
      load:
        - cuda/9.2.148
        purge: false
  cuda/10.0.130:

```

(continues on next page)

(continued from previous page)

```
cc: nvcc
cxx: nvcc
fc: None
module:
  load:
    - cuda/10.0.130
  purge: false
cuda/10.1.105:
cc: nvcc
cxx: nvcc
fc: None
module:
  load:
    - cuda/10.1.105
  purge: false
cuda/10.1.168:
cc: nvcc
cxx: nvcc
fc: None
module:
  load:
    - cuda/10.1.168
  purge: false
cuda/10.1.243:
cc: nvcc
cxx: nvcc
fc: None
module:
  load:
    - cuda/10.1.243
  purge: false
cuda/10.2.89:
cc: nvcc
cxx: nvcc
fc: None
module:
  load:
    - cuda/10.2.89
  purge: false
cuda/11.0.2:
cc: nvcc
cxx: nvcc
fc: None
module:
  load:
    - cuda/11.0.2
  purge: false
cuda/shifter:
cc: nvcc
cxx: nvcc
fc: None
module:
```

(continues on next page)

(continued from previous page)

```
    load:
      - cuda/shifter
    purge: false
upcxx:
  upcxx/2019.9.0:
    cc: upcxx
    cxx: upcxx
    fc: None
    module:
      load:
        - upcxx/2019.9.0
      purge: false
  upcxx/2020.3.0:
    cc: upcxx
    cxx: upcxx
    fc: None
    module:
      load:
        - upcxx/2020.3.0
      purge: false
  upcxx/2020.3.2:
    cc: upcxx
    cxx: upcxx
    fc: None
    module:
      load:
        - upcxx/2020.3.2
      purge: false
  upcxx/2020.3.8-snapshot:
    cc: upcxx
    cxx: upcxx
    fc: None
    module:
      load:
        - upcxx/2020.3.8-snapshot
      purge: false
  upcxx/2020.10.0:
    cc: upcxx
    cxx: upcxx
    fc: None
    module:
      load:
        - upcxx/2020.10.0
      purge: false
  upcxx/2020.11.0:
    cc: upcxx
    cxx: upcxx
    fc: None
    module:
      load:
        - upcxx/2020.11.0
      purge: false
```

(continues on next page)

(continued from previous page)

```
upcxx/bleeding-edge:
  cc: upcxx
  cxx: upcxx
  fc: None
  module:
    load:
      - upcxx/bleeding-edge
  purge: false
upcxx/nightly:
  cc: upcxx
  cxx: upcxx
  fc: None
  module:
    load:
      - upcxx/nightly
  purge: false
upcxx-bupc-narrow/2019.9.0:
  cc: upcxx
  cxx: upcxx
  fc: None
  module:
    load:
      - upcxx-bupc-narrow/2019.9.0
  purge: false
upcxx-bupc-narrow/2020.3.0:
  cc: upcxx
  cxx: upcxx
  fc: None
  module:
    load:
      - upcxx-bupc-narrow/2020.3.0
  purge: false
upcxx-bupc-narrow/2020.3.2:
  cc: upcxx
  cxx: upcxx
  fc: None
  module:
    load:
      - upcxx-bupc-narrow/2020.3.2
  purge: false
upcxx-bupc-narrow/2020.3.8-snapshot:
  cc: upcxx
  cxx: upcxx
  fc: None
  module:
    load:
      - upcxx-bupc-narrow/2020.3.8-snapshot
  purge: false
upcxx-bupc-narrow/2020.11.0:
  cc: upcxx
  cxx: upcxx
  fc: None
```

(continues on next page)

(continued from previous page)

```
module:
  load:
    - upcxx-bupc-narrow/2020.11.0
  purge: false
upcxx-bupc-narrow/bleeding-edge:
  cc: upcxx
  cxx: upcxx
  fc: None
module:
  load:
    - upcxx-bupc-narrow/bleeding-edge
  purge: false
upcxx-cs267/2020.10.0:
  cc: upcxx
  cxx: upcxx
  fc: None
module:
  load:
    - upcxx-cs267/2020.10.0
  purge: false
upcxx-extras/2020.3.0:
  cc: upcxx
  cxx: upcxx
  fc: None
module:
  load:
    - upcxx-extras/2020.3.0
  purge: false
upcxx-extras/2020.3.8:
  cc: upcxx
  cxx: upcxx
  fc: None
module:
  load:
    - upcxx-extras/2020.3.8
  purge: false
upcxx-extras/master:
  cc: upcxx
  cxx: upcxx
  fc: None
module:
  load:
    - upcxx-extras/master
  purge: false
```

Default Executor Settings

We can define default configurations for all executors using the `defaults` property.

```
executors:
  defaults:
    pollinterval: 10
    launcher: sbatch
    max_pend_time: 90
    account: nstaff
```

The `launcher` field is applicable for batch executors in this case, `launcher: sbatch` inherits **sbatch** as the job launcher for all slurm executors.

The `account: nstaff` will instruct buildtest to charge all jobs to account `nstaff` from Slurm Executors. The account option can be set in `defaults` field to all executors or defined per executor instance which overrides the default value.

Poll Interval

The `pollinterval` field is used to poll jobs at set interval in seconds when job is active in queue. The poll interval can be configured on command line using `buildtest build --poll-interval` which overrides the configuration value.

Note: `pollinterval`, `launcher` and `max_pend_time` have no effect on local executors.

Max Pend Time

The `max_pend_time` is **maximum** time job can be pending within an executor, if it exceeds the limit buildtest will cancel the job.

The `max_pend_time` option can be overridden per executor level for example the section below overrides the default to 300 seconds:

```
bigmem:
  description: bigmem jobs
  cluster: escori
  qos: bigmem
  max_pend_time: 300
```

The `max_pend_time` is used to cancel job only if job is pending in queue, it has no impact if job is running. buildtest starts a timer at job submission and every poll interval (`pollinterval` field) checks if job has exceeded **max_pend_time** only if job is pending. If job pendtime exceeds `max_pend_time` limit, buildtest will cancel job the job using the appropriate scheduler command like (`scancel`, `bkill`, `qdel`). Buildtest will remove cancelled jobs from poll queue, in addition cancelled jobs won't be reported in test report.

For more details on `max_pend_time` click [here](#).

Specifying QoS (Slurm)

At Cori, jobs are submitted via qos instead of partition so we model a slurm executor named by qos. The qos field instructs which Slurm QOS to use when submitting job. For example we defined a slurm executor named **haswell_debug** which will submit jobs to **debug** qos on the haswell partition as follows:

```
executors:
  slurm:
    haswell_debug:
      qos: debug
      cluster: cori
      options:
        - -C haswell
```

The cluster field specifies which slurm cluster to use (i.e `sbatch --clusters=<string>`). In-order to use bigmem, xfer, or gpu qos at Cori, we need to specify **escori** cluster (i.e `sbatch --clusters=escori`).

buildtest will detect slurm configuration and check qos, partition, cluster match with buildtest configuration. In addition, buildtest supports multi-cluster job submission and monitoring from remote cluster. This means if you specify cluster field buildtest will poll jobs using `sacct` with the cluster name as follows: `sacct -M <cluster>`.

The options field is use to specify any additional options to launcher (`sbatch`) on command line. For instance, slurm. gpu executor, we use the options: `-C gpu` to submit to Cori GPU cluster which requires `sbatch -M escori -C gpu`. Any additional **#SBATCH** options are defined in buildspec for more details see [batch scheduler support](#).

PBS Executors

buildtest supports **PBS** scheduler which can be defined in the `executors` section. Shown below is an example configuration using one pbs executor named `workq`. The property queue: `workq` defines the name of PBS queue that is available in your system.

```
1 system:
2   generic:
3     hostnames: ['.*']
4
5   moduletool: N/A
6   load_default_buildspecs: True
7   executors:
8     defaults:
9       pollinterval: 10
10      launcher: qsub
11      max_pend_time: 30
12    pbs:
13      workq:
14        queue: workq
15    compilers:
16      compiler:
17        gcc:
18          default:
19            cc: /usr/bin/gcc
20            cxx: /usr/bin/g++
21            fc: /usr/bin/gfortran
```

buildtest will detect the PBS queues in your system and determine if queues are active and enabled before submitting job to scheduler. buildtest will run `qstat -Q -f -F json` command to check for queue state which reports in JSON

format and check if queue has the fields `enabled: "True"` or `started: "True"` set in the queue definition. If these values are not set, buildtest will raise an exception.

Shown below is an example with one queue **workq** that is enabled and started.

```
1 $ qstat -Q -f -F json
2 {
3   "timestamp":1615924938,
4   "pbs_version":"19.0.0",
5   "pbs_server":"pbs",
6   "Queue":{
7     "workq":{
8       "queue_type":"Execution",
9       "total_jobs":0,
10      "state_count":"Transit:0 Queued:0 Held:0 Waiting:0 Running:0 Exiting:0_
↪ Begun:0 ",
11      "resources_assigned":{
12        "mem":"0kb",
13        "ncpus":0,
14        "nodect":0
15      },
16      "hasnodes":"True",
17      "enabled":"True",
18      "started":"True"
19    }
20  }
21 }
```

PBS Limitation

Note: Please note that buildtest PBS support relies on job history set because buildtest needs to query job after completion using `qstat -x`. This can be configured using `qmgr` by setting `set server job_history_enable=True`. For more details see section [13.15.5.1 Enabling Job History in PBS 2020.1 Admin Guide](#)

CDASH Configuration

buildtest can be configured to push test to [CDASH](#). The default configuration file provides a CDASH configuration for buildtest project is the following.

```
cdash:
url: https://my.cdash.org/
project: buildtest
site: generic
buildname: tutorials
```

The cdash section can be summarized as follows:

- `url`: URL to CDASH server
- `project`: Project Name in CDASH server
- `site`: Site name that shows up in CDASH entry. This should be name of your system name

- **buildname:** Build Name that shows up in CDASH, this can be any name you want.

The cdash settings can be used with `buildtest cdash` command. For more details see [CDASH Integration](#).

3.5.2 Defining Compilers at your site

buildtest provides a mechanism to declare compilers in your configuration file, this is defined in `compilers` top-level section. The compilers should reflect compilers installed at your site. The compilers are used if you are writing a builds spec with *compiler schema* that needs to reference a particular compiler. The compilers are declared within scope of a system since we assume compilers will vary across different HPC clusters.

Compiler Declaration

Shown below is a declaration of `builtin_gcc` provided by default.

```
compilers:
  compiler:
    gcc:
      builtin_gcc:
        cc: /usr/bin/gcc
        cxx: /usr/bin/g++
        fc: /usr/bin/gfortran
```

The compiler declaration is defined in section `compiler` followed by name of compiler in this case `gcc`. In the `gcc` section one can define all gnu compilers, which includes the name of the compiler in this example we call `builtin_gcc` as system compiler that defines C, C++ and Fortran compilers using `cc`, `cxx` and `fc`.

One can retrieve all compilers using `buildtest config compilers`, there are few options for this command.

```
$ buildtest config compilers --help
usage: buildtest [options] [COMMANDS] config compilers [-h] [-j] [-y] ...

optional arguments:
  -h, --help  show this help message and exit
  -j, --json  List compiler details in JSON format
  -y, --yaml  List compiler details in YAML format

subcommands:
  Find new compilers and add them to detected compiler section

  find      Find compilers
```

buildtest can represent compiler output in JSON, YAML using the `--json` and `--yaml`. Shown below is an example output with these options:

```
$ buildtest config compilers --json
{
  "gcc": {
    "builtin_gcc": {
      "cc": "/usr/bin/gcc",
      "cxx": "/usr/bin/g++",
      "fc": "/usr/bin/gfortran"
```

(continues on next page)

(continued from previous page)

```

    }
  }
}

$ buildtest config compilers --yaml
gcc:
  builtin_gcc:
    cc: /usr/bin/gcc
    cxx: /usr/bin/g++
    fc: /usr/bin/gfortran

$ buildtest config compilers
builtin_gcc

```

Detect Compilers (Experimental Feature)

buildtest can detect compilers based on modulefiles and generate compiler section that way you don't have to specify each compiler manually. This can be done via `buildtest config compilers find` command. Buildtest expects a key/value mapping when searching compiler names and regular expression using `re.match` for discovering compiler modules.

This can be demonstrated, by defining search pattern in the `find` section that expects a dictionary of key/value mapping between compiler names and their module names.

In example, below we define a pattern for gcc modules as `^(gcc)` which will find all modules that start with name `gcc`.

```

compilers:
  find:
    gcc: "^(gcc)"
  compiler:
    gcc:
      builtin:
        cc: /usr/bin/gcc
        cxx: /usr/bin/g++
        fc: /usr/bin/gfortran

```

In this system, we have two gcc modules installed via `spack` package manager, we will attempt to add both modules as compiler instance in buildtest.

```

$ module -t av gcc
/Users/siddiq90/projects/spack/share/spack/lmod/darwin-catalina-x86_64/Core:
gcc/9.3.0-n7p74fd
gcc/10.2.0-37fmsw7

```

Next we run `buildtest config compilers find` which will search all modules based on regular expression and add compilers in their respective group. In this example, buildtest automatically add `gcc/9.2.0-n7p74fd` and `gcc/10.2.0-37fmsw7` modules as compiler instance. Depending on the compiler group, buildtest will apply the compiler wrapper `cc`, `cxx`, `fc` however these can be updated manually. The module section is generated with the module to load. One can further tweak the module behavior along with purging or swap modules.

```

$ buildtest config compilers find
MODULEPATH: /Users/siddiq90/projects/spack/share/spack/lmod/darwin-catalina-x86_64/Core:/
↳usr/local/Cellar/lmod/8.4.12/modulefiles/Darwin:/usr/local/Cellar/lmod/8.4.12/
↳modulefiles/Core

```

(continues on next page)

(continued from previous page)

Configuration File: /Users/siddiq90/.buildtest/config.yml

```

-----
moduletool: lmod
load_default_buildspecs: true
executors:
  local:
    bash:
      description: submit jobs on local machine using bash shell
      shell: bash
    sh:
      description: submit jobs on local machine using sh shell
      shell: sh
    csh:
      description: submit jobs on local machine using csh shell
      shell: csh
    python:
      description: submit jobs on local machine using python shell
      shell: python
compilers:
  find:
    gcc: ^(gcc)
    pgi: ^(pgi)
  compiler:
    gcc:
      builtin_gcc:
        cc: /usr/bin/gcc
        cxx: /usr/bin/g++
        fc: /usr/local/bin/gfortran
      gcc/9.3.0-n7p74fd:
        cc: gcc
        cxx: g++
        fc: gfortran
        module:
          load:
            - gcc/9.3.0-n7p74fd
          purge: false
      gcc/10.2.0-37fmsw7:
        cc: gcc
        cxx: g++
        fc: gfortran
        module:
          load:
            - gcc/10.2.0-37fmsw7
          purge: false

```

```

-----
Updating settings file: /Users/siddiq90/.buildtest/config.yml

```

This feature relies on module system (Lmod, environment-modules) to search modulefiles and one must specify **moduletool** property to indicate how buildtest will search modules. If `moduletool: lmod` is set, buildtest will rely on Lmod spider using `Lmodule` API to detect and test all modules. If `moduletool: environment-modules` is set, buildtest will retrieve modules using output of `module -t av`.

3.5.3 Command Line Interface to buildtest configuration

Once you have implemented your buildtest configuration, you can query the configuration details using `buildtest config` command. Shown below is the command usage.

```
$ buildtest config --help
usage: buildtest [options] [COMMANDS] config [-h] ...

optional arguments:
  -h, --help  show this help message and exit

subcommands:
  Query information from buildtest configuration file

  executors
          Query executors from buildtest configuration
  view      View Buildtest Configuration File
  validate  Validate buildtest settings file with schema.
  summary   Provide summary of buildtest settings.
  systems   List all available systems
  compilers
          search or find compilers
```

Validate buildtest configuration

First thing you should do once you implement your configuration file is to make sure your configuration is valid with the schema. This can be achieved by running `buildtest config validate`. When you invoke this command, buildtest will load the configuration and attempt to validate the file with schema `settings.schema.json`. If validation is successful you will get the following message:

```
$ buildtest config validate
/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest/
↪ settings/config.yml is valid
```

Note: If you defined a user setting (`~/.buildtest/config.yml`) buildtest will validate this file instead of default one.

If there is an error during validation, the output from `jsonschema.exceptions.ValidationError` will be displayed in terminal. For example the error below indicates that `moduletool` property was expecting one of the values `[environment-modules, lmod, N/A]` but it recieved a value of `none`:

```
$ buildtest config validate
Traceback (most recent call last):
  File "/Users/siddiq90/Documents/buildtest/bin/buildtest", line 17, in <module>
    buildtest.main.main()
  File "/Users/siddiq90/Documents/buildtest/buildtest/main.py", line 39, in main
    buildtest_configuration = check_settings(settings_file, retrieve_settings=True)
  File "/Users/siddiq90/Documents/buildtest/buildtest/config.py", line 41, in check_
↪ settings
    validate(instance=user_schema, schema=config_schema)
```

(continues on next page)

(continued from previous page)

```

File "/Users/siddiq90/.local/share/virtualenvs/buildtest-1gHVG2Pd/lib/python3.7/site-
packages/jsonschema/validators.py", line 934, in validate
    raise error
jsonschema.exceptions.ValidationError: 'none' is not one of ['environment-modules', 'lmod', 'N/A']

Failed validating 'enum' in schema['properties']['moduletool']:
    {'description': 'Specify modules tool used for interacting with '
                    '`module` command. ',
      'enum': ['environment-modules', 'lmod', 'N/A'],
      'type': 'string'}

On instance['moduletool']:
    'none'

```

View buildtest configuration

If you want to view buildtest configuration you can run `buildtest config view` which will print content of buildtest configuration.

```

$ buildtest config view
hostnames:
- .*
description: Generic System
moduletool: N/A
load_default_buildspecs: true
executors:
  local:
    bash:
      description: submit jobs on local machine using bash shell
      shell: bash
    sh:
      description: submit jobs on local machine using sh shell
      shell: sh
    csh:
      description: submit jobs on local machine using csh shell
      shell: csh
    zsh:
      description: submit jobs on local machine using zsh shell
      shell: zsh
    python:
      description: submit jobs on local machine using python shell
      shell: python
compilers:
  compiler:
    gcc:
      builtin_gcc:
        cc: gcc
        fc: gfortran
        cxx: g++
cdash:

```

(continues on next page)

(continued from previous page)

```
url: https://my.cdash.org/
project: buildtest
site: generic
buildname: tutorials
```

Settings File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
 ↪10.0/buildtest/settings/config.yml

Note: buildtest config view will display contents of user buildtest settings ~/.buildtest/config.yml if found, otherwise it will display the default configuration

View Executors

You can use the command `buildtest config executors` to view executors from buildtest configuration file. Shown below is the command usage

```
$ buildtest config executors --help
usage: buildtest [options] [COMMANDS] config executors [-h] [-j] [-y]

optional arguments:
  -h, --help  show this help message and exit
  -j, --json  View executor in JSON format
  -y, --yaml  View executors in YAML format
```

You can run `buildtest config executors` without any options and it will report a list of named executors that you would reference in builds spec using the `executor` property. If you prefer json or yaml format you can use `--json` or `--yaml` option.

```
$ buildtest config executors
generic.local.bash
generic.local.sh
generic.local.csh
generic.local.zsh
generic.local.python
```

View Registered Systems

Your buildtest configuration may compose of one or more systems since you can define multiple systems in a single configuration file to run buildtest for different HPC clusters. You can use `buildtest config systems` to report all system details defined in your configuration file. In this example below we should the generic system. If you have multiple entries, you will see one entry per system record.

```
$ buildtest config systems
+-----+-----+-----+-----+
| system | description | hostnames | moduletool |
+=====+=====+=====+=====+
| generic | Generic System | ['.*'] | N/A |
+-----+-----+-----+-----+
```

Configuration Summary

You can get a summary of buildtest using `buildtest config summary`, this will display information from several sources into one single command along.

```
$ buildtest config summary
buildtest version: 0.9.6
buildtest Path: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest

Machine Details
-----
Operating System: darwin
Hostname: DOE-7086392.local
Machine: x86_64
Processor: i386
Python Path /Users/siddiq90/.local/share/virtualenvs/buildtest-KLOcDrW0/bin/python
Python Version: 3.7.3
User: siddiq90

Buildtest Settings
-----
Buildtest Settings: /Users/siddiq90/Documents/GitHubDesktop/buildtest/buildtest/settings/
↳ config.yml
Executors: ['local.bash', 'local.sh', 'local.csh', 'local.zsh', 'local.python']
Buildspec Cache File: /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/buildspecs/
↳ cache.json

Buildtest Schemas
-----
Available Schemas: ['script-v1.0.schema.json', 'compiler-v1.0.schema.json', 'global.
↳ schema.json', 'settings.schema.json']
```

Example Configurations

buildtest provides a few example configurations for configuring buildtest this can be retrieved by running `buildtest schema -n settings.schema.json --examples` or short option `(-e)`, which will validate each example with schema file `settings.schema.json`.

```
$ buildtest schema -n settings.schema.json -e
File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/schemas/examples/settings.schema.json/valid/slurm-example.yml

-----
system:
  generic:
    hostnames: ['.*']

    moduletool: lmod
    load_default_buildspecs: True
    buildspec_roots:
```

(continues on next page)

(continued from previous page)

```

- $HOME/buildtest-cori
testdir: /tmp/buildtest
executors:
  defaults:
    pollinterval: 20
    launcher: sbatch
    max_pend_time: 30
    account: admin
  slurm:
    normal:
      options: ["-C haswell"]
      qos: normal
      before_script: |
        time
        echo "commands run before job"

compilers:
  compiler:
    gcc:
      default:
        cc: /usr/bin/gcc
        cxx: /usr/bin/g++
        fc: /usr/bin/gfortran

```

File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
 ↪ buildtest/schemas/examples/settings.schema.json/valid/cobalt-example.yml

```

system:
  generic:
    hostnames: ['.*']

  moduletool: lmod
  load_default_buildspecs: True
  executors:
    defaults:
      launcher: qsub
      max_pend_time: 30

  cobalt:
    knl:
      queue: knl

    haswell:
      queue: haswell

  compilers:
    compiler:
      gcc:
        default:
          cc: /usr/bin/gcc
          cxx: /usr/bin/g++
          fc: /usr/bin/gfortran

```

File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
 ↪ buildtest/schemas/examples/settings.schema.json/valid/pbs-example.yml

(continues on next page)

(continued from previous page)

```

system:
  generic:
    hostnames: ['.*']

    moduletool: N/A
    load_default_buildspecs: True
    executors:
      defaults:
        pollinterval: 10
        launcher: qsub
        max_pend_time: 30
      pbs:
        workq:
          queue: workq
    compilers:
      compiler:
        gcc:
          default:
            cc: /usr/bin/gcc
            cxx: /usr/bin/g++
            fc: /usr/bin/gfortran
File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↪ buildtest/schemas/examples/settings.schema.json/valid/lsf-example.yml

```

```

system:
  generic:
    hostnames: ['.*']

    moduletool: lmod
    load_default_buildspecs: False
    executors:
      defaults:
        pollinterval: 10
        launcher: bsub
        max_pend_time: 45
      lsf:
        batch:
          description: "LSF Executor name 'batch' that submits jobs to 'batch' queue"
          queue: batch
          account: developer
          options: ["-W 20"]
          before_script: |
            time
            echo "commands run before job"
        test:
          description: "LSF Executor name 'test' that submits jobs to 'test' queue"
          launcher: bsub
          queue: test
          account: qa
          options: ["-W 20"]
    compilers:

```

(continues on next page)

(continued from previous page)

```

compiler:
  gcc:
    default:
      cc: /usr/bin/gcc
      cxx: /usr/bin/g++
      fc: /usr/bin/gfortran
File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/schemas/examples/settings.schema.json/valid/local-executor.yml

```

```

system:
  generic:
    hostnames: ['. *']

    logdir: $BUILDTEST_ROOT/logs
    testdir: $BUILDTEST_ROOT/tests

    moduletool: N/A
    load_default_buildspecs: False
    cdash:
      url: https://my.cdash.org
      project: buildtest
      site: laptop
    processor:
      numcpus: 8
      cores: 4
      threads_per_core: 2
      sockets: 1
      model: "Intel(R) Core(TM) i7-8569U CPU @ 2.80GHz"
    executors:
      local:
        bash:
          description: submit jobs on local machine using bash shell
          shell: bash
          before_script: |
            time
            echo "commands run before job"

        sh:
          description: submit jobs on local machine using sh shell
          shell: sh

        csh:
          description: submit jobs on local machine using csh shell
          shell: csh -x

        tcsh:
          description: submit jobs on local machine using tcsh shell
          shell: /bin/tcsh

        zsh:
          description: submit jobs on local machine using zsh shell
          shell: /bin/zsh

```

(continues on next page)

(continued from previous page)

```

python:
  description: submit jobs on local machine using python shell
  shell: python

compilers:
  find:
    gcc: "^(gcc|GCC|PrgEnv-gnu)"
    intel: "^(intel|Intel|PrgEnv-intel)"
    cray: "^(cray|PrgEnv-cray)"
    clang: "^(clang|Clang)"
    cuda: "^(cuda|CUDA)"
    pgi: "^(pgi|PGI|PrgEnv-pgi)"

  compiler:
    gcc:
      default:
        cc: /usr/bin/gcc
        cxx: /usr/bin/g++
        fc: /usr/bin/gfortran
      gcc@7.2.0:
        cc: 'cc'
        cxx: 'cxx'
        fc: 'fc'
        module:
          load:
            - gcc/7.2.0
    intel:
      intel@2019:
        cc: 'icc'
        cxx: 'icpc'
        fc: 'ifort'
        module:
          purge: True
          load:
            - gcc/7.2.0
            - intel/2019
    cray:
      craype@2.6.2:
        cc: 'cc'
        cxx: 'CC'
        fc: 'fc'
        module:
          load: [craype/2.6.2]
          swap: [PrgEnv-gnu, PrgEnv-cray]

    clang:
      clang@12.0.0:
        cc: 'clang'
        cxx: 'clang++'
        fc: 'None'
        module:

```

(continues on next page)

(continued from previous page)

```
        load: [clang/12.0]
    cuda:
      cuda@11.0:
        cc: 'nvcc'
        cxx: 'nvcc'
        fc: 'None'
        module:
          load: [cuda/11.0]
    pgi:
      pgi@18.0:
        cc: 'pgcc'
        cxx: 'pgc++'
        fc: 'pgfortran'
        module:
          load: [pgi/18.0]
```

File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↪ buildtest/schemas/examples/settings.schema.json/valid/combined_executor.yml

```
system:
  generic:
    hostnames: ['. *']

  moduletool: N/A
  load_default_buildspecs: True
  executors:
    local:
      bash:
        description: submit jobs on local machine
        shell: bash -v

    slurm:
      haswell:
        launcher: sbatch
        options:
          - "-p haswell"
          - "-t 00:10"

    lsf:
      batch:
        launcher: bsub
        queue: batch
        options:
          - "-q batch"
          - "-t 00:10"

    cobalt:
      normal:
        launcher: qsub
        queue: normal
        options:
          - "-n 1"
          - "-t 10"
```

(continues on next page)

(continued from previous page)

```

compilers:
  compiler:
    gcc:
      default:
        cc: /usr/bin/gcc
        cxx: /usr/bin/g++
        fc: /usr/bin/gfortran

```

If you want to retrieve full json schema file for buildtest configuration you can run `buildtest schema -n settings.schema.json --json` or short option `-j`.

3.5.4 Site Examples

Ascent @ OLCF

Ascent is a training system for Summit at OLCF, which is using a IBM Load Sharing Facility (LSF) as their batch scheduler. Ascent has two queues **batch** and **test**. To declare LSF executors we define them under `lsf` section within the `executors` section.

The default launcher is *bsub* which can be defined under `defaults`. The `pollinterval` will poll LSF jobs every 10 seconds using `bjobs`. The `pollinterval` accepts a range between **10 - 300** seconds as defined in schema. In order to avoid polling scheduler excessively pick a number that is best suitable for your site

```

system:
  ascent:
    moduletool: lmod
    load_default_buildspecs: false
    executors:
      defaults:
        launcher: bsub
        pollinterval: 10
        max_pend_time: 60
        account: gen014ecpci
      local:
        bash:
          description: submit jobs on local machine using bash shell
          shell: bash
        sh:
          description: submit jobs on local machine using sh shell
          shell: sh
        csh:
          description: submit jobs on local machine using csh shell
          shell: csh
        python:
          description: submit jobs on local machine using python shell
          shell: python
      lsf:
        batch:
          queue: batch
        test:
          queue: test

```

(continues on next page)

(continued from previous page)

```
compilers:
  find:
    gcc: ^(gcc)
    pgi: ^(pgi)
    cuda: ^(cuda)
  compiler:
    gcc:
      builtin_gcc:
        cc: /usr/bin/gcc
        cxx: /usr/bin/g++
        fc: /usr/bin/gfortran
```

JLSE @ ANL

Joint Laboratory for System Evaluation (JLSE) provides a testbed of emerging HPC systems, the default scheduler is Cobalt, this is defined in the `cobalt` section defined in the executor field.

We set default launcher to `qsub` defined with `launcher: qsub`. This is inherited for all batch executors. In each cobalt executor the queue property will specify the queue name to submit job, for instance the executor `yarrow` with `queue: yarrow` will submit job using `qsub -q yarrow` when using this executor.

```
system:
  jlse:
    hostnames:
      - jlselogin*
    moduletool: environment-modules
    load_default_buildspecs: false
    executors:
      defaults:
        launcher: qsub
        pollinterval: 10
        max_pend_time: 300
      local:
        bash:
          description: submit jobs on local machine using bash shell
          shell: bash
        sh:
          description: submit jobs on local machine using sh shell
          shell: sh
        csh:
          description: submit jobs on local machine using csh shell
          shell: csh
        python:
          description: submit jobs on local machine using python shell
          shell: python
      cobalt:
        yarrow:
          queue: yarrow
        yarrow_debug:
          queue: yarrow_debug
      iris:
```

(continues on next page)

(continued from previous page)

```

queue: iris
iris_debug:
queue: iris_debug

```

3.6 Writing buildspects

3.6.1 Buildspect Overview

What is a buildspect?

In buildtest, we refer to **buildspect** as a YAML file that defines your test that buildtest will parse using the provided schemas and build a shell script from the buildspect file. Every buildspect is validated with a global schema which you can find more if you click [here](#).

Example

Let's start off with a simple example that declares two variables **X** and **Y** and prints the sum of X+Y.

```

version: "1.0"
buildspecs:
  add_numbers:
    type: script
    executor: generic.local.bash
    description: Add X+Y
    tags: [tutorials]
    vars:
      X: 1
      Y: 2
    run: echo "$X+$Y=" $((X+Y))

```

buildtest will validate the entire file with `global.schema.json`, the schema requires **version** and **buildspect** in order to validate file. The **buildspect** is where you define each test. The name of the test is **add_numbers**. The test requires a **type** field which is the sub-schema used to validate the test section. In this example `type: script` informs buildtest to use the *Script Schema* when validating test section.

Each subschema has a list of field attributes that are supported, for example the fields: **type**, **executor**, **vars** and **run** are all valid fields supported by the *script* schema. The **version** field informs which version of subschema to use. Currently all sub-schemas are at version `1.0` where buildtest will validate with a schema `script-v1.0.schema.json`. In future, we can support multiple versions of subschema for backwards compatibility.

Let's look at a more interesting example, shown below is a multi line run example using the *script* schema with test name called **systemd_default_target**, shown below is the content of test:

```

version: "1.0"
buildspecs:
  systemd_default_target:
    executor: generic.local.bash
    type: script
    tags: [system]
    description: check if default target is multi-user.target

```

(continues on next page)

(continued from previous page)

```
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
```

The test name **systemd_default_target** defined in **buildspec** section is validated with the following pattern `^[A-Za-z_][A-Za-z0-9_]*$`. This test will use the executor **generic.local.bash** which means it will use the Local Executor with an executor name *bash* defined in the buildtest settings. The default buildtest settings will provide a bash executor as follows:

```
system:
  generic:
    hostnames: ["localhost"]
    executors:
      local:
        bash:
          description: submit jobs on local machine using bash shell
          shell: bash
```

The `shell: bash` indicates this executor will use *bash* to run the test scripts. To reference this executor use the format `<system>.<type>.<name>` in this case **generic.local.bash** refers to bash executor.

The `description` field is an optional key that can be used to provide a brief summary of the test. In this example we can a full multi-line run section, this is achieved in YAML using `run: |` followed by content of run section tab indented 2 spaces.

In this example we introduce a new field **status** that is used for controlling how buildtest will mark test state. By default, a returncode of **0** is **PASS** and non-zero is a **FAIL**. Currently buildtest reports only two states: **PASS**, **FAIL**. In this example, buildtest will match the actual returncode with one defined in key `returncode` in the status section.

Script Schema

The script schema is used for writing simple scripts (bash, sh, python) in Builds spec. To use this schema you must set `type: script`. The `run` field is responsible for writing the content of test.

Shown below is schema header for `script-v1.0.schema.json`.

```
{
  "$id": "script-v1.0.schema.json",
  "$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,
```

The `"type": "object"` means sub-schema is a JSON `object` where we define a list of key/value pair. The `"required"` field specifies a list of fields that must be specified in order to validate the Builds spec. In this example, `type`, `run`, and `executor` are required fields. The `additionalProperties: false` informs schema to reject any extra properties not defined in the schema. In our previous example, the JSON object is `variables`.

The **executor** key is required for all sub-schemas which instructs buildtest which executor to use when running the test. The executors are defined in *Configuring buildtest*. In our *first example* we define variables using the `vars` property which is a Key/Value pair for variable assignment. The **run** section is required for script schema which defines the content of the test script.

For more details on script schema see schema docs at <https://buildtesters.github.io/buildtest/>

Return Code Matching

buildtest can report PASS/FAIL based on returncode, by default a 0 exit code is PASS and everything else is FAIL. The returncode can be a list of exit codes to match. In this example we have four tests called `exit1_fail`, `exit1_pass`, `returncode_list_mismatch` and `returncode_int_match`. We expect `exit1_fail` and `returncode_mismatch` to FAIL while `exit1_pass` and `returncode_int_match` will PASS.

```
version: "1.0"
buildspecs:

exit1_fail:
  executor: generic.local.sh
  type: script
  description: exit 1 by default is FAIL
  tags: [tutorials, fail]
  run: exit 1

exit1_pass:
  executor: generic.local.sh
  type: script
  description: report exit 1 as PASS
  run: exit 1
  tags: [tutorials, pass]
  status:
    returncode: [1]

returncode_list_mismatch:
  executor: generic.local.sh
  type: script
  description: exit 2 failed since it failed to match returncode 1
  run: exit 2
  tags: [tutorials, fail]
  status:
    returncode: [1, 3]

returncode_int_match:
  executor: generic.local.sh
  type: script
  description: exit 128 matches returncode 128
  run: exit 128
  tags: [tutorials, pass]
  status:
    returncode: 128
```

To demonstrate we will build this test and pay close attention to the **status** column in output.

```
$ buildtest build -b tutorials/pass_returncode.yml

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:30
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳bin/buildtest build -b tutorials/pass_returncode.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
↳-----+
| Discovered Buildsspecs |
↳-----+
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳pass_returncode.yml |
+-----+
↳-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

+-----+
| Stage: Parsing Buildsspecs |
+-----+

  schemafile          | validstate | buildspec
+-----+-----+
↳-----+
  script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↳builds/buildtest/checkouts/v0.10.0/tutorials/pass_returncode.yml

name                  description
+-----+-----+
exit1_fail            exit 1 by default is FAIL
exit1_pass             report exit 1 as PASS
```

(continues on next page)

(continued from previous page)

```

returncode_list_mismatch  exit 2 failed since it failed to match returncode 1
returncode_int_match      exit 128 matches returncode 128

+-----+
| Stage: Building Test |
+-----+

name          | id          | type   | executor          | tags
↳ | testpath
+-----+
↳ +-----+
↳ -----
↳ -----
exit1_fail     | ae3ff33a   | script | generic.local.sh | ['tutorials', 'fail']
↳ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↳ tests/generic.local.sh/pass_returncode/exit1_fail/0/exit1_fail_build.sh
exit1_pass     | f8323253   | script | generic.local.sh | ['tutorials', 'pass']
↳ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↳ tests/generic.local.sh/pass_returncode/exit1_pass/0/exit1_pass_build.sh
returncode_list_mismatch | a2c8f242 | script | generic.local.sh | ['tutorials', 'fail']
↳ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↳ tests/generic.local.sh/pass_returncode/returncode_list_mismatch/0/returncode_list_
↳ mismatch_build.sh
returncode_int_match      | 19b15ae2 | script | generic.local.sh | ['tutorials', 'pass']
↳ | /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/
↳ tests/generic.local.sh/pass_returncode/returncode_int_match/0/returncode_int_match_
↳ build.sh

+-----+
| Stage: Running Test |
+-----+

name          | id          | executor          | status   | returncode
+-----+
exit1_fail     | ae3ff33a   | generic.local.sh | FAIL     | 1
exit1_pass     | f8323253   | generic.local.sh | PASS     | 1
returncode_list_mismatch | a2c8f242 | generic.local.sh | FAIL     | 2
returncode_int_match      | 19b15ae2 | generic.local.sh | PASS     | 128

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 2/4 Percentage: 50.000%
Failed Tests: 2/4 Percentage: 50.000%

Writing Logfile to: /tmp/buildtest_suysjl54.log

```

(continues on next page)

(continued from previous page)

A copy of logfile can be found at \$BUILDTTEST_ROOT/buildtest.log - /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log

The returncode field can be an integer or list of integers but it may not accept duplicate values. If you specify a list of exit codes, buildtest will check actual returncode with list of expected returncodes specified by *returncode* field.

Shown below are examples of invalid returncodes:

```
# empty list is not allowed
returncode: []

# floating point is not accepted in list
returncode: [1, 1.5]

# floating point not accepted
returncode: 1.5

# duplicates are not allowed
returncode: [1, 2, 5, 5]
```

Classifying tests with tags

The tags field can be used to classify tests which can be used to organize tests or if you want to *Building By Tags* (buildtest build --tags <TAGNAME>). Tags can be defined as a string or list of strings. In this example, the string_tag defines a tag name **network** while test list_of_strings_tags define a list of tags named network and ping.

```
version: "1.0"
buildspecs:
  string_tag:
    type: script
    executor: generic.local.bash
    description: tags can be a string
    tags: network
    run: hostname

  list_of_strings_tags:
    type: script
    executor: generic.local.bash
    description: tags can be a list of strings
    tags: [network, ping]
    run: ping -c 4 www.google.com
```

Each item in tags must be a string and no duplicates are allowed, for example in this test, we define a duplicate tag **network** which is not allowed.

```
version: "1.0"
buildspecs:
  duplicate_string_tags:
    type: script
    executor: generic.local.bash
    description: duplicate strings in tags list is not allowed
```

(continues on next page)

(continued from previous page)

```
tags: [network, network]
run: hostname
```

If we run this test and inspect the logs we will see an error message in schema validation:

```
2020-09-29 10:56:43,175 [parser.py:179 - _validate() ] - [INFO] Validating test -
→ 'duplicate_string_tags' with schemafile: script-v1.0.schema.json
2020-09-29 10:56:43,175 [buildspec.py:397 - parse_buildspecs() ] - [ERROR] ['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']
```

If tags is a list, it must contain one item, therefore an empty list (i.e tags: []) is invalid.

Declaring Environment Variables

You can define environment variables using the env property, this is compatible with shells: bash, sh, zsh, csh and tcsh. It does not work with shell: python. In example below we declare three tests using environment variable with default shell (bash), csh, and tcsh

```
version: "1.0"
buildspecs:
  bash_env_variables:
    executor: generic.local.bash
    description: Declare environment variables in default shell (bash)
    type: script
    env:
      FIRST_NAME: avocado
      LAST_NAME: dinosaur
    tags: [tutorials]
    run: |
      hostname
      whoami
      echo $USER
      printf "${FIRST_NAME} ${LAST_NAME}\n"

  csh_env_declaration:
    executor: generic.local.csh
    type: script
    description: "csh shell example to declare environment variables"
    shell: /bin/csh
    tags: [tutorials]
    env:
      SHELL_NAME: "csh"
    run: echo "This is running $SHELL_NAME"

  tcsh_env_declaration:
```

(continues on next page)

(continued from previous page)

```

executor: generic.local.csh
type: script
description: "tssh shell example to declare environment variables"
shell: /bin/tcsh
tags: [tutorials]
env:
  path: "/usr/local/bin:$PATH"
run: echo $path

```

This test can be run by issuing the following command: `buildtest build -b tutorials/environment.yml`. If we inspect one of the test script we will see that buildtest generates a build script that invokes the test using the shell wrapper `/bin/csh` for the csh test and gets the returncode.

```

#!/bin/bash

##### START VARIABLE DECLARATION #####
export BUILDTEST_TEST_NAME=csh_env_declaration
export BUILDTEST_TEST_ROOT=/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/
↳generic.local.csh/environment/csh_env_declaration/0
export BUILDTEST_BUILDSPEC_DIR=/Users/siddiq90/Documents/GitHubDesktop/buildtest/
↳tutorials
export BUILDTEST_STAGE_DIR=/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/
↳generic.local.csh/environment/csh_env_declaration/0/stage
export BUILDTEST_TEST_ID=501ec5d3-e614-4ae8-9c1e-4849ce340c76
##### END VARIABLE DECLARATION #####

# source executor startup script
source /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/executor/generic.local.csh/
↳before_script.sh
# Run generated script
/bin/csh /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.csh/
↳environment/csh_env_declaration/0/stage/csh_env_declaration.csh
# Get return code
returncode=$?
# Exit with return code
exit $returncode

```

This generated test looks something like this

```

#!/bin/csh
# Declare environment variables
setenv SHELL_NAME csh

# Content of run section
echo "This is running $SHELL_NAME"

```

Environment variables are defined using `export` in bash, sh, zsh while csh and tcsh use `setenv`.

Variable Declaration

Variables can be defined using `vars` property, this is compatible with all shells except for `python`. The variables are defined slightly different in `csh`, `tcsh` as pose to `bash`, `sh`, and `zsh`. In example below we define tests with `bash` and `csh`.

In YAML strings can be specified with or without quotes however in `bash`, variables need to be enclosed in quotes `"` if you are defining a multi word string (`name="First Last"`).

If you need define a literal string it is recommended to use the literal block `|` that is a special character in YAML. If you want to specify `"` or `'` in string you can use the escape character `\` followed by any of the special character. In example below we define several variables such as `X`, `Y` that contain numbers, variable `literalstring` is a literal string processed by YAML. The variable `singlequote` and `doublequote` defines a variable with the special character `'` and `"`. The variables `current_user` and `files_homedir` store result of a shell command. This can be done using `var=$(<command >)` or `var=` <command > `` where `<command>` is a Linux command.

Note: You can use the escape character `\` to set special character, for instance you can declare a variable in string with quotes by using `\`.

```
version: "1.0"
buildspecs:
  variables_bash:
    type: script
    executor: generic.local.bash
    description: Declare shell variables in bash
    tags: [tutorials]
    vars:
      X: 1
      Y: 2
      literalstring: |
        "this is a literal string ':' "
      singlequote: "'singlequote'"
      doublequote: "\"doublequote\""
      current_user: "$(whoami)"
      files_homedir: "`find $HOME -type f -maxdepth 1`"

  run: |
    echo "$X+$Y=" $((X+Y))
    echo $literalstring
    echo $singlequote
    echo $doublequote

    echo $current_user
    echo $files_homedir
```

Next we build this test by running `buildtest build -b $BUILDTEST_ROOT/tutorials/vars.yml`.

```
$ buildtest build -b $BUILDTEST_ROOT/tutorials/vars.yml
```

```
User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:30
```

(continues on next page)

(continued from previous page)

```
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳bin/buildtest build -b /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳checkouts/v0.10.0/tutorials/vars.yml
```

```
+-----+
| Stage: Discovering Buildsspecs |
+-----+
```

```
+-----+
↳-----+
| Discovered Buildsspecs |
↳      |
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳vars.yml |
+-----+
```

```
↳-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1
```

```
+-----+
| Stage: Parsing Buildsspecs |
+-----+
```

schemafile	validstate	buildspec
script-v1.0.schema.json	True	/home/docs/checkouts/readthedocs.org/user_
↳builds/buildtest/checkouts/v0.10.0/tutorials/vars.yml		

name	description
variables_bash	Declare shell variables in bash

```
+-----+
| Stage: Building Test |
+-----+
```

name	id	type	executor	tags	testpath
------	----	------	----------	------	----------

(continues on next page)

(continued from previous page)

```

-----+-----+-----+-----+-----+-----+-----+
↪ -----
↪ -----
variables_bash | 4d34c3aa | script | generic.local.bash | ['tutorials'] | /home/docs/
↪ checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/generic.
↪ local.bash/vars/variables_bash/0/variables_bash_build.sh

+-----+
| Stage: Running Test |
+-----+

name          | id          | executor          | status  | returncode
-----+-----+-----+-----+-----+
variables_bash | 4d34c3aa   | generic.local.bash | PASS    |          0

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 1/1 Percentage: 100.000%
Failed Tests: 0/1 Percentage: 0.000%

Writing Logfile to: /tmp/buildtest_dyv8b4j0.log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /home/docs/checkouts/
↪ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log

```

Let's check the generated script from the previous build, you will notice that buildtest will define the shell variables at top of script followed content defined in run section.

```

#!/bin/bash
# Declare shell variables
X=1
Y=2
literalstring="this is a literal string ':' "

singlequote='singlequote'
doublequote="doublequote"
current_user=$(whoami)
files_homedir=`find $HOME -type f -maxdepth 1`

# Content of run section
echo "$X+$Y=" $((X+Y))
echo $literalstring
echo $singlequote
echo $doublequote

```

(continues on next page)

(continued from previous page)

```
echo $current_user
echo $files_homedir
```

Customize Shell

buildtest will default to bash shell when running test, but we can configure shell option using the `shell` field. The shell field is defined in schema as follows:

```
"shell": {
  "type": "string",
  "description": "Specify a shell launcher to use when running jobs. This sets the
↳ shebang line in your test script. The ``shell`` key can be used with ``run`` section
↳ to describe content of script and how its executed",
  "pattern": "^(/bin/bash|/bin/sh|/bin/csh|/bin/tcsh|/bin/
↳ zsh|bash|sh|csh|tcsh|zsh|python).*"
},
```

The shell pattern is a regular expression where one can specify a shell name along with shell options. The shell will configure the `shebang` in the test-script. In this example, we illustrate a few tests using different shell field.

```
version: "1.0"
buildspecs:
  _bin_sh_shell:
    executor: generic.local.sh
    type: script
    description: "/bin/sh shell example"
    shell: /bin/sh
    tags: [tutorials]
    run: "bzip2 --help"

  _bin_bash_shell:
    executor: generic.local.bash
    type: script
    description: "/bin/bash shell example"
    shell: /bin/bash
    tags: [tutorials]
    run: "bzip2 -h"

  bash_shell:
    executor: generic.local.bash
    type: script
    description: "bash shell example"
    shell: bash
    tags: [tutorials]
    run: "echo $SHELL"

  sh_shell:
    executor: generic.local.sh
    type: script
    description: "sh shell example"
    shell: sh
```

(continues on next page)

(continued from previous page)

```

tags: [tutorials]
run: "echo $SHELL"

shell_options:
  executor: generic.local.sh
  type: script
  description: "shell options"
  shell: "sh -x"
  tags: [tutorials]
  run: |
    echo $SHELL
    hostname

```

The generated test-script for buildspec `_bin_sh_shell` will specify shebang `/bin/sh` because we specified `shell: /bin/sh`:

```

#!/bin/sh
# Content of run section
bzip2 --help

```

If you don't specify a shell path such as `shell: sh`, then buildtest will resolve path by looking in `$PATH` and build the shebang line.

In test `shell_options` we specify `shell: "sh -x"`, buildtest will tack on the shell options into the called script as follows:

```

#!/bin/bash

##### START VARIABLE DECLARATION #####
export BUILDTEST_TEST_NAME=shell_options
export BUILDTEST_TEST_ROOT=/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/
↳generic.local.sh/shell_examples/shell_options/0
export BUILDTEST_BUILDSPEC_DIR=/Users/siddiq90/Documents/GitHubDesktop/buildtest/
↳tutorials
export BUILDTEST_STAGE_DIR=/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/
↳generic.local.sh/shell_examples/shell_options/0/stage
export BUILDTEST_TEST_ID=95c11f54-bbb1-4154-849d-44313e4417c2
##### END VARIABLE DECLARATION #####

# source executor startup script
source /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/executor/generic.local.sh/
↳before_script.sh
# Run generated script
sh -x /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.sh/shell_
↳examples/shell_options/0/stage/shell_options.sh
# Get return code
returncode=$?
# Exit with return code
exit $returncode

```

If you prefer `csh` or `tcsh` for writing scripts just set `shell: csh` or `shell: tcsh`, note you will need to match this with appropriate executor. For now use `executor: generic.local.csh` to run your `csh/tcsh` scripts. In this

example below we define a script using csh, take note of `run` section we can write csh style.

```
version: "1.0"
buildspecs:
  csh_shell:
    executor: generic.local.csh
    type: script
    description: "csh shell example"
    shell: csh
    tags: [tutorials]
    vars:
      file: "/etc/csh.cshrc"
    run: |
      if (-e $file) then
        echo "$file file found"
      else
        echo "$file file not found"
        exit 1
      endif
```

Customize Shebang

You may customize the shebang line in testscript using `shebang` field. This takes precedence over the `shell` property which automatically detects the shebang based on shell path.

In next example we have two tests `bash_login_shebang` and `bash_nonlogin_shebang` which tests if shell is Login or Non-Login. The `#!/bin/bash -l` indicates we want to run in login shell and expects an output of `Login Shell` while test `bash_nonlogin_shebang` should run in default behavior which is non-login shell and expects output `Not Login Shell`. We match this with regular expression with stdout stream.

```
version: "1.0"
buildspecs:
  bash_login_shebang:
    type: script
    executor: generic.local.bash
    shebang: "#!/bin/bash -l"
    description: customize shebang line with bash login shell
    tags: tutorials
    run: shopt -q login_shell && echo 'Login Shell' || echo 'Not Login Shell'
    status:
      regex:
        exp: "^Login Shell$"
        stream: stdout

  bash_nonlogin_shebang:
    type: script
    executor: generic.local.bash
    shebang: "#!/bin/bash"
    description: customize shebang line with default bash (nonlogin) shell
    tags: tutorials
    run: shopt -q login_shell && echo 'Login Shell' || echo 'Not Login Shell'
    status:
      regex:
```

(continues on next page)

(continued from previous page)

```
exp: "^Not Login Shell$"
stream: stdout
```

Now let's run this test as we see the following.

```
$ buildtest build -b tutorials/shebang.yml

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:30
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳ python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ bin/buildtest build -b tutorials/shebang.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
↳ -----+
| Discovered Buildsspecs |
↳ |
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳ shebang.yml |
+-----+
↳ -----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

+-----+
| Stage: Parsing Buildsspecs |
+-----+

schemafile          | validstate | buildspec
+-----+
↳ -----+
script-v1.0.schema.json | True      | /home/docs/checkouts/readthedocs.org/user_
↳ builds/buildtest/checkouts/v0.10.0/tutorials/shebang.yml
```

(continues on next page)

(continued from previous page)

```

name                description
-----
bash_login_shebang  customize shebang line with bash login shell
bash_nonlogin_shebang customize shebang line with default bash (nonlogin) shell

+-----+
| Stage: Building Test |
+-----+

name                | id          | type   | executor          | tags          | testpath
-----
bash_login_shebang  | 5240851d   | script | generic.local.bash | tutorials    | /home/docs/
↳checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/generic.
↳local.bash/shebang/bash_login_shebang/0/bash_login_shebang_build.sh
bash_nonlogin_shebang | 33501085   | script | generic.local.bash | tutorials    | /home/docs/
↳checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/generic.
↳local.bash/shebang/bash_nonlogin_shebang/0/bash_nonlogin_shebang_build.sh

+-----+
| Stage: Running Test |
+-----+

name                | id          | executor          | status  | returncode
-----
bash_login_shebang  | 5240851d   | generic.local.bash | PASS    | 0
bash_nonlogin_shebang | 33501085   | generic.local.bash | PASS    | 0

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 2/2 Percentage: 100.000%
Failed Tests: 0/2 Percentage: 0.000%

Writing Logfile to: /tmp/buildtest_bzdf4whc.log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /home/docs/checkouts/
↳readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log

```

If we look at the generated test for **bash_login_shebang** we see the shebang line is passed into the script:

```

#!/bin/bash -l
# Content of run section
shopt -q login_shell && echo 'Login Shell' || echo 'Not Login Shell'

```

Python Shell

You can use **script** schema to write python scripts using the **run** property. This can be achieved if you use the `generic.local.python` executor assuming you have this defined in your buildtest configuration.

Here is a python example calculating area of circle

```
version: "1.0"
buildspecs:
  circle_area:
    executor: generic.local.python
    type: script
    shell: python
    description: "Calculate circle of area given a radius"
    tags: [tutorials, python]
    run: |
      import math
      radius = 2
      area = math.pi * radius * radius
      print("Circle Radius ", radius)
      print("Area of circle ", area)
```

The `shell: python` will let us write python script in the `run` section. The `tags` field can be used to classify test, the field expects an array of string items.

Note: Python scripts are very picky when it comes to formatting, in the `run` section if you are defining multiline python script you must remember to use 2 space indent to register multiline string. buildtest will extract the content from run section and inject in your test script. To ensure proper formatting for a more complex python script you may be better off writing a python script in separate file and call it in run section.

Skipping test

By default, buildtest will run all tests defined in `buildspecs` section, if you want to skip a test use the `skip` field which expects a boolean value. Shown below is an example test.

```
version: "1.0"
buildspecs:
  skip:
    type: script
    executor: generic.local.bash
    description: This test is skipped
    skip: Yes
    tags: [tutorials]
    run: hostname

  unskipped:
    type: script
    executor: generic.local.bash
    description: This test is not skipped
    skip: No
    tags: [tutorials]
    run: hostname
```

The first test **skip** will be ignored by buildtest because `skip: true` is defined while **unskipped** will be processed as usual.

Note: YAML and JSON have different representation for boolean. For json schema valid values are `true` and `false` see <https://json-schema.org/understanding-json-schema/reference/boolean.html> however YAML has many more representation for boolean see <https://yaml.org/type/bool.html>. You may use any of the YAML boolean, however it's best to stick with json schema values `true` and `false`.

Here is an example build, notice message `[skip] test is skipped` during the build stage

```
$ buildtest build -b tutorials/skip_tests.yml

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:31
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳bin/buildtest build -b tutorials/skip_tests.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
↳-----+
| Discovered Buildsspecs |
↳      |
+-----+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳skip_tests.yml |
+-----+
↳-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1
[skip](/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳tutorials/skip_tests.yml): test is skipped.

+-----+
| Stage: Parsing Buildsspecs |
+-----+
```

(continues on next page)

(continued from previous page)

```

schemafile          | validstate  | buildspect
-----+-----+-----
↪ -----
script-v1.0.schema.json | True          | /home/docs/checkouts/readthedocs.org/user_
↪ builds/buildtest/checkouts/v0.10.0/tutorials/skip_tests.yml

name      description
-----
unskipped This test is not skipped

+-----+
| Stage: Building Test |
+-----+

name      | id      | type  | executor          | tags          | testpath
-----+-----+-----+-----+-----+-----
↪ -----
↪ -----
unskipped | 0ec27f02 | script | generic.local.bash | ['tutorials'] | /home/docs/
↪ checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/generic.
↪ local.bash/skip_tests/unskipped/0/unskipped_build.sh

+-----+
| Stage: Running Test |
+-----+

name      | id      | executor          | status  | returncode
-----+-----+-----+-----+-----
unskipped | 0ec27f02 | generic.local.bash | PASS    | 0

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 1/1 Percentage: 100.000%
Failed Tests: 0/1 Percentage: 0.000%

Writing Logfile to: /tmp/buildtest_24fo3gnu.log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /home/docs/checkouts/
↪ readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log

```

run_only

The `run_only` property is used for running test given a specific condition has met. For example, you may want a test to run only if its particular system (Linux, Darwin), operating system, scheduler, etc...

run_only - user

buildtest will skip test if any of the conditions are not met. Let's take an example in this builds spec we define a test name **run_only_as_root** that requires **root** user to run test. The **run_only** is a property of key/value pairs and **user** is one of the field. buildtest will only build & run test if current user matches **user** field. We detect current user using `$USER` and match with input field **user**. buildtest will skip test if there is no match.

```
version: "1.0"
buildspecs:
  run_only_as_root:
    description: "This test will only run if current user is root"
    executor: generic.local.bash
    type: script
    tags: ["tutorials"]
    run_only:
      user: root
    run: echo $USER
```

Now if we run this test we see buildtest will skip test **run_only_as_root** because current user is not root.

```
$ buildtest build -b tutorials/root_user.yml

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:31
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳ python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ bin/buildtest build -b tutorials/root_user.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
↳ -----+
| Discovered Buildsspecs          |
↳ |                               |
```

(continues on next page)

(continued from previous page)

```

=====
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳root_user.yml |
+-----+
↳-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1
[run_only_as_root][/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↳v0.10.0/tutorials/root_user.yml]: test is skipped because this test is expected to run_
↳as user: root but detected user: None.
No buildsspecs to process because there are no valid buildsspecs

```

run_only - platform

Similarly, we can run test if it matches target platform. In this example we have two tests **run_only_platform_darwin** and **run_only_platform_linux** that are run if target platform is Darwin or Linux. This is configured using `platform` field which is a property of `run_only` object. buildtest will match target platform using `platform.system()` with field **platform**, if there is no match buildtest will skip test. In this test, we define a python shell using `shell: python` and `run platform.system()`. We expect the output of each test to have **Darwin** and **Linux** which we match with `stdout` using regular expression.

```

version: "1.0"
buildspecs:
  run_only_platform_darwin:
    description: "This test will only run if target platform is Darwin"
    executor: generic.local.python
    type: script
    tags: ["tutorials"]
    run_only:
      platform: Darwin
    shell: python
    run: |
      import platform
      print(platform.system())
    status:
      regex:
        stream: stdout
        exp: "^Darwin$"

  run_only_platform_linux:
    description: "This test will only run if target platform is Linux"
    executor: generic.local.python
    type: script
    tags: ["tutorials"]
    run_only:
      platform: Linux
    shell: python
    run: |
      import platform
      print(platform.system())

```

(continues on next page)

(continued from previous page)

```
status:
  regex:
    stream: stdout
    exp: "^Linux"
```

This test was ran on a MacOS (Darwin) so we expect test **run_only_platform_linux** to be skipped.

```
$ buildtest build -b tutorials/run_only_platform.yml

User: siddiq90
Hostname: DOE-7086392.local
Platform: Darwin
Current Time: 2021/07/06 18:54:27
buildtest path: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest
buildtest version: 0.9.6
python path: /Users/siddiq90/.local/share/virtualenvs/buildtest-KLOcDrW0/bin/python
python version: 3.7.3
Test Directory: /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests
Configuration File: /Users/siddiq90/Documents/GitHubDesktop/buildtest/buildtest/
↳ settings/config.yml
Command: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest build -b_
↳ tutorials/run_only_platform.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
| Discovered Buildsspecs |
+=====+
| /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/run_only_platform.yml |
+-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1
[run_only_platform_linux][Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/
↳ run_only_platform.yml]: test is skipped because this test is expected to run on_
↳ platform: Linux but detected platform: Darwin.

+-----+
| Stage: Parsing Buildsspecs |
+-----+

  schemafile          | validstate | buildspect
-----+-----+-----
↳ script-v1.0.schema.json | True      | /Users/siddiq90/Documents/GitHubDesktop/
↳ buildtest/tutorials/run_only_platform.yml
```

(continues on next page)

(continued from previous page)

```

name                description
-----
run_only_platform_darwin  This test will only run if target platform is Darwin

+-----+
| Stage: Building Test |
+-----+

name                | id          | type   | executor                | tags          |
--testpath
-----+-----+-----+-----+-----+
↳ -----
↳ -----
run_only_platform_darwin | 964e3016 | script | generic.local.python | ['tutorials'] | /
↳ Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.python/run_
↳ only_platform/run_only_platform_darwin/3/run_only_platform_darwin_build.sh

+-----+
| Stage: Running Test |
+-----+

name                | id          | executor                | status  | returncode
-----+-----+-----+-----+-----
run_only_platform_darwin | 964e3016 | generic.local.python | PASS    | 0

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 1/1 Percentage: 100.000%
Failed Tests: 0/1 Percentage: 0.000%

Writing Logfile to: /var/folders/1m/_jjv09h17k37mkktwnmbkmj0002t_q/T/buildtest__md43sa1.
↳ log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /Users/siddiq90/
↳ Documents/GitHubDesktop/buildtest/buildtest.log

```

run_only - scheduler

buildtest can run test if a particular scheduler is available. In this example, we introduce a new field `scheduler` that is part of `run_only` property. This field expects one of the following values: `[lsf, slurm, cobalt, pbs]` and buildtest will check if target system supports detects the scheduler. In this example we require `lsf` scheduler because this test runs `bmgroup` which is a LSF binary.

Note: buildtest assumes scheduler binaries are available in `$PATH`, if no scheduler is found buildtest sets this to an empty list

```
version: "1.0"
buildspecs:
  show_host_groups:
    type: script
    executor: generic.local.bash
    description: Show information about host groups using bmgroup
    tags: lsf
    run_only:
      scheduler: lsf
    run: bmgroup
```

If we build this test on a target system without LSF notice that buildtest skips test `show_host_groups`.

```
$ buildtest build -b general_tests/sched/lsf/bmggroups.yml

User: docs
Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:31
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳ python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ bin/buildtest build -b general_tests/sched/lsf/bmggroups.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
↳ -----+
| Discovered Buildsspecs |
↳ |
+=====
```

(continues on next page)

(continued from previous page)

```
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/general_
↳ tests/sched/lsf/bmgroups.yml |
+-----+
↳ -----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1
[show_host_groups][home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/
↳ v0.10.0/general_tests/sched/lsf/bmgroups.yml]: test is skipped because ['run_only']
↳ 'scheduler' got value: lsf but detected scheduler: [].
No buildsspecs to process because there are no valid buildsspecs
```

run_only - linux_distro

buildtest can run test if it matches a Linux distro, this is configured using `linux_distro` field that is a list of Linux distros that is returned via `distro.id()`. In this example, we run test only if host distro is darwin.

```
version: "1.0"
buildspecs:
  run_only_macos_distro:
    type: script
    executor: generic.local.bash
    description: "Run test only if distro is darwin."
    tags: [mac]
    run_only:
      linux_distro:
        - darwin
    run: uname
    status:
      regex:
        stream: stdout
        exp: "^Darwin$"

  run_only_linux_distro:
    type: script
    executor: generic.local.bash
    description: "Run test only if distro is CentOS."
    tags: [mac]
    run_only:
      linux_distro:
        - centos
    run: uname
```

This test will run successfully because this was ran on a Mac OS (darwin) system.

```
$ buildtest build -b tutorials/run_only_distro.yml
```

```
User: siddiq90
Hostname: DOE-7086392.local
Platform: Darwin
```

(continues on next page)

(continued from previous page)

```

Current Time: 2021/07/06 18:54:28
buildtest path: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest
buildtest version: 0.9.6
python path: /Users/siddiq90/.local/share/virtualenvs/buildtest-KLOcDrW0/bin/python
python version: 3.7.3
Test Directory: /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests
Configuration File: /Users/siddiq90/Documents/GitHubDesktop/buildtest/buildtest/
↳ settings/config.yml
Command: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest build -b_
↳ tutorials/run_only_distro.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
| Discovered Buildsspecs |
+-----+
| /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/run_only_distro.yml |
+-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1
[run_only_linux_distro][Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/run_
↳ only_distro.yml]: test is skipped because this test is expected to run on linux_
↳ distro: ['centos'] but detected linux distro: darwin.

+-----+
| Stage: Parsing Buildsspecs |
+-----+

  schemafilename | validstate | buildspectestpath
+-----+
↳ script-v1.0.schema.json | True | /Users/siddiq90/Documents/GitHubDesktop/
↳ buildtest/tutorials/run_only_distro.yml

name | description
+-----+
run_only_macos_distro | Run test only if distro is darwin.

+-----+
| Stage: Building Test |
+-----+

  name | id | type | executor | tags | testpath
+-----+
↳ run_only_macos_distro | 9d4d0d97 | script | generic.local.bash | ['mac'] | /Users/
↳ siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/run_only_
↳ distro/run_only_macos_distro/0/run_only_macos_distro_build.sh

```


(continued from previous page)

```

+-----+
| Stage: Running Test |
+-----+

name          | id          | executor          | status  | returncode
+-----+-----+-----+-----+-----+
run_only_macos_distro | 9d4d0d97 | generic.local.bash | PASS    |          0

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 1/1 Percentage: 100.000%
Failed Tests: 0/1 Percentage: 0.000%

Writing Logfile to: /var/folders/1m/_jjv09h17k37mkktwnmbkmj0002t_q/T/buildtest_6asbja74.
↳ log
A copy of logfile can be found at $BUILDTTEST_ROOT/buildtest.log - /Users/siddiq90/
↳ Documents/GitHubDesktop/buildtest/buildtest.log

```

Running test across multiple executors

The *executor* property can support regular expression to search for compatible executors, this can be used if you want to run a test across multiple executors. In buildtest, we use `re.fullmatch` with the input pattern defined by **executor** property against a list of available executors defined in configuration file. You can retrieve a list of executors by running `buildtest config executors`.

In example below we will run this test on *generic.local.bash* and *generic.local.sh* executor based on the regular expression.

```

version: "1.0"
buildspecs:
  executor_regex_script_schema:
    type: script
    executor: "generic.local.(bash|sh)"
    description: regular expression test with executor using script schema
    tags: [tutorials]
    run: date

```

If we build this test, notice that there are two tests, one for each executor.

```
$ buildtest build -b tutorials/executor_regex_script.yml
```

```

User: siddiq90
Hostname: DOE-7086392.local

```

(continues on next page)

(continued from previous page)

```

Platform: Darwin
Current Time: 2021/07/06 18:54:11
buildtest path: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest
buildtest version: 0.9.6
python path: /Users/siddiq90/.local/share/virtualenvs/buildtest-KLOcDrW0/bin/python
python version: 3.7.3
Test Directory: /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests
Configuration File: /Users/siddiq90/Documents/GitHubDesktop/buildtest/buildtest/
↳ settings/config.yml
Command: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest build -b_
↳ tutorials/executor_regex_script.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
| Discovered Buildsspecs |
+=====+
| /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/executor_regex_script.yml |
+-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

+-----+
| Stage: Parsing Buildsspecs |
+-----+

  schemafile          | validstate | buildspec
+-----+-----+
↳ script-v1.0.schema.json | True      | /Users/siddiq90/Documents/GitHubDesktop/
↳ buildtest/tutorials/executor_regex_script.yml

name                  description
+-----+-----+
executor_regex_script_schema regular expression test with executor using script schema
executor_regex_script_schema regular expression test with executor using script schema

+-----+
| Stage: Building Test |
+-----+

  name                  | id      | type | executor          | tags          |
↳ testpath
+-----+-----+-----+-----+-----+
↳
↳
executor_regex_script_schema | 225b1101 | script | generic.local.bash | ['tutorials'] |
↳ /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/
↳ executor_regex_script/executor_regex_script_schema/1/executor_regex_script_schema_
↳ build.sh

```

(continued from previous page)

```

executor_regex_script_schema | bdc3a622 | script | generic.local.sh | ['tutorials'] |
↪ /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.sh/executor_
↪ regex_script/executor_regex_script_schema/1/executor_regex_script_schema_build.sh

+-----+
| Stage: Running Test |
+-----+

name | id | executor | status | returncode
-----+-----+-----+-----+-----+
executor_regex_script_schema | 225b1101 | generic.local.bash | PASS | 0
executor_regex_script_schema | bdc3a622 | generic.local.sh | PASS | 0

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 2/2 Percentage: 100.000%
Failed Tests: 0/2 Percentage: 0.000%

Writing Logfile to: /var/folders/1m/_jjv09h17k37mkktwnmbkmj0002t_q/T/buildtest_n4z9jpc_.
↪ log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /Users/siddiq90/
↪ Documents/GitHubDesktop/buildtest/buildtest.log

```

Passing Test based on runtime

buildtest can determine state of test based on *runtime* property which is part of *status* object. This can be used if you want to control how test *PASS* or *FAIL* based on execution time of test. In example below we have five tests that make use of **runtime** property for passing a test. The runtime property support *min* and *max* property that can mark test pass based on minimum and maximum runtime. A test will pass if it's execution time is greater than *min* time and less than *max* time. If *min* is specified without *max* property the upperbound is not set, likewise *max* without *min* will pass if test is less than **max** time. The lower bound is not set, but test runtime will be greater than 0 sec.

In test **timelimit_min**, we sleep for 2 seconds and it will pass because minimum runtime is 1.0 seconds. Similarly, **timelimit_max** will pass because we sleep for 2 seconds with a max time of 5.0.

```

version: "1.0"
buildspecs:
  timelimit_min_max:
    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: ["tutorials"]
    run: sleep 2
    status:

```

(continues on next page)

(continued from previous page)

```
runtime:
  min: 1.0
  max: 3.0

timelimit_min:
  type: script
  executor: generic.local.sh
  description: "Run a sleep job for 2 seconds and test pass if its exceeds min time of ↵
↵1.0 sec"
  tags: ["tutorials"]
  run: sleep 2
  status:
    runtime:
      min: 1.0

timelimit_max:
  type: script
  executor: generic.local.sh
  description: "Run a sleep job for 2 seconds and test pass if it's within max time: 5.
↵0 sec"
  tags: ["tutorials"]
  run: 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 mintime of 10 second"
  tags: ["tutorials"]
  run: 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: ["tutorials"]
  run: sleep 3
  status:
    runtime:
      max: 1.0
```

```
$ buildtest build -b tutorials/runtime_status_test.yml
```

```
User: docs
```

(continues on next page)

(continued from previous page)

```

Hostname: build-14361642-project-280831-buildtest
Platform: Linux
Current Time: 2021/07/30 15:08:31
buildtest path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/bin/buildtest
buildtest version: 0.10.0
python path: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/envs/v0.10.0/bin/
↳ python
python version: 3.6.12
Test Directory: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.
↳ 10.0/var/tests
Configuration File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
↳ checkouts/v0.10.0/buildtest/settings/config.yml
Command: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ bin/buildtest build -b tutorials/runtime_status_test.yml

```

```

+-----+
| Stage: Discovering Buildsspecs |
+-----+

```

```

+-----+
↳ -----+
| Discovered Buildsspecs |
↳ |
+=====+
| /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/tutorials/
↳ runtime_status_test.yml |
+-----+

```

```

↳ -----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

```

```

+-----+
| Stage: Parsing Buildsspecs |
+-----+

```

```

schemafile          | validstate | buildspect
-----+-----+-----
↳ script-v1.0.schema.json | True       | /home/docs/checkouts/readthedocs.org/user_
↳ builds/buildtest/checkouts/v0.10.0/tutorials/runtime_status_test.yml

```

name	description
timelimit_min_max	Run a sleep job for 2 seconds and test pass if its within 1.0-3.0sec
timelimit_min	Run a sleep job for 2 seconds and test pass if its exceeds min time.↳
↳ of 1.0 sec	
timelimit_max	Run a sleep job for 2 seconds and test pass if it's within max time:↳
↳ 5.0 sec	

(continues on next page)

(continued from previous page)

timelimit_min_fail This test fails because it runs less than mintime of 10 second

timelimit_max_fail This test fails because it exceeds maxtime of 1.0 second

```

+-----+
| Stage: Building Test |
+-----+

```

name	id	type	executor	tags	testpath
timelimit_min_max	1ebfe81a	script	generic.local.sh	['tutorials']	/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/generic.local.sh/runtime_status_test/timelimit_min_max/0/timelimit_min_max_build.sh
timelimit_min	8d6de0cc	script	generic.local.sh	['tutorials']	/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/generic.local.sh/runtime_status_test/timelimit_min/0/timelimit_min_build.sh
timelimit_max	44821569	script	generic.local.sh	['tutorials']	/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/generic.local.sh/runtime_status_test/timelimit_max/0/timelimit_max_build.sh
timelimit_min_fail	d6e452ef	script	generic.local.sh	['tutorials']	/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/generic.local.sh/runtime_status_test/timelimit_min_fail/0/timelimit_min_fail_build.sh
timelimit_max_fail	9d91d66e	script	generic.local.sh	['tutorials']	/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/tests/generic.local.sh/runtime_status_test/timelimit_max_fail/0/timelimit_max_fail_build.sh

```

+-----+
| Stage: Running Test |
+-----+

```

name	id	executor	status	returncode
timelimit_min_max	1ebfe81a	generic.local.sh	PASS	0
timelimit_min	8d6de0cc	generic.local.sh	PASS	0
timelimit_max	44821569	generic.local.sh	PASS	0
timelimit_min_fail	d6e452ef	generic.local.sh	FAIL	0
timelimit_max_fail	9d91d66e	generic.local.sh	FAIL	0

```

+-----+
| Stage: Test Summary |
+-----+

```

Passed Tests: 3/5 Percentage: 60.000%

Failed Tests: 2/5 Percentage: 40.000%

Writing Logfile to: /tmp/buildtest_ok9a6o5b.log

(continues on next page)

(continued from previous page)

A copy of logfile can be found at \$BUILDTTEST_ROOT/buildtest.log - /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/buildtest.log

If we look at the test results, we expect the first three tests **timelimit_min**, **timelimit_max**, **timelimit_min_max** will pass while the last two tests fail because it fails to comply with runtime property.

```
$ buildtest report --filter buildspectutorialstutorials/runtime_status_test.yml --format name,id,
state,runtime --latest
Reading report file: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/
checkouts/v0.10.0/var/report.json
```

name	id	state	runtime
timelimit_min_max	1ebfe81a	PASS	2.0061
timelimit_min	8d6de0cc	PASS	2.00664
timelimit_max	44821569	PASS	2.00661
timelimit_min_fail	d6e452ef	FAIL	2.00665
timelimit_max_fail	9d91d66e	FAIL	3.00667

3.6.2 Global Schema

The global schema is validated with for all buildspects is the top-level schema when defining a buildspect file.

Please refer to [Global Schema Documentation](#) that provides a summary .

Schema Definition

Shown below is the start of the schema definition for **global.schema.json**

```
{
  "$id": "global.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "global schema",
  "description": "buildtest global schema is validated for all buildspects. The global
schema defines top-level structure of buildspect and defintions that are inherited for
sub-schemas",
  "type": "object",
  "required": ["version","buildspecs"],
  "additionalProperties": false,
```

This schema requires that every buildspect should have **version** and **buildspecs** fields. The version key is required to lookup an a sub-schema using the type field. The buildspecs is the start of test declaration.

Shown below is an example buildspect.

```
version: "1.0"
buildspecs:
  hello_world:
    executor: generic.local.bash
    type: script
    tags: tutorials
    description: "hello world example"
    run: echo "hello world!"
maintainers:
- "@shahzebsiddiqui"
```

The `maintainers` is an optional field that is an array test maintainers. To understand how buildtest validates the buildspec see *parsing buildspecs*.

In this example, the global schema validates the following section:

```
version: "1.0"
buildspecs:
  hello_world:

maintainers:
- "@shahzebsiddiqui"
```

The field `version` `buildspecs` and `maintainers` are validated with `global.schema.json` using `jsonschema.validate` method. The test section within `hello_world` is validated by sub-schema by looking up schema based on `type` field:

```
hello_world:
  executor: generic.local.bash
  type: script
  description: "hello world example"
  run: echo "hello world!"
```

Every sub-schema requires `type` field in this case, `type: script` directs buildtest to validate with the script schema. All type schemas have a version, currently buildtest supports **1.0** version for all type schemas. The `version: "1.0"` is used to select the version of the sub-schema, in this example we validate with the schema `script-v1.0.schema.json`.

Test Names

The `buildspecs` property is a JSON object that defines one or more test. This is defined in JSON as follows:

```
"buildspecs": {
  "type": "object",
  "description": "This section is used to define one or more tests (buildspecs). Each
↪test must be unique name",
  "propertyNames": {
    "pattern": "^[A-Za-z_.][A-Za-z0-9_.]*$",
    "maxLength": 32
  }
}
```

The test names take the following pattern `^[A-Za-z_.][A-Za-z0-9_.]*$` and limited to 32 characters. In previous example, the test name is `hello_world`. You must have unique testname in your `buildspecs` section, otherwise you will have an invalid buildspec file. The `description` field is used to document the test and limited to 80 characters.

Note: We refer to the entire YAML content as **buildspec file**, this is not to be confused with the **buildspecs** field.

Buildspec Structure

Shown below is an overview of buildspec file. In this diagram we define one test within `buildspecs` property named `systemd_default_target`. This test is using the script schema defined by `type: script`. The `executor` property is a required property that determines how test is run. The executors are defined in buildtest configuration see *Configuring buildtest* for more details.

The `run` property is used for defining content of script, this can a shell-script (bash,csh) or python script.

<code>version: "1.0"</code>	Schema Version
<code>buildspecs:</code>	Declaration of tests
<code> systemd_default_target:</code>	Name of Test
<code> executor: generic.local.bash</code>	Name of Executor
<code> type: script</code>	Schema Type
<code> tags: [system]</code>	Tag Name
<code> description: check if default target is multi-user.target</code>	Description of Test
<code> run: </code> 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	Script

Please proceed to *Buildspec Overview* to learn more about buildspecs.

3.6.3 Compiler Schema

The compiler schema is used for compilation of programs, currently we support single source file compilation. In order to use the compiler schema you must set `type: compiler` in your sub-schema. See *compiler schema docs*

Compilation Examples

We assume the reader has basic understanding of *Global Schema* validation. Shown below is the schema header definition for `compiler-v1.0.schema.json`:

```
{
  "$id": "compiler-v1.0.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "compiler schema version 1.0",
  "description": "The compiler schema is of ``type: compiler`` in sub-schema which is
  ↪used for compiling and running programs",
  "type": "object",
  "required": [
    "type",
    "source",
    "compilers",
    "executor"
  ],
}
```

The required fields for compiler schema are **type**, **compilers**, **source** and **executor**.

Shown below is a test name `hello_f` that compiles Fortran code with GNU compiler.

```
version: "1.0"
buildspecs:
  hello_f:
    type: compiler
    description: "Hello World Fortran Compilation"
    executor: generic.local.bash
    tags: [tutorials, compile]
    source: "src/hello.f90"
    compilers:
      name: ["^(builtin_gcc)$"]
      default:
        gcc:
          fflags: -Wall
```

The source property is used to specify input program for compilation, this can be a file relative to buildspec file or an absolute path. In this example the source file `src/hello.f90` is relative to buildspec file. The `compilers` section specifies compiler configuration, the `name` field is required property which is used to search compilers based on regular expression. In this example we use the `builtin_gcc` compiler as regular expression which is the system gcc compiler provided by buildtest. The `default` section specifies default compiler configuration applicable to a specific compiler group.

Shown below is an example build for the buildspec example

```
$ buildtest build -b tutorials/compilers/gnu_hello_fortran.yml

User: siddiq90
Hostname: DOE-7086392.local
Platform: Darwin
Current Time: 2021/07/06 18:54:28
buildtest path: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest
buildtest version: 0.9.6
python path: /Users/siddiq90/.local/share/virtualenvs/buildtest-KLOcDrW0/bin/python
python version: 3.7.3
Test Directory: /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests
Configuration File: /Users/siddiq90/Documents/GitHubDesktop/buildtest/buildtest/
↳ settings/config.yml
Command: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest build -b_
↳ tutorials/compilers/gnu_hello_fortran.yml
```

```
+-----+
| Stage: Discovering Buildsspecs |
+-----+
```

```
+-----+
↳ -----+
| Discovered Buildsspecs |
↳ |
+=====+
| /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/compilers/gnu_hello_
↳ fortran.yml |
+-----+
↳ -----+
```

(continues on next page)

(continued from previous page)

```

Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

+-----+
| Stage: Parsing Buildsspecs |
+-----+

  schemafile          | validstate | buildspec
+-----+-----+-----+
↪ compiler-v1.0.schema.json | True      | /Users/siddiq90/Documents/GitHubDesktop/
↪ buildtest/tutorials/compilers/gnu_hello_fortran.yml

name      description
-----
hello_f   Hello World Fortran Compilation

+-----+
| Stage: Building Test |
+-----+

name      | id          | type      | executor          | tags
↪ compiler | testpath
+-----+-----+-----+-----+-----+
↪
↪
hello_f | 5e3d8b5f | compiler | generic.local.bash | ['tutorials', 'compile'] | builtin_
↪ gcc | /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.bash/
↪ gnu_hello_fortran/hello_f/1/hello_f_build.sh

+-----+
| Stage: Running Test |
+-----+

name      | id          | executor          | status | returncode
+-----+-----+-----+-----+-----+
hello_f | 5e3d8b5f | generic.local.bash | FAIL   | 127

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 0/1 Percentage: 0.000%
Failed Tests: 1/1 Percentage: 100.000%

```

(continues on next page)

(continued from previous page)

```
Writing Logfile to: /var/folders/1m/_jjv09h17k37mkktwnmbkmj0002t_q/T/buildtest_ycbz5z6n.
↳ log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /Users/siddiq90/
↳ Documents/GitHubDesktop/buildtest/buildtest.log
```

The generated test for test name **hello_f** is the following:

```
#!/bin/bash

# name of executable
_EXECUTABLE=hello.f90.exe
# Compilation Line
gfortran -o $_EXECUTABLE /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/compilers/
↳ src/hello.f90

# Run executable
./$_EXECUTABLE
```

buildtest will use compiler wrappers specified in your settings to build the test, however these values can be overridden in buildspec file which will be discussed later.

The builtin_gcc compiler is defined below this can be retrieved by running `buildtest config compilers`. The `-y` will display compilers in YAML format.

```
$ buildtest config compilers -y
gcc:
  builtin_gcc:
    cc: /usr/bin/gcc
    cxx: /usr/bin/g++
    fc: /usr/bin/gfortran
```

buildtest will compile and run the code depending on the compiler flags. buildtest, will detect the file extension of source file (source property) to detect programming language and finally generate the appropriate C, C++, or Fortran compilation based on language detected. In this example, buildtest detects a **.f90** file extension and determines this is a Fortran program.

Shown below is the file extension table buildtest uses for determining the programming language.

Table 1: File Extension Language Mapping

Language	File Extension
C	.c
C++	.cc .cxx .cpp .c++
Fortran	.f90 .F90 .f95 .f .F .FOR .for .FTN .ftn

Compiler Selection

buildtest selects compiler based on name property which is a list of regular expression applied for available compilers defined in buildtest configuration. In example below we select all compilers with regular expression `^(builtin_gcc|gcc)` that is specified in line name: `["^(builtin_gcc|gcc)"]`

```
version: "1.0"
buildspecs:
  vecadd_gnu:
    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)"]
      default:
        gcc:
          cflags: -fopenacc
          ldflags: -lm
```

Currently, we have 3 compilers defined in buildtest settings, shown below is a listing of all compilers. We used `buildtest config compilers find` to *detect compilers*.

```
$ buildtest config compilers
builtin_gcc
gcc/9.3.0-n7p74fd
gcc/10.2.0-37fmsw7
```

Note: This example may vary on your machine depending on compilers available via `module` command.

We expect buildtest to select all three compilers based on our regular expression. In the following build, notice we have three tests for `vecadd_gnu` one for each compiler:

```
$ buildtest build -b tutorials/compilers/vecadd.yml

User: siddiq90
Hostname: DOE-7086392.local
Platform: Darwin
Current Time: 2021/06/10 21:52:32
buildtest path: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest
buildtest version: 0.9.5
python path: /Users/siddiq90/.local/share/virtualenvs/buildtest-KLOcDrW0/bin/python
python version: 3.7.3
Test Directory: /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests
Configuration File: /Users/siddiq90/.buildtest/config.yml
Command: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest build -b_
↳ tutorials/compilers/vecadd.yml

+-----+
| Stage: Discovering Buildsspecs |
```

(continues on next page)

(continued from previous page)

```

+-----+
+-----+
| Discovered Buildsspecs |
+=====+
| /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/compilers/vecadd.yml |
+-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

+-----+
| Stage: Parsing Buildsspecs |
+-----+

  schemafile          | validstate | buildspec
+-----+-----+-----+
↪ compiler-v1.0.schema.json | True      | /Users/siddiq90/Documents/GitHubDesktop/
↪ buildtest/tutorials/compilers/vecadd.yml

name      description
+-----+-----+
vecadd_gnu Vector Addition example with GNU compiler
vecadd_gnu Vector Addition example with GNU compiler
vecadd_gnu Vector Addition example with GNU compiler

+-----+
| Stage: Building Test |
+-----+

  name      | id      | type      | executor      | tags      |
↪ compiler  |         | testpath  |               |           |
+-----+-----+-----+-----+-----+
↪
↪
vecadd_gnu | 6f6b16e1 | compiler | generic.local.bash | ['tutorials', 'compile'] |
↪ builtin_gcc | /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/
↪ generic.local.bash/vecadd/vecadd_gnu/2/vecadd_gnu_build.sh
vecadd_gnu | a76dd163 | compiler | generic.local.bash | ['tutorials', 'compile'] | gcc/
↪ 9.3.0-n7p74fd | /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.
↪ local.bash/vecadd/vecadd_gnu/3/vecadd_gnu_build.sh
vecadd_gnu | 82360702 | compiler | generic.local.bash | ['tutorials', 'compile'] | gcc/
↪ 10.2.0-37fmsw7 | /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.
↪ local.bash/vecadd/vecadd_gnu/4/vecadd_gnu_build.sh

+-----+
| Stage: Running Test |

```

(continues on next page)

(continued from previous page)

+-----+				
name	id	executor	status	returncode
+-----+				
vecadd_gnu	6f6b16e1	generic.local.bash	PASS	0
vecadd_gnu	a76dd163	generic.local.bash	PASS	0
vecadd_gnu	82360702	generic.local.bash	PASS	0
+-----+				
Stage: Test Summary				
+-----+				
Passed Tests: 3/3 Percentage: 100.000%				
Failed Tests: 0/3 Percentage: 0.000%				
Writing Logfile to: /Users/siddiq90/buildtest/buildtest_b0jwyoyv.log				
A copy of logfile can be found at \$BUILDTEST_ROOT/buildtest.log - /Users/siddiq90/				
↪ Documents/GitHubDesktop/buildtest/buildtest.log				

buildtest will use compiler settings including module configuration from buildtest settings (`config.yml`). In example below we show the compiler definitions for the three gcc compilers. The `module` section is the declaration of modules to load, by default we disable purge (`purge: False`) which instructs buildtest to not insert `module purge`. The `load` is a list of modules to load via `module load`.

Shown below is the compiler configuration.

```

1 compilers:
2   find:
3     gcc: ^(gcc)
4   compiler:
5     gcc:
6       builtin_gcc:
7         cc: gcc
8         fc: gfortran
9         cxx: g++
10      gcc/9.3.0-n7p74fd:
11        cc: gcc
12        cxx: g++
13        fc: gfortran
14      module:
15        load:
16          - gcc/9.3.0-n7p74fd
17        purge: false
18      gcc/10.2.0-37fmsw7:
19        cc: gcc
20        cxx: g++
21        fc: gfortran
22      module:
23        load:
24          - gcc/10.2.0-37fmsw7
25        purge: false

```

If we take a closer look at the generated test we see the *module load* command in the test script.

```
1 #!/bin/bash
2
3
4 # name of executable
5 _EXEC=vecAdd.c.exe
6 # Loading modules
7 module load gcc/10.2.0-37fmsw7
8 # Compilation Line
9 gcc -fopenacc -o $_EXEC /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/
  ↪compilers/src/vecAdd.c -lm
10
11
12 # Run executable
13 ./$_EXEC
```

```
1 #!/bin/bash
2
3
4 # name of executable
5 _EXEC=vecAdd.c.exe
6 # Loading modules
7 module load gcc/9.3.0-n7p74fd
8 # Compilation Line
9 gcc -fopenacc -o $_EXEC /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/
  ↪compilers/src/vecAdd.c -lm
10
11
12 # Run executable
13 ./$_EXEC
```

Excluding Compilers

The `exclude` property is part of `compilers` section which allows one to exclude compilers upon discovery by name field. The `exclude` property is a list of compiler names that will be removed from test generation which is done prior to build phase. buildtest will exclude any compilers specified in `exclude` if they were found based on regular expression in name field. In this example, we slightly modified previous example by excluding `gcc/10.2.0-37fmsw7` compiler. This is specified by `exclude: [gcc/10.2.0-37fmsw7]`.

```
version: "1.0"
buildspecs:
  vecadd_gnu_exclude:
    type: compiler
    description: Vector Addition example with GNU compilers but exclude gcc@10.2.0
    tags: [tutorials, compile]
    executor: generic.local.bash
    source: src/vecAdd.c
    compilers:
      name: ["^(gcc)"]
      exclude: [gcc/10.2.0-37fmsw7]
      default:
```

(continues on next page)

(continued from previous page)

```
gcc:
  cflags: -fopenacc
  ldflags: -lm
```

Notice when we build this test, buildtest will exclude **gcc/10.2.0-37fmsw7** compiler and test is not created during build phase.

```
1 $ buildtest build -b tutorials/compilers/compiler_exclude.yml
2
3
4 User: siddiq90
5 Hostname: DOE-7086392.local
6 Platform: Darwin
7 Current Time: 2021/06/10 21:56:11
8 buildtest path: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest
9 buildtest version: 0.9.5
10 python path: /Users/siddiq90/.local/share/virtualenvs/buildtest-KLOcDrW0/bin/python
11 python version: 3.7.3
12 Test Directory: /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests
13 Configuration File: /Users/siddiq90/.buildtest/config.yml
14 Command: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest build -b_
↳ tutorials/compilers/compiler_exclude.yml
15
16 +-----+
17 | Stage: Discovering Buildsspecs |
18 +-----+
19
20 +-----+
↳ ----+
21 | Discovered Buildsspecs |
↳ |
22 +=====+
23 | /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/compilers/compiler_exclude.
↳ yml |
24 +-----+
↳ ----+
25 Discovered Buildsspecs: 1
26 Excluded Buildsspecs: 0
27 Detected Buildsspecs after exclusion: 1
28 Excluding compiler: gcc/10.2.0-37fmsw7 from test generation
29
30 +-----+
31 | Stage: Parsing Buildsspecs |
32 +-----+
33
34 schemafile          | validstate | buildspec
35 -----+-----+-----
↳ -----+
36 compiler-v1.0.schema.json | True      | /Users/siddiq90/Documents/GitHubDesktop/
↳ buildtest/tutorials/compilers/compiler_exclude.yml
37
38
```

(continues on next page)

(continued from previous page)

```

39
40 name                description
41 -----
42 vecadd_gnu_exclude  Vector Addition example with GNU compilers but exclude gcc@10.2.0
43
44 +-----+
45 | Stage: Building Test |
46 +-----+
47
48
49
50 name                | id          | type      | executor          | tags
51 ↪ | compiler          | testpath
52 -----+-----+-----+-----+-----
53 ↪ +-----+
54 ↪ -----
55 vecadd_gnu_exclude | a7373d09 | compiler | generic.local.bash | ['tutorials', 'compile
56 ↪'] | gcc/9.3.0-n7p74fd | /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/
57 ↪generic.local.bash/compiler_exclude/vecadd_gnu_exclude/0/vecadd_gnu_exclude_build.sh
58
59 +-----+
60 | Stage: Running Test |
61 +-----+
62
63
64
65 name                | id          | executor          | status | returncode
66 -----+-----+-----+-----+-----
67 vecadd_gnu_exclude | a7373d09 | generic.local.bash | PASS   | 0
68
69 +-----+
70 | Stage: Test Summary |
71 +-----+
72
73 Passed Tests: 1/1 Percentage: 100.000%
74 Failed Tests: 0/1 Percentage: 0.000%
75
76 Writing Logfile to: /Users/siddiq90/buildtest/buildtest_4szlay_j.log
77 A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /Users/siddiq90/
78 ↪Documents/GitHubDesktop/buildtest/buildtest.log

```

Compiler Defaults and Override Default Settings

Sometimes you may want to set default compiler flags (**cflags**, **fflags**, **cxxflags**), preprocessor (**cppflags**) or linker flags (**ldflags**) for compiler group (gcc, intel, pgi, etc...). This can be achieved using the `default` property that is part of **compilers** section.

The `default` field is organized into compiler groups, in example below we set default C compiler flags (`cflags: -O1`). In addition, we can override default settings using the `config` property where one must specify the compiler name to override. In example below we can override compiler settings for `gcc/9.3.0-n7p74fd` to use `-O2` and `gcc/10.2.0-37fmsw7` to use `-O3` for **cflags**.

```

version: "1.0"
buildspecs:
  hello_c:
    type: compiler
    description: "Hello World C Compilation"
    executor: generic.local.bash
    tags: [tutorials, compile]
    source: "src/hello.c"
    compilers:
      name: ["^(builtin_gcc|gcc)"]
      default:
        gcc:
          cflags: -O1
        config:
          gcc/9.3.0-n7p74fd:
            cflags: -O2
          gcc/10.2.0-37fmsw7:
            cflags: -O3

```

Next we run this test, and we get three tests for test name **hello_c**.

```

$ buildtest build -b tutorials/compilers/gnu_hello_c.yml

User: siddiq90
Hostname: DOE-7086392.local
Platform: Darwin
Current Time: 2021/06/10 22:00:08
buildtest path: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest
buildtest version: 0.9.5
python path: /Users/siddiq90/.local/share/virtualenvs/buildtest-KLOcDrW0/bin/python
python version: 3.7.3
Test Directory: /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests
Configuration File: /Users/siddiq90/.buildtest/config.yml
Command: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest build -b_
↳tutorials/compilers/gnu_hello_c.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
| Discovered Buildsspecs |
+=====+
| /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/compilers/gnu_hello_c.yml |
+-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

+-----+
| Stage: Parsing Buildsspecs |
+-----+

```

(continues on next page)

(continued from previous page)

```

schemafile          | validstate | buildspect
-----+-----+-----
↪-----+-----+-----
↪ compiler-v1.0.schema.json | True          | /Users/siddiq90/Documents/GitHubDesktop/
↪ buildtest/tutorials/compilers/gnu_hello_c.yml

name      description
-----+-----
hello_c   Hello World C Compilation
hello_c   Hello World C Compilation
hello_c   Hello World C Compilation

+-----+
| Stage: Building Test |
+-----+

name      | id          | type      | executor          | tags          |
↪ compiler          | testpath
-----+-----+-----+-----+-----+
↪-----+-----+-----+-----+-----+
↪-----+-----+-----+-----+-----+
hello_c | afa92b9d | compiler | generic.local.bash | ['tutorials', 'compile'] | builtin_
↪ gcc          | /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.
↪ bash/gnu_hello_c/hello_c/2/hello_c_build.sh
hello_c | 498010d3 | compiler | generic.local.bash | ['tutorials', 'compile'] | gcc/9.3.
↪ 0-n7p74fd | /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.local.
↪ bash/gnu_hello_c/hello_c/3/hello_c_build.sh
hello_c | ee753488 | compiler | generic.local.bash | ['tutorials', 'compile'] | gcc/10.
↪ 2.0-37fmsw7 | /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.
↪ local.bash/gnu_hello_c/hello_c/4/hello_c_build.sh

+-----+
| Stage: Running Test |
+-----+

name      | id          | executor          | status | returncode
-----+-----+-----+-----+-----+
hello_c | afa92b9d | generic.local.bash | PASS   | 0
hello_c | 498010d3 | generic.local.bash | PASS   | 0
hello_c | ee753488 | generic.local.bash | PASS   | 0

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 3/3 Percentage: 100.000%
Failed Tests: 0/3 Percentage: 0.000%

```

(continues on next page)

(continued from previous page)

```
Writing Logfile to: /Users/siddiq90/buildtest/buildtest_dtyx0ags.log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /Users/siddiq90/
↳ Documents/GitHubDesktop/buildtest/buildtest.log
```

```
Writing Logfile to: /private/tmp/buildtest/buildtest_hh9k7vm6.log
```

If we inspect the following test, we see the compiler flags are associated with the compiler. The test below is for *builtin_gcc* which use the default -O1 compiler flag as shown below.

```
1  #!/bin/bash
2
3
4  # name of executable
5  _EXEC=hello.c.exe
6  # Compilation Line
7  gcc -O1 -o $_EXEC /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/compilers/
8  ↳ src/hello.c
9
10 # Run executable
11 ./$_EXEC
```

The test for **gcc/10.2.0-37fmsw7** and **gcc/9.3.0-n7p74fd** have cflags -O3 and -O2 set in their respective tests.

```
1  #!/bin/bash
2
3
4  # name of executable
5  _EXEC=hello.c.exe
6  # Loading modules
7  module load gcc/10.2.0-37fmsw7
8  # Compilation Line
9  gcc -O3 -o $_EXEC /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/compilers/
10 ↳ src/hello.c
11
12 # Run executable
13 ./$_EXEC
```

```
1  #!/bin/bash
2
3
4  # name of executable
5  _EXEC=hello.c.exe
6  # Loading modules
7  module load gcc/9.3.0-n7p74fd
8  # Compilation Line
9  gcc -O2 -o $_EXEC /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/compilers/
10 ↳ src/hello.c
```

(continues on next page)

(continued from previous page)

```

10
11
12 # Run executable
13 ./$_EXEC

```

Setting environment variables

Environment variables can be set using `env` property which is a list of key/value pair to assign environment variables. This property can be used in `default` section within a compiler group. In example below we have an OpenMP Hello World example in C where we define `OMP_NUM_THREADS` environment variable which controls number of OpenMP threads to use when running program. In this example we use 2 threads for all gcc compiler group

```

version: "1.0"
buildspecs:
  openmp_hello_c_example:
    type: compiler
    description: OpenMP Hello World C example
    executor: generic.local.bash
    tags: [tutorials, compile]
    source: "src/hello_omp.c"
    compilers:
      name: ["^(gcc)"]
      default:
        gcc:
          cflags: -fopenmp
          env:
            OMP_NUM_THREADS: 2

```

Shown below is one of the generated test and notice that buildtest will set environment variable `OMP_NUM_THREADS`.

```

1  #!/bin/bash
2
3
4  # name of executable
5  _EXEC=hello_omp.c.exe
6  # Declare environment variables
7  export OMP_NUM_THREADS=2
8
9
10 # Loading modules
11 module load gcc/10.2.0-37fmsw7
12 # Compilation Line
13 gcc -fopenmp -o $_EXEC /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/
14 ↪compilers/src/hello_omp.c
15
16 # Run executable
17 ./$_EXEC

```

Similarly, one can define environment variables at the compiler level in `config` section. buildtest will override value defined in `default` section. In this example, we make slight modification to the test, so that `gcc/10.2.0-37fmsw7`

will use 4 threads when running program. This will override the default value of 2.

```
version: "1.0"
buildspecs:
  override_environmentvars:
    type: compiler
    description: override default environment variables
    executor: generic.local.bash
    tags: [tutorials, compile]
    source: "src/hello_omp.c"
    compilers:
      name: ["^(gcc)"]
      default:
        gcc:
          cflags: -fopenmp
          env:
            OMP_NUM_THREADS: 2
        config:
          gcc/10.2.0-37fmsw7:
            env:
              OMP_NUM_THREADS: 4
```

Next we build this test as follows:

```
$ buildtest build -b tutorials/compilers/envvar_override.yml

User: siddiq90
Hostname: DOE-7086392.local
Platform: Darwin
Current Time: 2021/06/10 22:04:19
buildtest path: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest
buildtest version: 0.9.5
python path: /Users/siddiq90/.local/share/virtualenvs/buildtest-KLOcDrW0/bin/python
python version: 3.7.3
Test Directory: /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests
Configuration File: /Users/siddiq90/.buildtest/config.yml
Command: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest build -b_
↳ tutorials/compilers/envvar_override.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
↳ ---+
| Discovered Buildsspecs |
↳ |
+-----+
| /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/compilers/envvar_override.
↳ yml |
+-----+
↳ ---+
Discovered Buildsspecs: 1
```

(continues on next page)

(continued from previous page)

```

Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

+-----+
| Stage: Parsing Buildsspecs |
+-----+

  schemafile          | validstate | buildspec
+-----+-----+-----+
↪ compiler-v1.0.schema.json | True      | /Users/siddiq90/Documents/GitHubDesktop/
↪ buildtest/tutorials/compiler/envvar_override.yml

name                  description
+-----+-----+
override_environmentvars  override default environment variables
override_environmentvars  override default environment variables

+-----+
| Stage: Building Test |
+-----+

  name                  | id          | type      | executor          | tags
↪      | compiler          | testpath
+-----+-----+-----+-----+-----+
↪
↪
↪
↪
override_environmentvars | 72619a4b | compiler | generic.local.bash | ['tutorials',
↪ 'compile'] | gcc/9.3.0-n7p74fd | /Users/siddiq90/Documents/GitHubDesktop/buildtest/
↪ var/tests/generic.local.bash/envvar_override/override_environmentvars/0/override_
↪ environmentvars_build.sh
override_environmentvars | 31098506 | compiler | generic.local.bash | ['tutorials',
↪ 'compile'] | gcc/10.2.0-37fmsw7 | /Users/siddiq90/Documents/GitHubDesktop/buildtest/
↪ var/tests/generic.local.bash/envvar_override/override_environmentvars/1/override_
↪ environmentvars_build.sh

+-----+
| Stage: Running Test |
+-----+

  name                  | id          | executor          | status | returncode
+-----+-----+-----+-----+-----+
override_environmentvars | 72619a4b | generic.local.bash | PASS   | 0
override_environmentvars | 31098506 | generic.local.bash | PASS   | 0

+-----+
| Stage: Test Summary |
+-----+

```

(continues on next page)

(continued from previous page)

+-----+

Passed Tests: 2/2 Percentage: 100.000%

Failed Tests: 0/2 Percentage: 0.000%

Writing Logfile to: /Users/siddiq90/buildtest/buildtest_p3wdn11t.log

A copy of logfile can be found at \$BUILDTEST_ROOT/buildtest.log - /Users/siddiq90/
→Documents/GitHubDesktop/buildtest/buildtest.log

Now let's inspect the test by running `buildtest inspect name override_environmentvars` and we notice there are two test records for `override_environmentvars` using `gcc/9.3.0-n7p74fd` and `gcc/10.2.0-37fmsw7`.

1 \$ buildtest inspect name override_environmentvars

2 Reading Report File: /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/report.json

```

3 {
4   "override_environmentvars": [
5     {
6       "id": "72619a4b",
7       "full_id": "72619a4b-3ed2-489c-aebd-2e0cacbf2d6a",
8       "description": "override default environment variables",
9       "schemafile": "compiler-v1.0.schema.json",
10      "executor": "generic.local.bash",
11      "compiler": "gcc/9.3.0-n7p74fd",
12      "hostname": "DOE-7086392.local",
13      "user": "siddiq90",
14      "testroot": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.
15      →local.bash/envvar_override/override_environmentvars/0",
16      "testpath": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.
17      →local.bash/envvar_override/override_environmentvars/0/stage/override_environmentvars.sh
18      →",
19      "stagedir": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.
20      →local.bash/envvar_override/override_environmentvars/0/stage",
21      "command": "sh override_environmentvars_build.sh",
22      "outfile": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.
23      →local.bash/envvar_override/override_environmentvars/0/override_environmentvars.out",
24      "errfile": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.
25      →local.bash/envvar_override/override_environmentvars/0/override_environmentvars.err",
26      "buildspec_content": "version: \"1.0\"\nbuildspecs:\n  override_environmentvars:\n
27      →  type: compiler\n    description: override default environment variables\n
28      →  executor: generic.local.bash\n    tags: [tutorials, compile]\n    source: \"src/hello_
29      →omp.c\"\n    compilers:\n      name: [\"^(gcc)\"]\n      default:\n        gcc:\n
30      →  cflags: -fopenmp\n      env:\n        OMP_NUM_THREADS: 2\n      config:\n
31      →  gcc/10.2.0-37fmsw7:\n      env:\n        OMP_NUM_THREADS: 4",
32      "test_content": "#!/bin/bash\n_EXEC=hello_omp.c.exe\nexport OMP_NUM_THREADS=2\n
33      →nmodule load gcc/9.3.0-n7p74fd\ngcc -fopenmp -o $_EXEC /Users/siddiq90/Documents/
34      →GitHubDesktop/buildtest/tutorials/compilers/src/hello_omp.c\n./$_EXEC",
35      "logpath": "/Users/siddiq90/buildtest/buildtest_p3wdn11t.log",
36      "tags": "tutorials compile",
37      "starttime": "2021/06/10 22:04:19",
38      "endtime": "2021/06/10 22:04:20",

```

(continues on next page)

(continued from previous page)

```

27     "runtime": 0.727095,
28     "state": "PASS",
29     "returncode": 0,
30     "output": "Hello World from thread = 0\nHello World from thread = 1\n",
31     "error": "The following have been reloaded with a version change:\n 1) gcc/10.2.0-
↳ 37fmsw7 => gcc/9.3.0-n7p74fd\n",
32     "job": null,
33     "build_script": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/
↳ generic.local.bash/envvar_override/override_environmentvars/0/override_environmentvars_
↳ build.sh"
34 },
35 {
36     "id": "31098506",
37     "full_id": "31098506-2bbf-4a50-8386-2fcd5bcddff5",
38     "description": "override default environment variables",
39     "schemafile": "compiler-v1.0.schema.json",
40     "executor": "generic.local.bash",
41     "compiler": "gcc/10.2.0-37fmsw7",
42     "hostname": "DOE-7086392.local",
43     "user": "siddiq90",
44     "testroot": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.
↳ local.bash/envvar_override/override_environmentvars/1",
45     "testpath": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.
↳ local.bash/envvar_override/override_environmentvars/1/stage/override_environmentvars.sh
↳ ",
46     "stagedir": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.
↳ local.bash/envvar_override/override_environmentvars/1/stage",
47     "command": "sh override_environmentvars_build.sh",
48     "outfile": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.
↳ local.bash/envvar_override/override_environmentvars/1/override_environmentvars.out",
49     "errfile": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/generic.
↳ local.bash/envvar_override/override_environmentvars/1/override_environmentvars.err",
50     "buildspec_content": "version: \"1.0\"\nbuilder:\n  override_environmentvars:\n
↳ type: compiler\n  description: override default environment variables\n
↳ executor: generic.local.bash\n  tags: [tutorials, compile]\n  source: \"src/hello_
↳ omp.c\"\n  compilers:\n    name: [\"^(gcc)\"]\n    default:\n      gcc:\n
↳ cflags: -fopenmp\n    env:\n      OMP_NUM_THREADS: 2\n    config:\n
↳ gcc/10.2.0-37fmsw7:\n      env:\n        OMP_NUM_THREADS: 4",
51     "test_content": "#!/bin/bash\n_EXEC=hello_omp.c.exe\nexport OMP_NUM_THREADS=4\n
↳ module load gcc/10.2.0-37fmsw7\ngcc -fopenmp -o $_EXEC /Users/siddiq90/Documents/
↳ GitHubDesktop/buildtest/tutorials/compilers/src/hello_omp.c\n./$_EXEC",
52     "logpath": "/Users/siddiq90/buildtest/buildtest_p3wdn11t.log",
53     "tags": "tutorials compile",
54     "starttime": "2021/06/10 22:04:20",
55     "endtime": "2021/06/10 22:04:20",
56     "runtime": 0.482645,
57     "state": "PASS",
58     "returncode": 0,
59     "output": "Hello World from thread = 1\nHello World from thread = 3\nHello World
↳ from thread = 2\nHello World from thread = 0\n",
60     "error": "",
61     "job": null,

```

(continues on next page)

(continued from previous page)

```

62     "build_script": "/Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests/
↳ generic.local.bash/envvar_override/override_environmentvars/1/override_environmentvars_
↳ build.sh"
63   }
64 ]
65 }

```

Tweak how test are passed

The status property can be used to determine how buildtest will pass the test. By default, buildtest will use returncode to determine if test PASS or FAIL with exitcode 0 as PASS and anything else is FAIL.

Sometimes, it may be useful check output of test to determine using regular expression. This can be done via status property. In this example, we define two tests, the first one defines status property in the default **gcc** group. This means all compilers that belong to gcc group will be matched with the regular expression.

In second example we override the status regex property for **gcc/10.2.0-37fmsw7**. We expect the test to produce an output of final result: 1.000000 so we expect one failure from **gcc/10.2.0-37fmsw7**.

```

version: "1.0"
buildspecs:
  default_status_regex:
    type: compiler
    description: Regular expression check in stdout for gcc group
    tags: [tutorials, compile]
    executor: generic.local.bash
    source: src/vecAdd.c
    compilers:
      name: ["^(gcc)"]
      default:
        gcc:
          cflags: -fopenacc
          ldflags: -lm
          status:
            regex:
              stream: stdout
              exp: "^final result: 1.000000$"

  override_status_regex:
    type: compiler
    description: Override regular expression for compiler gcc/10.2.0-37fmsw7
    tags: [tutorials, compile]
    executor: generic.local.bash
    source: src/vecAdd.c
    compilers:
      name: ["^(gcc)"]
      default:
        gcc:
          cflags: -fopenacc
          ldflags: -lm
          status:
            regex:

```

(continues on next page)

(continued from previous page)

```

    stream: stdout
    exp: "^final result: 1.000000$"
  config:
    gcc/10.2.0-37fmsw7:
      status:
        regex:
          stream: stdout
          exp: "^final result: 0.99$"

```

If we build this test, notice that test id **9320ca41** failed which corresponds to gcc/10.2.0-37fmsw7 compiler test. The test fails because it fails to pass on regular expression even though we have a returncode of 0.

```

1  $ buildtest build -b tutorials/compilers/compiler_status_regex.yml
2
3
4  User: siddiq90
5  Hostname: DOE-7086392.local
6  Platform: Darwin
7  Current Time: 2021/06/10 22:08:03
8  buildtest path: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest
9  buildtest version: 0.9.5
10 python path: /Users/siddiq90/.local/share/virtualenvs/buildtest-KLOcDrW0/bin/python
11 python version: 3.7.3
12 Test Directory: /Users/siddiq90/Documents/GitHubDesktop/buildtest/var/tests
13 Configuration File: /Users/siddiq90/.buildtest/config.yml
14 Command: /Users/siddiq90/Documents/GitHubDesktop/buildtest/bin/buildtest build -b_
    ↳ tutorials/compilers/compiler_status_regex.yml
15
16 +-----+
17 | Stage: Discovering Buildsspecs |
18 +-----+
19
20 +-----+
21 ↳ -----+
22 | Discovered Buildsspecs |
23 ↳ |
24 +=====+
25 | /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/compilers/compiler_status_
26 ↳ regex.yml |
27 +-----+
28 ↳ -----+
29 Discovered Buildsspecs: 1
30 Excluded Buildsspecs: 0
31 Detected Buildsspecs after exclusion: 1
32
33 +-----+
34 | Stage: Parsing Buildsspecs |
35 +-----+
36
37 schemafile          | validstate | buildspec
38 +-----+
39 ↳ -----+

```

(continues on next page)

(continued from previous page)

```

35 compiler-v1.0.schema.json | True          | /Users/siddiq90/Documents/GitHubDesktop/
    ↪ buildtest/tutorials/compiler/compiler_status_regex.yml
36
37
38
39 name                description
40 -----
41 default_status_regex Regular expression check in stdout for gcc group
42 default_status_regex Regular expression check in stdout for gcc group
43 override_status_regex Override regular expression for compiler gcc/10.2.0-37fmsw7
44 override_status_regex Override regular expression for compiler gcc/10.2.0-37fmsw7
45
46 +-----+
47 | Stage: Building Test |
48 +-----+
49
50
51
52 name                | id          | type      | executor          | tags
53 ↪ | compiler          | testpath
54 -----+-----+-----+-----+-----
55 ↪ +-----+-----+-----+-----+-----
56 ↪ +-----+-----+-----+-----+-----
57 ↪ +-----+-----+-----+-----+-----
58
59 default_status_regex | a023a2c2 | compiler | generic.local.bash | ['tutorials',
    ↪ 'compile'] | gcc/9.3.0-n7p74fd | /Users/siddiq90/Documents/GitHubDesktop/buildtest/
    ↪ var/tests/generic.local.bash/compiler_status_regex/default_status_regex/0/default_
    ↪ status_regex_build.sh
60 default_status_regex | 155865c3 | compiler | generic.local.bash | ['tutorials',
    ↪ 'compile'] | gcc/10.2.0-37fmsw7 | /Users/siddiq90/Documents/GitHubDesktop/buildtest/
    ↪ var/tests/generic.local.bash/compiler_status_regex/default_status_regex/1/default_
    ↪ status_regex_build.sh
61 override_status_regex | 3411bddf | compiler | generic.local.bash | ['tutorials',
    ↪ 'compile'] | gcc/9.3.0-n7p74fd | /Users/siddiq90/Documents/GitHubDesktop/buildtest/
    ↪ var/tests/generic.local.bash/compiler_status_regex/override_status_regex/0/override_
    ↪ status_regex_build.sh
62 override_status_regex | 295310a4 | compiler | generic.local.bash | ['tutorials',
    ↪ 'compile'] | gcc/10.2.0-37fmsw7 | /Users/siddiq90/Documents/GitHubDesktop/buildtest/
    ↪ var/tests/generic.local.bash/compiler_status_regex/override_status_regex/1/override_
    ↪ status_regex_build.sh
63
64 +-----+
65 | Stage: Running Test |
66 +-----+
67
68
69 name                | id          | executor          | status | returncode
70 -----+-----+-----+-----+-----
71 default_status_regex | a023a2c2 | generic.local.bash | PASS   | 0
72 default_status_regex | 155865c3 | generic.local.bash | PASS   | 0
73 override_status_regex | 3411bddf | generic.local.bash | PASS   | 0
74 override_status_regex | 295310a4 | generic.local.bash | FAIL   | 0

```

(continues on next page)

(continued from previous page)

```

70 +-----+
71 | Stage: Test Summary |
72 +-----+
73
74 Passed Tests: 3/4 Percentage: 75.000%
75 Failed Tests: 1/4 Percentage: 25.000%
76
77
78 Writing Logfile to: /Users/siddiq90/buildtest/buildtest_hp7_gpbn.log
79 A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /Users/siddiq90/
  ↪ Documents/GitHubDesktop/buildtest/buildtest.log

```

Single Test Multiple Compilers

It's possible to run single test across multiple compilers (gcc, intel, cray, etc...). In the next example, we will build an OpenMP reduction test using gcc, intel and cray compilers. In this test, we use `name` field to select compilers that start with **gcc**, **intel** and **PrgEnv-cray** as compiler names. The `default` section is organized by compiler groups which inherits compiler flags for all compilers. OpenMP flag for gcc, intel and cray differ for instance one must use `-fopenmp` for gcc, `--qopenmp` for intel and `-h omp` for cray.

```

1  version: "1.0"
2  buildspecs:
3    reduction:
4      type: compiler
5      executor: local.bash
6      source: src/reduction.c
7      description: OpenMP reduction example using gcc, intel and cray compiler
8      tags: [openmp]
9      compilers:
10       name: ["^(gcc|intel|PrgEnv-cray)"]
11       default:
12         all:
13           env:
14             OMP_NUM_THREADS: 4
15           gcc:
16             cflags: -fopenmp
17           intel:
18             cflags: -qopenmp
19           cray:
20             cflags: -h omp

```

In this example `OMP_NUM_THREADS` environment variable under the `all` section which will be used for all compiler groups. This example was built on Cori, we expect this test to run against every gcc, intel and PrgEnv-cray compiler module:

```
$ buildtest build -b buildspecs/apps/openmp/reduction.yml
```

```

User: siddiq90
Hostname: cori02
Platform: Linux

```

(continues on next page)

(continued from previous page)

```

Current Time: 2021/06/11 08:42:54
buildtest path: /global/homes/s/siddiq90/github/buildtest/bin/buildtest
buildtest version: 0.9.5
python path: /global/homes/s/siddiq90/.conda/envs/buildtest/bin/python
python version: 3.8.8
Test Directory: /global/u1/s/siddiq90/github/buildtest/var/tests
Configuration File: /global/u1/s/siddiq90/.buildtest/config.yml
Command: /global/homes/s/siddiq90/github/buildtest/bin/buildtest build -b buildsspecs/
↳ apps/openmp/reduction.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
| Discovered Buildsspecs |
+=====+
| /global/u1/s/siddiq90/github/buildtest-cori/buildspecs/apps/openmp/reduction.yml |
+-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

+-----+
| Stage: Parsing Buildsspecs |
+-----+

  schemafilename  | validstate  | buildspec
-----+-----+-----
↳ compiler-v1.0.schema.json | True        | /global/u1/s/siddiq90/github/buildtest-cori/
↳ buildsspecs/apps/openmp/reduction.yml

name      description
-----
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler

```

(continues on next page)

(continued from previous page)

```

reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler
reduction OpenMP reduction example using gcc, intel, PrgEnv-cray compiler

+-----+
| Stage: Building Test |
+-----+

name      | id      | type      | executor      | tags      | compiler      |
↪         |         | testpath  |               |           |               |
-----+-----+-----+-----+-----+-----+
↪         +-----+
↪         |
reduction | fd93fdcb | compiler | cori.local.bash | ['openmp'] | gcc/6.1.0      |
↪         |         |         |                 |           | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
↪reduction/reduction/25/reduction_build.sh
reduction | 43737191 | compiler | cori.local.bash | ['openmp'] | gcc/7.3.0      |
↪         |         |         |                 |           | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
↪reduction/reduction/26/reduction_build.sh
reduction | 6e2e95cd | compiler | cori.local.bash | ['openmp'] | gcc/8.1.0      |
↪         |         |         |                 |           | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
↪reduction/reduction/27/reduction_build.sh
reduction | c48a8d8d | compiler | cori.local.bash | ['openmp'] | gcc/8.2.0      |
↪         |         |         |                 |           | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
↪reduction/reduction/28/reduction_build.sh
reduction | a6201c48 | compiler | cori.local.bash | ['openmp'] | gcc/8.3.0      |
↪         |         |         |                 |           | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
↪reduction/reduction/29/reduction_build.sh
reduction | aa06b1be | compiler | cori.local.bash | ['openmp'] | gcc/9.3.0      |
↪         |         |         |                 |           | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
↪reduction/reduction/30/reduction_build.sh
reduction | 02b8e7aa | compiler | cori.local.bash | ['openmp'] | gcc/10.1.0     |
↪         |         |         |                 |           | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
↪reduction/reduction/31/reduction_build.sh
reduction | bd9abd7e | compiler | cori.local.bash | ['openmp'] | gcc/6.3.0      |
↪         |         |         |                 |           | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
↪reduction/reduction/32/reduction_build.sh
reduction | 9409a86f | compiler | cori.local.bash | ['openmp'] | gcc/8.1.1-openacc-gcc-
↪8-branch-20190215 | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
↪reduction/reduction/33/reduction_build.sh
reduction | b9700a0f | compiler | cori.local.bash | ['openmp'] | PrgEnv-cray/6.0.5
↪         |         |         |                 |           | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
↪reduction/reduction/34/reduction_build.sh

```

(continues on next page)

(continued from previous page)

```

reduction | a605c970 | compiler | cori.local.bash | ['openmp'] | PrgEnv-cray/6.0.7
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/35/reduction_build.sh
reduction | 9ef915a9 | compiler | cori.local.bash | ['openmp'] | PrgEnv-cray/6.0.9
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/36/reduction_build.sh
reduction | 4f9e4242 | compiler | cori.local.bash | ['openmp'] | intel/19.0.3.199
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/37/reduction_build.sh
reduction | e37befed | compiler | cori.local.bash | ['openmp'] | intel/19.1.2.254
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/38/reduction_build.sh
reduction | 1e9b0ab5 | compiler | cori.local.bash | ['openmp'] | intel/16.0.3.210
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/39/reduction_build.sh
reduction | 4e6d6f8a | compiler | cori.local.bash | ['openmp'] | intel/17.0.1.132
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/40/reduction_build.sh
reduction | ad1e44af | compiler | cori.local.bash | ['openmp'] | intel/17.0.2.174
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/41/reduction_build.sh
reduction | 49acf44b | compiler | cori.local.bash | ['openmp'] | intel/18.0.1.163
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/42/reduction_build.sh
reduction | 4192750c | compiler | cori.local.bash | ['openmp'] | intel/18.0.3.222
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/43/reduction_build.sh
reduction | 06584529 | compiler | cori.local.bash | ['openmp'] | intel/19.0.0.117
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/44/reduction_build.sh
reduction | 82fd9bab | compiler | cori.local.bash | ['openmp'] | intel/19.0.8.324
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/45/reduction_build.sh
reduction | 6140e8b4 | compiler | cori.local.bash | ['openmp'] | intel/19.1.0.166
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/46/reduction_build.sh
reduction | ac509e2e | compiler | cori.local.bash | ['openmp'] | intel/19.1.1.217
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/47/reduction_build.sh
reduction | 9c39818e | compiler | cori.local.bash | ['openmp'] | intel/19.1.2.275
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/48/reduction_build.sh
reduction | 2cb3acd1 | compiler | cori.local.bash | ['openmp'] | intel/19.1.3.304
→ | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.local.bash/
→reduction/reduction/49/reduction_build.sh

+-----+
| Stage: Running Test |
+-----+

name      | id      | executor      | status  | returncode
-----+-----+-----+-----+-----

```

(continues on next page)

(continued from previous page)

reduction	fd93fdcb	cori.local.bash	PASS	0
reduction	43737191	cori.local.bash	PASS	0
reduction	6e2e95cd	cori.local.bash	PASS	0
reduction	c48a8d8d	cori.local.bash	PASS	0
reduction	a6201c48	cori.local.bash	PASS	0
reduction	aa06b1be	cori.local.bash	PASS	0
reduction	02b8e7aa	cori.local.bash	PASS	0
reduction	bd9abd7e	cori.local.bash	PASS	0
reduction	9409a86f	cori.local.bash	PASS	0
reduction	b9700a0f	cori.local.bash	PASS	0
reduction	a605c970	cori.local.bash	PASS	0
reduction	9ef915a9	cori.local.bash	PASS	0
reduction	4f9e4242	cori.local.bash	PASS	0
reduction	e37befed	cori.local.bash	PASS	0
reduction	1e9b0ab5	cori.local.bash	PASS	0
reduction	4e6d6f8a	cori.local.bash	PASS	0
reduction	ad1e44af	cori.local.bash	PASS	0
reduction	49acf44b	cori.local.bash	PASS	0
reduction	4192750c	cori.local.bash	PASS	0
reduction	06584529	cori.local.bash	PASS	0
reduction	82fd9bab	cori.local.bash	PASS	0
reduction	6140e8b4	cori.local.bash	PASS	0
reduction	ac509e2e	cori.local.bash	PASS	0
reduction	9c39818e	cori.local.bash	PASS	0
reduction	2cb3acd1	cori.local.bash	PASS	0


```

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 25/25 Percentage: 100.000%
Failed Tests: 0/25 Percentage: 0.000%

Writing Logfile to: /tmp/buildtest_sq87154s.log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /global/homes/s/
↪siddiq90/github/buildtest/buildtest.log

```

If we inspect one of these tests from each compiler group (gcc, intel) we will see OMP_NUM_THREADS is set in all tests along with the appropriate compiler flag.

```

1  #!/bin/bash
2  _EXEC=reduction.c.exe
3  export OMP_NUM_THREADS=4
4  module load intel/19.1.3.304
5  icc -qopenmp -o $_EXEC /global/u1/s/siddiq90/github/buildtest-cori/buildspecs/apps/
↪openmp/src/reduction.c
6  ./$_EXEC

```

```

1  #!/bin/bash
2  _EXEC=reduction.c.exe
3  export OMP_NUM_THREADS=4

```

(continues on next page)

(continued from previous page)

```

4 module load gcc/6.1.0
5 gcc -fopenmp -o $_EXEC /global/u1/s/siddiq90/github/buildtest-cori/buildspecs/apps/
  ↪ openmp/src/reduction.c
6 ./$_EXEC

```

Customize Run Line

buildtest will define variable `_EXEC` in the job script that can be used to reference the generated binary. By default, buildtest will run the program standalone, but sometimes you may want to customize how job is run. This may include passing arguments or running binary through a job/mpi launcher. The `run` property expects user to specify how to launch program. buildtest will change directory to the called script before running executable. The compiled executable will be present in local directory which can be accessed via `./$_EXEC`. In example below we pass arguments `1 3 5` for gcc group and `100 200` for compiler `gcc/10.2.0-37fmsw7`.

```

version: "1.0"
buildspecs:
  custom_run_by_compilers:
    type: compiler
    description: Customize binary launch based on compiler
    executor: generic.local.bash
    tags: [tutorials, compile]
    source: "src/argc.c"
    compilers:
      name: ["^(builtin_gcc|gcc)"]
      default:
        gcc:
          run: ./$_EXEC 1 3 5
        config:
          gcc/10.2.0-37fmsw7:
            run: ./$_EXEC 100 120

```

If we build this test and see generated test, we notice buildtest customized the run line for launching binary. buildtest will directly replace content in `run` section into the shell-script. If no `run` field is specified buildtest will run the binary in standalone mode (`./$_EXEC`).

```

1  #!/bin/bash
2
3
4  # name of executable
5  _EXEC=argc.c.exe
6  # Loading modules
7  module load gcc/10.2.0-37fmsw7
8  # Compilation Line
9  gcc -o $_EXEC /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/compilers/src/
  ↪ argc.c
10
11
12 # Run executable
13 ./$_EXEC 100 120

```

MPI Example

In this example we run a MPI Laplace code using 4 process on a KNL node using the intel/19.1.2.254 compiler. This test is run on Cori through batch queue system. We can define **#SBATCH** parameters using `sbatch` property. This program is compiled using `mpiicc` wrapper this can be defined using `cc` parameter.

Currently, buildtest cannot detect if program is serial or MPI to infer appropriate compiler wrapper. If `cc` wasn't specified, buildtest would infer `icc` as compiler wrapper for C program. This program is run using `srun` job launcher, we can control how test is executed using the `run` property. This test required we swap intel modules and load `impi/2020` module.

```

1 version: "1.0"
2 buildspecs:
3   laplace_mpi:
4     type: compiler
5     description: Laplace MPI code in C
6     executor: slurm.knl_debug
7     tags: ["mpi"]
8     source: src/laplace_mpi.c
9     compilers:
10      name: ["^(intel/19.1.2.254)$"]
11      default:
12        all:
13          sbatch: ["-N 1", "-n 4"]
14          run: srun -n 4 $_EXEC
15          intel:
16            cc: mpiicc
17            cflags: -O3
18          config:
19            intel/19.1.2.254:
20              module:
21                load: [impi/2020]
22                swap: [intel, intel/19.1.2.254]

```

Shown below is a sample build for this buildspec, buildtest will dispatch and poll job until its complete.

```

$ buildtest build -b buildspecs/apps/mpi/laplace_mpi.yml

User: siddiq90
Hostname: cori02
Platform: Linux
Current Time: 2021/06/11 09:11:16
buildtest path: /global/homes/s/siddiq90/github/buildtest/bin/buildtest
buildtest version: 0.9.5
python path: /global/homes/s/siddiq90/.conda/envs/buildtest/bin/python
python version: 3.8.8
Test Directory: /global/u1/s/siddiq90/github/buildtest/var/tests
Configuration File: /global/u1/s/siddiq90/.buildtest/config.yml
Command: /global/homes/s/siddiq90/github/buildtest/bin/buildtest build -b buildspecs/
↪ apps/mpi/laplace_mpi.yml

+-----+
| Stage: Discovering Buildsspecs |

```

(continues on next page)

(continued from previous page)

```

+-----+
+-----+
| Discovered Buildsspecs |
+-----+
| /global/u1/s/siddiq90/github/buildtest-cori/buildspecs/apps/mpi/laplace_mpi.yml |
+-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

+-----+
| Stage: Parsing Buildsspecs |
+-----+

  schemafile          | validstate | buildspec
+-----+-----+-----+
↪ compiler-v1.0.schema.json | True      | /global/u1/s/siddiq90/github/buildtest-cori/
↪ buildspecs/apps/mpi/laplace_mpi.yml

name      description
+-----+-----+
laplace_mpi Laplace MPI code in C

+-----+
| Stage: Building Test |
+-----+

name      | id      | type      | executor          | tags      | compiler      |
+-----+-----+-----+-----+-----+-----+
↪ testpath

↪ -----
laplace_mpi | a6087b86 | compiler | cori.slurm.knl_debug | ['mpi'] | intel/19.1.2.254 |
↪ /global/u1/s/siddiq90/github/buildtest/var/tests/cori.slurm.knl_debug/laplace_mpi/
↪ laplace_mpi/0/laplace_mpi_build.sh

+-----+
| Stage: Running Test |
+-----+

[laplace_mpi] JobID: 43308598 dispatched to scheduler
name      | id      | executor          | status | returncode
+-----+-----+-----+-----+-----+
laplace_mpi | a6087b86 | cori.slurm.knl_debug | N/A    | -1

```

(continues on next page)

(continued from previous page)

Polling Jobs in 30 seconds

Job Queue: [43308598]

Pending Jobs

name	executor	jobID	jobstate
laplace_mpi	cori.slurm.knl_debug	43308598	COMPLETED

Polling Jobs in 30 seconds

Job Queue: []

Completed Jobs

name	executor	jobID	jobstate
laplace_mpi	cori.slurm.knl_debug	43308598	COMPLETED

Stage: Final Results after Polling all Jobs

name	id	executor	status	returncode
laplace_mpi	a6087b86	cori.slurm.knl_debug	PASS	0

Stage: Test Summary

Passed Tests: 1/1 Percentage: 100.000%

Failed Tests: 0/1 Percentage: 0.000%

Writing Logfile to: /tmp/buildtest_wgptyp8v.log

A copy of logfile can be found at \$BUILDTEST_ROOT/buildtest.log - /global/homes/s/
 ↳siddiq90/github/buildtest/buildtest.log

The generated test is as follows, note that buildtest will insert the #SBATCH directives at the top of script, and module load are done before module swap command.

```

1 #!/bin/bash
2 #SBATCH -N 1
3 #SBATCH -n 4
4 #SBATCH --job-name=laplace_mpi
5 #SBATCH --output=laplace_mpi.out
6 #SBATCH --error=laplace_mpi.err
7 _EXEC=laplace_mpi.c.exe
8 module load impi/2020
9 module swap intel intel/19.1.2.254
10 mpiicc -O3 -o $_EXEC /global/u1/s/siddiq90/github/buildtest-cori/buildspecs/apps/mpi/src/
   ↳ laplace_mpi.c
11 srun -n 4 $_EXEC

```

The master script that buildtest will invoke is the following, notice that our generated script (shown above) is invoked via *sbatch* with its options. The options *sbatch -q debug --clusters=cori -C knl,quad,cache* was inserted by our executor configuration. We add the *--parsable* option for Slurm jobs in order to get the JobID when this script is invoked so that buildtest can poll the job.

```

1 #!/bin/bash
2 source /global/u1/s/siddiq90/github/buildtest/var/executor/cori.slurm.knl_debug/before_
   ↳ script.sh
3 sbatch --parsable -q debug --clusters=cori -C knl,quad,cache /global/u1/s/siddiq90/
   ↳ github/buildtest/var/tests/cori.slurm.knl_debug/laplace_mpi/laplace_mpi/0/stage/
   ↳ laplace_mpi.sh
4 returncode=$?
5 exit $returncode

```

Pre/Post sections for build and run section

The compiler schema comes with *pre_build*, *post_build*, *pre_run* and *post_run* fields where you can insert commands before and after build or run section. The **build** section is where we compile code, and **run** section is where compiled binary is executed.

Shown below is an example buildspec with pre/post section.

```

version: "1.0"
buildspecs:
  pre_post_build_run:
    type: compiler
    description: example using pre_build, post_build, pre_run, post_run example
    executor: generic.local.bash
    tags: [tutorials, compile]
    source: "src/hello.c"
    compilers:
      name: ["^(builtin_gcc)$"]
      default:
        gcc:
          cflags: -Wall
        all:
          pre_build: |
            echo "This is a pre-build section"
            gcc --version

```

(continues on next page)

(continued from previous page)

```

post_build: |
    echo "This is post-build section"
pre_run: |
    echo "This is pre-run section"
    export FOO=BAR
post_run: |
    echo "This is post-run section"

```

The format of the test structure is as follows.

```

#!/{shebang path} -- defaults to #!/bin/bash depends on executor name (local.bash, local.
↳ sh)
{job directives} -- sbatch or bsub field
{environment variables} -- env field
{variable declaration} -- vars field
{module commands} -- modules field

{pre build commands} -- pre_build field
{compile program} -- build field
{post build commands} -- post_build field

{pre run commands} -- pre_run field
{run executable} -- run field
{post run commands} -- post_run field

```

The generated test for this builds spec is the following:

```

#!/bin/bash

# name of executable
_EXECUTABLE=hello.c.exe
### START OF PRE BUILD SECTION ###
echo "This is a pre-build section"
gcc --version

### END OF PRE BUILD SECTION ###

# Compilation Line
gcc -Wall -o $_EXECUTABLE /Users/siddiq90/Documents/GitHubDesktop/buildtest/tutorials/
↳ compilers/src/hello.c

### START OF POST BUILD SECTION ###
echo "This is post-build section"

### END OF POST BUILD SECTION ###

### START OF PRE RUN SECTION ###
echo "This is pre-run section"

```

(continues on next page)

(continued from previous page)

```
export FOO=BAR

### END OF PRE RUN SECTION ###

# Run executable
./$_EXEC

### START OF POST RUN SECTION ###
echo "This is post-run section"

### END OF POST RUN SECTION ###
```

3.6.4 Spack Schema

Note: This feature is in active development.

buildtest can generate tests for the `spack` package manager which can be used if you want to install or test packages as part of a repeatable process. You must set `type: spack` property in buildspect to use the spack schema for validating the buildspect test. Currently, we have `spack-v1.0.schema.json` JSON schema that defines the structure of how tests are to be written in buildspect. Shown below is the schema header. The **required** properties are `type`, `executor` and `spack`.

```
"$id": "spack-v1.0.schema.json",
"$schema": "http://json-schema.org/draft-07/schema#",
"title": "spack schema version 1.0",
"description": "The spack schema is referenced using ``type: spack`` which is used for
↳ generating tests using spack package manager",
"type": "object",
"required": [
  "type",
  "executor",
  "spack"
],
```

Install Specs

Let's start off with a simple example where we create a test that can `spack install zlib`. Shown below is a test named `install_zlib`. The `spack` keyword is a JSON object, in this test we define the root of spack using the `root` keyword which informs buildtest where spack is located. buildtest will automatically check the path and source the startup script. The `install` field is a JSON object that contains a `specs` property which is a list of strings types that are name of spack packages to install. Each item in the `specs` property will be added as a separate `spack install` command.

The schema is designed to mimic spack commands which will be clear with more examples.

```
version: "1.0"
buildspecs:
```

(continues on next page)

(continued from previous page)

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

If you build this test and inspect the generated script, buildtest will source spack startup script - **source \$SPACK_ROOT/share/spack/setup-env.sh** based on the `root` property. In this example, we have spack cloned in **\$HOME/spack** which is **/Users/siddiq90/spack** and buildtest will find the startup script which is in **share/spack/setup-env.sh**.

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

Spack Environment

buildtest can generate scripts to make use of **spack environments** which can be useful if you want to install or test specs in an isolated environment.

Currently, we can create spack environment (**spack env create**) via name, directory and manifest file (**spack.yaml**, **spack.lock**) and pass any options to **spack env create** command. Furthermore, we can activate existing spack environment via name or directory using **spack env activate** and pass options to the command. buildtest can remove spack environments automatically before creating spack environment or one can explicitly specify by name.

Activate Spack Environment

In this next example, we will activate an existing environment **m4** and add spec for **m4** and concretize the spack environment. The `env` is an object that mimics the **spack env** command. The `activate` field maps to **spack env activate** command. The **name** property is of type: `string` which is name of spack environment you want to activate. The `specs` property in **env** section maps to **spack add <specs** instead of **spack install**.

The property `concretize: true` will run **spack concretize** command that is only available as part of the `env` object since this command is only applicable in spack environments.

```
version: "1.0"
buildspecs:
  concretize_m4_in_spack_env:
    type: spack
    executor: generic.local.sh
    description: "Concretize m4 in a spack environment named m4"
    tags: [spack]
    spack:
      root: $HOME/spack
      env:
        specs:
          - 'm4'
```

(continues on next page)

(continued from previous page)

```
activate:
  name: m4
  concretize: true
```

If we build this test and inspect the generated test we see that spack will activate a spack environment **m4**, add specs in spack environment via `spack add m4` and concretize the environment. The `concretize` is a boolean type, if its `true` we will run `spack concretize -f`, if its `false` this command will not be in script.

```
#!/bin/bash
source /Users/siddiq90/spack/share/spack/setup-env.sh
spack env activate m4
spack add m4
spack concretize -f
```

If we inspect the output file we see that m4 was concretized in the spack environment.

```
==> Package m4 was already added to m4
==> Concretized m4
[+] volmsbn m4@1.4.19%apple-clang@11.0.3+sigsegv arch=darwin-bigsur-skylake
[+] bc6kuc4 ^libsigsegv@2.13%apple-clang@11.0.3 arch=darwin-bigsur-skylake
```

Create a Spack Environment by name

In this next example, we will create a spack environment named `m4_zlib` that will install `m4` and `zlib` spec. The `create` field is a JSON object that maps to `spack env create` command which can pass some arguments in the form of key/value pairs. The `name` property in `create` section is used to create a spack environment by name.

The `compiler_find: true` is a boolean that determines if we need to find compilers in spack via `spack compiler find`. This can be useful if you need to find compilers so spack can install specs with a preferred compiler otherwise spack may have issues concretizing or install specs. buildtest will run **spack compiler find** after sourcing spack.

Note: The `compiler_find` option may not be useful if your compilers are already defined in one of your configuration scopes or `spack.yaml` that is part of your spack environment.

The `option` field can pass any command line arguments to `spack install` command and this field is available for other properties.

```
version: "1.0"
buildspecs:
  install_m4_zlib_in_spack_env:
    type: spack
    executor: generic.local.sh
    description: "Install m4 and zlib in a spack environment named m4_zlib"
    tags: [spack]
    spack:
      root: $HOME/spack
      compiler_find: true
      env:
        create:
          name: 'm4_zlib'
        specs:
```

(continues on next page)

(continued from previous page)

```

- 'm4'
- 'zlib'
activate:
  name: m4_zlib
  concretize: true
install:
  option: '--keep-prefix'

```

If we build this test and see generated test we see that buildtest will create a spack environment *m4_zlib* and activate the environment, add **m4** and **zlib**, concretize the environment and install the specs.

```

#!/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

```

Now let's examine the output of this test, shown below is the summary of this test, as you can see we have successfully installed **m4** and **zlib** in a spack environment *m4_zlib*.

```

==> Found no new compilers
==> Compilers are defined in the following files:
    /Users/siddiq90/.spack/darwin/compilers.yaml
==> Updating view at /Users/siddiq90/spack/var/spack/environments/m4_zlib/.spack-env/view
==> Created environment 'm4_zlib' in /Users/siddiq90/spack/var/spack/environments/m4_zlib
==> You can activate this environment with:
==>   spack env activate m4_zlib
==> Adding m4 to environment m4_zlib
==> Adding zlib to environment m4_zlib
==> Concretized m4
[+] volmsbn m4@1.4.19%apple-clang@11.0.3+sigsegv arch=darwin-bigsur-skylake
[+] bc6kuc4 ^libsigsegv@2.13%apple-clang@11.0.3 arch=darwin-bigsur-skylake
==> Concretized zlib
- 2hw3hzd zlib@1.2.11%apple-clang@11.0.3+optimize+pic+shared arch=darwin-bigsur-
↳skylake
==> Updating view at /Users/siddiq90/spack/var/spack/environments/m4_zlib/.spack-env/view
==> Installing environment m4_zlib
==> Installing zlib-1.2.11-2hw3hzdfy7e2ndzojgqoq472m5flsloj
==> No binary for zlib-1.2.11-2hw3hzdfy7e2ndzojgqoq472m5flsloj found: installing from
↳source
==> Fetching https://mirror.spack.io/_source-cache/archive/c3/
↳c3e5e9fdd5004dcb542feda5ee4f0ff0744628baf8ed2dd5d66f8ca1197cb1a1.tar.gz
==> No patches needed for zlib
==> zlib: Executing phase: 'install'
==> zlib: Successfully installed zlib-1.2.11-2hw3hzdfy7e2ndzojgqoq472m5flsloj
    Fetch: 0.84s. Build: 6.98s. Total: 7.82s.
[+] /Users/siddiq90/spack/opt/spack/darwin-bigsur-skylake/apple-clang-11.0.3/zlib-1.2.11-
↳2hw3hzdfy7e2ndzojgqoq472m5flsloj
==> Updating view at /Users/siddiq90/spack/var/spack/environments/m4_zlib/.spack-env/view

```

Creating Spack Environment from Directory

We can create spack environment from a directory using the `dir` property that is available as part of `create` and `activate` field. In this next example we create a spack environment in our `$HOME` directory and concretize `m4` in the spack environment

```
version: "1.0"
buildspecs:
  spack_env_directory:
    type: spack
    executor: generic.local.sh
    description: "Concretize m4 in a spack environment named m4"
    tags: [spack]
    spack:
      root: $HOME/spack
      env:
        create:
          dir: $HOME/spack-envs/m4
        activate:
          dir: $HOME/spack-envs/m4
        specs:
          - 'm4'
      concretize: true
```

When creating spack environment using directory, buildtest will automatically add the `-d` option which is required when creating spack environments. However, one can also pass this using the `option` field. Shown below is the generated script for the above test.

```
#!/bin/bash
source /Users/siddiq90/spack/share/spack/setup-env.sh
spack env create -d /Users/siddiq90/spack-envs/m4
spack env activate -d /Users/siddiq90/spack-envs/m4
spack add m4
spack concretize -f
```

buildtest will create environment first followed by activating the spack environment.

Create Spack Environment from Manifest File (`spack.yaml`, `spack.lock`)

Spack can create environments from `spack.yaml` or `spack.lock` which can be used if you have a spack configuration that works for your system and want to write a buildspec. While creating a spack environment, you can use the `manifest` property to specify path to your `spack.yaml` or `spack.lock`.

Note: buildtest will not enforce that manifest names be `spack.yaml` or `spack.lock` since spack allows one to create spack environment from arbitrary name so long as it is a valid spack configuration.

Shown below is an example buildspec that generates a test from a manifest file. The `manifest` property is of type: `string` and this is only available as part of `create` property.

```
version: "1.0"
buildspecs:
```

(continues on next page)

(continued from previous page)

```

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

```

If we build this test and inspect the generated script we see `spack env create` command will create an environment **manifest_example** using the manifest file that we provided.

```

#!/bin/bash
source /Users/siddiq90/spack/share/spack/setup-env.sh
spack env create manifest_example /Users/siddiq90/Documents/GitHubDesktop/buildtest/
↪tutorials/spack/example/spack.yaml
spack env activate manifest_example
spack concretize -f

```

Removing Spack Environments

buildtest can remove spack environments which can be used if you are periodically running the same test where one is creating the same environment. buildtest can automatically remove spack environment using the property `remove_environment` which will remove the environment before creating it with same name. This field is part of the `create` field and only works if one is creating spack environments by name.

Alternately, buildtest provides the `rm` field which can be used for removing environment explicitly. In the `rm` field, the `name` is a required field which is the name of the spack environment to remove. The `name` field is of type: `string`. Shown below are two example tests where we remove spack environment using the **remove_environment** and **rm** field.

```

version: "1.0"
buildspecs:
  remove_environment_automatically:
    type: spack
    executor: generic.local.sh
    description: "remove spack environment automatically before creating a new_
↪environment"
    tags: [spack]
    spack:
      root: $HOME/spack
      env:
        create:
          remove_environment: true
          name: remove_environment
        activate:

```

(continues on next page)

(continued from previous page)

```

        name: remove_environment
    specs:
        - 'bzip2'
    concretize: true

remove_environment_explicit:
    type: spack
    executor: generic.local.sh
    description: "remove spack environment explicitly using the 'rm' property"
    tags: [spack]
    spack:
        root: $HOME/spack
        env:
            rm:
                name: dummy
            create:
                name: dummy
            activate:
                name: dummy
        specs:
            - 'bzip2'
        concretize: true

```

If we look at the generated test, we notice that spack will remove environments names: **remove_environment**, **dummy**.

```

#!/bin/bash
source /Users/siddiq90/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

```

```

#!/bin/bash
source /Users/siddiq90/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

```

Pre and Post Commands

The spack schema supports ability to write arbitrary shell script content using the `pre_cmds` and `post_cmds` field that are of type: `string` and buildtest will insert the content into the test exactly as it is defined by these two fields.

In this next example, we will test an installation of *zlib* by cloning spack from upstream and use `pre_cmds` field to specify where we will clone spack. In this example, we will clone spack under `/tmp`. Since we don't have a valid root of spack since test hasn't been run, we can ignore check for spack paths by specifying `verify_spack: false` which informs buildtest to skip spack path check. Generally, buildtest will raise an exception if path specified by `root` is invalid and if `$SPACK_ROOT/share/spack/setup-env.sh` doesn't exist since this is the file that must be sourced.

The `pre_cmds` are shell commands that are run before sourcing `spack`, whereas the `post_cmds` are run at the very end of the script. In the `post_cmds`, we will `spack find` that will be run after `spack install`. We remove `spack root` (`$SPACK_ROOT`) so that this test can be rerun again.

```
version: "1.0"
buildspecs:
  run_pre_post_commands:
    type: spack
    executor: generic.local.sh
    description: "Install zlib"
    tags: [spack]
    pre_cmds: |
      cd /tmp
      git clone https://github.com/spack/spack
    spack:
      root: /tmp/spack
      verify_spack: false
      install:
        specs: ['zlib']
    post_cmds: |
      spack find
      rm -rf $SPACK_ROOT
```

If we build this test and inspect the generated script we see the following

```
#!/bin/bash

##### START OF PRE COMMANDS #####
cd /tmp
git clone https://github.com/spack/spack

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

source /private/tmp/spack/share/spack/setup-env.sh
spack install  zlib

##### START OF POST COMMANDS #####
spack find
rm -rf $SPACK_ROOT
##### END OF POST COMMANDS #####
```

If we inspect the output, we see that `zlib` is installed as shown in output from `spack find`

```
==> Installing zlib-1.2.11-2hw3hzdfy7e2ndzojgqoq472m5flsloj
==> No binary for zlib-1.2.11-2hw3hzdfy7e2ndzojgqoq472m5flsloj found: installing from
↳ source
==> Fetching https://mirror.spack.io/_source-cache/archive/c3/
↳ c3e5e9fdd5004dcb542feda5ee4f0ff0744628baf8ed2dd5d66f8ca1197cb1a1.tar.gz
==> No patches needed for zlib
==> zlib: Executing phase: 'install'
==> zlib: Successfully installed zlib-1.2.11-2hw3hzdfy7e2ndzojgqoq472m5flsloj
```

(continues on next page)

(continued from previous page)

```

Fetch: 0.50s. Build: 5.90s. Total: 6.40s.
[+] /private/tmp/spack/opt/spack/darwin-bigsur-skylake/apple-clang-11.0.3/zlib-1.2.11-
↪2hw3hzdfy7e2ndzojgqoq472m5flsloj
-- darwin-bigsur-skylake / apple-clang@11.0.3 -----
zlib@1.2.11

```

Specifying Scheduler Directives

The spack schema supports all of the *scheduler directives* such as `sbatch`, `bsub`, `pbs`, `cobalt`, and `batch` property in the buildspect.

The directives are applied at top of script. Shown below is a toy example that will define directives using **sbatch** and **batch** property. Note, this test won't submit job to scheduler since we are not using the a slurm executor.

```

version: "1.0"
buildspecs:
  spack_sbatch_example:
    type: spack
    executor: generic.local.sh
    description: "sbatch directives can be defined in spack schema"
    tags: [spack]
    sbatch: ["-N 1"]
    batch:
      cpucount: "8"
      timelimit: "30"
    spack:
      root: $HOME/spack
      env:
        specs:
          - 'm4'
        activate:
          name: m4
        concretize: true

```

buildtest will generate the shell script with the job directives and set the name, output and error files based on name of test. If we build this test, and inspect the generated test we see that **#SBATCH** directives are written based on the **sbatch** and **batch** field.

```

#!/bin/bash

##### START OF SCHEDULER DIRECTIVES #####
#SBATCH -N 1
#SBATCH --ntasks=8
#SBATCH --time=30
#SBATCH --job-name=spack_sbatch_example
#SBATCH --output=spack_sbatch_example.out
#SBATCH --error=spack_sbatch_example.err
##### END OF SCHEDULER DIRECTIVES #####

source /Users/siddiq90/spack/share/spack/setup-env.sh
spack env activate m4

```

(continues on next page)

(continued from previous page)

```
spack add m4
spack concretize -f
```

Configuring Spack Mirrors

We can add [mirrors](#) in the spack instance or spack environment using the `mirror` property which is available in the `spack` and `env` section. If the `mirror` property is part of the `env` section, the mirror will be added to spack environment. The `mirror` is an object that expects a Key/Value pair where the key is the name of mirror and value is location of the spack mirror.

In this next example, we will define a mirror name `e4s` that points to <https://cache.e4s.io> as the mirror location. Internally, this translates to `spack mirror add e4s https://cache.e4s.io` command.

```
version: "1.0"
buildspecs:
  add_mirror:
    type: spack
    executor: generic.local.sh
    description: Declare spack mirror
    tags: [spack]
    spack:
      root: $HOME/spack
      mirror:
        e4s: https://cache.e4s.io
    post_cmds: |
      spack mirror list

  add_mirror_in_spack_env:
    type: spack
    executor: generic.local.sh
    description: Declare spack mirror in spack environment
    tags: [spack]
    spack:
      root: $HOME/spack
      env:
        create:
          name: spack_mirror
        activate:
          name: spack_mirror
        mirror:
          e4s: https://cache.e4s.io
    post_cmds: |
      spack mirror list
```

If we look at the generated script for both tests, we see that mirror is added for both tests. Note that one can have mirrors defined in their `spack.yaml` or one of the [configuration scopes](#) defined by spack.

```
#!/bin/bash
source /Users/siddiq90/spack/share/spack/setup-env.sh
spack mirror add e4s https://cache.e4s.io
```

(continues on next page)

(continued from previous page)

```
##### START OF POST COMMANDS #####
spack mirror list
##### END OF POST COMMANDS #####
```

```
#!/bin/bash
source /Users/siddiq90/spack/share/spack/setup-env.sh
spack env create spack_mirror
spack env activate spack_mirror
spack mirror add e4s https://cache.e4s.io
```

```
##### START OF POST COMMANDS #####
spack mirror list
##### END OF POST COMMANDS #####
```

Spack Test

buildtest can run tests using `spack test run` that can be used for testing installed specs with tests provided by spack. In order to run tests, you need to declare the `test` section which is of type: `object` in JSON and `run` is a required property. The `run` section maps to `spack test run` that is responsible for running tests for a list of specs that are specified using the `specs` property.

Upon running the tests, we can retrieve results using `spack test results` which is configured using the `results` property. The `results` property expects one to specify the `specs` or `suite` or both in order to retrieve results.

The `suite` property is used to retrieve test results based on suite name, whereas `specs` property can be used to retrieve based on spec format. Both properties are a list of string types.

In example below we install `bzip2` and run the test using `spack test run bzip2`.

```
version: "1.0"
buildspecs:
  spack_test:
    type: spack
    executor: generic.local.sh
    description: "Install bzip2 and run spack test and report results"
    tags: [spack]
    pre_cmds: |
      cd /tmp
      git clone https://github.com/spack/spack
    spack:
      root: /tmp/spack
      verify_spack: false
      install:
        specs: ['bzip2']
      test:
        run:
          specs: ['bzip2']
        results:
          suite: ['bzip2']
```

(continues on next page)

(continued from previous page)

```
post_cmds: |
    spack find
    rm -rf $SPACK_ROOT
```

If we look at the generated test, buildtest will automatically set `--alias` option to define name of suite, otherwise spack will generate a random text for suite name which you won't know at time of writing test that is required by `spack test results` to fetch the results.

```
#!/bin/bash

##### START OF PRE COMMANDS #####
cd /tmp
git clone https://github.com/spack/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 #####
```

Shown below is the example output of this test.

```
==> libiconv: Executing phase: 'configure'
==> libiconv: Executing phase: 'build'
==> libiconv: Executing phase: 'install'
==> libiconv: Successfully installed libiconv-1.16-xgemfyqy3gsdz3lk7wy3ejudfaksja4x
Fetch: 1.54s. Build: 33.03s. Total: 34.57s.
[+] /private/tmp/spack/opt/spack/darwin-bigsur-skylake/apple-clang-11.0.3/libiconv-1.16-
↳xgemfyqy3gsdz3lk7wy3ejudfaksja4x
==> Installing diffutils-3.7-3dfrh6li733xxcenwyjhwyta7xkh3udq
==> No binary for diffutils-3.7-3dfrh6li733xxcenwyjhwyta7xkh3udq found: installing from
↳source
==> Fetching https://mirror.spack.io/_source-cache/archive/b3/
↳b3a7a6221c3dc916085f0d205abf6b8e1ba443d4dd965118da364a1dc1cb3a26.tar.xz
==> No patches needed for diffutils
==> diffutils: Executing phase: 'autoreconf'
==> diffutils: Executing phase: 'configure'
==> diffutils: Executing phase: 'build'
==> diffutils: Executing phase: 'install'
==> diffutils: Successfully installed diffutils-3.7-3dfrh6li733xxcenwyjhwyta7xkh3udq
Fetch: 1.32s. Build: 52.35s. Total: 53.67s.
[+] /private/tmp/spack/opt/spack/darwin-bigsur-skylake/apple-clang-11.0.3/diffutils-3.7-
↳3dfrh6li733xxcenwyjhwyta7xkh3udq
```

(continues on next page)

(continued from previous page)

```

==> Installing bzip2-1.0.8-avjwvsoaivuflugopwk4ap7rffhejxzu
==> No binary for bzip2-1.0.8-avjwvsoaivuflugopwk4ap7rffhejxzu found: installing from
↳ source
==> Fetching https://mirror.spack.io/_source-cache/archive/ab/
↳ ab5a03176ee106d3f0fa90e381da478ddae405918153cca248e682cd0c4a2269.tar.gz
==> Ran patch() for bzip2
==> bzip2: Executing phase: 'install'
==> bzip2: Successfully installed bzip2-1.0.8-avjwvsoaivuflugopwk4ap7rffhejxzu
    Fetch: 1.42s. Build: 1.84s. Total: 3.26s.
[+] /private/tmp/spack/opt/spack/darwin-bigsur-skylake/apple-clang-11.0.3/bzip2-1.0.8-
↳ avjwvsoaivuflugopwk4ap7rffhejxzu
==> Spack test bzip2
==> Testing package bzip2-1.0.8-avjwvso
==> Results for test suite 'bzip2':
==> bzip2-1.0.8-avjwvso PASSED
-- darwin-bigsur-skylake / apple-clang@11.0.3 -----
bzip2@1.0.8
diffutils@3.7
libiconv@1.16

```

We can search for test results using the spec format instead of suite name. In the `results` property we can use `specs` field instead of `suite` property to specify a list of spec names to run. In spack, you can retrieve the results using `spack test results -- <spec>`, note that double dash `--` is in front of spec name. We can pass options to `spack test results` using the `option` property which is available for `results` and `run` property. Currently, spack will write test results in `$HOME/.spack/tests` and we can use `spack test remove` to clear all test results. This can be done in `buildspec` using the `remove_tests` field which is a boolean. If this is set to **True** buildtest will run `spack test remove -y` to remove all test suites before running the tests.

```

version: "1.0"
buildspecs:
  spack_test_results_specs_format:
    type: spack
    executor: generic.local.sh
    description: "Run spack test results with spec format"
    tags: [spack]
    pre_cmds: |
      cd /tmp
      git clone https://github.com/spack/spack
  spack:
    root: /tmp/spack
    verify_spack: false
    install:
      specs: ['bzip2']
    test:
      remove_tests: true
      run:
        specs: ['bzip2']
      results:
        option: '-l'
        specs: ['bzip2']
    post_cmds: |
      spack find

```

(continues on next page)

(continued from previous page)

```
rm -rf $SPACK_ROOT
```

In the generated test, we see that buildtest will remove all testsuites using `spack test remove -y` and query results based on spec format. The options are passed into `spack test results` based on the option field specified under the `results` section.

```
#!/bin/bash

##### START OF PRE COMMANDS #####
cd /tmp
git clone https://github.com/spack/spack

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

source /private/tmp/spack/share/spack/setup-env.sh
spack install bzip2
spack test remove -y
spack test run --alias bzip2 bzip2
spack test results -l -- bzip2

##### START OF POST COMMANDS #####
spack find
rm -rf $SPACK_ROOT
##### END OF POST COMMANDS #####
```

3.6.5 Batch Scheduler Support

Slurm

buildtest can submit jobs to [Slurm](#) assuming you have slurm executors defined in your configuration file. The `SlurmExecutor` class is responsible for managing slurm jobs which will perform the following action

1. Check slurm binary `sbatch` and `sacct`.
2. Dispatch Job and acquire job ID using `sacct`.
3. Poll all slurm jobs until all have finished
4. Gather Job results once job is complete via `sacct`.

buildtest will dispatch slurm jobs and poll all jobs until all jobs are complete. If job is in **PENDING** or **RUNNING** state, then buildtest will keep polling at a set interval defined by `pollinterval` setting in buildtest. Once job is not in **PENDING** or **RUNNING** stage, buildtest will gather job results and wait until all jobs have finished.

In this example we have a slurm executor `cori.slurm.knl_debug`, in addition we can specify **#SBATCH** directives using `sbatch` field. The `sbatch` field is a list of string types, buildtest will insert **#SBATCH** directive in front of each value.

Shown below is an example buildspect

```

1 version: "1.0"
2 buildspecs:
3   slurm_metadata:
4     description: Get metadata from compute node when submitting job
5     type: script
6     executor: cori.slurm.knl_debug
7     tags: [jobs]
8     sbatch:
9       - "-t 00:05"
10      - "-N 1"
11   run: |
12     export SLURM_JOB_NAME="firstjob"
13     echo "jobname:" $SLURM_JOB_NAME
14     echo "slurmdb host:" $SLURMD_NODENAME
15     echo "pid:" $SLURM_TASK_PID
16     echo "submit host:" $SLURM_SUBMIT_HOST
17     echo "nodeid:" $SLURM_NODEID
18     echo "partition:" $SLURM_JOB_PARTITION

```

buildtest will add the #SBATCH directives at top of script followed by content in the run section. Shown below is the example test content. Every slurm will insert #SBATCH --job-name, #SBATCH --output and #SBATCH --error line which is determined by the name of the test.

```

1 #!/bin/bash
2 #SBATCH -t 00:05
3 #SBATCH -N 1
4 #SBATCH --job-name=slurm_metadata
5 #SBATCH --output=slurm_metadata.out
6 #SBATCH --error=slurm_metadata.err
7 export SLURM_JOB_NAME="firstjob"
8 echo "jobname:" $SLURM_JOB_NAME
9 echo "slurmdb host:" $SLURMD_NODENAME
10 echo "pid:" $SLURM_TASK_PID
11 echo "submit host:" $SLURM_SUBMIT_HOST
12 echo "nodeid:" $SLURM_NODEID
13 echo "partition:" $SLURM_JOB_PARTITION

```

The cori.slurm.knl_debug executor in our configuration file is defined as follows

```

1 system:
2   cori:
3     executors:
4       slurm:
5         knl_debug:
6           qos: debug
7           cluster: cori
8           options:
9             - -C knl,quad,cache
10          description: debug queue on KNL partition

```

With this setting, any buildspec test that use cori.slurm.knl_debug executor will result in the following launch option: sbatch --qos debug --clusters=cori -C knl,quad,cache </path/to/script.sh>.

Unlike the LocalExecutor, the **Run Stage**, will dispatch the slurm job and poll until job is completed. Once job is

complete, it will gather the results and terminate. In Run Stage, buildtest will mark test status as N/A because job is submitted to scheduler and pending in queue. In order to get job result, we need to wait until job is complete then we gather results and determine test state. buildtest keeps track of all buildsspecs, testscripts to be run and their results. A test using LocalExecutor will run test in **Run Stage** and returncode will be retrieved and status can be calculated immediately. For Slurm Jobs, buildtest dispatches the job and process next job. buildtest will show output of all tests after **Polling Stage** with test results of all tests. A slurm job with exit code 0 will be marked with status PASS.

Shown below is an example build for this test

```
$ buildtest build -b buildspecs/jobs/metadata.yml

User: siddiq90
Hostname: cori02
Platform: Linux
Current Time: 2021/06/11 09:24:44
buildtest path: /global/homes/s/siddiq90/github/buildtest/bin/buildtest
buildtest version: 0.9.5
python path: /global/homes/s/siddiq90/.conda/envs/buildtest/bin/python
python version: 3.8.8
Test Directory: /global/u1/s/siddiq90/github/buildtest/var/tests
Configuration File: /global/u1/s/siddiq90/.buildtest/config.yml
Command: /global/homes/s/siddiq90/github/buildtest/bin/buildtest build -b buildspecs/
↪ jobs/metadata.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

+-----+
| Discovered Buildsspecs |
+=====+
| /global/u1/s/siddiq90/github/buildtest-cori/buildspecs/jobs/metadata.yml |
+-----+
Discovered Buildsspecs: 1
Excluded Buildsspecs: 0
Detected Buildsspecs after exclusion: 1

+-----+
| Stage: Parsing Buildsspecs |
+-----+

  schemafilename | validstate | buildspec
-----+-----+-----
↪ script-v1.0.schema.json | True | /global/u1/s/siddiq90/github/buildtest-cori/
↪ buildspecs/jobs/metadata.yml

name            description
-----
slurm_metadata  Get metadata from compute node when submitting job
```

(continues on next page)

(continued from previous page)

```

+-----+
| Stage: Building Test |
+-----+

name          | id          | type   | executor          | tags      | testpath
+-----+-----+-----+-----+-----+-----+
↪ -----
↪ -----
slurm_metadata | 722b3291 | script | cori.slurm.knl_debug | ['jobs'] | /global/u1/s/
↪ siddiq90/github/buildtest/var/tests/cori.slurm.knl_debug/metadata/slurm_metadata/0/
↪ slurm_metadata_build.sh

+-----+
| Stage: Running Test |
+-----+

[slurm_metadata] JobID: 43308838 dispatched to scheduler
name          | id          | executor          | status  | returncode
+-----+-----+-----+-----+-----+
slurm_metadata | 722b3291 | cori.slurm.knl_debug | N/A     | -1

Polling Jobs in 30 seconds

Job Queue: [43308838]

Pending Jobs

+-----+-----+-----+-----+
| name          | executor          | jobID  | jobstate |
+-----+-----+-----+-----+
| slurm_metadata | cori.slurm.knl_debug | 43308838 | COMPLETED |
+-----+-----+-----+-----+

Polling Jobs in 30 seconds

Job Queue: []

Completed Jobs

+-----+-----+-----+-----+
| name          | executor          | jobID  | jobstate |
+-----+-----+-----+-----+

```

(continues on next page)

(continued from previous page)

```

| slurm_metadata | cori.slurm.knl_debug | 43308838 | COMPLETED |
+-----+-----+-----+-----+-----+
|
+-----+
| Stage: Final Results after Polling all Jobs |
+-----+

name          | id          | executor          | status  | returncode
+-----+-----+-----+-----+-----+
slurm_metadata | 722b3291   | cori.slurm.knl_debug | PASS    | 0

+-----+
| Stage: Test Summary |
+-----+

Passed Tests: 1/1 Percentage: 100.000%
Failed Tests: 0/1 Percentage: 0.000%

Writing Logfile to: /tmp/buildtest_u8ehladd.log
A copy of logfile can be found at $BUILDTEST_ROOT/buildtest.log - /global/homes/s/
↪siddiq90/github/buildtest/buildtest.log

```

The **SlurmExecutor** class is responsible for processing slurm job that may include: dispatch, poll, gather, or cancel job. The SlurmExecutor will gather job metrics via **sacct** using the following format fields: **Account**, **AllocNodes**, **AllocTRES**, **ConsumedEnergyRaw**, **CPUTimeRaw**, **Elapsed**, **End**, **ExitCode**, **JobID**, **JobName**, **NCPUS**, **NNodes**, **QOS**, **ReqGRES**, **ReqMem**, **ReqNodes**, **ReqTRES**, **Start**, **State**, **Submit**, **UID**, **User**, **WorkDir**. For a complete list of format fields see `sacct -e`. For now, we support only these fields of interest for reporting purpose.

buildtest can check status based on Slurm Job State, this is defined by **State** field in `sacct`. In next example, we introduce field `slurm_job_state` which is part of `status` field. This field expects one of the following values: `[COMPLETED, FAILED, OUT_OF_MEMORY, TIMEOUT]` This is an example of simulating fail job by expecting a return code of 1 with job state of `FAILED`.

```

1 version: "1.0"
2 buildspecs:
3   wall_timeout:
4     type: script
5     executor: cori.slurm.debug
6     sbatch: [ "-t 2", "-C haswell", "-n 1" ]
7     run: sleep 300
8     status:
9       slurm_job_state: "TIMEOUT"

```

If we run this test, buildtest will mark this test as `PASS` because the slurm job state matches with expected result defined by field `slurm_job_state`. This job will be `TIMEOUT` because we requested 2 mins while this job will sleep 300sec (5min).

Completed Jobs

```

+-----+-----+-----+-----+-----+

```

(continues on next page)

(continued from previous page)

name	executor	jobID	jobstate
wall_timeout	cori.slurm.haswell_debug	43309265	TIMEOUT

Stage: Final Results after Polling all Jobs

name	id	executor	status	returncode
wall_timeout	3b43850c	cori.slurm.haswell_debug	PASS	0

Stage: Test Summary

Passed Tests: 1/1 Percentage: 100.000%

Failed Tests: 0/1 Percentage: 0.000%

Writing Logfile to: /tmp/buildtest_k6h246yx.log

A copy of logfile can be found at \$BUILDTEST_ROOT/buildtest.log - /global/homes/s/siddiq90/github/buildtest/buildtest.log

If you examine the logfile `buildtest.log` you will see an entry of `sacct` command run to gather results followed by list of field and value output:

```
2021-06-11 09:52:27,826 [slurm.py:292 - poll() ] - [DEBUG] Querying JobID: '43309265'
↪ Job State by running: 'sacct -j 43309265 -o State -n -X -P --clusters=cori'
2021-06-11 09:52:27,826 [slurm.py:296 - poll() ] - [DEBUG] JobID: '43309265' job
↪ state:TIMEOUT
```

LSF

buildtest can support job submission to **IBM Spectrum LSF** if you have defined LSF executors in your configuration file.

The `bsub` property can be used to specify **#BSUB** directive into job script. This example will use the executor `ascent.lsf.batch` executor that was defined in buildtest configuration.

```
1 version: "1.0"
2 buildspecs:
3   hostname:
4     type: script
5     executor: ascent.lsf.batch
6     bsub: [ "-W 10", "-nnodes 1" ]
7
8   run: jsrun hostname
```

The `LSFExecutor` poll jobs and retrieve job state using `bjobs -noheader -o 'stat' <JOBID>`. The `LSFExecutor` will poll job so long as they are in **PEND** or **RUN** state. Once job is not in any of the two states, `LSFExecutor` will gather

job results. buildtest will retrieve the following format fields using bjobs: **job_name**, **stat**, **user**, **user_group**, **queue**, **proj_name**, **pids**, **exit_code**, **from_host**, **exec_host**, **submit_time**, **start_time**, **finish_time**, **nthreads**, **exec_home**, **exec_cwd**, **output_file**, **error_file** to get job record.

PBS

buildtest can support job submission to [PBS Pro](#) or [OpenPBS](#) scheduler. Assuming you have configured [PBS Executors](#) in your configuration file you can submit jobs to the PBS executor by selecting the appropriate pbs executor via **executor** property in buildspect. The **#PBS** directives can be specified using **pbs** field which is a list of PBS options that get inserted at top of script. Shown below is an example buildspect using the *script* schema.

```
version: "1.0"
buildspecs:
  pbs_sleep:
    type: script
    executor: generic.pbs.workq
    pbs: ["-l nodes=1", "-l walltime=00:02:00"]
    run: sleep 10
```

buildtest will poll PBS jobs using `qstat -x -f -F json <jobID>` until job is finished. Note that we use **-x** option to retrieve finished jobs which is required inorder for buildtest to detect job state upon completion. Please see [PBS Limitation](#) to ensure your PBS cluster supports job history.

Shown below is an example build of the buildspect using PBS scheduler.

```
[pbsuser@pbs buildtest]$ buildtest build -b general_tests/sched/pbs/hostname.yml
```

```
+-----+
| Stage: Discovering Buildspects |
+-----+
```

Discovered Buildspects:

```
/tmp/Documents/buildtest/general_tests/sched/pbs/hostname.yml
```

```
+-----+
| Stage: Parsing Buildspects |
+-----+
```

schemafile	validstate	buildspec
script-v1.0.schema.json	True	/tmp/Documents/buildtest/general_tests/sched/pbs/hostname.yml

```
+-----+
| Stage: Building Test |
+-----+
```

name	id	type	executor	tags	testpath

(continues on next page)

(continued from previous page)

```
pbs_sleep | 2adfc3c1 | script | generic.pbs.workq |      | /tmp/Documents/buildtest/
↳ var/tests/generic.pbs.workq/hostname/pbs_sleep/3/stage/generate.sh
```

```
+-----+
| Stage: Running Test |
+-----+
```

```
[pbs_sleep] JobID: 40.pbs dispatched to scheduler
```

```
name      | id      | executor      | status  | returncode | testpath
```

```
-----+-----+-----+-----+-----+-----
```

```
↳ -----
pbs_sleep | 2adfc3c1 | generic.pbs.workq | N/A      |      -1 | /tmp/Documents/
↳ buildtest/var/tests/generic.pbs.workq/hostname/pbs_sleep/3/stage/generate.sh
```

Polling Jobs in 10 seconds

```
-----
Job Queue: ['40.pbs']
```

Completed Jobs

```
-----
| name      | executor      | jobID      | jobstate |
```

Pending Jobs

```
-----
| name      | executor      | jobID      | jobstate |
| pbs_sleep | generic.pbs.workq | 40.pbs    | R        |
```

Polling Jobs in 10 seconds

```
-----
Job Queue: ['40.pbs']
```

Completed Jobs

(continues on next page)

(continued from previous page)

name	executor	jobID	jobstate
------	----------	-------	----------

Pending Jobs

name	executor	jobID	jobstate
pbs_sleep	generic.pbs.workq	40.pbs	F

Polling Jobs in 10 seconds

Job Queue: []

Completed Jobs

name	executor	jobID	jobstate
pbs_sleep	generic.pbs.workq	40.pbs	F

Pending Jobs

name	executor	jobID	jobstate
------	----------	-------	----------

+-----+
| Stage: Final Results after Polling all Jobs |
+-----+

name	id	executor	status	returncode	testpath
pbs_sleep	2adfc3c1	generic.pbs.workq	PASS	0	/tmp/Documents/ buildtest/var/tests/generic.pbs.workq/hostname/pbs_sleep/3/stage/generate.sh

(continues on next page)

(continued from previous page)

```

+-----+
| Stage: Test Summary |
+-----+

Executed 1 tests
Passed Tests: 1/1 Percentage: 100.000%
Failed Tests: 0/1 Percentage: 0.000%

Writing Logfile to: /tmp/buildtest_mu285m58.log

```

buildtest will preserve the job record from `qstat -x -f -F json <jobID>` in the test report if job was complete. If we take a look at the test result using **buildtest inspect** you will see the job section is prepopulated from the JSON record provided by **qstat**.

```

1 [pbsuser@pbs buildtest]$ buildtest inspect id 2adfc3c1
2 {
3   "id": "2adfc3c1",
4   "full_id": "2adfc3c1-1c81-43d0-a151-6fa1a9818eb4",
5   "testroot": "/tmp/Documents/buildtest/var/tests/generic.pbs.workq/hostname/pbs_sleep/3
6   ↪",
7   "testpath": "/tmp/Documents/buildtest/var/tests/generic.pbs.workq/hostname/pbs_sleep/3/
8   ↪stage/generate.sh",
9   "stagedir": "/tmp/Documents/buildtest/var/tests/generic.pbs.workq/hostname/pbs_sleep/3/
10  ↪stage",
11  "rundir": "/tmp/Documents/buildtest/var/tests/generic.pbs.workq/hostname/pbs_sleep/3/
12  ↪run",
13  "command": "qsub -q workq /tmp/Documents/buildtest/var/tests/generic.pbs.workq/
14  ↪hostname/pbs_sleep/3/stage/generate.sh",
15  "outfile": "/tmp/Documents/buildtest/var/tests/generic.pbs.workq/hostname/pbs_sleep/3/
16  ↪stage/pbs_sleep.o40",
17  "errfile": "/tmp/Documents/buildtest/var/tests/generic.pbs.workq/hostname/pbs_sleep/3/
18  ↪stage/pbs_sleep.e40",
19  "schemafile": "script-v1.0.schema.json",
20  "executor": "generic.pbs.workq",
21  "tags": "",
22  "starttime": "Wed Mar 17 20:36:48 2021",
23  "endtime": "Wed Mar 17 20:36:48 2021",
24  "runtime": "00:00:10",
25  "state": "PASS",
26  "returncode": 0,
27  "output": "",
28  "error": "",
29  "job": {
30    "timestamp": 1616013438,
31    "pbs_version": "19.0.0",
32    "pbs_server": "pbs",
33    "Jobs": {
34      "40.pbs": {
35        "Job_Name": "pbs_sleep",
36        "Job_Owner": "pbsuser@pbs",

```

(continues on next page)

(continued from previous page)

```

30     "resources_used": {
31         "cpupercent": 0,
32         "cput": "00:00:00",
33         "mem": "5620kb",
34         "ncpus": 1,
35         "vmem": "25632kb",
36         "walltime": "00:00:10"
37     },
38     "job_state": "F",
39     "queue": "workq",
40     "server": "pbs",
41     "Checkpoint": "u",
42     "ctime": "Wed Mar 17 20:36:48 2021",
43     "Error_Path": "pbs:/tmp/Documents/buildtest/var/tests/generic.pbs.workq/hostname/
↪pbs_sleep/3/stage/pbs_sleep.e40",
44     "exec_host": "pbs/0",
45     "exec_vnode": "(pbs:ncpus=1)",
46     "Hold_Types": "n",
47     "Join_Path": "n",
48     "Keep_Files": "n",
49     "Mail_Points": "a",
50     "mtime": "Wed Mar 17 20:36:58 2021",
51     "Output_Path": "pbs:/tmp/Documents/buildtest/var/tests/generic.pbs.workq/
↪hostname/pbs_sleep/3/stage/pbs_sleep.o40",
52     "Priority": 0,
53     "qtime": "Wed Mar 17 20:36:48 2021",
54     "Rerunable": "True",
55     "Resource_List": {
56         "ncpus": 1,
57         "nodect": 1,
58         "nodes": 1,
59         "place": "scatter",
60         "select": "1:ncpus=1",
61         "walltime": "00:02:00"
62     },
63     "stime": "Wed Mar 17 20:36:48 2021",
64     "session_id": 7154,
65     "jobdir": "/home/pbsuser",
66     "substate": 92,
67     "Variable_List": {
68         "PBS_O_HOME": "/home/pbsuser",
69         "PBS_O_LANG": "en_US.utf8",
70         "PBS_O_LOGNAME": "pbsuser",
71         "PBS_O_PATH": "/tmp/Documents/buildtest/bin:/tmp/Documents/github/buildtest/
↪bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/pbs/bin:/home/pbsuser/.
↪local/bin:/home/pbsuser/bin",
72         "PBS_O_MAIL": "/var/spool/mail/pbsuser",
73         "PBS_O_SHELL": "/bin/bash",
74         "PBS_O_WORKDIR": "/tmp/Documents/buildtest/var/tests/generic.pbs.workq/
↪hostname/pbs_sleep/3/stage",
75         "PBS_O_SYSTEM": "Linux",
76         "PBS_O_QUEUE": "workq",

```

(continues on next page)

(continued from previous page)

```

77     "PBS_O_HOST": "pbs"
78   },
79   "comment": "Job run at Wed Mar 17 at 20:36 on (pbs:ncpus=1) and finished",
80   "etime": "Wed Mar 17 20:36:48 2021",
81   "run_count": 1,
82   "Stageout_status": 1,
83   "Exit_status": 0,
84   "Submit_arguments": "-q workq /tmp/Documents/buildtest/var/tests/generic.pbs.
↪workq/hostname/pbs_sleep/3/stage/generate.sh",
85   "history_timestamp": 1616013418,
86   "project": "_pbs_project_default"
87 }
88 }
89 }
90 }

```

Output File

```
-----
```

Error File

```
-----
```

Test Content

```
-----
```

```

108 #!/bin/bash
109 #PBS -l nodes=1
110 #PBS -l walltime=00:02:00
111 #PBS -N pbs_sleep
112 source /tmp/Documents/buildtest/var/executors/generic.pbs.workq/before_script.sh
113 sleep 10
114 source /tmp/Documents/buildtest/var/executors/generic.pbs.workq/after_script.sh
115
116
117
118 buildspec: /tmp/Documents/buildtest/general_tests/sched/pbs/hostname.yml
119 -----
120 version: "1.0"
121 buildspecs:
122   pbs_sleep:
123     type: script
124     executor: generic.pbs.workq
125     pbs: ["-l nodes=1", "-l walltime=00:02:00"]
126     run: sleep 10

```

You can use batch property to define schedule configuration that is translated into **#PBS** directives. To learn more

about *batch* property see *Scheduler Agnostic Configuration*.

In this example we show how one can use *batch* property with the PBS executor instead of using *pbs* property. You may specify *batch* and *pbs* property to define PBS directives. This example will allocate 1 node, 1 cpu, 500mb memory with 2min timelimit and send email notification.

```
version: "1.0"
buildspecs:
  pbs_sleep:
    type: script
    executor: generic.pbs.workq
  batch:
    nodecount: "1"
    cpucount: "1"
    memory: "500mb"
    email-address: "shahzebmsiddiqui@gmail.com"
    timelimit: "00:02:00"
run: sleep 15
```

buidtest will translate the *batch* property into #PBS directives if their is an equivalent option. Shown below is the generated test using the *batch* property.

```
#!/bin/bash
#PBS -l nodes=1
#PBS -l ncpus=1
#PBS -l mem=500mb
#PBS -WMail_Users=shahzebmsiddiqui@gmail.com
#PBS -l walltime=00:02:00
#PBS -N pbs_sleep
source /tmp/Documents/buidtest/var/executors/generic.pbs.workq/before_script.sh
sleep 15
source /tmp/Documents/buidtest/var/executors/generic.pbs.workq/after_script.sh
```

Cobalt

Cobalt is a job scheduler developed by [Argonne National Laboratory](#) that runs on compute resources and IBM BlueGene series. Cobalt resembles PBS in terms of command line interface such as *qsub*, *qacct* however they slightly differ in their behavior.

Cobalt support has been tested on JLSE and [Theta](#) system. Cobalt directives are specified using #COBALT this can be specified using *cobalt* property which accepts a list of strings. Shown below is an example using *cobalt* property.

```
1 version: "1.0"
2 buildspecs:
3   yarrow_hostname:
4     executor: jlse.cobalt.yarrow
5     type: script
6     cobalt: ["-n 1", "--proccount 1", "-t 10"]
7   run: hostname
```

In this example, we allocate 1 node with 1 processor for 10min. This is translated into the following job script.

```
#!/usr/bin/bash
#COBALT -n 1
```

(continues on next page)

(continued from previous page)

```
#COBALT --proccount 1
#COBALT -t 10
#COBALT --jobname yarrow_hostname
source /home/shahzebsiddiqui/buildtest/var/executors/cobalt.yarrow/before_script.sh
hostname
source /home/shahzebsiddiqui/buildtest/var/executors/cobalt.yarrow/after_script.sh
```

Let's run this test and notice the job states.

```
$ buildtest build -b yarrow_hostname.yml

+-----+
| Stage: Discovering Buildsspecs |
+-----+

Discovered Buildsspecs:

/home/shahzebsiddiqui/jlse_tests/yarrow_hostname.yml

+-----+
| Stage: Parsing Buildsspecs |
+-----+

  schemafile          | validstate | buildspec
+-----+-----+-----+
↪-----
script-v1.0.schema.json | True      | /home/shahzebsiddiqui/jlse_tests/yarrow_
↪hostname.yml
+-----+
| Stage: Building Test |
+-----+

  name          | id          | type   | executor      | tags   | testpath
+-----+-----+-----+-----+-----+-----+
↪-----
yarrow_hostname | f86b93f6   | script | cobalt.yarrow |        | /home/shahzebsiddiqui/
↪buildtest/var/tests/cobalt.yarrow/yarrow_hostname/yarrow_hostname/3/stage/generate.sh
+-----+
| Stage: Running Test |
+-----+

[yarrow_hostname] JobID: 284752 dispatched to scheduler
  name          | id          | executor      | status   | returncode | testpath
+-----+-----+-----+-----+-----+-----+
↪-----
↪-----
yarrow_hostname | f86b93f6   | cobalt.yarrow | N/A      | -1         | /home/
↪shahzebsiddiqui/buildtest/var/tests/cobalt.yarrow/yarrow_hostname/yarrow_hostname/3/
↪stage/generate.sh
```

(continues on next page)

(continued from previous page)

Polling Jobs in 10 seconds

```
builder: yarrow_hostname in None
[yarrow_hostname]: JobID 284752 in starting state
```

Polling Jobs in 10 seconds

```
builder: yarrow_hostname in starting
[yarrow_hostname]: JobID 284752 in starting state
```

Polling Jobs in 10 seconds

```
builder: yarrow_hostname in starting
[yarrow_hostname]: JobID 284752 in running state
```

Polling Jobs in 10 seconds

```
builder: yarrow_hostname in running
[yarrow_hostname]: JobID 284752 in exiting state
```

Polling Jobs in 10 seconds

```
builder: yarrow_hostname in done
```

```
+-----+
| Stage: Final Results after Polling all Jobs |
+-----+
```

name	id	executor	status	returncode	testpath
yarrow_hostname	f86b93f6	cobalt.yarrow	PASS	0	/home/shahzebsiddiqui/buildtest/var/tests/cobalt.yarrow/yarrow_hostname/yarrow_hostname/3/stage/generate.sh

```
+-----+
| Stage: Test Summary |
+-----+
```

```
Executed 1 tests
Passed Tests: 1/1 Percentage: 100.000%
Failed Tests: 0/1 Percentage: 0.000%
```

When job starts, Cobalt will write a cobalt log file <JOBID>.cobaltlog which is provided by scheduler for troubleshooting. The output and error file are generated once job finishes. Cobalt job progresses through job state starting → pending → running → exiting. buildtest will capture Cobalt job details using `qstat -lf <JOBID>`

and this is updated in the report file.

buildtest will poll job at set interval, where we run `qstat --header State <JobID>` to check state of job, if job is finished then we gather results. Once job is finished, `qstat` will not be able to poll job this causes an issue where buildtest can't poll job since `qstat` will not return anything. This is a transient issue depending on when you poll job, generally at ALCF `qstat` will not report existing job within 30sec after job is terminated. buildtest will assume if it's able to poll job and is in *exiting* stage that job is complete, if its unable to retrieve this state we check for output and error file. If file exists we assume job is complete and buildtest will gather the results.

buildtest will determine exit code by parsing cobalt log file, the file contains a line such as

```
Thu Nov 05 17:29:30 2020 +0000 (UTC) Info: task completed normally with an exit code of 0; initiating job cleanup and removal
```

`qstat` has no job record for capturing returncode so buildtest must rely on Cobalt Log file.:

Scheduler Agnostic Configuration

The `batch` field can be used for specifying scheduler agnostic configuration based on your scheduler. buildtest will translate the input into the appropriate script directive supported by the scheduler. Shown below is a translation table for the **batch** field

Table 2: Batch Translation Table

Field	Slurm	LSF	PBS	Cobalt
account	--account	-P	project	--project
begin	--begin	-b	N/A	N/A
cpucount	--ntasks	-n	-l ncpus	--proccount
email-address	--mail-user	-u	-WMail_Users	--notify
exclusive	--exclusive=user	-x	N/A	N/A
memory	--mem	-M	-l mem	N/A
network	--network	-network	N/A	N/A
nodecount	--nodes	-nnodes	-l nodes	--nodecount
qos	--qos	N/A	N/A	N/A
queue	--partition	-q	-q	--queue
tasks-per-core	--ntasks-per-core	N/A	N/A	N/A
tasks-per-node	--ntasks-per-node	N/A	N/A	N/A
tasks-per-socket	--ntasks-per-socket	N/A	N/A	N/A
timelimit	--time	-W	-l walltime	--time

In this example, we rewrite the LSF buildspec to use `batch` instead of `bsub` field.

```

1 version: "1.0"
2 buildspecs:
3   hostname:
4     type: script
5     executor: ascent.lsf.batch
6   batch:
7     timelimit: "10"
8     nodecount: "1"
9   run: jsrun hostname

```

buildtest will translate the `batch` field into `#BSUB` directive as you can see in the generated test. buildtest will automatically name the job based on the testname therefore you will see that buildtest will insert `#BSUB -J`, `#BSUB -o` and `#BSUB -e` directives in the test.

```
#!/usr/bin/bash
#BSUB -W 10
#BSUB -nnodes 1
#BSUB -J hostname
#BSUB -o hostname.out
#BSUB -e hostname.err
jsrun hostname
```

In next example we use batch field with on a Slurm cluster that submits a sleep job as follows.

```
1 version: "1.0"
2 buildspecs:
3   sleep:
4     type: script
5     executor: cori.slurm.knl_debug
6     description: sleep 2 seconds
7     tags: [tutorials]
8   batch:
9     nodecount: "1"
10    cpucount: "1"
11    timelimit: "5"
12    memory: "5MB"
13    exclusive: true
14
15   vars:
16     SLEEP_TIME: 2
17   run: sleep $SLEEP_TIME
```

The exclusive field is used for getting exclusive node access, this is a boolean instead of string. You can instruct buildtest to stop after build phase by using `--stage=build` which will build the script but not run it. If we inspect the generated script we see the following.

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --time=5
#SBATCH --mem=5MB
#SBATCH --exclusive=user
SLEEP_TIME=2
sleep $SLEEP_TIME
```

The batch property can translate some fields into #COBALT directives. buildtest will support fields that are applicable with scheduler. Shown below is an example with 1 node using 10min that runs hostname using executor *jlse.cobalt.iris*.

```
version: "1.0"
buildspecs:
  iris_hostname:
    executor: jlse.cobalt.iris
    type: script
    batch:
      nodecount: "1"
      timelimit: "10"
    run: hostname
```

If we build the buildspec and inspect the testscript we see the following.

```
#!/usr/bin/bash
#COBALT --nodelist 1
#COBALT --time 10
#COBALT --jobname iris_hostname
hostname
```

The first two lines `#COBALT --nodelist 1` and `#COBALT --time 10` are translated based on input from `batch` field. buildtest will automatically add `#COBALT --jobname` based on the name of the test.

You may leverage `batch` with `sbatch`, `bsub`, or `cobalt` field to specify your job directives. If a particular field is not available in `batch` property then utilize `sbatch`, `bsub`, `cobalt` field to fill in rest of the arguments.

Jobs exceeds `max_pend_time`

Recall from [Configuring buildtest](#) that `max_pend_time` will cancel jobs if job exceed timelimit. buildtest will start a timer for each job right after job submission and keep track of time duration, and if job is in **pending** state and it exceeds `max_pend_time`, then job will be cancelled.

We can also override `max_pend_time` configuration via command line `--max-pend-time`. To demonstrate, here is an example where job was cancelled after job was pending and exceeds `max_pend_time`. Note that cancelled job is not reported in final output nor updated in report hence it won't be present in the report (buildtest report). In this example, we only had one test so upon job cancellation we found there was no tests to report hence, buildtest will terminate after run stage.

```
1 $ buildtest build -b buildspects/queues/shared.yml --max-pend-time 15 --poll-interval 5 -k
2
3
4 User: siddiq90
5 Hostname: cori08
6 Platform: Linux
7 Current Time: 2021/06/11 13:31:46
8 buildtest path: /global/homes/s/siddiq90/github/buildtest/bin/buildtest
9 buildtest version: 0.9.5
10 python path: /global/homes/s/siddiq90/.conda/envs/buildtest/bin/python
11 python version: 3.8.8
12 Test Directory: /global/u1/s/siddiq90/github/buildtest/var/tests
13 Configuration File: /global/u1/s/siddiq90/.buildtest/config.yml
14 Command: /global/homes/s/siddiq90/github/buildtest/bin/buildtest build -b buildspects/
15 ↪ queues/shared.yml --max-pend-time 15 --poll-interval 5 -k
16
17 +-----+
18 | Stage: Discovering Buildspects |
19 +-----+
20
21 +-----+
22 | Discovered Buildspects |
23 +-----+
24 | /global/u1/s/siddiq90/github/buildtest-cori/buildspects/queues/shared.yml |
25 +-----+
26 Discovered Buildspects: 1
27 Excluded Buildspects: 0
28 Detected Buildspects after exclusion: 1
```

(continues on next page)

(continued from previous page)

```

28 +-----+
29 | Stage: Parsing Buildsspecs |
30 +-----+
31
32
33 schemafile          | validstate  | buildspec
34 +-----+-----+-----+
35 script-v1.0.schema.json | True      | /global/u1/s/siddiq90/github/buildtest-cori/
36 ↳ buildsspecs/queues/shared.yml
37
38
39 name                description
40 +-----+-----+
41 shared_qos_haswell_hostname run hostname through shared qos on Haswell
42
43 +-----+
44 | Stage: Building Test |
45 +-----+
46
47 name                | id          | type   | executor          | tags
48 ↳                  | testpath
49 +-----+-----+-----+-----+-----+
50 ↳
51 ↳
52 ↳
53 shared_qos_haswell_hostname | 94b2de5d | script | cori.slurm.haswell_shared | ['queues',
54 ↳ 'jobs', 'reframe'] | /global/u1/s/siddiq90/github/buildtest/var/tests/cori.slurm.
55 ↳ haswell_shared/shared/shared_qos_haswell_hostname/2/shared_qos_haswell_hostname_build.
56 ↳ sh
57
58 +-----+
59 | Stage: Running Test |
60 +-----+
61
62 [shared_qos_haswell_hostname] JobID: 43313766 dispatched to scheduler
63 name                | id          | executor          | status |
64 ↳ returncode
65 +-----+-----+-----+-----+-----+
66 ↳
67 shared_qos_haswell_hostname | 94b2de5d | cori.slurm.haswell_shared | N/A    |
68 ↳ -1
69
70 Polling Jobs in 5 seconds
71
72 Job Queue: [43313766]
73
74 Pending Jobs

```

(continues on next page)

(continued from previous page)

```

69 -----
70
71
72 +-----+-----+-----+-----+
73 |          name          |      executor      |  jobID  | jobstate |
74 +-----+-----+-----+-----+
75 | shared_qos_haswell_hostname | cori.slurm.haswell_shared | 43313766 | PENDING |
76 +-----+-----+-----+-----+

```

Polling Jobs in 5 seconds

```

80 -----
81 Job Queue: [43313766]
82
83

```

Pending Jobs

```

85 -----
86
87
88 +-----+-----+-----+-----+
89 |          name          |      executor      |  jobID  | jobstate |
90 +-----+-----+-----+-----+
91 | shared_qos_haswell_hostname | cori.slurm.haswell_shared | 43313766 | PENDING |
92 +-----+-----+-----+-----+

```

Polling Jobs in 5 seconds

```

96 -----
97 Job Queue: [43313766]
98
99

```

Pending Jobs

```

101 -----
102
103
104 +-----+-----+-----+-----+
105 |          name          |      executor      |  jobID  | jobstate |
106 +-----+-----+-----+-----+
107 | shared_qos_haswell_hostname | cori.slurm.haswell_shared | 43313766 | PENDING |
108 +-----+-----+-----+-----+

```

Polling Jobs in 5 seconds

```

112 -----
113 Cancelling Job because duration time: 21.177340 sec exceeds max pend time: 15 sec
114 Job Queue: [43313766]
115
116

```

Pending Jobs

```

118 -----
119
120

```

(continues on next page)

(continued from previous page)

```

121 +-----+-----+-----+-----+
122 |          name          |      executor      |  jobID  | jobstate |
123 +-----+-----+-----+-----+
124 | shared_qos_haswell_hostname | cori.slurm.haswell_shared | 43313766 | CANCELLED |
125 +-----+-----+-----+-----+
126
127
128 Polling Jobs in 5 seconds
129 -----
130 Job Queue: []
131 Cancelled Tests:
132 shared_qos_haswell_hostname
133 After polling all jobs we found no valid builders to process

```

Cray Burst Buffer & Data Warp

For Cray systems, you may want to stage-in or stage-out into your burst buffer this can be configured using the `#DW` directive. For a list of data warp examples see section on [DataWarp Job Script Commands](#)

In buildtest we support properties `BB` and `DW` which is a list of job directives that get inserted as `#BW` and `#DW` into the test script. To demonstrate let's start off with an example where we create a persistent burst buffer named `databuffer` of size 10GB striped. We access the burst buffer using the `DW` directive. Finally we `cd` into the `databuffer` and write a 5GB random file.

Note: `BB` and `DW` directives are generated after scheduler directives. The `#BB` comes before `#DW`. buildtest will automatically add the directive `#BB` and `#DW` when using properties `BB` and `DW`

```

1  version: "1.0"
2  buildspecs:
3    create_burst_buffer:
4      type: script
5      executor: cori.slurm.debug
6      batch:
7        nodecount: "1"
8        timelimit: "5"
9        cpucount: "1"
10     sbatch: ["-C knl"]
11     description: Create a burst buffer
12     tags: [jobs]
13     BB:
14       - create_persistent name=databuffer capacity=10GB access_mode=striped type=scratch
15     DW:
16       - persistentdw name=databuffer
17     run: |
18       cd $DW_PERSISTENT_STRIPED_databuffer
19       pwd
20       dd if=/dev/urandom of=random.txt bs=1G count=5 iflags=fullblock
21       ls -lh $DW_PERSISTENT_STRIPED_databuffer/

```

Next we run this test and inspect the generated test we will see that `#BB` and `#DW` directives are inserted after the scheduler directives

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --time=5
#SBATCH --ntasks=1
#SBATCH --job-name=create_burst_buffer
#SBATCH --output=create_burst_buffer.out
#SBATCH --error=create_burst_buffer.err
#BB create_persistent name=databuffer capacity=10GB access_mode=striped type=scratch
#DW persistentdw name=databuffer
cd $DW_PERSISTENT_STRIPED_databuffer
pwd
dd if=/dev/urandom of=random.txt bs=1G count=5 iflag=fullblock
ls -lh $DW_PERSISTENT_STRIPED_databuffer
```

We can confirm there is an active burst buffer by running the following

```
$ scontrol show burst | grep databuffer
    Name=databuffer CreateTime=2020-10-29T13:06:21 Pool=wlm_pool Size=20624MiB
    ↪State=allocated UserID=siddiq90(92503)
```

3.6.6 Buildtest Schemas

Schema Naming Convention

All schema files use the file extension **.schema.json** to distinguish itself as a json schema definition from an ordinary json file. The schema files are located in `buildtest/schemas` directory.

Schema Examples

The schema examples are a great way to help write your buildsspecs and help you understand the edge cases that can lead to an invalid buildspec. The schema examples are used in buildtest regression test for validating the schemas. We expose the examples through buildtest client so its accessible for everyone.

In order to view an example you can run:

```
buildtest schema -n <schema> --example
```

Definition Schema

This schema is used for declaring `definitions` that need to be reused in multiple schemas. We use `$ref` keyword to reference definitions from this file.

Schema Content

```
$ buildtest schema -n definitions.schema.json --json
{
  "$id": "definitions.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "JSON Schema Definitions File. ",
  "description": "This file is used for declaring definitions that are referenced from_
↳ other schemas",
  "definitions": {
    "list_of_strings": {
      "type": "array",
      "uniqueItems": true,
      "minItems": 1,
      "items": {
        "type": "string"
      }
    },
    "string_or_list": {
      "oneOf": [
        {
          "type": "string"
        },
        {
          "$ref": "#/definitions/list_of_strings"
        }
      ]
    },
    "list_of_ints": {
      "type": "array",
      "uniqueItems": true,
      "minItems": 1,
      "items": {
        "type": "integer"
      }
    },
    "int_or_list": {
      "oneOf": [
        {
          "type": "integer"
        },
        {
          "$ref": "#/definitions/list_of_ints"
        }
      ]
    },
    "env": {
      "type": "object",
      "description": "One or more key value pairs for an environment (key=value)",
      "minItems": 1,
      "items": {
        "type": "object",
```

(continues on next page)

(continued from previous page)

```

    "minItems": 1,
    "propertyNames": {
      "pattern": "^[A-Za-z_][A-Za-z0-9_]*$"
    }
  },
  "description": {
    "type": "string",
    "description": "The ``description`` field is used to document what the test is.↵
↵doing",
    "maxLength": 80
  },
  "tags": {
    "description": "Classify tests using a tag name, this can be used for categorizing.↵
↵test and building tests using ``--tags`` option",
    "$ref": "#/definitions/string_or_list"
  },
  "skip": {
    "type": "boolean",
    "description": "The ``skip`` is a boolean field that can be used to skip tests.↵
↵during builds. By default buildtest will build and run all tests in your buildspeg.↵
↵file, if ``skip: True`` is set it will skip the buildspeg."
  },
  "executor": {
    "type": "string",
    "description": "Select one of the executor name defined in your configuration file.↵
↵(``config.yml``). Every buildspeg must have an executor which is responsible for.↵
↵running job. "
  },
  "run_only": {
    "type": "object",
    "description": "A set of conditions to specify when running tests. All conditions.↵
↵must pass in order to process test.",
    "additionalProperties": false,
    "properties": {
      "scheduler": {
        "type": "string",
        "description": "Test will run only if scheduler is available. We assume.↵
↵binaries are available in $PATH",
        "enum": [
          "lsf",
          "slurm",
          "cobalt",
          "pbs"
        ]
      },
      "user": {
        "type": "string",
        "description": "Test will run only if current user matches this field,↵
↵otherwise test will be skipped"
      },
      "platform": {

```

(continues on next page)

(continued from previous page)

```

        "type": "string",
        "description": "This test will run if target system is Linux or Darwin. We
↪check target system using ``platform.system()`` and match with input field",
        "enum": [
            "Linux",
            "Darwin"
        ]
    },
    "linux_distro": {
        "type": "array",
        "description": "Specify a list of Linux Distros to check when processing test.
↪If target system matches one of input field, test will be processed.",
        "uniqueItems": true,
        "minItems": 1,
        "items": {
            "type": "string",
            "enum": [
                "darwin",
                "ubuntu",
                "debian",
                "rhel",
                "centos",
                "fedora",
                "sles",
                "opensuse",
                "amazon",
                "arch"
            ]
        }
    }
},
"batch": {
    "type": "object",
    "description": "The ``batch`` field is used to specify scheduler agnostic
↪directives that are translated to #SBATCH or #BSUB based on your scheduler. This is an
↪experimental feature that supports a subset of scheduler parameters.",
    "additionalProperties": false,
    "properties": {
        "account": {
            "type": "string",
            "description": "Specify Account to charge job"
        },
        "begintime": {
            "type": "string",
            "description": "Specify begin time when job will start allocation"
        },
        "cpucount": {
            "type": "string",
            "description": "Specify number of CPU to allocate"
        },
        "email-address": {

```

(continues on next page)

(continued from previous page)

```

        "type": "string",
        "description": "Email Address to notify on Job State Changes"
    },
    "exclusive": {
        "type": "boolean",
        "description": "Specify if job needs to run in exclusive mode"
    },
    "memory": {
        "type": "string",
        "description": "Specify Memory Size for Job"
    },
    "network": {
        "type": "string",
        "description": "Specify network resource requirement for job"
    },
    "nodecount": {
        "type": "string",
        "description": "Specify number of Nodes to allocate"
    },
    "qos": {
        "type": "string",
        "description": "Specify Quality of Service (QOS)"
    },
    "queue": {
        "type": "string",
        "description": "Specify Job Queue"
    },
    "tasks-per-core": {
        "type": "string",
        "description": "Request number of tasks to be invoked on each core. "
    },
    "tasks-per-node": {
        "type": "string",
        "description": "Request number of tasks to be invoked on each node. "
    },
    "tasks-per-socket": {
        "type": "string",
        "description": "Request the maximum tasks to be invoked on each socket. "
    },
    "timelimit": {
        "type": "string",
        "description": "Specify Job timelimit"
    }
}
},
"status": {
    "type": "object",
    "description": "The status section describes how buildtest detects PASS/FAIL on ↵
↵test. By default returncode 0 is a PASS and anything else is a FAIL, however buildtest ↵
↵can support other types of PASS/FAIL conditions.",
    "additionalProperties": false,
    "properties": {

```

(continues on next page)

(continued from previous page)

```

    "slurm_job_state": {
        "type": "string",
        "enum": [
            "COMPLETED",
            "FAILED",
            "OUT_OF_MEMORY",
            "TIMEOUT"
        ],
        "description": "This field can be used for checking Slurm Job State, if there_
↪ is a match buildtest will report as ``PASS`` "
    },
    "returncode": {
        "description": "Specify a list of returncodes to match with script's exit code.
↪ buildtest will PASS test if script's exit code is found in list of returncodes. You_
↪ must specify unique numbers as list and a minimum of 1 item in array",
        "$ref": "#/definitions/int_or_list"
    },
    "regex": {
        "type": "object",
        "description": "Perform regular expression search using ``re.search`` python_
↪ module on stdout/stderr stream for reporting if test ``PASS``. ",
        "properties": {
            "stream": {
                "type": "string",
                "enum": [
                    "stdout",
                    "stderr"
                ],
            },
            "description": "The stream field can be stdout or stderr. buildtest will_
↪ read the output or error stream after completion of test and check if regex matches in_
↪ stream"
        },
        "exp": {
            "type": "string",
            "description": "Specify a regular expression to run with input stream_
↪ specified by ``stream`` field. buildtest uses re.search when performing regex"
        }
    },
    "required": [
        "stream",
        "exp"
    ],
    "runtime": {
        "type": "object",
        "description": "The runtime section will pass test based on min and max values_
↪ and compare with actual runtime. ",
        "properties": {
            "min": {
                "type": "number",
                "minimum": 0,
                "description": "Specify a minimum runtime in seconds. The test will PASS_
↪ if actual runtime exceeds min time."
            }
        }
    }

```

(continues on next page)

(continued from previous page)

```

    },
    "max": {
      "type": "number",
      "minimum": 0,
      "description": "Specify a maximum runtime in seconds. The test will PASS_
→if actual runtime is less than max time"
    }
  }
}
}
}
}
}
}
}
}

```

Settings Schema

This schema defines how *buildtest configuration* file is validated.

Schema Content

```

$ buildtest schema -n settings.schema.json --json
{
  "$id": "settings.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "buildtest configuration schema",
  "type": "object",
  "required": [
    "system"
  ],
  "additionalProperties": false,
  "properties": {
    "system": {
      "type": "object",
      "patternProperties": {
        "^.*$": {
          "$ref": "#/definitions/system"
        }
      }
    }
  },
  "definitions": {
    "system": {
      "required": [
        "executors",
        "moduletool",
        "load_default_buildspecs",
        "hostnames",
        "compilers"
      ],

```

(continues on next page)

(continued from previous page)

```

    "additionalProperties": false,
    "type": "object",
    "properties": {
      "hostnames": {
        "type": "array",
        "description": "Specify a list of hostnames to check where buildtest can run.↵
↵for the given system record",
        "items": {
          "type": "string"
        }
      },
      "description": {
        "type": "string",
        "description": "system description field"
      },
      "buildspec_roots": {
        "type": "array",
        "items": {
          "type": "string"
        },
        "description": "Specify a list of directory paths to search buildspecs. This↵
↵field can be used with ``buildtest buildspec find`` to rebuild buildspec cache or↵
↵build tests using ``buildtest build`` command"
      },
      "load_default_buildspecs": {
        "type": "boolean",
        "description": "Specify whether buildtest should automatically load ↵
↵buildspecs provided in buildtest repo into buildspec cache"
      },
      "testdir": {
        "type": "string",
        "description": "Specify full path to test directory where buildtest will write↵
↵tests."
      },
      "logdir": {
        "type": "string",
        "description": "Specify location where buildtest will write log files"
      },
      "moduletool": {
        "type": "string",
        "description": "Specify modules tool used for interacting with ``module``↵
↵command. ",
        "enum": [
          "environment-modules",
          "lmod",
          "N/A"
        ]
      },
      "processor": {
        "type": "object",
        "description": "Specify processor information",
        "additionalProperties": false,

```

(continues on next page)

(continued from previous page)

```

"properties": {
  "numcpus": {
    "type": "integer",
    "minimum": 1,
    "description": "Specify Total Number of CPUs"
  },
  "sockets": {
    "type": "integer",
    "minimum": 1,
    "description": "Specify Number of CPU Sockets"
  },
  "cores": {
    "type": "integer",
    "minimum": 1,
    "description": "Specify Number of Physical Cores"
  },
  "threads_per_core": {
    "type": "integer",
    "minimum": 1,
    "description": "Specify Threads per Core"
  },
  "core_per_socket": {
    "type": "integer",
    "minimum": 1,
    "description": "Specify Cores per Socket"
  },
  "model": {
    "type": "string",
    "description": "Specify Processor Model"
  },
  "arch": {
    "type": "string",
    "description": "Specify processor architecture"
  },
  "vendor": {
    "type": "string",
    "description": "Vendor Name"
  }
},
"compilers": {
  "type": "object",
  "description": "Declare compiler section for defining system compilers that
↳ can be referenced in buildspec.",
  "additionalProperties": false,
  "properties": {
    "find": {
      "type": "object",
      "additionalProperties": false,
      "description": "Find compilers by specifying regular expression that is
↳ applied to modulefile names",
      "properties": {

```

(continues on next page)

(continued from previous page)

```

        "gcc": {
            "type": "string",
            "description": "Specify a regular expression to search for gcc_
↪compilers from your module stack"
        },
        "intel": {
            "type": "string",
            "description": "Specify a regular expression to search for intel_
↪compilers from your module stack"
        },
        "cray": {
            "type": "string",
            "description": "Specify a regular expression to search for cray_
↪compilers from your module stack"
        },
        "clang": {
            "type": "string",
            "description": "Specify a regular expression to search for clang_
↪compilers from your module stack"
        },
        "cuda": {
            "type": "string",
            "description": "Specify a regular expression to search for cuda_
↪compilers from your module stack"
        },
        "pgi": {
            "type": "string",
            "description": "Specify a regular expression to search for pgi_
↪compilers from your module stack"
        },
        "upcxx": {
            "type": "string",
            "description": "Specify a regular expression to search for upcxx_
↪compilers from your module stack"
        }
    },
    "compiler": {
        "type": "object",
        "additionalProperties": false,
        "description": "Start of compiler declaration",
        "properties": {
            "gcc": {
                "description": "Declaration of one or more GNU compilers where we_
↪define C, C++ and Fortran compiler. The GNU compiler wrapper are ``gcc``, ``g++`` and_
↪``gfortran``. ",
                "type": "object",
                "patternProperties": {
                    "^.*$": {
                        "$ref": "#/definitions/compiler_section"
                    }
                }
            }
        }
    }

```

(continues on next page)

(continued from previous page)

```

    },
    "intel": {
        "description": "Declaration of one or more Intel compilers where we
↪define C, C++ and Fortran compiler. The Intel compiler wrapper are ``icc``, ``icpc``
↪and ``ifort``.",
        "type": "object",
        "patternProperties": {
            "^.*$": {
                "$ref": "#/definitions/compiler_section"
            }
        }
    },
    "cray": {
        "description": "Declaration of one or more Cray compilers where we
↪define C, C++ and Fortran compiler. The Cray compiler wrapper are ``cc``, ``CC`` and
↪``ftn``.",
        "type": "object",
        "patternProperties": {
            "^.*$": {
                "$ref": "#/definitions/compiler_section"
            }
        }
    },
    "pgi": {
        "description": "Declaration of one or more PGI compilers where we
↪define C, C++ and Fortran compiler. The PGI compiler wrapper are ``pgcc``, ``pgc++``
↪and ``pgfortran``.",
        "type": "object",
        "patternProperties": {
            "^.*$": {
                "$ref": "#/definitions/compiler_section"
            }
        }
    },
    "clang": {
        "description": "Declaration of one or more Clang compilers where we
↪define C, C++ compiler. The Clang compiler wrapper are ``clang``, ``clang++``.",
        "type": "object",
        "patternProperties": {
            "^.*$": {
                "$ref": "#/definitions/compiler_section"
            }
        }
    },
    "cuda": {
        "description": "Declaration of one or more Cuda compilers where we
↪define C compiler. The Cuda compiler wrapper is ``nvcc``.",
        "type": "object",
        "patternProperties": {
            "^.*$": {
                "$ref": "#/definitions/compiler_section"
            }
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

    }
  },
  "upcxx": {
    "description": "Declaration of one or more UPCXX compilers where we
↪define C, C++ compiler. The UPCXX compiler wrapper are ``upcxx``.",
    "type": "object",
    "patternProperties": {
      "^.*$": {
        "$ref": "#/definitions/compiler_section"
      }
    }
  }
}
},
"executors": {
  "type": "object",
  "additionalProperties": false,
  "description": "The executor section is used for declaring your executors that
↪are responsible for running jobs. The executor section can be ``local``, ``lsf``,
↪``slurm``, ``cobalt``. The executors are referenced in builds spec using ``executor``
↪field.",
  "properties": {
    "defaults": {
      "type": "object",
      "description": "Specify default executor settings for all executors",
      "additionalProperties": false,
      "properties": {
        "pollinterval": {
          "type": "integer",
          "description": "Specify poll interval in seconds after job submission,
↪where buildtest will sleep and poll all jobs for job states. This field can be
↪configured based on your preference. Excessive polling every few seconds can result in
↪system degradation. ",
          "minimum": 10,
          "maximum": 300,
          "default": 30
        },
        "launcher": {
          "type": "string",
          "enum": [
            "sbatch",
            "bsub",
            "qsub"
          ],
          "description": "Specify batch launcher to use when submitting jobs,
↪this is applicable for LSF and Slurm Executors."
        },
        "max_pend_time": {
          "$ref": "#/definitions/max_pend_time"
        }
      }
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

        "account": {
            "$ref": "#/definitions/account"
        }
    },
    "local": {
        "type": "object",
        "description": "The ``local`` section is used for declaring local_
↳executors for running jobs on local machine",
        "patternProperties": {
            "^.*$": {
                "$ref": "#/definitions/local"
            }
        }
    },
    "lsf": {
        "type": "object",
        "description": "The ``lsf`` section is used for declaring LSF executors_
↳for running jobs using LSF scheduler",
        "patternProperties": {
            "^.*$": {
                "$ref": "#/definitions/lsf"
            }
        }
    },
    "slurm": {
        "type": "object",
        "description": "The ``slurm`` section is used for declaring Slurm_
↳executors for running jobs using Slurm scheduler",
        "patternProperties": {
            "^.*$": {
                "$ref": "#/definitions/slurm"
            }
        }
    },
    "cobalt": {
        "type": "object",
        "description": "The ``cobalt`` section is used for declaring Cobalt_
↳executors for running jobs using Cobalt scheduler",
        "patternProperties": {
            "^.*$": {
                "$ref": "#/definitions/cobalt"
            }
        }
    },
    "pbs": {
        "type": "object",
        "description": "The ``pbs`` section is used for declaring PBS executors_
↳for running jobs using PBS scheduler",
        "patternProperties": {
            "^.*$": {
                "$ref": "#/definitions/pbs"
            }
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

        }
    }
}
},
"cdash": {
    "type": "object",
    "description": "Specify CDASH configuration used to upload tests via
↳ 'buildtest cdash' command",
    "required": [
        "url",
        "project",
        "site"
    ],
    "properties": {
        "url": {
            "type": "string",
            "description": "Url to CDASH server"
        },
        "project": {
            "type": "string",
            "description": "Name of CDASH project"
        },
        "site": {
            "type": "string",
            "description": "Site Name reported in CDASH"
        }
    }
}
}
},
"cc": {
    "description": "Specify path to C compiler wrapper. You may specify a compiler
↳ wrapper such as ``gcc`` assuming its in $PATH or you can use ``modules`` property to
↳ resolve path to compiler wrapper.",
    "type": "string"
},
"cxx": {
    "type": "string",
    "description": "Specify path to C++ compiler wrapper. You may specify a compiler
↳ wrapper such as ``g++`` assuming its in $PATH or you can use ``modules`` property to
↳ resolve path to compiler wrapper."
},
"fc": {
    "type": "string",
    "description": "Specify path to Fortran compiler wrapper. You may specify a
↳ compiler wrapper such as ``gfortran`` assuming its in $PATH or you can use ``modules``
↳ property to resolve path to compiler wrapper."
},
"compiler_section": {
    "description": "A compiler section is composed of ``cc``, ``cxx`` and ``fc``
↳ wrapper these are required when you need to specify compiler wrapper.",

```

(continues on next page)

(continued from previous page)

```

    "type": "object",
    "additionalProperties": false,
    "required": [
        "cc",
        "cxx",
        "fc"
    ],
    "properties": {
        "cc": {
            "$ref": "#/definitions/cc"
        },
        "cxx": {
            "$ref": "#/definitions/cxx"
        },
        "fc": {
            "$ref": "#/definitions/fc"
        },
        "module": {
            "$ref": "#/definitions/module"
        }
    }
},
"unique_string_array": {
    "type": "array",
    "uniqueItems": true,
    "items": {
        "type": "string"
    }
},
"module": {
    "type": "object",
    "additionalProperties": false,
    "properties": {
        "purge": {
            "type": "boolean",
            "description": "Run ``module purge`` if purge is set"
        },
        "load": {
            "$ref": "definitions.schema.json#/definitions/list_of_strings",
            "description": "Load one or more modules via ``module load``"
        },
        "swap": {
            "description": "Swap modules using ``module swap``. The swap property expects_
↪ 2 unique modules.",
            "type": "array",
            "uniqueItems": true,
            "minItems": 2,
            "maxItems": 2,
            "items": {
                "type": "string"
            }
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

    }
  },
  "script": {
    "type": "array",
    "additionalProperties": false,
    "items": {
      "type": "string"
    }
  },
  "max_pend_time": {
    "type": "integer",
    "description": "Cancel job if it is still pending in queue beyond max_pend_time",
    "minimum": 10,
    "default": 90
  },
  "account": {
    "type": "string",
    "description": "Specify Job Account for charging resources"
  },
  "local": {
    "type": "object",
    "description": "An instance object of local executor",
    "additionalProperties": false,
    "required": [
      "shell"
    ],
    "properties": {
      "description": {
        "type": "string",
        "description": "description field for documenting your executor"
      },
      "shell": {
        "type": "string",
        "description": "Specify the shell launcher you want to use when running tests.
↳ locally",
        "pattern": "^(/bin/bash|/bin/sh|/bin/csh|/bin/tcsh|/bin/
↳ zsh|sh|bash|csh|tcsh|zsh|python).*"
      },
      "before_script": {
        "#ref": "#/definitions/script"
      }
    }
  },
  "slurm": {
    "type": "object",
    "additionalProperties": false,
    "description": "An instance object of slurm executor",
    "properties": {
      "description": {
        "type": "string",
        "description": "description field for documenting your executor"
      }
    },

```

(continues on next page)

(continued from previous page)

```

    "launcher": {
      "type": "string",
      "enum": [
        "sbatch"
      ],
      "description": "Specify the slurm batch scheduler to use. This overrides the
↪ default ``launcher`` field. This must be ``sbatch``. "
    },
    "options": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "description": "Specify any other options for ``sbatch`` used by this executor
↪ for running all jobs."
    },
    "cluster": {
      "type": "string",
      "description": "Specify the slurm cluster you want to use ``-M <cluster>``"
    },
    "partition": {
      "type": "string",
      "description": "Specify the slurm partition you want to use ``-p <partition>``"
    },
    "qos": {
      "type": "string",
      "description": "Specify the slurm qos you want to use ``-q <qos>``"
    },
    "before_script": {
      "description": "The ``before_script`` section can be used to specify commands
↪ before start of test. The script will be sourced in active shell.",
      "#ref": "#/definitions/script"
    },
    "after_script": {
      "description": "The ``after_script`` section can be used to specify commands
↪ at end of test. The script will be sourced in active shell.",
      "#ref": "#/definitions/script"
    },
    "max_pend_time": {
      "description": "overrides default ``max_pend_time`` value",
      "$ref": "#/definitions/max_pend_time"
    },
    "account": {
      "description": "overrides default ``account`` value",
      "$ref": "#/definitions/account"
    }
  },
  "lsf": {
    "type": "object",
    "description": "An instance object of lsf executor",
    "additionalProperties": false,

```

(continues on next page)

(continued from previous page)

```

    "required": [
        "queue"
    ],
    "properties": {
        "description": {
            "type": "string",
            "description": "description field for documenting your executor"
        },
        "launcher": {
            "type": "string",
            "enum": [
                "bsub"
            ],
            "description": "Specify the lsf batch scheduler to use. This overrides the
↪ default ``launcher`` field. It must be ``bsub``. "
        },
        "options": {
            "type": "array",
            "items": {
                "type": "string"
            },
            "description": "Specify any options for ``bsub`` for this executor when
↪ running all jobs associated to this executor"
        },
        "queue": {
            "type": "string",
            "description": "Specify the lsf queue you want to use ``-q <queue>``"
        },
        "before_script": {
            "description": "The ``before_script`` section can be used to specify commands
↪ before start of test. The script will be sourced in active shell.",
            "#ref": "#/definitions/script"
        },
        "after_script": {
            "description": "The ``after_script`` section can be used to specify commands
↪ at end of test. The script will be sourced in active shell.",
            "#ref": "#/definitions/script"
        },
        "max_pend_time": {
            "description": "overrides default ``max_pend_time`` value",
            "$ref": "#/definitions/max_pend_time"
        },
        "account": {
            "description": "overrides default ``account`` value",
            "$ref": "#/definitions/account"
        }
    }
},
"cobalt": {
    "type": "object",
    "description": "An instance object of cobalt executor",
    "additionalProperties": false,

```

(continues on next page)

(continued from previous page)

```

    "required": [
        "queue"
    ],
    "properties": {
        "description": {
            "type": "string",
            "description": "description field for documenting your executor"
        },
        "launcher": {
            "type": "string",
            "enum": [
                "qsub"
            ],
            "description": "Specify the cobalt batch scheduler to use. This overrides the
↪ default ``launcher`` field. It must be ``qsub``. "
        },
        "options": {
            "type": "array",
            "items": {
                "type": "string"
            },
            "description": "Specify any options for ``qsub`` for this executor when
↪ running all jobs associated to this executor"
        },
        "queue": {
            "type": "string",
            "description": "Specify the lsf queue you want to use ``-q <queue>``"
        },
        "before_script": {
            "description": "The ``before_script`` section can be used to specify commands
↪ before start of test. The script will be sourced in active shell.",
            "#ref": "#/definitions/script"
        },
        "after_script": {
            "description": "The ``after_script`` section can be used to specify commands
↪ at end of test. The script will be sourced in active shell.",
            "#ref": "#/definitions/script"
        },
        "max_pend_time": {
            "description": "overrides default ``max_pend_time`` value",
            "$ref": "#/definitions/max_pend_time"
        },
        "account": {
            "description": "overrides default ``account`` value",
            "$ref": "#/definitions/account"
        }
    }
},
"pbs": {
    "type": "object",
    "description": "An instance object of cobalt executor",
    "additionalProperties": false,

```

(continues on next page)

(continued from previous page)

```

    "required": [
        "queue"
    ],
    "properties": {
        "description": {
            "type": "string",
            "description": "description field for documenting your executor"
        },
        "launcher": {
            "type": "string",
            "enum": [
                "qsub"
            ],
            "description": "Specify the pbs batch scheduler to use. This overrides the
↳ default ``launcher`` field. It must be ``qsub``. "
        },
        "options": {
            "type": "array",
            "items": {
                "type": "string"
            },
            "description": "Specify any options for ``qsub`` for this executor when
↳ running all jobs associated to this executor"
        },
        "queue": {
            "type": "string",
            "description": "Specify the lsf queue you want to use ``-q <queue>``"
        },
        "before_script": {
            "description": "The ``before_script`` section can be used to specify commands
↳ before start of test. The script will be sourced in active shell.",
            "#ref": "#/definitions/script"
        },
        "after_script": {
            "description": "The ``after_script`` section can be used to specify commands
↳ at end of test. The script will be sourced in active shell.",
            "#ref": "#/definitions/script"
        },
        "max_pend_time": {
            "description": "overrides default ``max_pend_time`` value",
            "$ref": "#/definitions/max_pend_time"
        },
        "account": {
            "description": "overrides default ``account`` value",
            "$ref": "#/definitions/account"
        }
    }
}
}
}

```

Schema Examples

```
$ buildtest schema -n settings.schema.json --example
File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/schemas/examples/settings.schema.json/valid/slurm-example.yml
```

```
system:
  generic:
    hostnames: ['.*']

    moduletool: lmod
    load_default_buildspecs: True
    buildspec_roots:
      - $HOME/buildtest-cori
    testdir: /tmp/buildtest
    executors:
      defaults:
        pollinterval: 20
        launcher: sbatch
        max_pend_time: 30
        account: admin
      slurm:
        normal:
          options: ["-C haswell"]
          qos: normal
          before_script: |
            time
            echo "commands run before job"

    compilers:
      compiler:
        gcc:
          default:
            cc: /usr/bin/gcc
            cxx: /usr/bin/g++
            fc: /usr/bin/gfortran
```

```
File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/schemas/examples/settings.schema.json/valid/cobalt-example.yml
```

```
system:
  generic:
    hostnames: ['.*']

    moduletool: lmod
    load_default_buildspecs: True
    executors:
      defaults:
        launcher: qsub
        max_pend_time: 30

    cobalt:
      knl:
```

(continues on next page)

(continued from previous page)

```

    queue: knl

    haswell:
      queue: haswell

  compilers:
    compiler:
      gcc:
        default:
          cc: /usr/bin/gcc
          cxx: /usr/bin/g++
          fc: /usr/bin/gfortran
File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/schemas/examples/settings.schema.json/valid/pbs-example.yml

```

```

system:
  generic:
    hostnames: ['.*']

  moduletool: N/A
  load_default_buildspecs: True
  executors:
    defaults:
      pollinterval: 10
      launcher: qsub
      max_pend_time: 30
    pbs:
      workq:
        queue: workq
  compilers:
    compiler:
      gcc:
        default:
          cc: /usr/bin/gcc
          cxx: /usr/bin/g++
          fc: /usr/bin/gfortran
File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/schemas/examples/settings.schema.json/valid/lsf-example.yml

```

```

system:
  generic:
    hostnames: ['.*']

  moduletool: lmod
  load_default_buildspecs: False
  executors:
    defaults:
      pollinterval: 10
      launcher: bsub
      max_pend_time: 45
    lsf:
      batch:

```

(continues on next page)

(continued from previous page)

```

    description: "LSF Executor name 'batch' that submits jobs to 'batch' queue"
    queue: batch
    account: developer
    options: ["-W 20"]
    before_script: |
        time
        echo "commands run before job"
  test:
    description: "LSF Executor name 'test' that submits jobs to 'test' queue"
    launcher: bsub
    queue: test
    account: qa
    options: ["-W 20"]
  compilers:
    compiler:
      gcc:
        default:
          cc: /usr/bin/gcc
          cxx: /usr/bin/g++
          fc: /usr/bin/gfortran
File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↪ buildtest/schemas/examples/settings.schema.json/valid/local-executor.yml

```

```

system:
  generic:
    hostnames: ['.*']

    logdir: $BUILDTEST_ROOT/logs
    testdir: $BUILDTEST_ROOT/tests

    moduletool: N/A
    load_default_buildspecs: False
    cdash:
      url: https://my.cdash.org
      project: buildtest
      site: laptop
    processor:
      numcpus: 8
      cores: 4
      threads_per_core: 2
      sockets: 1
      model: "Intel(R) Core(TM) i7-8569U CPU @ 2.80GHz"
    executors:
      local:
        bash:
          description: submit jobs on local machine using bash shell
          shell: bash
          before_script: |
            time
            echo "commands run before job"

      sh:

```

(continues on next page)

(continued from previous page)

```

description: submit jobs on local machine using sh shell
shell: sh

csh:
description: submit jobs on local machine using csh shell
shell: csh -x

tcsh:
description: submit jobs on local machine using tcsh shell
shell: /bin/tcsh

zsh:
description: submit jobs on local machine using zsh shell
shell: /bin/zsh

python:
description: submit jobs on local machine using python shell
shell: python

compilers:
find:
gcc: "^(gcc|GCC|PrgEnv-gnu)"
intel: "^(intel|Intel|PrgEnv-intel)"
cray: "^(cray|PrgEnv-cray)"
clang: "^(clang|Clang)"
cuda: "^(cuda|CUDA)"
pgi: "^(pgi|PGI|PrgEnv-pgi)"

compiler:
gcc:
default:
cc: /usr/bin/gcc
cxx: /usr/bin/g++
fc: /usr/bin/gfortran
gcc@7.2.0:
cc: 'cc'
cxx: 'c++'
fc: 'fc'
module:
load:
- gcc/7.2.0
intel:
intel@2019:
cc: 'icc'
cxx: 'icpc'
fc: 'ifort'
module:
purge: True
load:
- gcc/7.2.0
- intel/2019
cray:

```

(continues on next page)

(continued from previous page)

```

craype@2.6.2:
  cc: 'cc'
  cxx: 'CC'
  fc: 'fc'
  module:
    load: [craype/2.6.2]
    swap: [PrgEnv-gnu, PrgEnv-cray]

clang:
  clang@12.0.0:
    cc: 'clang'
    cxx: 'clang++'
    fc: 'None'
    module:
      load: [clang/12.0]
  cuda:
    cuda@11.0:
      cc: 'nvcc'
      cxx: 'nvcc'
      fc: 'None'
      module:
        load: [cuda/11.0]
  pgi:
    pgi@18.0:
      cc: 'pgcc'
      cxx: 'pgc++'
      fc: 'pgfortran'
      module:
        load: [pgi/18.0]

```

File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
 ↪ buildtest/schemas/examples/settings.schema.json/valid/combined_executor.yml

```

system:
  generic:
    hostnames: ['.*']

    moduletool: N/A
    load_default_buildspecs: True
    executors:
      local:
        bash:
          description: submit jobs on local machine
          shell: bash -v

      slurm:
        haswell:
          launcher: sbatch
          options:
            - "-p haswell"
            - "-t 00:10"

```

(continues on next page)

(continued from previous page)

```
lsf:
  batch:
    launcher: bsub
    queue: batch
    options:
      - "-q batch"
      - "-t 00:10"
cobalt:
  normal:
    launcher: qsub
    queue: normal
    options:
      - "-n 1"
      - "-t 10"

compilers:
  compiler:
    gcc:
      default:
        cc: /usr/bin/gcc
        cxx: /usr/bin/g++
        fc: /usr/bin/gfortran
```

Global Schema

This schema is used for validating buildspec file and validates outer level structure of test. This is referred as *Global Schema*

Schema Content

```
$ buildtest schema -n global.schema.json --json
{
  "$id": "global.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "global schema",
  "description": "buildtest global schema is validated for all buildspecs. The global
↪ schema defines top-level structure of buildspec and definitions that are inherited for
↪ sub-schemas",
  "type": "object",
  "required": [
    "version",
    "buildspecs"
  ],
  "additionalProperties": false,
  "properties": {
    "version": {
      "type": "string",
      "description": "The semver version of the schema to use (x.x).",
    },
  },
}
```

(continues on next page)

(continued from previous page)

```

    "maintainers": {
      "type": "array",
      "description": "One or more maintainers or aliases",
      "uniqueItems": true,
      "minItems": 1,
      "items": {
        "type": "string"
      }
    },
    "buildspecs": {
      "type": "object",
      "description": "This section is used to define one or more tests (buildspecs).  

↳ Each test must be unique name",
      "propertyNames": {
        "pattern": "^[A-Za-z_.][A-Za-z0-9_.]*$",
        "maxLength": 32
      }
    }
  }
}

```

Schema Examples

```

$ buildtest schema -n global.schema.json --example
File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/schemas/examples/global.schema.json/valid/examples.yml

```

```

-----
version: "1.0"

```

```

buildspecs:

```

```

  # testing all caps
  ABCDEFGHIJKLMNOPQRSTUVWXYZ:
    type: script
    run: "hostname"

```

```

  # testing all lowercase letters
  abcdefghijklmnopqrstuvwxyz:
    type: script
    run: "hostname"

```

```

  # testing '_' in beginning followed by all numbers
  _0123456789:
    type: script
    run: "hostname"

```

```

  # testing '_' in combination with caps, lowercase and numbers
  _ABCDEFabcdef0123456789:
    type: script
    run: "hostname"

```

(continues on next page)

(continued from previous page)

```
# testing '_' at end of word
abcdefghijklmnpqrstuvwxyz_:
  type: script
  run: "hostname"

# testing '.' in beginning of word
.helloworld:
  type: script
  run: hostname

# testing '.' in middle of word
hello.world:
  type: script
  run: hostname

# testing '.' at end of word
helloworld.:
  type: script
  run: hostname
```

File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/schemas/examples/global.schema.json/invalid/missing-version.yml

```
buildspecs:
  # Shell would be accepted to indicate a single line shell command (or similar)
  login_node_check:
    type: script
    run: "ping login 1"
```

File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/schemas/examples/global.schema.json/invalid/exceed_testname_length.yml

```
# this test fails because it exceeds 32 character length for test name
version: "1.0"
buildspecs:
  this_test_exceeds_32_character_length:
    type: script
    run: hostname
```

File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/schemas/examples/global.schema.json/invalid/unique_maintainers.yml

```
version: "1.0"
maintainers: [shahzebsiddiqui, shahzebsiddiqui]
buildspecs:
  hostname:
    type: script
    run: "hostname"
```

File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/schemas/examples/global.schema.json/invalid/maintainers_type_mismatch.yml

```
version: "1.0"
# wrong type for maintainers key, expects a string
maintainers: 1
```

(continues on next page)

(continued from previous page)

```

buildspecs:
  hostname:
    type: script
    run: "hostname"
File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/schemas/examples/global.schema.json/invalid/invalid_pattern.yml
-----
version: "1.0"
buildspecs:
  # invalid pattern for test. Must be matching regex "[A-Za-z_.][A-Za-z0-9_]*$" when
  ↳ declaring a dict
  (badname:
    type: script
    run: "ping login 1"

```

Script Schema

This is the script schema used for writing scripts (bash, csh, sh, zsh, tcsh, python) and this is used for validating test instance when type: script is specified. For more details on script schema see [Script Schema](#).

Schema Content

```

$ buildtest schema -n script-v1.0.schema.json --json
{
  "$id": "script-v1.0.schema.json",
  "$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,
  "properties": {
    "type": {
      "type": "string",
      "pattern": "^script$",
      "description": "Select schema type to use when validating buildspec. This must be
      ↳ of set to 'script'"
    },
    "description": {
      "$ref": "definitions.schema.json#/definitions/description"
    },
    "sbatch": {
      "$ref": "definitions.schema.json#/definitions/list_of_strings",
      "description": "This field is used for specifying #SBATCH options in test script."
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

    },
    "bsub": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings",
        "description": "This field is used for specifying #BSUB options in test script."
    },
    "cobalt": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings",
        "description": "This field is used for specifying #COBALT options in test script."
    },
    "pbs": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings",
        "description": "This field is used for specifying #PBS directives in test script."
    },
    "batch": {
        "$ref": "definitions.schema.json#/definitions/batch"
    },
    "BB": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings",
        "description": "Create burst buffer space, this specifies #BB options in your test."
    },
    "DW": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings",
        "description": "Specify Data Warp option (#DW) when using burst buffer."
    },
    "env": {
        "$ref": "definitions.schema.json#/definitions/env"
    },
    "vars": {
        "$ref": "definitions.schema.json#/definitions/env"
    },
    "executor": {
        "$ref": "definitions.schema.json#/definitions/executor"
    },
    "run_only": {
        "$ref": "definitions.schema.json#/definitions/run_only"
    },
    "shell": {
        "type": "string",
        "description": "Specify a shell launcher to use when running jobs. This sets the
        ↪ shebang line in your test script. The ``shell`` key can be used with ``run`` section
        ↪ to describe content of script and how its executed",
        "pattern": "^(/bin/bash|/bin/sh|/bin/csh|/bin/tcsh|/bin/
        ↪ zsh|bash|sh|csh|tcsh|zsh|python).*"
    },
    "shebang": {
        "type": "string",
        "description": "Specify a custom shebang line. If not specified buildtest will
        ↪ automatically add it in the test script."
    },
    "run": {
        "type": "string",

```

(continues on next page)

(continued from previous page)

```

    "description": "A script to run using the default shell."
  },
  "status": {
    "$ref": "definitions.schema.json#/definitions/status"
  },
  "skip": {
    "$ref": "definitions.schema.json#/definitions/skip"
  },
  "tags": {
    "$ref": "definitions.schema.json#/definitions/tags"
  }
}
}

```

Schema Examples

```

$ buildtest schema -n script-v1.0.schema.json --example
File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
→ buildtest/schemas/examples/script-v1.0.schema.json/valid/examples.yml

```

```

-----
version: "1.0"
buildspecs:
  multiline_run:
    executor: generic.local.bash
    type: script
    description: multiline run command
    run: |
      echo "1"
      echo "2"

  single_command_run:
    executor: generic.local.bash
    type: script
    description: single command as a string for run command
    run: "hostname"

  declare_env:
    executor: generic.local.bash
    type: script
    description: declaring environment variables
    env:
      FOO: BAR
      X: 1
    run: |
      echo $FOO
      echo $X

  declare_vars:
    executor: generic.local.bash
    type: script

```

(continues on next page)

(continued from previous page)

```
description: declaring variables
vars:
  First: Bob
  Last:  Bill
run: |
  echo "First:" $First
  echo "Last:" $Last

declare_shell_sh:
  executor: generic.local.sh
  type: script
  description: declare shell name to sh
  shell: sh
  run: hostname

declare_shell_bash:
  executor: generic.local.bash
  type: script
  description: declare shell name to bash
  shell: bash
  run: hostname

declare_shell_python:
  executor: generic.local.python
  type: script
  description: declare shell name to python
  shell: python
  run: |
    print("Hello World")

declare_shell_bin_bash:
  executor: generic.local.bash
  type: script
  description: declare shell name to /bin/bash
  shell: "/bin/bash -e"
  run: hostname

declare_shell_name_bin_sh:
  executor: generic.local.sh
  type: script
  description: declare shell name to /bin/sh
  shell: "/bin/sh -e"
  run: hostname

declare_shell_opts:
  executor: generic.local.sh
  type: script
  description: declare shell name to sh
  shell: "sh -e"
  run: hostname
```

(continues on next page)

(continued from previous page)

```
declare_shell_bin_zsh:
  executor: generic.local.zsh
  type: script
  description: declare shell zsh
  shell: "zsh"
  run: hostname

declare_shell_zsh:
  executor: generic.local.zsh
  type: script
  description: declare shell /bin/zsh
  shell: "zsh"
  run: hostname

declare_shell_bin_csh:
  executor: generic.local.csh
  type: script
  description: declare shell /bin/csh
  shell: "/bin/csh"
  run: hostname

declare_shell_csh:
  executor: generic.local.csh
  type: script
  description: declare shell /bin/tcsh
  shell: "csh"
  run: hostname

declare_shell_bin_tcsh:
  executor: generic.local.csh
  type: script
  description: declare shell /bin/tcsh
  shell: "/bin/tcsh"
  run: hostname

declare_shell_tcsh:
  executor: generic.local.csh
  type: script
  description: declare shell tcsh
  shell: "tcsh"
  run: hostname

declare_shebang:
  executor: generic.local.bash
  type: script
  description: declare shell name to sh
  shebang: "#!/usr/bin/env bash"
  run: hostname

status_returncode_list:
  executor: generic.local.bash
```

(continues on next page)

(continued from previous page)

```

type: script
description: The returncode can be a list of integers
run: exit 0
status:
  returncode: [0]

status_returncode_int:
  executor: generic.local.bash
  type: script
  description: The returncode can be an integer to match with single returncode
  run: exit 0
  status:
    returncode: 0

status_regex:
  executor: generic.local.bash
  type: script
  description: This test pass with a regular expression status check
  run: hostname
  status:
    regex:
      stream: stdout
      exp: "^$"

status_regex_returncode:
  executor: generic.local.bash
  type: script
  description: This test fails because returncode and regex specified
  run: hostname
  status:
    returncode: [0]
    regex:
      stream: stdout
      exp: "^hello"

status_runtime_min_max:
  type: script
  executor: generic.local.sh
  description: "Run a sleep job for 2 seconds and test pass if its within 1.0-4.0sec"
  tags: ["tutorials"]
  run: sleep 2
  status:
    runtime:
      min: 1.0
      max: 4.0

status_runtime_min:
  type: script
  executor: generic.local.sh
  description: "Run a sleep job for 2 seconds and test pass if exceeds mintime of 1.
↪0sec"

```

(continues on next page)

(continued from previous page)

```

tags: ["tutorials"]
run: sleep 2
status:
  runtime:
    min: 1.0

status_runtime_max:
  type: script
  executor: generic.local.sh
  description: "Run a sleep job for 2 seconds and test pass if less than maxtime of 4.
↪0sec"
tags: ["tutorials"]
run: sleep 2
status:
  runtime:
    max: 4.0

sbatch_example:
  type: script
  executor: generic.local.bash
  description: This test runs hostname using sbatch directives
  sbatch:
    - "-t 10:00:00"
    - "-p normal"
    - "-N 1"
    - "-n 8"
  run: hostname

bsub_example:
  type: script
  executor: generic.local.bash
  description: This test runs hostname using bsub directives
  bsub:
    - "-W 00:30"
    - "-N 1"
  run: hostname

cobalt_example:
  type: script
  executor: generic.local.bash
  description: This test runs hostname using cobalt directives
  cobalt:
    - "-t 30"
    - "-n 1"
  run: hostname

skip_example:
  type: script
  executor: generic.local.bash
  description: this test is skip
  skip: true
  run: hostname

```

(continues on next page)

(continued from previous page)

```

tag_str_example:
  type: script
  executor: generic.local.bash
  description: tags can be defined as string
  tags: network
  run: hostname

tag_list_example:
  type: script
  executor: generic.local.bash
  description: This is a tag example using list
  sbatch:
    - "-t 10:00:00"
    - "-p normal"
    - "-N 1"
    - "-n 8"
  tags: ["slurm"]
  run: hostname

run_only_example:
  type: script
  executor: generic.local.bash
  description: run_only example that runs with user1 on Linux system (rhel, centos)↵
↵with LSF
  run_only:
    user: user1
    scheduler: lsf
    platform: Linux
    linux_distro:
      - rhel
      - centos
  run: |
    uname -av
    lsinfo
File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↵buildtest/schemas/examples/script-v1.0.schema.json/invalid/examples.yml

```

```

version: "1.0"
buildspecs:
  invalid_test_name_&!@#%$:
    type: script
    executor: generic.local.bash
    description: "invalid test name"

  invalid_bash:
    type: script
    executor: generic.local.bash
    shell: "bash-missing-run"

  missing_run_key:
    type: script

```

(continues on next page)

(continued from previous page)

```
executor: generic.local.bash
description: invalid key name roon, missing run key
roon: |
    systemctl is-active slurmd
    systemctl is-enabled slurmd | grep enabled

invalid_env_type:
  type: script
  executor: generic.local.bash
  description: env key should be a dictionary
  env:
    - FOO=BAR
  run: echo $FOO

invalid_vars_type:
  type: script
  executor: generic.local.bash
  description: var key should be a dictionary
  vars:
    - FOO=BAR
  run: echo $FOO

invalid_description:
  type: script
  executor: generic.local.bash
  description:
    - "Multi Line description"
    - "is not accepted"

invalid_regex_stream:
  type: script
  executor: generic.local.bash
  description: This test fails because of invalid regex stream
  run: hostname
  status:
    regex:
      stream: file
      exp: "world$"

missing_regex_exp:
  type: script
  executor: generic.local.bash
  description: This test fails because of missing key 'exp' in regex
  run: hostname
  status:
    regex:
      stream: stdout

invalid_returncode_type:
  type: script
  executor: generic.local.bash
```

(continues on next page)

(continued from previous page)

```
description: This test fails because of invalid return code type
run: hostname
status:
  returncode: ["1"]

empty_returncode_list:
  type: script
  executor: generic.local.bash
  description: An empty returncode list will cause an error
  run: hostname
  status:
    returncode: []

non_int_returncodes:
  type: script
  executor: generic.local.bash
  description: The returncode must be an int and not a number
  run: exit 1
  status:
    returncode: 1.01

non_int_returncodes_list:
  type: script
  executor: generic.local.bash
  description: The returncode must be a list of integers and no numbers
  run: exit 1
  status:
    returncode: [1, 2.230]

invalid_shell_usr_bin_bash:
  type: script
  executor: generic.local.bash
  description: invalid shell name '/usr/bin/bash'
  shell: /usr/bin/bash
  run: hostname

invalid_shell_type:
  type: script
  executor: generic.local.bash
  description: invalid shell type must be a string
  shell: ["/bin/bash"]
  run: hostname

invalid_type_shell_shebang:
  type: script
  executor: generic.local.bash
  description: invalid type for shell shebang, must be a string
  shebang: ["#!/bin/bash"]
  run: hostname

invalid_skip_value:
  type: script
```

(continues on next page)

(continued from previous page)

```

    executor: generic.local.bash
    description: invalid value for skip, must be boolean
    skip: 1
    run: hostname

empty_tags:
    type: script
    executor: generic.local.bash
    description: tag list can't be empty, requires one item.
    tags: []
    run: hostname

non_unique_tags:
    type: script
    executor: local.bash
    description: tag names must be unique
    tags: ["network", "network"]
    run: hostname

invalid_tags_value:
    type: script
    executor: generic.local.bash
    description: invalid tag value must be all string items
    tags: ["network", 400 ]
    run: hostname

additionalProperties_test:
    type: script
    executor: generic.local.bash
    description: additional properties are not allowed so any invalid key/value pair
↳ will result in error
    FOO: BAR
    run: hostname

additionalProperties_status:
    type: script
    executor: generic.local.bash
    description: test additional properties in status object. This is not allowed
    sbatch: [ "-n 2", "-q normal", "-t 10"]
    run: hostname
    status:
        slurm_job_state: "COMPLETED"
        FOO: BAR

invalid_runtime_min:
    type: script
    executor: generic.local.sh
    description: "Invalid type for min property in runtime. Must be an integer or float
↳ not a string"
    run: sleep 2
    status:
        runtime:

```

(continues on next page)

(continued from previous page)

```

        min: "1"

runtime_min_must_exceed_0:
    type: script
    executor: generic.local.sh
    description: "The runtime must exceed 0"
    run: sleep 2
    status:
        runtime:
            min: -1

invalid_slurm_job_state:
    type: script
    executor: generic.local.sh
    description: invalid value for slurm_job_state, should raise error with enum values.
    sbatch:
        - "-n 2"
        - "-q normal"
        - "-t 10"
    run: hostname
    status:
        slurm_job_state: "FINISH"

duplicate_linux_distro:
    type: script
    executor: generic.local.bash
    description: Duplicate items in linux_distro is not allowed
    run_only:
        linux_distro:
            - rhel
            - rhel
    run: uname -av

empty_list_linux_distro:
    type: script
    executor: generic.local.bash
    description: Empty List in linux_distro is not allowed. Requires atleast 1 item
    run_only:
        linux_distro: []
    run: uname -av

additionalProperties_run_only:
    type: script
    executor: generic.local.bash
    description: additional Properties not allowed in run_only field. Invalid field_
↪python
    run_only:
        user: root
        python: 3.5
    run: hostname

```

Compiler Schema

This is the compiler schema used for validating buildsspecs that define test using `type: compiler`. This schema is used for compiling a single source code. For more details see [Compiler Schema](#)

Schema Content

```
$ buildtest schema -n compiler-v1.0.schema.json --json
{
  "$id": "compiler-v1.0.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "compiler schema version 1.0",
  "description": "The compiler schema is of ``type: compiler`` in sub-schema which is
  ↪ used for compiling and running programs",
  "type": "object",
  "required": [
    "type",
    "source",
    "compilers",
    "executor"
  ],
  "definitions": {
    "cc": {
      "type": "string",
      "description": "Set C compiler wrapper"
    },
    "fc": {
      "type": "string",
      "description": "Set Fortran compiler wrapper"
    },
    "cxx": {
      "type": "string",
      "description": "Set C++ compiler wrapper"
    },
    "cflags": {
      "type": "string",
      "description": "Set C compiler flags."
    },
    "fflags": {
      "type": "string",
      "description": "Set Fortran compiler flags."
    },
    "cxxflags": {
      "type": "string",
      "description": "Set C++ compiler flags."
    },
    "ldflags": {
      "type": "string",
      "description": "Set linker flags"
    },
    "cppflags": {
      "type": "string",

```

(continues on next page)

(continued from previous page)

```

    "description": "Set C or C++ preprocessor flags"
  },
  "pre_build": {
    "type": "string",
    "description": "Run commands before building program"
  },
  "post_build": {
    "type": "string",
    "description": "Run commands after building program"
  },
  "pre_run": {
    "type": "string",
    "description": "Run commands before running program"
  },
  "post_run": {
    "type": "string",
    "description": "Run commands after running program"
  },
  "run": {
    "type": "string",
    "description": "Run command for launching compiled binary"
  },
  "default_compiler_all": {
    "type": "object",
    "description": "Specify compiler configuration for all compiler groups. Use the
    ↪ ``all`` property if configuration is shared across compiler groups. This property can
    ↪ be overridden in compiler group or named compiler in ``config`` section.",
    "additionalProperties": false,
    "properties": {
      "sbatch": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings"
      },
      "bsub": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings"
      },
      "cobalt": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings"
      },
      "pbs": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings"
      },
      "batch": {
        "$ref": "definitions.schema.json#/definitions/batch"
      },
      "BB": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings"
      },
      "DW": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings"
      },
      "env": {
        "$ref": "definitions.schema.json#/definitions/env"
      }
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

    },
    "vars": {
      "$ref": "definitions.schema.json#/definitions/env"
    },
    "status": {
      "$ref": "definitions.schema.json#/definitions/status"
    },
    "pre_build": {
      "$ref": "#definitions/pre_build"
    },
    "post_build": {
      "$ref": "#definitions/post_build"
    },
    "pre_run": {
      "$ref": "#definitions/pre_run"
    },
    "post_run": {
      "$ref": "#definitions/post_run"
    },
    "run": {
      "$ref": "#definitions/run"
    }
  }
},
"default_compiler_config": {
  "type": "object",
  "description": "Specify compiler configuration for group of compilers. Use this_
↪property if you want to define common configuration for all compilers of same group._
↪This property overrides ``all`` property. ",
  "properties": {
    "cc": {
      "$ref": "#definitions/cc"
    },
    "fc": {
      "$ref": "#definitions/fc"
    },
    "cxx": {
      "$ref": "#definitions/cxx"
    },
    "cflags": {
      "$ref": "#definitions/cflags"
    },
    "fflags": {
      "$ref": "#definitions/fflags"
    },
    "cxxflags": {
      "$ref": "#definitions/cxxflags"
    },
    "ldflags": {
      "$ref": "#definitions/ldflags"
    },
    "cppflags": {

```

(continues on next page)

(continued from previous page)

```

    "$ref": "#definitions/cppflags"
  },
  "sbatch": {
    "$ref": "definitions.schema.json#/definitions/list_of_strings"
  },
  "bsub": {
    "$ref": "definitions.schema.json#/definitions/list_of_strings"
  },
  "cobalt": {
    "$ref": "definitions.schema.json#/definitions/list_of_strings"
  },
  "pbs": {
    "$ref": "definitions.schema.json#/definitions/list_of_strings"
  },
  "batch": {
    "$ref": "definitions.schema.json#/definitions/batch"
  },
  "BB": {
    "$ref": "definitions.schema.json#/definitions/list_of_strings"
  },
  "DW": {
    "$ref": "definitions.schema.json#/definitions/list_of_strings"
  },
  "env": {
    "$ref": "definitions.schema.json#/definitions/env"
  },
  "vars": {
    "$ref": "definitions.schema.json#/definitions/env"
  },
  "status": {
    "$ref": "definitions.schema.json#/definitions/status"
  },
  "pre_build": {
    "$ref": "#definitions/pre_build"
  },
  "post_build": {
    "$ref": "#definitions/post_build"
  },
  "pre_run": {
    "$ref": "#definitions/pre_run"
  },
  "post_run": {
    "$ref": "#definitions/post_run"
  },
  "run": {
    "$ref": "#definitions/run"
  }
},
"compiler_declaration": {
  "type": "object",
  "description": "Specify compiler configuration at compiler level. The ``config``
  section has highest precedence when searching compiler configuration. This overrides
  fields found in compiler group and ``all`` property",

```

(continued from previous page)

```

"additionalProperties": false,
"properties": {
  "cc": {
    "$ref": "#definitions/cc"
  },
  "fc": {
    "$ref": "#definitions/fc"
  },
  "cxx": {
    "$ref": "#definitions/cxx"
  },
  "cflags": {
    "$ref": "#definitions/cflags"
  },
  "fflags": {
    "$ref": "#definitions/fflags"
  },
  "cxxflags": {
    "$ref": "#definitions/cxxflags"
  },
  "ldflags": {
    "$ref": "#definitions/ldflags"
  },
  "cppflags": {
    "$ref": "#definitions/cppflags"
  },
  "sbatch": {
    "$ref": "definitions.schema.json#/definitions/list_of_strings"
  },
  "bsub": {
    "$ref": "definitions.schema.json#/definitions/list_of_strings"
  },
  "cobalt": {
    "$ref": "definitions.schema.json#/definitions/list_of_strings"
  },
  "pbs": {
    "$ref": "definitions.schema.json#/definitions/list_of_strings"
  },
  "batch": {
    "$ref": "definitions.schema.json#/definitions/batch"
  },
  "BB": {
    "$ref": "definitions.schema.json#/definitions/list_of_strings"
  },
  "DW": {
    "$ref": "definitions.schema.json#/definitions/list_of_strings"
  },
  "env": {
    "$ref": "definitions.schema.json#/definitions/env"
  },
  "vars": {
    "$ref": "definitions.schema.json#/definitions/env"
  }
}

```

(continues on next page)

(continued from previous page)

```

    },
    "status": {
      "$ref": "definitions.schema.json#/definitions/status"
    },
    "pre_build": {
      "$ref": "#definitions/pre_build"
    },
    "post_build": {
      "$ref": "#definitions/post_build"
    },
    "pre_run": {
      "$ref": "#definitions/pre_run"
    },
    "post_run": {
      "$ref": "#definitions/post_run"
    },
    "run": {
      "$ref": "#definitions/run"
    },
    "module": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "purge": {
          "type": "boolean",
          "description": "Run ``module purge`` if purge is set"
        },
        "load": {
          "$ref": "definitions.schema.json#/definitions/list_of_strings",
          "description": "Load one or more modules via ``module load``"
        },
        "restore": {
          "description": "Load a collection name via ``module restore``",
          "type": "string"
        },
        "swap": {
          "description": "Swap modules using ``module swap``. The swap property_
↪ expects 2 unique modules.",
          "type": "array",
          "uniqueItems": true,
          "minItems": 2,
          "maxItems": 2,
          "items": {
            "type": "string"
          }
        }
      }
    },
    "additionalProperties": false,

```

(continues on next page)

(continued from previous page)

```

"properties": {
  "type": {
    "type": "string",
    "pattern": "^compiler$",
    "description": "Select schema type to use when validating buildspec. This must be
↳ of set to ``compiler``."
  },
  "description": {
    "$ref": "definitions.schema.json#/definitions/description"
  },
  "compilers": {
    "type": "object",
    "required": [
      "name"
    ],
    "additionalProperties": false,
    "properties": {
      "name": {
        "description": "Specify a list of regular expression to search compiler
↳ instance from buildtest settings.",
        "$ref": "definitions.schema.json#/definitions/list_of_strings"
      },
      "exclude": {
        "description": "Specify a list of named compilers to exclude when building
↳ test based on regular expression specified in ``name`` property. The ``exclude``
↳ property has no effect if named compiler not found based on regular expression.",
        "$ref": "definitions.schema.json#/definitions/list_of_strings"
      },
      "default": {
        "type": "object",
        "additionalProperties": false,
        "properties": {
          "all": {
            "$ref": "#definitions/default_compiler_all"
          },
          "gcc": {
            "$ref": "#definitions/default_compiler_config"
          },
          "intel": {
            "$ref": "#definitions/default_compiler_config"
          },
          "pgi": {
            "$ref": "#definitions/default_compiler_config"
          },
          "cray": {
            "$ref": "#definitions/default_compiler_config"
          },
          "clang": {
            "$ref": "#definitions/default_compiler_config"
          },
          "cuda": {
            "$ref": "#definitions/default_compiler_config"
          }
        }
      }
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

    },
    "upcxx": {
      "$ref": "#definitions/default_compiler_config"
    }
  },
  "config": {
    "type": "object",
    "description": "Specify compiler configuration based on named compilers.",
    "patternProperties": {
      "^.*$": {
        "$ref": "#definitions/compiler_declaration"
      }
    }
  }
},
"source": {
  "type": "string",
  "description": "Specify a source file for compilation, the file can be relative,
↳ path to buildspect or an absolute path"
},
"executor": {
  "$ref": "definitions.schema.json#/definitions/executor"
},
"run_only": {
  "$ref": "definitions.schema.json#/definitions/run_only"
},
"skip": {
  "$ref": "definitions.schema.json#/definitions/skip"
},
"tags": {
  "$ref": "definitions.schema.json#/definitions/tags"
}
}
}

```

Schema Examples

```

$ buildtest schema -n compiler-v1.0.schema.json --example
File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
↳ buildtest/schemas/examples/compiler-v1.0.schema.json/valid/examples.yml

```

```

version: "1.0"
buildspecs:
  gnu_example:
    executor: local.bash
    type: compiler
    description: "gnu example with modules, and cflags example"
    source: src/hello.c

```

(continues on next page)

(continued from previous page)

```

compilers:
  name: [gcc]
  config:
    gcc@8.4.0:
      cflags: "-O3"

intel_example:
  executor: local.bash
  type: compiler
  description: "intel example using cflags"
  source: src/hello.c
  compilers:
    name: [intel]
    config:
      intel@2018:
        cflags: "-O1"

clang_example:
  executor: local.bash
  type: compiler
  description: "clang example using cflags"
  source: src/hello.c
  compilers:
    name: [clang]
    default:
      clang:
        cflags: "-O1"
    config:
      clang@11:
        cflags: "-O2"

upcxx_example:
  executor: local.bash
  type: compiler
  description: "upcxx compiler declaration in default and config section "
  source: src/hello.c
  compilers:
    name: [upcxx]
    default:
      upcxx:
        cflags: "-g aries"
    config:
      upcxx@2020:
        cflags: "-O1 -g aries"

pgi_example:
  executor: local.bash
  type: compiler
  description: "pgi example using cxxflags, ldflags in default and config section"
  source: src/hello.cpp
  compilers:
    name: ["^(pgi|PrgEnv)"]

```

(continues on next page)

(continued from previous page)

```

default:
  pgi:
    cxxflags: "-O1"
    ldflags: "-lm"
  config:
    pgi@18.1:
      module:
        swap: [PrgEnv-gnu, PrgEnv-pgi]
        load: [pgi/18.1]
    pgi@18.2:
      module:
        swap: [PrgEnv-gnu, PrgEnv-pgi]
        load: [pgi/18.2]

cray_example:
  executor: local.bash
  type: compiler
  description: "cray example using fflags and cppflags"
  source: src/hello.f90
  compilers:
    name: ["PrgEnv-cray"]
    default:
      cray:
        fflags: "-O1"
    config:
      PrgEnv-cray@2.6.2:
        module:
          swap: [PrgEnv-intel, PrgEnv-cray/2.6.2]

sbatch_example_all_compiler_groups:
  type: compiler
  description: sbatch example to for all compiler groups
  executor: local.bash
  source: src/hello.f90
  compilers:
    name: ["PrgEnv-cray"]
    default:
      cray:
        fflags: "-O1"
      all:
        sbatch: ["-t 10", "-n 2", "-C haswell" ]
    config:
      PrgEnv-cray@2.6.2:
        module:
          swap: [PrgEnv-intel, PrgEnv-cray/2.6.2]

bsub_all_compiler_groups:
  type: compiler
  description: bsub example for all compiler groups
  executor: local.bash
  source: "src/hello.cpp"

```

(continues on next page)

(continued from previous page)

```

compilers:
  name: [intel]
  default:
    all:
      bsub: ["-W 00:30", "-n 2"]
  config:
    intel@2019:
      cxxflags: "-O1"

cobalt_all_compiler_groups:
  type: compiler
  description: cobalt example for all compiler groups
  executor: local.bash
  source: "src/hello.cpp"
  compilers:
    name: [intel]
    default:
      all:
        cobalt: ["-t 30", "-n 1"]
  config:
    intel@2019:
      cxxflags: "-O1"

sbatch_compiler_group:
  type: compiler
  description: sbatch example in multiple compiler groups.
  executor: local.bash
  source: src/hello.f90
  compilers:
    name: ["^(gcc|intel)"]
    default:
      gcc:
        fflags: "-O1"
        sbatch: ["-t 10", "-n 2", "-C haswell" ]
      intel:
        fflags: "-O2"
        sbatch: ["-t 10", "-n 2", "-C knl" ]
  config:
    gcc@8.1.0:
      sbatch: ["-t 60", "-n 2", "-C knl"]
    module:
      swap: [PrgEnv-intel, PrgEnv-gnu/6.1.0]

bsub_compiler_group:
  type: compiler
  description: bsub example in multiple compiler groups.
  executor: local.bash
  source: src/hello.f90
  compilers:
    name: ["^(gcc|intel)"]
    default:

```

(continues on next page)

(continued from previous page)

```

gcc:
  fflags: "-O1"
  bsub: ["-W 00:30", "-n 2" ]
intel:
  fflags: "-O2"
  bsub: ["-W 00:30", "-n 4" ]
config:
gcc@8.1.0:
  bsub: ["-W 00:30", "-n 6" ]
  module:
    swap: [PrgEnv-intel, PrgEnv-gnu/6.1.0]

batch_example:
  type: compiler
  description: example using batch field
  executor: local.bash
  source: "src/hello.cpp"
  compilers:
    name: [intel]
    default:
      all:
        batch:
          "timelimit": "30"
          "nodecount": "2"
          "queue": "batch"
          "account": "biology"
    config:
      intel@2019:
        cxxflags: "-O1"

env_example:
  type: compiler
  description: Setting environment variables
  executor: local.bash
  source: "src/hello.cpp"
  compilers:
    name: ["^(gcc)"]
    default:
      all:
        env:
          OMP_NUM_THREADS: 2
          run: $_EXEC 1 2 4
    config:
      gcc@10.2.0:
        cxxflags: "-fopenmp"

custom_env_by_compiler_group:
  type: compiler
  description: Setting environment variables in compiler groups
  executor: local.bash
  source: "src/hello.cpp"
  compilers:

```

(continues on next page)

(continued from previous page)

```

name: ["^(gcc|pgi)"]
default:
  all:
    run: $_EXEC 1 2 4
  gcc:
    cxxflags: "-fopenmp"
    env:
      OMP_NUM_THREADS: 4
  pgi:
    cxxflags: "-mp"
    env:
      OMP_NUM_THREADS: 6
config:
  gcc@10.2.0:
    env:
      OMP_NUM_THREADS: 6

  gcc@9.2.0:
    env:
      OMP_NUM_THREADS: 8

  pgi@9.2.0:
    env:
      OMP_NUM_THREADS: 10

vars_example:
  type: compiler
  description: Setting shell variables
  executor: local.bash
  source: "src/hello.cpp"
  compilers:
    name: ["^(gcc)"]
    default:
      all:
        vars:
          OUTFILE: /tmp/file1.txt
        run: $_EXEC > $OUTFILE
    config:
      gcc@10.2.0:
        cxxflags: "-fopenmp"

pass_args_run:
  type: compiler
  description: Passing arguments to executable in run section
  executor: local.bash
  source: "src/hello.cpp"
  compilers:
    name: [intel]
    default:
      all:
        run: $_EXEC 1 2 4
    config:

```

(continues on next page)

(continued from previous page)

```

    intel@2019:
        cxxflags: "-O1"

mpi_launcher_example:
    type: compiler
    description: mpi launcher example
    executor: local.bash
    source: "src/hello.cpp"
    compilers:
        name: [gcc]
        default:
            all:
                run: mpirun -np 2 $_EXEC
    config:
        gcc@7.3.0:
            cflags: "-O3"
            cxx: mpicxx

status_returncode_example:
    type: compiler
    description: Status returncode match example
    executor: local.bash
    source: "src/hello.cpp"
    compilers:
        name: [gnu]
        default:
            all:
                vars:
                    OUTFILE: /tmp/file1.txt
                run: $_EXEC > $OUTFILE
                status:
                    returncode: 1
    config:
        gcc@10.2.0:
            cxxflags: "-fopenmp"

pre_post_build_run_sections:
    type: compiler
    description: Run commands pre and post build section
    executor: local.bash
    source: "src/hello.cpp"
    compilers:
        name: ["^(gcc)"]
        default:
            all:
                pre_build: echo "pre-build section for ALL compilers"
                post_build: echo "post-build section for ALL Compilers"
                pre_run: echo "pre-run section for ALL compilers"
                post_run: echo "post-run section for ALL Compilers"
            gcc:
                pre_build: echo "pre-build section for GCC compilers"
                post_build: echo "post-build section for GCC compilers"

```

(continues on next page)

(continued from previous page)

```

    pre_run: echo "pre-run section for ALL compilers"
    post_run: echo "post-run section for ALL Compilers"
config:
  gcc@7.3.0:
    pre_build: echo "pre-build section for gcc@7.3.0"
    post_build: echo "post-build section for gcc@7.3.0"
    pre_run: echo "pre-run section for ALL compilers"
    post_run: echo "post-run section for ALL Compilers"
    cflags: "-O3"
  gcc@8.2.0:
    pre_build: echo "gcc --version"
    cflags: "-O3"

multi_compilers:
  type: compiler
  description: Select one or more compilers to run test
  executor: local.bash
  source: "src/hello.cpp"
  compilers:
    name: ["^(gcc|intel|pgi|cray)"]
    exclude: [intel@18]
    default:
      gcc:
        cflags: "-fopenmp"
      intel:
        cflags: "-qopenmp"
      pgi:
        cflags: "-fopenmp"
      cray:
        cflags: "-h omp"
  config:
    gcc@7.5.0:
      cflags: "-O3"
      module:
        load: [gcc/7.5.0]
    intel@17:
      module:
        load: [intel/2017]
    intel@18:
      module:
        load: [intel/2018]
    pgi/18.0:
      module:
        load: [pgi/18.0]
    craype/2.6.2:
      module:
        swap: [PrgEnv-intel, PrgEnv-cray]
        load: [craype/2.6.2]

```

File: /home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/
 ↪ buildtest/schemas/examples/compiler-v1.0.schema.json/invalid/examples.yml

(continues on next page)

(continued from previous page)

```
-----
version: "1.0"
buildspecs:
  missing_type:
    executor: local.bash
    description: "type key is missing, this is a required field"
    source: "src/hello.c"
    compilers:
      name: [intel]

  missing_required_compilers:
    executor: local.bash
    type: compiler
    description: "missing required field compilers "
    source: "src/hello.c"

  missing_required_source:
    executor: local.bash
    type: compiler
    description: "missing required field 'source' "
    compilers:
      name: [gcc]

  invalid_type_value:
    executor: local.bash
    type: script
    description: "invalid value for type field must be 'compiler' "
    source: src/hello.c
    compilers:
      name: [gcc]

  invalid_description_value:
    executor: local.bash
    type: compiler
    description: 1
    source: src/hello.c
    compilers:
      name: [gcc]

  invalid_type_module:
    executor: local.bash
    type: compiler
    description: "type for 'module' key, expecting a property but received 'string' "
    source: src/hello.c
    compilers:
      name: [gcc]
    config:
      gcc/9.2.0:
        module: "module load gcc/9.2.0"

  module_purge_invalid_type:
    executor: local.bash
```

(continues on next page)

(continued from previous page)

```
type: compiler
description: "The purge property module is invalid. Expects bool got an int"
source: src/hello.c
compilers:
  name: [gcc]
  config:
    gcc/9.2.0:
      module:
        purge: 1

module_swap_duplicate_check:
  executor: local.bash
  type: compiler
  description: "The swap property expects two unique items"
  source: src/hello.c
  compilers:
    name: [gcc]
    config:
      gcc/9.2.0:
        module:
          swap: [gcc/8.0, gcc/8.0]

module_swap_min_items:
  executor: local.bash
  type: compiler
  description: "The swap property expects a minimum of 2 items"
  source: src/hello.c
  compilers:
    name: [gcc]
    config:
      gcc/9.2.0:
        module:
          swap: [gcc/8.0]

module_swap_max_items:
  executor: local.bash
  type: compiler
  description: "The swap property expects a maximum of 2 items"
  source: src/hello.c
  compilers:
    name: [gcc]
    config:
      gcc/9.2.0:
        module:
          swap: [gcc/8.0, gcc/9.0, gcc/10.0]

module_load_duplicate_items:
  executor: local.bash
  type: compiler
  description: "The load property expects unique items"
  source: src/hello.c
  compilers:
```

(continues on next page)

(continued from previous page)

```
name: [gcc]
config:
  gcc/9.2.0:
    module:
      load: [gcc/9.2.0, gcc/9.2.0]

module_load_min_items:
  executor: local.bash
  type: compiler
  description: "The load property expects a minimum of 1 item"
  source: src/hello.c
  compilers:
    name: [gcc]
    config:
      gcc/9.2.0:
        module:
          load: []

additionalProperties_main:
  executor: local.bash
  type: compiler
  description: "test additionalProperties in main schema"
  foo: bar
  source: src/hello.c
  compilers:
    name: [gcc]

missing_required_compiler_name:
  executor: local.bash
  type: compiler
  description: "'name' field in compilers section is required field"
  source: src/hello.f90
  compilers:
  default:
    cray:
      fflags: "-O1"
  config:
    PrgEnv-cray@2.6.2:
      module:
        swap: [PrgEnv-intel, PrgEnv-cray/2.6.2]

uniqueItems_compiler_name:
  executor: local.bash
  type: compiler
  description: "Test unique items in 'name' field in compilers section"
  source: src/hello.f90
  compilers:
    name: ["^(PrgEnv-cray)", "^(PrgEnv-cray)"]
    config:
      PrgEnv-cray@2.6.2:
        fflags: "-O1"
        module:
```

(continues on next page)

(continued from previous page)

```

        swap: [PrgEnv-intel, PrgEnv-cray/2.6.2]

additionalProperties_compiler:
  executor: local.bash
  type: compiler
  description: "Test additionalProperties in compiler section"
  source: src/hello.f90
  compilers:
    name: ["PrgEnv-cray"]
    FOO: BAR
    default:
      all:
        env:
          X: 1
    config:
      PrgEnv-cray@2.6.2:
        fflags: "-O1"
        module:
          swap: [PrgEnv-intel, PrgEnv-cray/2.6.2]

additionalProperties_compiler_default_all:
  executor: local.bash
  type: compiler
  description: "Test additionalProperties in compiler default all section"
  source: src/hello.f90
  compilers:
    name: ["PrgEnv-cray"]
    default:
      all:
        XYZ: 123
    config:
      PrgEnv-cray@2.6.2:
        fflags: "-O1"
        module:
          swap: [PrgEnv-intel, PrgEnv-cray/2.6.2]

additionalProperties_compiler_config:
  executor: local.bash
  type: compiler
  description: "Test additionalProperties in compiler config section, FOO: BAR"
  source: src/hello.f90
  compilers:
    name: ["PrgEnv-cray"]
    config:
      PrgEnv-cray@2.6.2:
        FOO: BAR
        fflags: "-O1"
        module:
          swap: [PrgEnv-intel, PrgEnv-cray/2.6.2]

```

Spack Schema

This schema is used for writing tests with `spack package manager` using `type: spack` field. For more details see [Spack Schema](#).

Schema Content

```
$ buildtest schema -n spack-v1.0.schema.json --json
{
  "$id": "spack-v1.0.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "spack schema version 1.0",
  "description": "The spack schema is referenced using ``type: spack`` which is used for
↳ generating tests using spack package manager",
  "type": "object",
  "required": [
    "type",
    "executor",
    "spack"
  ],
  "additionalProperties": false,
  "properties": {
    "type": {
      "type": "string",
      "pattern": "^spack$",
      "description": "Select schema type to use when validating buildspec. This must be
↳ set to 'spack'"
    },
    "description": {
      "$ref": "definitions.schema.json#/definitions/description"
    },
    "executor": {
      "$ref": "definitions.schema.json#/definitions/executor"
    },
    "sbatch": {
      "$ref": "definitions.schema.json#/definitions/list_of_strings",
      "description": "This field is used for specifying #SBATCH options in test script."
    },
    "bsub": {
      "$ref": "definitions.schema.json#/definitions/list_of_strings",
      "description": "This field is used for specifying #BSUB options in test script."
    },
    "cobalt": {
      "$ref": "definitions.schema.json#/definitions/list_of_strings",
      "description": "This field is used for specifying #COBALT options in test script."
    },
    "pbs": {
      "$ref": "definitions.schema.json#/definitions/list_of_strings",
      "description": "This field is used for specifying #PBS directives in test script."
    },
    "batch": {
      "$ref": "definitions.schema.json#/definitions/batch"
```

(continues on next page)

(continued from previous page)

```

},
"skip": {
  "$ref": "definitions.schema.json#/definitions/skip"
},
"tags": {
  "$ref": "definitions.schema.json#/definitions/tags"
},
"pre_cmds": {
  "type": "string",
  "description": "Shell commands run before spack"
},
"post_cmds": {
  "type": "string",
  "description": "Shell commands run after spack"
},
"status": {
  "$ref": "definitions.schema.json#/definitions/status"
},
"spack": {
  "type": "object",
  "description": "Entry point to spack configuration",
  "required": [
    "root"
  ],
  "additionalProperties": false,
  "properties": {
    "root": {
      "type": "string"
    },
    "compiler_find": {
      "type": "boolean",
      "description": "Run ``spack compiler find`` if set to ``True``. This is run_
↳right after sourcing spack startup script."
    },
    "mirror": {
      "$ref": "definitions.schema.json#/definitions/env",
      "description": "Add mirror by running ``spack mirror add``"
    },
    "env": {
      "$ref": "#definitions/env",
      "description": "Manage spack environments via ``spack env`` command"
    },
    "install": {
      "$ref": "#definitions/install",
      "description": "Install spack packages by running ``spack install``. "
    },
    "verify_spack": {
      "type": "boolean",
      "description": "This boolean will determine if we need to check for file_
↳existence where spack is cloned via ``root`` property and file **$SPACK_ROOT/share/_
↳spack/setup-env.sh** exists. These checks can be disabled by setting this to ``False``_
↳which can be useful if you dont want buildtest to raise exception during test_
↳generation process and test is skipped."
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

        "default": true
    },
    "test": {
        "$ref": "#definitions/test",
        "description": "Entry point to ``spack test``"
    }
}
},
"definitions": {
    "env": {
        "additionalProperties": false,
        "type": "object",
        "description": "Used for managing spack environment using ``spack env`` command. ",
        "properties": {
            "create": {
                "additionalProperties": false,
                "description": "Create a spack environment via ``spack env create``,
↪environment. If set to ``True`` we will run ``spack env rm -y <name>``,
                "type": "object",
                "properties": {
                    "remove_environment": {
                        "type": "boolean",
                        "description": "Remove existing spack environment before creating new
↪environment. If set to ``True`` we will run ``spack env rm -y <name>``,
                        "default": false
                    },
                    "name": {
                        "type": "string",
                        "description": "Name of spack environment to create"
                    },
                    "manifest": {
                        "type": "string",
                        "description": "Specify path to spack manifest file (``spack.yaml`` or
↪``spack.lock``) when creating environment"
                    },
                    "options": {
                        "type": "string",
                        "description": "Pass options to ``spack env create`` command"
                    },
                    "dir": {
                        "type": "string",
                        "description": "Create a spack environment in a specific directory. This
↪will run ``spack env create -d <dir>``. Directory path does not have to exist prior to
↪execution however user must have appropriate ACL in-order to create directory."
                    }
                }
            },
            "activate": {
                "additionalProperties": false,
                "type": "object",
                "description": "Activate a spack environment via ``spack env activate``,
                "properties": {

```

(continues on next page)

(continued from previous page)

```

    "name": {
      "type": "string",
      "description": "Name of spack environment to activate. In order to
↳ activate spack environment ``my-project`` you need to run ``spack env activate my-
↳ project`` which is specified by ``name: my-project``."
    },
    "options": {
      "type": "string",
      "description": "Pass options to ``spack env activate`` command"
    },
    "dir": {
      "type": "string",
      "description": "Activate spack environment from directory."
    }
  },
  "rm": {
    "additionalProperties": false,
    "description": "Remove an existing spack environment via ``spack env rm``.",
    "type": "object",
    "required": [
      "name"
    ],
    "properties": {
      "name": {
        "type": "string",
        "description": "Remove spack environment by name. This will run ``spack
↳ env rm -y <name>``."
      }
    }
  },
  "mirror": {
    "$ref": "definitions.schema.json#/definitions/env",
    "description": "Add mirror in spack environment by running ``spack mirror add``
↳ "
  },
  "specs": {
    "$ref": "definitions.schema.json#/definitions/list_of_strings",
    "description": "Add specs to environment by running ``spack add <specs>``. The
↳ ``specs`` is a list of string which expect the argument to be name of spack package."
  },
  "concretize": {
    "type": "boolean",
    "description": "If ``concretize: true`` is set, we will concretize spack
↳ environment by running ``spack concretize -f`` otherwise this line will be ignored."
  }
},
"install": {
  "description": "Install spack packages using ``spack install`` command",
  "additionalProperties": false,
  "type": "object",

```

(continues on next page)

(continued from previous page)

```

    "properties": {
      "options": {
        "type": "string",
        "description": "Pass options to ``spack install`` command"
      },
      "specs": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings",
        "description": "List of specs to install using ``spack install`` command"
      }
    },
    "test": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "run"
      ],
      "properties": {
        "remove_tests": {
          "type": "boolean",
          "description": "Remove all test suites in spack before running test via
↳ ``spack test run``. If set to ``True`` we will run ``spack test remove -y`` which will
↳ remove all test suites."
        },
        "run": {
          "description": "Run tests using spack via ``spack test run`` command. This
↳ command requires specs are installed in your spack instance prior to running tests.",
          "type": "object",
          "required": [
            "specs"
          ],
          "additionalProperties": false,
          "properties": {
            "option": {
              "type": "string",
              "description": "Pass options to ``spack test run``"
            },
            "specs": {
              "$ref": "definitions.schema.json#/definitions/list_of_strings",
              "description": "List of specs to run tests by running ``spack test run
↳ <specs>``."
            }
          }
        },
        "results": {
          "type": "object",
          "description": "View test results via ``spack test results`` after running
↳ tests via ``spack test run``. Results can be viewed using suite name or installed specs
↳ or both.",
          "additionalProperties": false,
          "anyOf": [
            {

```

(continues on next page)

(continued from previous page)

```

    "executor",
    "spack"
  ],
  "additionalProperties": false,
  "properties": {
    "type": {
      "type": "string",
      "pattern": "^spack$",
      "description": "Select schema type to use when validating buildspec. This must be
↪set to 'spack'"
    },
    "description": {
      "$ref": "definitions.schema.json#/definitions/description"
    },
    "executor": {
      "$ref": "definitions.schema.json#/definitions/executor"
    },
    "sbatch": {
      "$ref": "definitions.schema.json#/definitions/list_of_strings",
      "description": "This field is used for specifying #SBATCH options in test script."
    },
    "bsub": {
      "$ref": "definitions.schema.json#/definitions/list_of_strings",
      "description": "This field is used for specifying #BSUB options in test script."
    },
    "cobalt": {
      "$ref": "definitions.schema.json#/definitions/list_of_strings",
      "description": "This field is used for specifying #COBALT options in test script."
    },
    "pbs": {
      "$ref": "definitions.schema.json#/definitions/list_of_strings",
      "description": "This field is used for specifying #PBS directives in test script."
    },
    "batch": {
      "$ref": "definitions.schema.json#/definitions/batch"
    },
    "skip": {
      "$ref": "definitions.schema.json#/definitions/skip"
    },
    "tags": {
      "$ref": "definitions.schema.json#/definitions/tags"
    },
    "pre_cmds": {
      "type": "string",
      "description": "Shell commands run before spack"
    },
    "post_cmds": {
      "type": "string",
      "description": "Shell commands run after spack"
    },
    "status": {
      "$ref": "definitions.schema.json#/definitions/status"
    }
  }

```

(continues on next page)

(continued from previous page)

```

    },
    "spack": {
      "type": "object",
      "description": "Entry point to spack configuration",
      "required": [
        "root"
      ],
      "additionalProperties": false,
      "properties": {
        "root": {
          "type": "string"
        },
        "compiler_find": {
          "type": "boolean",
          "description": "Run ``spack compiler find`` if set to ``True``. This is run_
↳right after sourcing spack startup script."
        },
        "mirror": {
          "$ref": "definitions.schema.json#/definitions/env",
          "description": "Add mirror by running ``spack mirror add``"
        },
        "env": {
          "$ref": "#definitions/env",
          "description": "Manage spack environments via ``spack env`` command"
        },
        "install": {
          "$ref": "#definitions/install",
          "description": "Install spack packages by running ``spack install``. "
        },
        "verify_spack": {
          "type": "boolean",
          "description": "This boolean will determine if we need to check for file_
↳existence where spack is cloned via ``root`` property and file **$SPACK_ROOT/share/_
↳spack/setup-env.sh** exists. These checks can be disabled by setting this to ``False``_
↳which can be useful if you dont want buildtest to raise exception during test_
↳generation process and test is skipped.",
          "default": true
        },
        "test": {
          "$ref": "#definitions/test",
          "description": "Entry point to ``spack test``"
        }
      }
    },
    "definitions": {
      "env": {
        "additionalProperties": false,
        "type": "object",
        "description": "Used for managing spack environment using ``spack env`` command. ",
        "properties": {
          "create": {

```

(continues on next page)

(continued from previous page)

```

    "additionalProperties": false,
    "description": "Create a spack environment via ``spack env create``,
    "type": "object",
    "properties": {
      "remove_environment": {
        "type": "boolean",
        "description": "Remove existing spack environment before creating new
↪environment. If set to ``True`` we will run ``spack env rm -y <name>``,
        "default": false
      },
      "name": {
        "type": "string",
        "description": "Name of spack environment to create"
      },
      "manifest": {
        "type": "string",
        "description": "Specify path to spack manifest file (``spack.yaml`` or
↪``spack.lock``) when creating environment"
      },
      "options": {
        "type": "string",
        "description": "Pass options to ``spack env create`` command"
      },
      "dir": {
        "type": "string",
        "description": "Create a spack environment in a specific directory. This
↪will run ``spack env create -d <dir>``. Directory path does not have to exist prior to
↪execution however user must have appropriate ACL in-order to create directory."
      }
    }
  },
  "activate": {
    "additionalProperties": false,
    "type": "object",
    "description": "Activate a spack environment via ``spack env activate``,
    "properties": {
      "name": {
        "type": "string",
        "description": "Name of spack environment to activate. In order to
↪activate spack environment ``my-project`` you need to run ``spack env activate my-
↪project`` which is specified by ``name: my-project``."
      },
      "options": {
        "type": "string",
        "description": "Pass options to ``spack env activate`` command"
      },
      "dir": {
        "type": "string",
        "description": "Activate spack environment from directory."
      }
    }
  }
},

```

(continues on next page)

(continued from previous page)

```

"rm": {
  "additionalProperties": false,
  "description": "Remove an existing spack environment via ``spack env rm``.",
  "type": "object",
  "required": [
    "name"
  ],
  "properties": {
    "name": {
      "type": "string",
      "description": "Remove spack environment by name. This will run ``spack_
↳ env rm -y <name>``."
    }
  },
  "mirror": {
    "$ref": "definitions.schema.json#/definitions/env",
    "description": "Add mirror in spack environment by running ``spack mirror add``
↳ "
  },
  "specs": {
    "$ref": "definitions.schema.json#/definitions/list_of_strings",
    "description": "Add specs to environment by running ``spack add <specs>``. The_
↳ ``specs`` is a list of string which expect the argument to be name of spack package."
  },
  "concretize": {
    "type": "boolean",
    "description": "If ``concretize: true`` is set, we will concretize spack_
↳ environment by running ``spack concretize -f`` otherwise this line will be ignored."
  },
  "install": {
    "description": "Install spack packages using ``spack install`` command",
    "additionalProperties": false,
    "type": "object",
    "properties": {
      "options": {
        "type": "string",
        "description": "Pass options to ``spack install`` command"
      },
      "specs": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings",
        "description": "List of specs to install using ``spack install`` command"
      }
    },
    "test": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "run"
      ]
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

],
  "properties": {
    "remove_tests": {
      "type": "boolean",
      "description": "Remove all test suites in spack before running test via
↳ ``spack test run``. If set to ``True`` we will run ``spack test remove -y`` which will
↳ remove all test suites."
    },
    "run": {
      "description": "Run tests using spack via ``spack test run`` command. This
↳ command requires specs are installed in your spack instance prior to running tests.",
      "type": "object",
      "required": [
        "specs"
      ],
      "additionalProperties": false,
      "properties": {
        "option": {
          "type": "string",
          "description": "Pass options to ``spack test run``"
        },
        "specs": {
          "$ref": "definitions.schema.json#/definitions/list_of_strings",
          "description": "List of specs to run tests by running ``spack test run
↳ <specs>``."
        }
      }
    },
    "results": {
      "type": "object",
      "description": "View test results via ``spack test results`` after running
↳ tests via ``spack test run``. Results can be viewed using suite name or installed specs,
↳ or both.",
      "additionalProperties": false,
      "anyOf": [
        {
          "required": [
            "specs"
          ]
        },
        {
          "required": [
            "suite"
          ]
        },
        {
          "required": [
            "specs",
            "suite"
          ]
        }
      ]
    }
  ],

```

(continues on next page)

(continued from previous page)

```

    "properties": {
      "option": {
        "type": "string",
        "description": "Pass options to ``spack test results``"
      },
      "suite": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings",
        "description": "Report results by suite name by running ``spack test_
↪results <suite>``."
      },
      "specs": {
        "$ref": "definitions.schema.json#/definitions/list_of_strings",
        "description": "Report result by spec name by running ``spack test run --
↪<specs>``."
      }
    }
  }
}

```

3.7 Build and Test Process

The *buildtest build* command is responsible for building and running tests. Every buildspect goes through a pipeline that discovers buildspects, validates the buildspect and builds and runs the test. The buildspect must go through each stage of the pipeline, if it fails in one of the stage, the buildspect will be ignored.



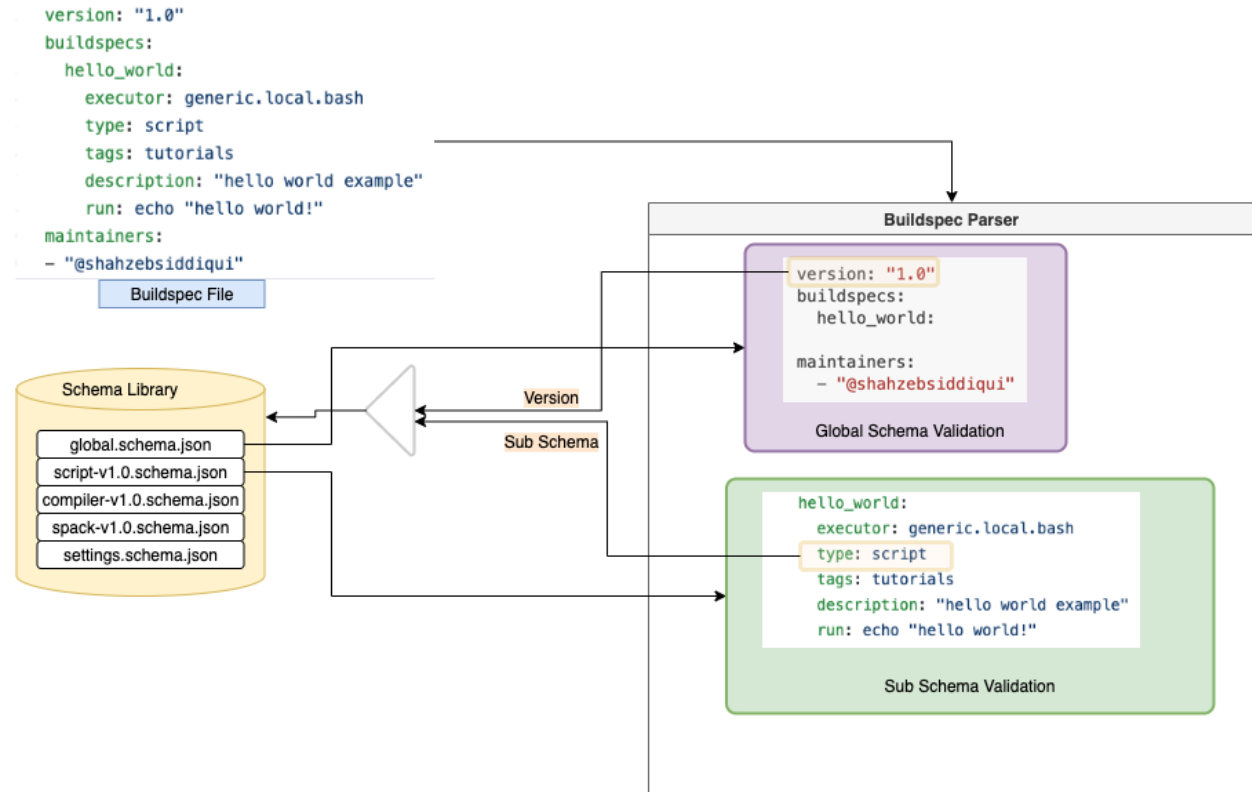
3.7.1 Discover Buildspects

buildtest will discover buildspects based on command line arguments since you can build by file, directory, executor, or tags. In **discover** stage, buildtest will detect buildspects which is discussed in *Discover Buildspects*.

For every discovered buildspects, buildtest will validate the buildspects in the **parse** stage which is performed using `jsonschema.validate` library. The parser will validate every buildspect with the global schema named `global.schema.json` and one of the sub-schemas, check *parsing buildspects* section for more details.

3.7.2 Parse Buildspects

A buildspec file may contain one or more test sections specified via `buildspecs` field. Each test is validated by a sub-schema specified by `type` field. buildtest will validate the buildspec with global schema first followed by sub-schema by using the `version` field to look up the schema version for sub-schema. buildtest will look up the schema from its schema library and validate the test section `hello_world` with schema `script-v1.0.schema.json`.



Buildspecs will be ignored if it fails validation process for instance you may have an *Invalid Buildspects*. Invalid buildspects won't be sent to **build** stage since we can't reliably build a test-script.

3.7.3 Building Buildspects

buildtest will send all valid buildspects to **build** phase which is responsible for building a shell-script from the buildspec file. In this stage, we create a **Builder** object that is an instance of `BuilderBase` class that is a base class for building a buildspect. There is a sub-class for `BuilderBase` class such as `ScriptBuilder` and `CompilerBuilder` that implements how to build a test-script based on the sub-schema selection (`type: compiler`).

During build phase, there are additional checks on buildspects to ensure we can generate a test-script. In the event of failure, buildtest will raise an exception and buildspect will be ignored. The ignored buildspects are not sent to **run** stage

3.7.4 Running Buildsspecs

In this stage, we run the test based on *executors* defined in configuration file. buildtest will select the executor defined by *executor* property in buildspect which is responsible for running the test. There is a *BaseExecutor* that is a base-class for all executors. We have sub-class for each executor type (Local, Slurm, Cobalt, PBS, Cobalt). In this stage, we run the test and get output, error, returncode and detect status of test (PASS, FAIL). If test is run via scheduler, we submit job to scheduler and poll jobID until it is finished.

Upon completion of test, we update the **Builder** object with the test results which is written to report file.

3.8 Using buildtest at HPC sites

We assume you have read the *Getting Started* and *Configuring buildtest* and now you want to use buildtest at your site. This document will highlight some points to consider before you start.

To get started, you should consider standing up an empty repository where you will host your tests. This can be GitHub, GitLab, bitbucket, etc. . .

3.8.1 Picking a version of buildtest

If you are going to use buildtest, you should consider if you want to use the bleeding edge (*devel*), stable release (*master*) or a *tag release*. Generally, we recommend you start off with stable release and then incrementally update your buildtest with new *releases* as they come out and check the *CHANGELOG.rst* for updates between version release.

Please make sure to read the appropriate version documentation based on the version of buildtest.

- Devel Docs: <https://buildtest.readthedocs.io/en/devel/index.html>
- Stable Docs: <https://buildtest.readthedocs.io/en/latest/>
- v0.9.2 Docs: <https://buildtest.readthedocs.io/en/v0.9.2/>

3.8.2 Configuring buildtest for your site

Once you have picked a version of buildtest, you need to configure buildtest for your site, this requires you see *Configuring buildtest*. We recommend you see *buildtest-cori configuration* that provides how buildtest is configured at NERSC. Once you have defined your configuration file you should make sure your configuration is valid by running:

```
buildtest config validate
```

3.8.3 Writing Test

If you are going to write test, we assume you have read *Writing buildspects* section which covers how to write buildspects. You should consider reviewing the Schema Documentation: <https://buildtesters.github.io/buildtest/> which goes in detail about each schema and buildspect attributes.

If you are writing tests, it's generally good practice to *define tags* in your test so you can group tests by a tagname and run them via `buildtest build --tags`. If you plan to use tags to run your tests, you should document tags and how they are meant to be used.

3.9 Conference and Publications

3.9.1 Talks

Conference	Date	Link
ECP Annual Meeting 2021	Apr 15, 2021	PDF , VIDEO
High Performance Computing & Simulation 2020 at HP-Bench	Mar 26, 2021	PDF
SEA Improving Scientific Software 2021	Mar 23, 2021	PDF , VIDEO
FOSDEM21	Feb 7, 2021	PDF
6th Easybuild User Meeting	Jan 29, 2021	PDF , VIDEO
FOSDEM20	Feb 2, 2020	PDF , VIDEO
5th Easybuild User Meeting	Jan 30, 2020	PDF , VIDEO
SC19 @ HUST workshop	Nov 18, 2019	PDF
HPCKP'18	June 22, 2018	PDF
HPCKP'17	June 15, 2017	PDF

3.9.2 Publications

- Siddiqui S. (2020) [Buildtest: A Software Testing Framework with Module Operations for HPC Systems](#) . In: Juckeland G., Chandrasekaran S. (eds) [Tools and Techniques for High Performance Computing](#). HUST 2019, SE-HER 2019, WIHPC 2019. Communications in Computer and Information Science, vol 1190. Springer, Cham

3.9.3 Article

- <https://www.hpcwire.com/2019/01/17/pfizer-hpc-engineer-aims-to-automate-software-stack-testing/>

3.10 Contributing Guide

This guide is geared for developers and maintainers of buildtest who want to contribute back to buildtest project. There are many ways you can help contribute to buildtest that may include:

- Improve user documentation
- Increase test coverage of buildtest regression tests.
- Work on an [existing issue](#)
- Report a bug or new feature requests at <https://github.com/buildtesters/buildtest/issues>

3.10.1 Overview

buildtest codebase is written in Python 3, so if you are new to Python you will want to check out the python 3 tutorial: <https://docs.python.org/3/tutorial/>. This is a good starting point to understand python basics. If you are familiar with Python 2 you may want to review the [Python 2-3 cheat sheet](#).

buildtest relies on [YAML](#) and [JSON Schema](#), you should review [Understanding JSON Schema](#) article as it provides a thorough overview of JSON Schema. There are several resources to help you learn YAML for instance you can check out:

- <https://www.tutorialspoint.com/yaml/index.htm>
- <https://learnxinyminutes.com/docs/yaml/>

buildtest has a regression test that is run via [pytest](#). You should be familiar with pytest and it's usage and documentation as it will help you write regression test. The regression test makes use of [coverage](#) to measure code coverage of buildtest source code. This is configured using `.coveragerc` file located in top of repo. The coverage data is pushed to [codecov](#) at <https://codecov.io/gh/buildtesters/buildtest/>.

buildtest has several CI checks written in GitHub workflows. These are found in `.github/workflows` directory of buildtest. You should familiarize yourself with [github workflow syntax](#) if you want to contribute back to github workflows.

Git is essential to code contribution so we recommend you get comfortable using *git* as it will be discussed in [code contributing guide](#). We recommend you review one the following guides to help you learn *git*:

- <https://guides.github.com/introduction/git-handbook/>
- <https://git-scm.com/docs/gittutorial>
- <https://guides.github.com/>
- <https://lab.github.com/>

buildtest documentation is built on [sphinx](#) and hosted via [readthedocs](#). Be sure to check out [documentation on readthedocs](#) to understand how it works. The buildtest project is hosted at <https://readthedocs.org/projects/buildtest/> which hosts the public documentation at <https://buildtest.readthedocs.io/>. The documentation pages are written in [reStructured Text \(rST\)](#) which is Sphinx's markup language when hosting the docs.

3.10.2 Contributing Topics

Code Contribution Guide

This guide will walk through the code contribution guide, we expect you have a [github account](#) and experience using *git* and familiarity with GitHub interface.

GitHub Account

If you don't have a GitHub account please [register](#) your account.

Fork the repo

First, you'll need to fork the repo <https://github.com/buildtesters/buildtest>

You might need to setup your SSH keys in your git profile if you are using ssh option for cloning. For more details on setting up SSH keys in your profile, follow instruction found in <https://help.github.com/articles/connecting-to-github-with-ssh/>

SSH key will help you pull and push to repository without requesting for password for every commit. Once you have forked the repo, clone your local repo:

```
git clone git@github.com:YOUR\_GITHUB\_LOGIN/buildtest.git
```

Adding Upstream Remote

First you need to add the upstream repo, to do this you can issue the following:

```
git remote add upstream git@github.com/buildtesters/buildtest.git
```

The upstream tag is used to sync changes from upstream repo to keep your repo in sync before you contribute back.

Make sure you have set your user name and email set properly in git configuration. We don't want commits from unknown users. This can be done by setting the following:

```
git config user.name "First Last"
git config user.email "abc@example.com"
```

For more details see [First Time Git Setup](#)

Sync your branch from upstream

The devel from upstream will get Pull Requests from other contributors, in-order to sync your forked repo with upstream, run the commands below:

```
git checkout devel
git fetch upstream devel
git pull upstream devel
```

Once the changes are pulled locally you can sync devel branch with your fork as follows:

```
git checkout devel
git push origin devel
```

Repeat this same operation with master branch if you want to sync it with upstream repo

Contribution Workflow

If you want to contribute back, you should create a feature branch from *devel* and add your files, commit and push them to your fork. The workflow can be summarized as follows:

```
git checkout devel
git checkout -b featureX
git add <file1> <file2> ...
git commit -m "commit message"
git push origin featureX
```

Once the branch is created in your fork, you can [create a Pull Request](https://github.com/buildtesters/buildtest) with the destination branch *devel* at <https://github.com/buildtesters/buildtest> and base branch which is your feature branch pushed at your fork.

Note: Do not push to master or devel branch on your fork or upstream.

Pull Request Review

Once you have submitted a Pull Request, please check the automated checks that are run for your PR to ensure checks are passed. Most common failures in CI checks are black and pyflakes issue, this can be done by [configuring black](#) and running [pyflakes](#). Once all checks have passed, maintainer will review your PR and provide feedback so please be patient. Please coordinate with maintainer through PR or Slack.

Resolving PR Merge Conflicts

Often times, you may start a feature branch and your PR get's out of sync with *devel* branch which may lead to conflicts, this is a result of merging incoming PRs that may cause upstream *HEAD* to change over time which can cause merge conflicts. This may be confusing at first, but don't worry we are here to help. For more details about merge conflicts click [here](#).

Syncing your feature branch with *devel* is out of scope for this documentation, however you can use the steps below as a *guide* when you run into this issue.

You may want to take the steps to first sync *devel* branch and then selectively rebase or merge *devel* into your feature branch.

First go to *devel* branch and fetch changes from upstream:

```
git checkout devel
git fetch upstream devel
```

Note you shouldn't be making any changes to your local *devel* branch, if `git fetch` was successful you can merge your *devel* with upstream as follows:

```
git merge upstream/devel
```

Next, navigate to your feature branch and sync feature changes with *devel*:

```
git checkout <feature-branch>
git merge devel
```

Note: Running above command will sync your feature branch with `devel` but you may have some file conflicts depending on files changed during PR. You will need to resolve them manually before pushing your changes

Instead of merge from `devel` you can rebase your commits interactively when syncing with `devel`. This can be done by running:

```
git rebase -i devel
```

Once you have synced your branch push your changes and check if file conflicts are resolved in your Pull Request:

```
git push origin <feature-branch>
```

General Tips

1. It's good practice to link PR to an issue during commit message. Such as stating `Fix #132` for fixing issue 132.
2. If you have an issue, ask your question in slack before reporting issue. If your issue is not resolved check any open issues for resolution before creating a new issue.
3. For new features or significant code refactor please notify maintainers and open an issue before working on task to keep everyone informed.
4. If you open an issue, please respond back during discussion, if there is no activity the issue will be closed.
5. Please refrain from opening duplicate issue, check if there is an existing issue addressing similar problem, instead you can participate in discussion in the issue or contact appropriate individuals directly in slack.
6. There should not be any branches other than `master` or `devel`. Feature branches should be pushed to your fork and not to origin.

Configuring Black Pre-Commit Hook

To configure pre-commit hook, make sure you install `pre-commit` via `pip install pre-commit`. The *pre-commit* utility should be available if you install extra dependencies from buildtest (`pip install -r docs/requirements.txt`).

You can configure `.pre-commit-config.yaml` with the version of python you are using. It is currently setup to run for python 3.7 version as follows:

```
language_version: python3.7
```

Alter this value based on python version you are using or refer to [black version control integration](#).

To install the pre-commit hook run:

```
$ pre-commit install
pre-commit installed at .git/hooks/pre-commit
```

This will invoke hook `.git/hooks/pre-commit` prior to `git commit`. Shown below we attempt to commit which resulted in pre commit hook and caused black to format code.

```
$ git commit -m "test black commit with precommit"
black.....Failed
- hook id: black
```

(continues on next page)

(continued from previous page)

```
- files were modified by this hook

reformatted buildtest/config.py
All done!
1 file reformatted.
```

If you are interested in running black locally to see diff result from black without auto-formatting code, you can do the following:

```
$ black --check --diff .
--- tests/test_inspect.py      2020-02-25 18:58:58.360360 +0000
+++ tests/test_inspect.py      2020-02-25 18:59:07.336414 +0000
@@ -18,11 +18,11 @@
 def test_distro_short():
     assert "rhel" == distro_short("Red Hat Enterprise Linux Server")
     assert "centos" == distro_short("CentOS")
     assert "suse" == distro_short("SUSE Linux Enterprise Server")
-    x=0+1*3
+    x = 0 + 1 * 3
```

The changes will be shown with lines removed or added via - and +. For more details refer to [black documentation](#).

isort

isort is a python utility that will sort python imports alphabetically. We use isort as part of the CI checks, there is a `.isort.cfg` that defines the isort configuration that is compatible with **black** utility. We have setup a pre-commit hook that can be used to automatically run isort as part of your `git commit` process. This is defined in pre-commit configuration file `.pre-commit-config.yaml` that can be installed by running `pre-commit install`. Once this is setup, you will see **isort** and **black** checks are run during the commit process.

```
$ git commit
isort.....Passed
black.....Passed
[sphinx_fix 85d9d42c] fix issue with rendering bullet points in sphinx. This is solved.
↳by downgrading docutils to version 0.16.
2 files changed, 5 insertions(+)
```

Please make sure you run `pip install -r docs/requirements.txt` to get the development dependencies that includes isort.

If you want to run isort, you can use the `-c` and `-diff` option to check and see diff between files. For instance in example below we see isort reports changes to `import` statement

```
$ isort -c --diff profile black buildtest/main.py
ERROR: /Users/siddiq90/Documents/GitHubDesktop/buildtest/buildtest/main.py Imports are
↳incorrectly sorted and/or formatted.
--- /Users/siddiq90/Documents/GitHubDesktop/buildtest/buildtest/main.py:before      2021-
↳07-13 16:53:42.722718
+++ /Users/siddiq90/Documents/GitHubDesktop/buildtest/buildtest/main.py:after      2021-
↳07-13 16:54:12.135986
@@ -1,8 +1,7 @@
"""Entry point for buildtest"""
```

(continues on next page)

(continued from previous page)

```
+import os
import webbrowser
-import os
-

from buildtest.cli import get_parser
from buildtest.cli.build import BuildTest
Broken 2 paths
```

If you want to apply the changes you can get rid of `-c` and `--diff` option and isort will apply the changes. Please see https://pycqa.github.io/isort/docs/configuration/black_compatibility.html and https://black.readthedocs.io/en/stable/guides/using_black_with_other_tools.html#isort for documentation regarding black and isort compatibility.

pyflakes

`pyflakes` is a program that checks for python source code for errors such as unused imports. We have configured an automated check to test your incoming PR using pyflakes. pyflakes should be available in your python environment if you installed buildtest extra dependencies in requirements.txt (`pip install -r docs/requirements.txt`).

You can run pyflakes against any file or directory the ones of importance is running pyflakes against buildtest source code and regression test. You can do that by running:

```
pyflakes buildtest tests
```

GitHub Integrations

buildtest has several CI checks that are run when you create a Pull Request, it is your responsibility to review the CI checks and make sure all checks are passing. Each pull request will show the CI checks, you can see the [github actions](#) that are also typically linked as part of the pull request.

Coverage

We use `coverage` to measure code coverage of buildtest when running regression test. We use CodeCov to display coverage reports through web interface. The coverage configuration is managed by `.coveragerc` file found in the root of the repo.

Whenever you add new feature to buildtest, please add regression test with test coverage to help maintainers review new feature request. For more details on running coverage tests see [Running test via coverage](#).

CodeCov

Codecov report coverage details in web-browser. CodeCov can perform [pull request comments](#) after coverage report is uploaded to Codecov which is useful for reviewer and assignee to see status of coverage report during PR review process. The codecov file `.codecov.yml` is used for configuration codecov. For more details on codecov yaml file see <https://docs.codecov.io/docs/codecov-yaml>.

Gitlab CI checks

buildtest has automated CI checks on gitlab servers: <https://software.nersc.gov> and <https://code.ornl.gov>. The gitlab pipelines are stored in `.gitlab` directory found in root of repository.

The `mirror.yml` github workflow is responsible for mirroring and trigger CI check and return result back to github PR. Currently, we are using github action `stenongithub/gitlab-mirror-and-ci-action` to perform pull mirroring and triggering CI job.

The gitlab server <https://software.nersc.gov> is hosted at NERSC. The following steps were taken to setup pipeline

1. Create a Personal Access token with **read_api**, **read_repository**, **write_repository** scope at https://software.nersc.gov/-/profile/personal_access_tokens
2. Define a secret **CORI_GITLAB_PASSWORD** at <https://github.com/buildtesters/buildtest/settings/secrets/actions> with token value generated in step 1
3. Import buildtest project from github at <https://software.nersc.gov/siddiq90/buildtest>
4. Add variable **SECRET_CODECOV_TOKEN** in https://software.nersc.gov/siddiq90/buildtest/-/settings/ci_cd that contains codecov token found at <https://app.codecov.io/gh/buildtesters/buildtest/settings>
5. Change gitlab CI configuration file to `.gitlab/cori.yml` under **Settings > CI/CD > General pipelines**. For more details see <https://docs.gitlab.com/ee/ci/pipelines/settings.html#custom-cicd-configuration-path>

The gitlab server <https://code.ornl.gov> is hosted at OLCF which has access to systems like Summit and Ascent. We performed similar steps as shown above with slight modification

1. Create a Personal access token with same scope at https://code.ornl.gov/-/profile/personal_access_tokens
2. Define a secret **OLCF_GITLAB_PASSWORD** at <https://github.com/buildtesters/buildtest/settings/secrets/actions>
3. Import buildtest project at <https://code.ornl.gov/ecpcitest/buildtest>. Currently, all projects in `ecpcitest` project group has access to gitlab runners.
4. Add variable **SECRET_CODECOV_TOKEN** in https://code.ornl.gov/ecpcitest/buildtest/-/settings/ci_cd that contains codecov token found at <https://app.codecov.io/gh/buildtesters/buildtest/settings>
5. Change gitlab CI configuration file to `.gitlab/olcf.yml`

Currently, the gitlab pipelines are triggered manually which requires a user to have access to the gitlab project to run the pipeline. The pipelines can be run manually at <https://software.nersc.gov/siddiq90/buildtest/-/pipelines> and <https://code.ornl.gov/ecpcitest/buildtest/-/pipelines>

The github workflow `mirror.yml` defines gitlab configuration for each mirror. Any changes to mirror path must be addressed in this workflow to ensure pull mirroring is done properly.

GitHub Bots

buildtest has a few bots to do various operations that are described below.

- **Stale** - stale bot is used to close outdated issues. This is configured in `.github/stale.yml`. If there is no activity on a issue after certain time period, **probot-stale** will mark the issue and project maintainers can close it manually. For more details on Stale refer to the [documentation](#)
- **CodeCov** - The codecov bot will report codecov report from the issued pull request once coverage report is complete. The configuration for codecov is defined in `.codecov.yml` found in root of repo.
- **Pull Request Size** - is a bot that labels Pull Request by number of **changed** lines of code.

Building Documentation

The buildtest documentation is written in `reStructuredText` using sphinx. You should be familiar with rst if you want to contribute to user documentation.

ReadTheDocs

buildtest `documentation` is hosted by ReadTheDocs at <https://readthedocs.org> which is a documentation platform for building and hosting your docs.

buildtest project can be found at <https://readthedocs.org/projects/buildtest/> which will show the recent builds and project setting. If you are interested in becoming a maintainer, please contact **Shahzeb Siddiqui** (shahzebmsiddiqui@gmail.com) to grant access to this project.

Setup

buildtest documentation is located in top-level `docs` directory. If you want to build the documentation you will need to make sure your python environment has all the packages defined in `docs/requirements.txt`. If your environment is already setup as described in *Installing buildtest* then you can skip this step.

To install your python packages, you can run the following:

```
pip install -r docs/requirements.txt
```

Building docs locally

To build your documentation simply run the following:

```
cd docs
make clean
make html
```

It is best practice to run `make clean` to ensure sphinx will remove old html content from previous builds, but it is ok to skip this step if you are making minor changes.

Running `make html` will build the sphinx project and generate all the html files in `docs/_build/html`. Once this process is complete you may want to view the documentation. If you have `firefox` in your system you can simply run the following:

```
make view
```

This will open a `firefox` session to the root of your documentation that was recently generated. Make sure you have X11 forwarding in order for firefox to work properly. Refer to the `Makefile` to see all of the make tags or run `make help` for additional help.

Automate Documentation Examples

buildtest has a script in top-level folder `script/docgen.py` to automate documentation examples. This script can be run as follows:

```
python script/docgen.py
```

This assumes your buildtest environment is setup, the script will write documentation test examples in `docs/docgen`. Consider running this script when **adding**, **modifying**, or **removing** documentation examples. Once the test are complete, you will want to add the tests, commit and push as follows:

```
git add docs/docgen
git commit -m <MESSAGE>
git push
```

Regression Tests

buildtest has a suite of regression tests to verify the state of buildtest. These tests are located in the top-level directory `tests`. buildtest is using `pytest` for running the regression tests.

Getting Started

In order to write regression tests, you should have `pytest` and `coverage` installed in your python environment. You can do this by installing all dependencies found in `requirements` file:

```
pip install -r docs/requirements.txt
```

Writing Regression Tests

If you want to write a new regression test, you should be familiar with `coverage` report that is pushed to `codecov`. The coverage report will give a detailed line-line coverage of source code HIT/MISS when running the regression test. Increasing coverage report would be great way to write a new regression test.

The `tests` directory is structured in a way that each source file has a corresponding test file that starts with `test_`. For instance, if you want to write a test for `buildtest/utils/command.py`, there will be a corresponding test under `tests/utils/test_command.py`.

If you adding a new directory, make sure the name corresponds to one found under `buildtest` directory and add a `__init__.py` in the new directory. This is required by `pytest` for test discovery. All test methods must start with `test_` in order for `pytest` to run your regression test.

Shown below is a simple test that always passes

```
def test_regression_example1():
    assert True
```

For more details on writing tests with `pytest` see [Getting-Started](#).

Running Regression Test

The recommended way to run regression test is via:

```
$ python $BUILDTEST_ROOT/scripts/regtest.py
```

This script is a wrapper to *pytest* and *coverage*. We have a *pytest.ini* found in top-level folder that defines *pytest* configuration. If you want to run tests natively via *pytest* without using the script you can just run *pytest* and it will run with options defined in *pytest.ini* file.

If you want to run all schema tests you can use the *schema* marker as follows:

```
pytest -v -m schema
```

To see a list of *pytest* markers see *pytest.ini* or run:

```
pytest --markers
```

For a complete list of options refer to *pytest* [documentation](#) or run *pytest --help*.

Running test via coverage

There is a coverage configuration file *.coveragerc* located in root of buildtest that is read by **coverage** utility. The *regtest.py* script will collect coverage details upon completion of regression test which is equivalent to running *coverage run -m pytest* but we make some additional checks when running the script. Upon completion of tests you can run *coverage report* to show coverage results of your regression test run locally. Shown below is an example output:

```
$ coverage report
```

Name	Stmts	Miss	Branch	BrPart	Cover
-----	-----	-----	-----	-----	-----
buildtest/__init__.py	3	3	0	0	0.00%
buildtest/defaults.py	17	17	0	0	0.00%
buildtest/executors/slurm.py	110	93	28	0	12.32%
buildtest/executors/cobalt.py	110	93	22	0	12.88%
buildtest/executors/pbs.py	96	81	14	0	13.64%
buildtest/executors/lsf.py	103	85	16	0	15.13%
buildtest/utils/timer.py	15	9	4	0	31.58%
buildtest/menu/__init__.py	29	16	10	0	33.33%
buildtest/executors/setup.py	108	60	60	8	35.71%
buildtest/menu/compiler.py	107	60	50	3	38.22%
buildtest/config.py	158	72	76	10	47.86%
buildtest/system.py	155	70	38	11	50.78%
buildtest/docs.py	5	2	0	0	60.00%
buildtest/log.py	19	7	0	0	63.16%
buildtest/buildsystem/base.py	185	45	50	8	67.23%
buildtest/menu/build.py	421	117	208	22	70.59%
buildtest/buildsystem/batch.py	75	17	44	7	71.43%
buildtest/buildsystem/compilerbuilder.py	193	36	52	10	77.14%
buildtest/buildsystem/builders.py	107	24	60	8	77.25%
buildtest/utils/tools.py	19	2	12	2	80.65%
buildtest/exceptions.py	7	2	4	0	81.82%
buildtest/menu/buildspec.py	356	46	188	22	83.82%
buildtest/executors/local.py	49	3	10	4	88.14%

(continues on next page)

(continued from previous page)

buildtest/buildsystem/scriptbuilder.py	41	3	10	3	88.24%
buildtest/utils/file.py	68	11	30	0	88.78%
buildtest/menu/report.py	193	16	114	14	89.58%
buildtest/executors/base.py	61	4	18	2	89.87%
buildtest/utils/command.py	68	3	20	5	90.91%
buildtest/menu/config.py	73	4	12	1	94.12%
buildtest/buildsystem/parser.py	51	2	18	2	94.20%
buildtest/menu/schema.py	26	0	16	2	95.24%
buildtest/menu/inspect.py	63	2	46	3	95.41%
buildtest/schemas/defaults.py	32	0	0	0	100.00%
buildtest/schemas/utils.py	26	0	8	0	100.00%
buildtest/utils/shell.py	30	0	8	0	100.00%

TOTAL	3179	1005	1246	147	66.19%

4 empty files skipped.

If you want to view the coverage details locally in a browser you can run: `coverage html` which will write the results to directory **htmlcov**. You can open the file `open htmlcov/index.html` and it will show you a summary of coverage results that you would see from codecov.

Coverage report

×

+

← → ↺

File | /Users/siddiq90/Documents/github/buildtest/htmlcov/index.html

Apps

NERSC

ECP

Benefits

ANL

GitHub

Conference

Links

Coverage report: 66.19%

Module ↑	statements	missing	excluded	branches	partial	coverage
buildtest/__init__.py	3	3	0	0	0	0.00%
buildtest/buildsystem/base.py	185	45	4	50	8	67.23%
buildtest/buildsystem/batch.py	75	17	0	44	7	71.43%
buildtest/buildsystem/builders.py	107	24	0	60	8	77.25%
buildtest/buildsystem/compilerbuilder.py	193	36	0	52	10	77.14%
buildtest/buildsystem/parser.py	51	2	4	18	2	94.20%
buildtest/buildsystem/scriptbuilder.py	41	3	0	10	3	88.24%
buildtest/config.py	158	72	0	76	10	47.86%
buildtest/defaults.py	17	17	0	0	0	0.00%
buildtest/docs.py	5	2	0	0	0	60.00%
buildtest/exceptions.py	7	2	2	4	0	81.82%
buildtest/executors/base.py	61	4	4	18	2	89.87%
buildtest/executors/cobalt.py	110	93	0	22	0	12.88%
buildtest/executors/local.py	49	3	0	10	4	88.14%
buildtest/executors/lsf.py	103	85	0	16	0	15.13%
buildtest/executors/pbs.py	96	81	0	14	0	13.64%
buildtest/executors/setup.py	108	60	4	60	8	35.71%
buildtest/executors/slurm.py	110	93	0	28	0	12.32%
buildtest/log.py	19	7	0	0	0	63.16%
buildtest/menu/__init__.py	29	16	113	10	0	33.33%
buildtest/menu/build.py	421	117	0	208	22	70.59%
buildtest/menu/buildspec.py	356	46	0	188	22	83.82%
buildtest/menu/compilers.py	107	60	8	50	3	38.22%
buildtest/menu/config.py	73	4	0	12	1	94.12%
buildtest/menu/inspect.py	63	2	0	46	3	95.41%
buildtest/menu/report.py	193	16	0	114	14	89.58%
buildtest/menu/schema.py	26	0	0	16	2	95.24%
buildtest/schemas/defaults.py	32	0	0	0	0	100.00%
buildtest/schemas/utils.py	26	0	0	8	0	100.00%
buildtest/system.py	155	70	0	38	11	50.78%
buildtest/utils/command.py	68	3	0	20	5	90.91%
buildtest/utils/file.py	68	11	0	30	0	88.78%
buildtest/utils/shell.py	30	0	4	8	0	100.00%
buildtest/utils/timer.py	15	9	0	4	0	31.58%
buildtest/utils/tools.py	19	2	2	12	2	80.65%
Total	3179	1005	145	1246	147	66.19%

For more details on coverage please refer to [coverage documentation](#).

Tox

buildtest provides a [tox.ini](#) configuration to allow user to test regression test in isolated virtual environment. To get started install tox:

```
pip install tox
```

Refer to [tox documentation](#) for more details. To run tox for all environment you can run:

```
tox
```

If your system has one python instance let's say python 3.7 you can test for python 3.7 environment by running `tox -e py37`.

Contributing to Schemas

Schema Docs

Schema Documentation are hosted on branch [gh-pages](#) which is hosted via GitHub Pages at <https://buildtesters.github.io/buildtest/>.

There is an automated workflow [jsonschema2md](#) which publishes schemas, documentation and examples. If you want to edit top-level page [README.md](#) please send a pull-request to *gh-pages* branch.

Adding a new schema

If you want to add a new schema to buildtest you need to do the following:

1. Add schema file in [buildtest/schemas](#) and schema file must end in **.schema.json**. If it's a sub-schema it must in format `<name>-<version>.schema.json`. For example a schema name `script-v2.0.schema.json` will be sub-schema script and version 2.0.
2. Their should be a folder that corresponds to name of schema in [examples](#) directory.
3. There should be a list of invalid and valid examples for schema.
4. There should be regression testfile in [schema_tests](#) to test the schema.

Be sure to update properties and take account for:

- a property being required or not
- Make use of *additionalProperties*: *false* when defining properties so that additional keys in properties are not passed in.
- requirements for the values provided (types, lengths, etc.)
- If you need help, see [Resources](#) or reach out to someone in Slack.

Running Schema Tests

The schema tests are found in folder `tests/schema_tests` which has regression test for each schema. The purpose for schema test is to ensure Buildsspecs are written according to specification outlined in schemas. Furthermore, we have edge cases to test invalid Buildspec recipes to ensure schemas are working as expected.

To run all schema test you can run via marker:

```
pytest -v -m schema
```

JSON Definitions

We store all JSON definitions in `definitions.schema.json` which are fields need to be reused in other schemas. A JSON definition is defined under `definitions` field, in this example we define a definition anchor **list_of_strings** that declares an array of string:

```
{
  "definitions": {
    "list_of_strings": {
      "type": "array",
      "uniqueItems": true,
      "minItems": 1,
      "items": {"type": "string"}
    },
  },
}
```

A definition anchor can be referenced using **\$ref** keyword. In example below we declare a definitions **string_or_list** that uses **\$ref** that points to anchor `list_of_strings`:

```
"string_or_list": {
  "oneOf": [
    {"type": "string"},
    {"$ref": "#/definitions/list_of_strings"}
  ],
},
```

For example the `tags` field is defined in `definitions.schema.json` that references definition `string_or_list`:

```
"tags": {
  "description": "Classify tests using a tag name, this can be used for categorizing_
↪ test and building tests using ``--tags`` option",
  "$ref": "#/definitions/string_or_list"
},
```

The `tags` field is used in other schemas like `compiler-v1.0.schema.json` and `script-v1.0.schema.json`. In this example we declare **tags** field and reference tags anchor from `definitions.schema.json`:

```
"tags": {
  "$ref": "definitions.schema.json#/definitions/tags"
}
```

It's worth noting each schema must have a **\$id** in order for JSON to resolve references (**\$ref**). For example the definitions schema has the following id:

```
"$id": "definitions.schema.json"
```

It's recommended each schema has a **\$schema**, **\$title**, **description** field for each schema. Currently, we support JSON Schema Draft7 so our schema field must be set to the following:

```
"$schema": "http://json-schema.org/draft-07/schema#",
```

Resources

The following sites (along with the files here) can be useful to help with your development of a schema.

- json-schema.org
- [json schema readthedocs](#)

If you have issues with writing json schema please join the [JSON-SCHEMA Slack Channel](#)

Maintainer Guide

This is a guide for buildtest maintainers

Incoming Pull Request

These are just a few points to consider when dealing with incoming pull requests

1. Any incoming Pull Request should be assigned to one or more maintainers for review.
2. Upon approval, the PR should be **Squash and Merge**. If it's important to preserve a few commits during PR then **Rebase and Merge** is acceptable.
3. The final commit PR commit, either Squash Commit or Rebase, should have meaningful comments and if possible link to the github issue.
4. Maintainers can request user to put meaningful commit if author has not provided a meaningful message (i.e `git commit --amend`)
5. Maintainers are requested that committer name and email is from a valid Github account. If not please request the committer to fix the author name and email.
6. All incoming PRs should be pushed to devel branch, if you see any PR sent to any other branch please inform code owner to fix it

Release Process

Every buildtest release will be tagged with a version number using format **X.Y.Z**. Every release will have a git tags such as **v1.2.3** to correspond to release **1.2.3**. Git tags should be pushed to upstream by **release manager** only. The process for pushing git tags can be described in the following article: [Git Basics - Tagging](#)

We will create annotated tags as follows:

```
git tag -a v1.2.3 -m "buildtest version 1.2.3"
```

Once tag is created you can view the tag details by running either:

```
git tag
git show v1.2.3
```

We have created the tag locally, next we must push the tag to the upstream repo by doing the following:

```
git push origin v.1.2.3
```

Every release must have a release note that is maintained in file [CHANGELOG.rst](#)

Under buildtest [releases](#) a new release can be created that corresponds to the git tag. In the release summary, just direct with a message stating **refer to CHANGELOG.rst for more details**

Once the release is published, make sure to open a pull request from devel -> master and **Rebase and Merge** to master branch. If there are conflicts during merge for any reason, then simply remove master and create a master branch from devel.

Default Branch

The default branch should be devel which should be protected branch.

Branch Settings

All maintainers are encouraged to view branch [settings](#) for devel and master. If something is not correct please consult with the maintainers.

The master and devel branches should be protected branches and master should be enabled as default branch. Shown below is the expected configuration.

We have disabled Merge Commits for the Merge button in Pull Request. This was done because we wanted a linear history as a requirement for devel branch. This avoids having a maintainer accidentally merge a PR with Merge Commit which adds an extra commit.

3.10. Contributing Guide

Merge button

When merging pull requests, you can allow any combination of merge commits, squashing, or rebasing. At least one option must be enabled. If you have linear history requirement enabled on any protected branch, you must enable squashing or rebasing.

<input type="checkbox"/>	Allow merge commits Add all commits from the head branch to the base branch with a merge commit.
<input checked="" type="checkbox"/>	Allow squash merging Combine all commits from the head branch into a single commit in the base branch.
<input checked="" type="checkbox"/>	Allow rebase merging Add all commits from the head branch onto the base branch individually.

If you notice a deviation, please consult with the maintainers.

Google Analytics

The buildtest site is tracked via Google Analytics, if you are interested in get access contact **Shahzeb Siddiqui** (@shahzebsiddiqui)

Read The Docs Access

buildtest project for readthedocs can be found at <https://readthedocs.org/projects/buildtest/>. If you need to administer project configuration, please contact **Shahzeb Siddiqui** @shahzebsiddiqui to gain access.

Slack Admin Access

If you need admin access to Slack Channel please contact **Shahzeb Siddiqui** @shahzebsiddiqui. The slack admin link is <https://hpcbuildtest.slack.com/admin>

New Maintainers Checklist

Onboarding Email

This guide is to help onboard new maintainers into the buildtest project. To get started send an invitation email as follows:

We are pleased to invite you to the buildtest project **and** become a buildtesters (a.k.a buildtest maintainer). We understand your time **is** valuable; therefore we request a minimal effort of **2-3hrs** per week towards buildtest.

As a buildtesters, you will be working on the following:

(continues on next page)

(continued from previous page)

- * Monitor **and** triage issues
- * Assist user **in** slack channel (*#general*)
- * Update documentation
- * Review **or** triage Pull Request
- * Issue new pull request
- * Troubleshoot build errors **in** regression test **or** CI checks

As a buildtesters you may be granted elevated privilege to the following services: GitHub, ReadTheDocs, Slack, **and** Google Analytics. As a buildtesters, you agree to be accessible on Slack **as** our primary communication channel.

If you agree to these terms, you will be assigned to work **with** another buildtest maintainer **in** your first two weeks. Once you are confident **in** your duties, we will let you work independently at your own pace, should you need help please contact one of the buildtesters.

Please review the contributing guide: <https://buildtest.readthedocs.io/en/devel/contributing.html>

if you are unsure about your responsibilities **as** a buildtesters.

If you agree to these terms **and** conditions, please reply **"I CONFIRM"**.

Thanks,
buildtest

Onboarding Checklist

- Please make sure the maintainer has a GitHub account if not please create an account at <https://github.com/join>.
- Ensure user has setup two-factor authentication (2FA) with GitHub.
- Invite member to buildtesters organization.
- Add member to buildtest repository with **Role: Maintain**.
- Invite member to join slack channel and preferably install Slack on your workstation and phone. Please follow instructions to download slack for **Windows**, **Mac**, or **Android**. Slack is available on Apple Store and Google Play Store.
- Once member is added to Slack, ensure member has the appropriate account type. Generally you will want member to be a **Workspace Admin** for more details see [Slack Roles & Permissions](#).
- Ensure member has an account at ReadTheDocs if not please request member to create an account at <https://readthedocs.org/accounts/signup/>. Once member has an account please add member to buildtest readthedocs project at <https://readthedocs.org/dashboard/buildtest/users/>. This will ensure user has ability to access readthedocs platform when troubleshooting build errors related to documentation.

3.11 API Reference

This page contains auto-generated API reference documentation¹.

3.11.1 buildtest

Subpackages

`buildtest.buildsystem`

Submodules

`buildtest.buildsystem.base`

BuilderBase class is an abstract class that defines common functions for any types of builders. Each type schema (script, compiler) is implemented as separate Builder.

ScriptBuilder class implements ‘type: script’ CompilerBuilder class implements ‘type: compiler’

Module Contents

Classes

<i>BuilderBase</i>	The BuilderBase is an abstract class that implements common functions for
--------------------	---

class `buildtest.buildsystem.base.BuilderBase(name, recipe, buildspec, executor, buildexecutor, testdir)`

Bases: `abc.ABC`

The BuilderBase is an abstract class that implements common functions for any kind of builder.

__repr__(*self*)

Return repr(self).

__str__(*self*)

Return str(self).

_build_setup(*self*)

This method is the setup operation to get ready to build test which includes getting unique build id, setting up metadata object to store test details such as where test will be located and directory of test. This section cannot be reached without a valid, loaded recipe.

_default_test_variables(*self*)

Return a list of lines inserted in testscript that define buildtest specific variables that can be referenced when writing tests. The buildtest variables all start with BUILDTEST_*

_emit_command(*self*)

This method will return a shell command used to invoke the script that is used for tests that use local executors

¹ Created with `sphinx-autoapi`

_generate_unique_id(*self*)

Generate a unique build id using `uuid.uuid4()`.

_get_burst_buffer(*self*, *burstbuffer*)

Get Burst Buffer directives (#BB) lines specified by BB property

Parameters **burstbuffer** (*dict*, *required*) – Burst Buffer configuration specified by BB property

Returns list of burst buffer directives

Return type list

_get_data_warp(*self*, *datawarp*)

Get Cray Data Warp directives (#DW) lines specified by DW property.

Parameters **datawarp** (*dict*, *required*) – Data Warp configuration specified by DW property

Returns list of data warp directives

Return type list

_get_environment(*self*, *env*)

Retrieve a list of environment variables defined in builds spec and return them as list with the shell equivalent command

Returns list of environment variable lines to add to test script.

Return type list

_get_scheduler_directives(*self*, *bsub*=None, *sbatch*=None, *cobalt*=None, *pbs*=None, *batch*=None)

Get Scheduler Directives for LSF, Slurm, PBS or Cobalt if we are processing test with one of the executor types. This method will return a list of string containing scheduler directives generally found at top of script. If test is local executor we return an empty list

Parameters

- **bsub** – bsub property from builds spec
- **sbatch** – sbatch property from builds spec
- **cobalt** – cobalt property from builds spec
- **pbs** – pbs property from builds spec
- **batch** – batch property from builds spec

_get_variables(*self*, *variables*)

Retrieve a list of variables defined in builds spec and return them as list with the shell equivalent command.

Returns list of variables variable lines to add to test script.

Return type list

_set_execute_perm(*self*, *fname*)

Apply chmod 755 to input file name. The path must be an absolute path to script

_set_metadata_values(*self*)

This method sets `self.metadata` that contains metadata for each builder object.

_write_build_script(*self*)

This method will write the build script used for running the test

_write_test(self)

This method is responsible for invoking `generate_script` that formulates content of testscript which is implemented in each subclass. Next we write content to file and apply 755 permission on script so it has executable permission.

build(self)

This method is responsible for invoking setup, creating test directory and writing test. This method is called from an instance object of this class that does `builder.build()`.

complete(self)

This method is invoked to indicate that builder job is complete after polling job.

copy_stage_files(self)

Copy output and error file into test root directory since stage directory will be removed.

endtime(self)

This method is called upon termination of job, we get current time using `datetime.datetime.now()` and calculate runtime of job

abstract generate_script(self)

Build the testscript content implemented in each subclass

get_test_extension(self)

Return the test extension, which depends on the shell used. Based on the value of `shell` key we return the shell extension.

shell: bash → sh (default)

Returns returns test extension based on shell type

Return type str

incomplete(self)

This method indicates that builder job is not complete after polling job either job was cancelled by scheduler or job failed to run.

run(self)

Run the test and record the starttime and start timer. We also return the instance object of type BuildTest-Command which is used by Executors for processing output and error

run_command(self)

Command used to run the build script. buildtest will change into the stage directory (`self.stage_dir`) before running the test.

runtime(self)**start(self)**

Keep internal time for start of test. We start timer by calling Timer class

starttime(self)

This method will record the starttime when job starts execution by using `datetime.datetime.now()`

stop(self)

Stop timer of test and calculate duration.

buildtest.buildsystem.batch

Module Contents

Classes

BatchScript

CobaltBatchScript

LSFBatchScript

PBSBatchScript

SlurmBatchScript

class buildtest.buildsystem.batch.**BatchScript**

get_headers(*self*)

class buildtest.buildsystem.batch.**CobaltBatchScript**(*batch=None, cobalt=None*)

 Bases: *BatchScript*

batch_translation

build_header(*self*)

class buildtest.buildsystem.batch.**LSFBatchScript**(*batch=None, bsub=None*)

 Bases: *BatchScript*

batch_translation

build_header(*self*)

 Generate BSUB directive that will be part of the script

class buildtest.buildsystem.batch.**PBSBatchScript**(*batch=None, pbs=None*)

 Bases: *BatchScript*

batch_translation

build_header(*self*)

class buildtest.buildsystem.batch.**SlurmBatchScript**(*batch=None, sbatch=None*)

 Bases: *BatchScript*

batch_translation

build_header(*self*)

 Generate SBATCH directive that will be part of the script

buildtest.buildsystem.builders

This file implements the Builder class that is responsible for getting builders from a builds spec file. The Builder class is invoked once builds spec file has parsed validation via Builds specParser.

Module Contents

Classes

Builder

```
class buildtest.buildsystem.builders.Builder(bp, buildexecutor, filters, testdir, configuration,
                                             buildtest_system=None, rebuild=1)
```

_build_compilers(*self*, *name*, *recipe*)

This method will perform regular expression with ‘name’ field in compilers section and retrieve one or more compiler that were defined in buildtest configuration. If any compilers were retrieved we return one or more builder objects that call CompilerBuilder

Parameters

- **bp** (*BuildspecParser*) – an instance of BuilderspecParser class
- **recipe** (*dict*) – loaded test recipe

_generate_builders(*self*, *recipe*, *name*, *compiler_name=None*)

This method is responsible for generating builders by applying regular expression specified by *executor* field in builds spec with list of executors. If their is a match we generate a builder.

Parameters

- **name** (*str*) – Name of test in builds spec file
- **recipe** (*object*) – Loaded test recipe from a test section.
- **compiler_name** (*str*, *optional*) – Name of compiler

Returns A list of builder objects

_skip_tests_by_tags(*self*, *recipe*, *name*)

This method determines if test should be skipped based on tag names specified in filter field that is specified on command line via `buildtest build --tags`

Parameters

- **recipe** (*dict*) – loaded builds spec recipe as dictionary
- **name** (*str*) – An instance of test from builds spec file

Returns Returns a boolean True/False which determines if test is skipped.

Return type bool

_skip_tests_run_only(*self*, *recipe*, *name*)

This method will skip tests based on `run_only` field from builds spec. Checks are performed based on conditionals and if any conditional is not met we skip test.

Parameters

- **recipe** (*dict*, *required*) – loaded builds spec recipe as dictionary
- **name** (*str*, *required*) – name of test from builds spec file

Returns Returns a boolean to see if test is skipped based on `run_only` property

Return type bool

get_builders(*self*)

get_test_names(*self*)

Return the list of test names for the loaded Builds spec recipe. This can be retrieved by returning a list of keys under ‘buildspecs’ property

Returns A list of test names in builds spec file

Return type list

`buildtest.buildsystem.compilerbuilder`

Module Contents

Classes

CompilerBuilder

The BuilderBase is an abstract class that implements common functions for

```
class buildtest.buildsystem.compilerbuilder.CompilerBuilder(name, recipe, builds spec,
                                                           buildexecutor, executor,
                                                           configuration, compiler=None,
                                                           testdir=None)
```

Bases: `buildtest.buildsystem.base.BuilderBase`

The BuilderBase is an abstract class that implements common functions for any kind of builder.

cc

cflags

cppflags

cxx

cxxflags

fc

fflags

lang_ext_table

ldflags

type = compiler

_compile_cmd(*self*)

This method generates the compilation line and returns the output as a list. The compilation line depends on the the language detected that is stored in variable `self.lang`.

_detect_lang(*self*, *sourcefile*)

This method will return the Programming Language based by looking up file extension of source file.

`_get_modules(self, modules)`

Return a list of module command as a list of instructions based on module property.

param modules 'module' property specified in builds spec used for loading/swapping modules

type modules object

`_process_compiler_config(self)`

This method is responsible for setting cc, fc, cxx class variables based on compiler selection. The order of precedence is config, default, then buildtest setting. Compiler settings in 'config' takes highest precedence, this overrides any configuration in 'default'. Finally we resort to compiler configuration in buildtest setting if none defined. This method is responsible for setting cc, fc, cxx, cflags, cxxflags, fflags, ldflags, and cppflags.

`_resolve_source(self)`

This method resolves full path to source file, it checks for absolute path first before checking relative path that is relative to Builds spec recipe.

`_run_cmd(self)`

This method builds the run command which refers to how to run the generated binary after compilation.

`generate_script(self)`

This method is responsible for generating test script for compiler schema. The method `generate_script` is implemented in each subclass because implementation on test generation differs across schema types.

This method will add the lines into list which comprise content of test. The method will return a list containing lines of test script.

`get_cc(self)`

`get_cflags(self)`

`get_cppflags(self)`

`get_cxx(self)`

`get_cxxflags(self)`

`get_fc(self)`

`get_fflags(self)`

`get_ldflags(self)`

`get_path(self)`

This method returns the full path for C, C++, Fortran compilers

`set_cc(self, cc)`

`set_cflags(self, cflags)`

`set_cppflags(self, cppflags)`

`set_cxx(self, cxx)`

`set_cxxflags(self, cxxflags)`

`set_fc(self, fc)`

`set_fflags(self, fflags)`

`set_ldflags(self, ldflags)`

setup(self)

The setup method is responsible for process compiler section, getting modules pre_build, post_build, pre_run, post_run section and generate compilation and run command. This method invokes other methods and set values in class variables. This method is called by self.generate_script method.

buildtest.buildsystem.parser

BuildspecParser is intended to read in a Buildspec file with one or more test blocks, and then generate builders based on the type of each. The BuilderBase is the base class for all builders that expose functions to run builds.

Module Contents**Classes***BuildspecParser*

A BuildspecParser is a base class for loading and validating a Buildspec file.

class buildtest.buildsystem.parser.BuildspecParser(buildspec, buildexecutor)

A BuildspecParser is a base class for loading and validating a Buildspec file. The type (e.g., script) and version are derived from reading in the file, and then matching to a Buildspec schema.

The schemas are located in buildtest/schemas, we load the schema dictionary and validate each buildspec with global schema and a sub-schema based on the type field. If the schema fails validation check, then we stop immediately.

__repr__(self)

Return repr(self).

__str__(self)

Return str(self).

_check_executor(self, test)

This method checks if executor property is not None and executor value is found in list of available executors.

Parameters test (str, required) – name of test in buildspecs property in buildspec file

_check_schema_type(self, test)

Check type field is a valid sub-schema and verify type + version will resolve to a schema file.

_validate(self)

This method will validate the entire buildspec file with global schema and each test section with a sub-schema. The global validation ensures that the overall structure of the file is sound for further parsing. We load in the global.schema.json for this purpose.

A buildspec is composed of one or more tests, each section is validated with a sub-schema. The type field is used for sub-schema lookup from schema library. Finally we validate loaded recipe with sub-schema.

buildtest.buildsystem.scriptbuilder

Module Contents

Classes

<i>ScriptBuilder</i>	The BuilderBase is an abstract class that implements common functions for
----------------------	---

class buildtest.buildsystem.scriptbuilder.**ScriptBuilder**(*name, recipe, buildspec, executor, buildexecutor, testdir*)

Bases: *buildtest.buildsystem.base.BuilderBase*

The BuilderBase is an abstract class that implements common functions for any kind of builder.

type = **script**

generate_script(*self*)

This method builds the testscript content based on the builder type. For ScriptBuilder we need to add the shebang, environment variables and the run section. If shell is python we write a python script and return immediately. The variables, environment section are not applicable for python scripts

Returns return content of test script

Return type list

write_python_script(*self*)

This method is used for writing python script when shell: python is set. The content from run section is added into a python script. The file is written to run directory and we simply invoke python script by running python script.py

buildtest.buildsystem.spack

This method defines the Spack buildsystem for the spack package manager (<https://spack.readthedocs.io/en/latest/>) by generating scripts that will do various spack operation. The SpackBuilder class will generate a test script using the schema definition 'spack-v1.0.schema.json' that defines how buildspecs are written.

Module Contents

Classes

<i>SpackBuilder</i>	The BuilderBase is an abstract class that implements common functions for
---------------------	---

class buildtest.buildsystem.spack.**SpackBuilder**(*name, recipe, buildspec, buildexecutor, executor, testdir=None*)

Bases: *buildtest.buildsystem.base.BuilderBase*

The BuilderBase is an abstract class that implements common functions for any kind of builder.

type = **spack**

`_resolve_spack_root(self, path, verify_spack=True)`

Given a path find the startup spack setup script to source.

`_spack_environment(self, spack_env)`

This method is responsible for creating a spack environment, activate an existing spack environment, create a spack environment from a directory and a manifest file (spack.yaml, spack.lock)

`generate_script(self)`

Method responsible for generating the content of test script for spack buildsystem

buildtest.cli

buildtest cli: include functions to build, get test configurations, and interact with a global configuration for buildtest.

Submodules

buildtest.cli.build

This module contains all the methods related to “buildtest build” which is used for building test scripts from a Builds spec

Module Contents

Classes

<i>BuildTest</i>	This class is an interface to building tests via “buildtest build” command.
------------------	---

Functions

<i>discover_buildspecs</i> (buildspecs=None, exclude_buildspecs=None, executors=None, tags=None)	ex- This method discovers all buildspecs based on –buildspecs, –tags, –executor
<i>discover_buildspecs_by_executor</i> (executors)	This method discovers buildspecs by executor name, using <code>buildtest build --executor</code>
<i>discover_buildspecs_by_tags</i> (tagnames)	This method discovers buildspecs by tags, using <code>--tags</code> option
<i>discover_by_buildspecs</i> (buildspec)	Given a buildspec file specified by the user with <code>buildtest build --buildspec,</code>
<i>print_discovered_buildspecs</i> (buildspec_dict)	This method will print the discovered buildspecs in the table format
<i>update_report</i> (valid_builders, port_file=BUILD_REPORT)	re- This method will update BUILD_REPORT after every test run performed

Attributes

logger

```
class buildtest.cli.build.BuildTest(configuration=None, buildsspecs=None, exclude_buildspecs=None,
                                     tags=None, executors=None, testdir=None, stage=None,
                                     filter_tags=None, rebuild=None, buildtest_system=None,
                                     report_file=None, max_pend_time=None, poll_interval=None,
                                     keep_stage_dir=None)
```

This class is an interface to building tests via “buildtest build” command.

_print_build_phase(self, invalid_builders, table)

_print_jobs_after_poll(self, valid_builders)

Print table of all tests after polling

_print_test_summary(self)

Print a summary of total pass and fail test with percentage breakdown.

_update_build_history(self)

Write a build history file that is stored in **\$BUILDTEST_ROOT/var/.history** directory summarizing output of build. The history file is a json file named *build.json* which contains a copy of the build log for troubleshooting. buildtest will create a sub-directory that is incremented such as 0, 1, 2 in **\$BUILDTEST_ROOT/var/.history** which is used to differentiate builds.

build(self)

This method is responsible for discovering buildsspecs based on input argument. Then we parse the buildsspecs and retrieve builder objects for each test. Each builder object will invoke *build* which will build the test script, and then we run the test and update report.

build_phase(self)

This method will build all tests by invoking class method *build* for each builder that generates testscript in the test directory.

parse_buildspecs(self)

Parse all buildsspecs by passing buildspec file to *BuildspecParser* class. If buildspec fails validation we skip the buildspec and print all skipped buildsspecs. If buildspec passes validation we get all builders by invoking *Builder* class that is responsible for creating builder objects for each test.

Returns A list of builder objects which are instances of *BuilderBase* class

Return type list

poll_jobs(self, poll_queue, valid_builders)

This method will poll jobs by processing all jobs in *poll_queue*. If job is cancelled by scheduler, we remove this from *valid_builders* list. This method will return a list of *valid_builders* after polling. If there are no *valid_builders* after polling, the method will return None

Parameters

- **poll_queue** (*list*, *required*) – a list of jobs that need to be polled. The jobs will poll using poll method from executor
- **valid_builders** (*list*, *required*) – list of valid builders

resolve_testdirectory(self, cli_testdir=None)

This method resolves which test directory to select. For example, one can specify test directory via command line *buildtest build --testdir <path>* or path in configuration file. The default is *\$HOME/.buildtest/var/tests*

Parameters `cli_testdir` (*str*) – test directory from command line `buildtest build --testdir`

Returns Path to test directory to use

Return type *str*

run_phase(*self*)

This method will run all builders with the appropriate executor. The executor argument is an instance of `BuildExecutor` that is responsible for orchestrating builder execution to the appropriate executor class. The executor contains a list of executors picked up from buildtest configuration. For tests running locally, we get the test metadata and count PASS/FAIL test state which is printed at end in Test Summary. For tests that need to run via scheduler, the first stage of run will dispatch job, and state will be *N/A*. We first dispatch all jobs and later poll jobs until they are complete. The poll section is skipped if all tests are run locally. In poll section we regenerate table with all `valid_builders` and updated test state and returncode and recalculate total pass/fail tests. Finally we return a list of `valid_builders` which are tests that ran through one of the executors. Any test that failed to run or be dispatched will be skipped during run stage and not added in `valid_builders`. The `valid_builders` contains the test meta-data that is used for updating test report in next stage.

Returns A list of valid builders

Return type *list*

`buildtest.cli.build.discover_buildspecs`(*buildspecs=None, exclude_buildspecs=None, executors=None, tags=None*)

This method discovers all buildspecs based on `--buildspecs`, `--tags`, `--executor` and excluding buildspecs (`--exclude`).

Parameters

- **buildspecs** (*list*) – List of input buildspecs passed by argument `buildtest build --buildspec`
- **exclude_buildspecs** (*list*) – List of excluded buildspecs by argument `buildtest build --exclude`
- **tags** (*list*) – List of input tags for discovering buildspecs by argument `buildtest build --tags`
- **executors** (*list*) – List of input executors for discovering buildspecs by argument `buildtest build --executor`

`buildtest.cli.build.discover_buildspecs_by_executor`(*executors*)

This method discovers buildspecs by executor name, using `buildtest build --executor` command. This method will read `BUILDSPEC_CACHE_FILE` and search for `executor` key in buildspec recipe and match with input executor name. The return is a list of matching buildspec with executor name to process.

Parameters **executors** (*list*) – List of input executor name from command line argument `buildtest build --executor <name>`

Returns a list of buildspec files that match tag name

Return type *list*

`buildtest.cli.build.discover_buildspecs_by_tags`(*tagnames*)

This method discovers buildspecs by tags, using `--tags` option from `buildtest build` command. This method will read `BUILDSPEC_CACHE_FILE` and search for `tags` key in buildspec recipe and match with input tag. Since `tags` field is a list, we check if input tag is in `list` and if so we add the entire buildspec into a list. The return is a list of buildspec files to process.

Parameters **input_tag** (*list*) – List of input tags from command line argument `buildtest build --tags <tags>`

Returns a list of buildspec files that match tag name

Return type list

`buildtest.cli.build.discover_by_buildspecs(buildspec)`

Given a buildspec file specified by the user with `buildtest build --buildspec`, discover one or more files and return a list for buildtest to process. This method is called once per argument of `--buildspec` or `--exclude` option. If its a directory path we recursively find all buildspecs with option. If its a directory path we recursively find all buildspecs with `.yaml` extension. If filepath doesn't exist or file extension is not `.yaml` we return `None` and capture error in log.

file path `buildtest build --buildspec tutorials/hello.sh.yaml`

directory path `buildtest build --buildspec tutorials`

Parameters `buildspec` (*str*) – Input argument from `buildtest build --buildspec`

Returns A list of discovered buildspec with resolved path, if its invalid we return `None`

Return type list or `None`

`buildtest.cli.build.logger`

`buildtest.cli.build.print_discovered_buildspecs(buildspec_dict)`

This method will print the discovered buildspecs in the table format

`buildtest.cli.build.update_report(valid_builders, report_file=BUILD_REPORT)`

This method will update `BUILD_REPORT` after every test run performed by `buildtest build`. If `BUILD_REPORT` is not created, we will create file and update json file by extracting contents from `builder.metadata`

Parameters

- **valid_builders** (*instance of BuilderBase (subclass)*) – builder object that were successful during build and able to execute test
- **report_file** (*str*) – specify location to report file

`buildtest.cli.buildspec`

Module Contents

Classes

BuildspecCache

Functions

buildspec_find(args, configuration)

Entry point for buildtest buildspec find command

buildspec_validate(configuration, buildspecs=None, excluded_buildspecs=None, tags=None, executors=None)

Entry point for buildtest buildspec validate. This method is responsible for discovering buildspec

Attributes

logger

```
class buildtest.cli.buildspec.BuildspecCache(configuration, rebuild=False, filterfields=None,
                                             formatfields=None, roots=None)
```

```
    default_format_fields = ['name', 'type', 'executor', 'tags', 'description']
```

```
    filter_fields = ['type', 'executor', 'tags']
```

```
    format_fields
```

```
    table
```

```
    _discover_buildspecs(self)
```

This method retrieves buildspecs based on `self.paths` which is a list of directory paths to search. If `--root` is specified we process each argument and recursively find all .yaml files

```
    _filter_buildspecs(self, executor, tags, schema_type)
```

This method will return a boolean True/False that determines if buildspec test entry is skipped as part of filter process. The filter are done based on executor, tags, type field. True indicates test needs to be skipped.

Parameters

- **executor** (*str*, *required*) – ‘executor’ field from buildspec recipe
- **tags** (*str or list*, *required*) – ‘tags’ field from buildspec recipe
- **schema_type** (*str*, *required*) – ‘type’ field from buildspec recipe

Returns boolean to determine if we need to skip buildspec

Return type bool

```
    _validate_buildspecs(self, buildspecs)
```

Given a list of buildspec files, validate each buildspec using BuildspecParser and return a list of valid buildspecs. Any invalid buildspecs are added to separate list

```
    _write_buildspec_cache(self)
```

This method is responsible for writing buildspec cache to file

```
    build(self)
```

This method will build buildspec cache file. If user requests to rebuild cache we remove the file and recreate cache. If cache file exists, we simply load from cache

```
    build_cache(self)
```

This method will rebuild the buildspec cache file by recursively searching all .yaml files specified by input argument `paths` which is a list of directory roots. The buildspecs are validated and cache file is updated”

Parameters **paths** (*list*) – A list of directory roots to process buildspecs files.

Returns Rebuild cache file

```
    check_filter_fields(self)
```

This method checks filter fields are valid. The filter fields are specified as “buildtest buildspec find –filter <KEY1>=<VAL1>,<KEY2>=<VAL2>,...

```
    check_format_fields(self)
```

This method will check if all format fields are valid. Format fields are passed comma separated as –format field1,field2,field3,...

find_buildspecs(*self*)

This method will find buildspecs based on cache content. We skip any tests based on executor filter, tag filter or type filter and build a table of tests that will be printed using `print_buildspecs` method.

get_buildspecfiles(*self*, *terse=None*)

This method implements `buildtest buildspect find --buildspec` which reports all buildspec files in cache.

Parameters *terse* (*bool*) – This argument controls output of `buildtest buildspect find --buildspec` which is a boolean. If its `True` we print output in raw format otherwise we print in table format

get_cache(*self*)

Returns cache file as loaded dictionary

get_executors(*self*, *terse=None*)

This method implements `buildtest buildspect find --executors` which reports all executors from cache.

Parameters *terse* (*bool*) – This argument controls output of `buildtest buildspect find --executor` which is a boolean. If its `True` we print output in raw format otherwise we print in table format

get_tags(*self*, *terse=None*)

This method implements `buildtest buildspect find --tags` which reports a list of unique tags from all buildspecs in cache file.

Parameters *terse* (*bool*) – This argument controls output of `buildtest buildspect find --tags` which is a boolean. If its `True` we print output in raw format otherwise we print in table format

load_paths(*self*)

Add all paths to search for buildspecs. We must read configuration file and check property `buildspec_roots` for list of directories to search. We check all directories exist, if any fail we don't add them to path. In addition, we add the default buildspec path where we find tutorials and general tests.

print_buildspecs(*self*)

Print buildspec table

print_by_executors(*self*)

This method prints executors by tests and implements `buildtest buildspect find --group-by-executor` command

print_by_tags(*self*)

This method prints tags by tests and implements `buildtest buildspect find --group-by-tags` command

static print_filter_fields()

This method prints filter fields available for buildspec cache. This method implements command `buildtest buildspect find --helpfilter`

static print_format_fields()

This method prints format fields available for buildspec cache. This method implements command `buildtest buildspect find --helpformat`

print_maintainer(*self*, *terse=None*)

This method prints maintainers from buildspec cache file which implements `buildtest buildspect find --maintainers` command.

Parameters *terse* (*bool*) – This argument controls output of `buildtest buildspect find --maintainers` which is a boolean. If its `True` we print output in raw format otherwise we

print in table format

print_maintainers_by_buildspecs(*self*)

This method prints maintainers breakdown by buildspecs. This method implements buildtest buildspec find --maintainers-by-buildspecs

print_paths(*self*)

This method print buildspec paths, this implements command buildtest buildspec find --paths

buildtest.cli.buildspec.**buildspec_find**(*args, configuration*)

Entry point for buildtest buildspec find command

buildtest.cli.buildspec.**buildspec_validate**(*configuration, buildspecs=None, excluded_buildspecs=None, tags=None, executors=None*)

Entry point for buildtest buildspec validate. This method is responsible for discovering buildspec with same options used for building buildspecs that includes --buildspec, --exclude, --tag, and --executor. Upon discovery we pass each buildspec to BuildspecParser class to validate buildspec and report any errors during validation which is raised as exceptions.

Parameters

- **configuration** (*instance of SiteConfiguration*) – An instance of SiteConfiguration class which is the loaded buildtest configuration used for validating the buildspecs.
- **buildspecs** (*List, optional*) – List of paths to buildspec file which can be a file or directory
- **excluded_buildspecs** (*List, optional*) –
- **tags** – List of tag names to search for buildspec
- **executors** (*List, optional*) – List of executor names to search for buildspecs

buildtest.cli.buildspec.**logger**

buildtest.cli.cdash

Module Contents

Functions

<code>cdash_cmd</code> (<i>args, default_configuration=None, open_browser=True</i>)	This method is entry point for buildtest cdash command which implements uploading
<code>upload_test_cdash</code> (<i>build_name, configuration, site=None, report_file=None</i>)	This method is responsible for reading report file and pushing results to CDASH

buildtest.cli.cdash.**cdash_cmd**(*args, default_configuration=None, open_browser=True*)

This method is entry point for buildtest cdash command which implements uploading results to CDASH server and command line interface to open CDASH project.

Parameters

- **args** (*ArgumentParser*) – Instance of ArgumentParser that contains arguments for buildtest cdash command
- **default_configuration** (*SiteConfiguration, optional*) – The loaded default configuration which is an instance of SiteConfiguration
- **open_browser** (*optional*) – boolean to control if we open page in web browser using

webbrowser.open. This is enabled by default, but can be turned off especially when running regression test where we don't want to see the page

`buildtest.cli.cdash.upload_test_cdash(build_name, configuration, site=None, report_file=None)`

This method is responsible for reading report file and pushing results to CDASH server. User can specify cdash settings in configuration file or pass them in command line. The command `buildtest cdash upload` will upload results to CDASH.

Parameters

- **build_name** – build name that shows up in CDASH
- **configuration** (`SiteConfiguration`) – Instance of `SiteConfiguration` class that contains the configuration file
- **site** (*str*, *optional*) – site name that shows up in CDASH
- **report** (*str*, *optional*) – Path to report file when uploading results. This is specified via `buildtest cdash upload -r` command

Returns

`buildtest.cli.compilers`

Module Contents

Classes

BuildtestCompilers

Functions

compiler_cmd(args, configuration)

<i>compiler_find</i> (args, configuration)	This method implements <code>buildtest config compilers find</code> which detects
--	---

`class buildtest.cli.compilers.BuildtestCompilers(configuration, settings_file=None, debug=False)`

`compiler_table`

`_update_compiler_section(self)`

This method will update the compiler section by adding new compilers if found

Returns Updated compiler section for buildtest configuration

Return type dict

`_validate_modules(self, module_dict)`

This method will validate modules by running `module load` test for all discovered modules specified in parameter `discovered_modules`. This method returns a list of modules that were valid, if all tests pass we return the same list. A module test pass if we get a returncode 0.

find_compilers(*self*)

This method returns compiler modules discovered depending on your module system. If you have Lmod system we use spider utility to detect modules, this is leveraging Lmodule API. If you have environment-modules we parse output of `module av -t`.

Returns return a list of compiler modules detected based on module key name.

Return type dict

list(*self*)

Return all compilers defined in buildtest configuration

print_compilers(*self*)

This method implements `buildtest config compilers` which prints all compilers from buildtest configuration

print_json(*self*)

Prints compiler section in JSON, this implements `buildtest config compilers --json`

print_yaml(*self*)

Prints compiler section in YAML, this implements `buildtest config compilers --yaml`

`buildtest.cli.compilers.compiler_cmd(args, configuration)`

`buildtest.cli.compilers.compiler_find(args, configuration)`

This method implements `buildtest config compilers find` which detects new compilers based on module names defined in configuration. If system has Lmod we use Lmodule API to detect the compilers. For environment-modules we search for all modules in current `$MODULEPATH`.

buildtest.cli.config**Module Contents****Functions**

`config_cmd(args, configuration)`

`validate_config(configuration)`

This method implements `buildtest config validate` which attempts to

`view_configuration(configuration)`

View buildtest configuration file. This implements `buildtest config view`

`view_executors(configuration, buildexecutor, json_format=False, yaml_format=False)`

Display executors from buildtest configuration. This implements `buildtest config executors` command.

`view_summary(configuration, buildtestsystem=None)`

This method implements `buildtest config summary` option. In this method

`view_system(configuration)`

This method implements command `buildtest config systems` which displays

`buildtest.cli.config.config_cmd(args, configuration)`

`buildtest.cli.config.validate_config(configuration)`

This method implements `buildtest config validate` which attempts to validate buildtest settings with schema. If it's not validate an exception is raised which could be `ValidationError` or `ConfigurationError`.

`buildtest.cli.config.view_configuration(configuration)`

View buildtest configuration file. This implements `buildtest config view`

`buildtest.cli.config.view_executors(configuration, buildexecutor, json_format=False, yml_format=False)`

Display executors from buildtest configuration. This implements `buildtest config executors` command. If no option is specified we display output in JSON format

`buildtest.cli.config.view_summary(configuration, buildtestsystem=None)`

This method implements `buildtest config summary` option. In this method we will display a summary of System Details, Buildtest settings, Schemas, Repository details, Buildsspecs files and test names.

Parse buildtestsystem instance of class BuildTestSystem, optional

`buildtest.cli.config.view_system(configuration)`

This method implements command `buildtest config systems` which displays system details from configuration file in table format.

buildtest.cli.help

Module Contents

Functions

<code>buildtest_help()</code>	Entry point for <code>buildtest help</code> which display a summary of how to use buildtest commands
-------------------------------	--

`buildtest.cli.help.buildtest_help()`

Entry point for `buildtest help` which display a summary of how to use buildtest commands

buildtest.cli.history

Module Contents

Functions

<code>build_history(args)</code>	This is the entry point for command <code>buildtest build history</code> command which reports
<code>list_builds()</code>	
<code>query_builds(build_id, log_option)</code>	This method is called when user runs <code>buildtest history query</code> which will

Attributes

logger

`buildtest.cli.history.build_history(args)`

This is the entry point for command `buildtest build history` command which reports

`buildtest.cli.history.list_builds()`

`buildtest.cli.history.logger`

`buildtest.cli.history.query_builds(build_id, log_option)`

This method is called when user runs `buildtest history query` which will report the build.json and logfile.

Parameters

- **build_id** (*int*, *required*) – Input argument `buildtest history query <id>`
- **log_option** (*bool*, *required*) – Input argument `buildtest history query <id> -log`

`buildtest.cli.inspect`

This module implements methods for `buildtest inspect` command that can be used to retrieve test record from report file in JSON format.

Module Contents

Functions

<code>get_all_ids(report)</code>	Return all unique test ids from report cache
<code>inspect_by_id(test_ids, report, args)</code>	
<code>inspect_by_name(report, names)</code>	Implements command <code>buildtest inspect name</code> which will print all test records
<code>inspect_cmd(args)</code>	Entry point for <code>buildtest inspect</code> command
<code>inspect_list(test_ids, parse=None)</code>	Implements method <code>buildtest inspect list</code>

`buildtest.cli.inspect.get_all_ids(report)`

Return all unique test ids from report cache

Parameters **report** (*dict*) – loaded report file in JSON format

Returns a dictionary returning a list of unique test IDs

Return type *dict*

`buildtest.cli.inspect.inspect_by_id(test_ids, report, args)`

`buildtest.cli.inspect.inspect_by_name(report, names)`

Implements command `buildtest inspect name` which will print all test records by given name in JSON format.

`buildtest.cli.inspect.inspect_cmd(args)`

Entry point for `buildtest inspect` command

`buildtest.cli.inspect.inspect_list(test_ids, parse=None)`
Implements method `buildtest inspect list`

`buildtest.cli.report`

Module Contents

Classes

Report

Functions

is_int(val)

report_cmd(args)

Attributes

logger

class `buildtest.cli.report.Report`(*filter_args, format_args, latest, oldest, report_file=BUILD_REPORT*)

display_table

filter_fields = ['buildspec', 'name', 'executor', 'state', 'tags', 'returncode']

format_fields = ['buildspec', 'command', 'compiler', 'endtime', 'errfile',
'executor', 'full_id', 'hostname', ...]

_check_filter_fields(*self*)

_check_format_fields(*self*)

Check all format arguments (`--format`) are valid, the arguments are specified in format (`--format key1=val1,key2=val2`). We make sure each key is valid format field.

_filter_by_executor(*self, test*)

_filter_by_names(*self, name*)

Filter test by name of test. This method will return True if record should be processed, otherwise returns False

Parameters **name** (*str*) – Name of test

_filter_by_returncode(*self, test*)

_filter_by_state(*self, test*)

`_filter_by_tags(self, test)`

This method will return a boolean (True/False) to check if test should be skipped from report. Given an input test, we check if test has 'tags' property in buildspect and if tagnames specified by `--filter tags` are found in the test. If there is a match we return False. A True indicates the test will be filtered out.

Parameters `test (dict)` – test record

Returns Return True if test is filtered out, otherwise return False

Return type bool

`filter_buildspecs_from_report(self)`

This method filters the report table input filter `--filter buildspect`. If entry found in buildspect cache we add to list

`get_report_file(self)`

`load(self)`

This method is responsible for loading report file. If file not found or report is empty dictionary we raise an error. The report file is loaded using `json.loads` and return value is a dictionary containing entire report of all tests.

`print_display_table(self)`

`print_filter_fields(self)`

Implements command `buildtest report --helpfilter`

`print_format_fields(self)`

Implements command `buildtest report --helpformat`

`process_report(self)`

`buildtest.cli.report.is_int(val)`

`buildtest.cli.report.logger`

`buildtest.cli.report.report_cmd(args)`

`buildtest.cli.schema`

Module Contents

Functions

`schema_cmd(args)`

This method implements command `buildtest schema` which shows a list

`buildtest.cli.schema.schema_cmd(args)`

This method implements command `buildtest schema` which shows a list of schemas, their json content and list of schema examples. The input `args` is an instance of `argparse` class that contains user selection via command line. This method can do the following

`buildtest schema` - Show all schema names `buildtest schema --name <NAME> -j ```. View json content of a specified schema ```buildtest schema --name <NAME> -e`. Show schema examples
Parameters:

Parameters `args (<class 'argparse.Namespace'>)` – instance of `argparse` class

Result output of json schema on console

Package Contents

Functions

<i>build_menu</i> (subparsers)	This method implements command line menu for <code>buildtest build</code> command.
<i>buildspec_menu</i> (subparsers)	This method implements <code>buildtest buildspec</code> command
<i>cdash_menu</i> (subparsers)	This method builds arguments for <i>buildtest cdash</i> command.
<i>config_menu</i> (subparsers)	This method adds <code>argparse</code> argument for <code>buildtest config</code>
<i>get_parser</i> ()	
<i>handle_kv_string</i> (val)	This method is used as type field in <code>-filter</code> argument in <code>buildtest buildspec find</code> .
<i>history_menu</i> (subparsers)	This method builds the command line menu for <code>buildtest history</code> command
<i>inspect_menu</i> (subparsers)	This method builds argument for <i>buildtest inspect</i> command
<i>positive_number</i> (value)	Checks if input value is positive value and within range of 1-50. This method
<i>report_menu</i> (subparsers)	This method implements the <code>buildtest report</code> command options
<i>schema_menu</i> (subparsers)	This method builds menu for <code>buildtest schema</code>

Attributes

<i>BUILDTEST_COPYRIGHT</i>
<i>BUILDTEST_VERSION</i>
<i>BUILD_REPORT</i>
<i>schema_table</i>

`buildtest.cli.BUILDTEST_COPYRIGHT` = Copyright (c) 2021, The Regents of the University of California, through Lawrence Berkeley...

`buildtest.cli.BUILDTEST_VERSION` = 0.10.0

`buildtest.cli.BUILD_REPORT`

`buildtest.cli.build_menu`(subparsers)
This method implements command line menu for `buildtest build` command.

`buildtest.cli.buildspec_menu`(subparsers)
This method implements `buildtest buildspec` command

`buildtest.cli.cdash_menu`(subparsers)
This method builds arguments for *buildtest cdash* command.

`buildtest.cli.config_menu(subparsers)`

This method adds argparse argument for `buildtest config`

`buildtest.cli.get_parser()`

`buildtest.cli.handle_kv_string(val)`

This method is used as type field in `-filter` argument in `buildtest buildspect find`. This method returns a dict of key,value pair where input is in format `key1=val1,key2=val2,key3=val3`

Parameters `val (str)` – input value

Returns dictionary of key/value pairs

Return type dict

`buildtest.cli.history_menu(subparsers)`

This method builds the command line menu for `buildtest history` command

`buildtest.cli.inspect_menu(subparsers)`

This method builds argument for `buildtest inspect` command

`buildtest.cli.positive_number(value)`

Checks if input value is positive value and within range of 1-50. This method is used for `-rebuild` option

`buildtest.cli.report_menu(subparsers)`

This method implements the `buildtest report` command options

`buildtest.cli.schema_menu(subparsers)`

This method builds menu for `buildtest schema`

`buildtest.cli.schema_table`

`buildtest.executors`

Submodules

`buildtest.executors.base`

BuildExecutor: manager for test executors

Module Contents

Classes

BaseExecutor

The BaseExecutor is an abstract base class for all executors.

class `buildtest.executors.base.BaseExecutor(name, settings, site_configs)`

The BaseExecutor is an abstract base class for all executors.

type = base

__repr__(self)

Return repr(self).

__str__(self)

Return str(self).

_check_regex(*self*, *builder*)

This method conducts a regular expression check using `re.search` with regular expression defined in `Buildspec`. User must specify an output stream (`stdout`, `stderr`) to select when performing regex. In `buildtest`, this would read the `.out` or `.err` file based on stream and run the regular expression to see if there is a match. This method will return a boolean `True` indicates there is a match otherwise `False` if `regex` object not defined or `re.search` doesn't find a match.

Parameters **builder** (`BuilderBase` (*subclass*)) – instance of `BuilderBase` class

Returns A boolean return `True/False` based on if `re.search` is successful or not

Return type `bool`

_check_runtime(*self*, *builder*)

This method will return a boolean (`True/False`) based on runtime specified in `buildspec` and check with test runtime. User can specify both *min* and *max*, or just specify *min* or *max*.

_returncode_check(*self*, *builder*)

Check status check of `returncode` field if specified in `status` property.

Parameters **builder** (`BuilderBase` (*subclass*)) – instance of `BuilderBase` class

check_test_state(*self*, *builder*)

This method is responsible for detecting state of test (`PASS/FAIL`) based on `returncode` or regular expression.

Parameters **builder** (`BuilderBase` (*subclass*)) – instance of `BuilderBase` class

load(*self*)

Load a particular configuration based on the name. This method should set defaults for the executor, and will vary based on the class.

run(*self*)

The run step basically runs the build. This is run after setup so we are sure that the builder is defined. This is also where we set the result to return.

buildtest.executors.cobalt

This method implements `CobaltExecutor` class which defines how cobalt executor submit job to Cobalt scheduler.

Module Contents**Classes**

<code>CobaltExecutor</code>	The <code>CobaltExecutor</code> class is responsible for submitting jobs to Cobalt Scheduler.
<code>CobaltJob</code>	The <code>CobaltJob</code> class performs operation on cobalt job upon job submission such

Attributes

logger

class buildtest.executors.cobalt.**CobaltExecutor**(*name, settings, site_configs, max_pend_time=None*)
 Bases: [buildtest.executors.base.BaseExecutor](#)

The CobaltExecutor class is responsible for submitting jobs to Cobalt Scheduler. The class implements the following methods:

- **load**: load Cobalt executors from configuration file
- **dispatch**: submit Cobalt job to scheduler
- **poll**: poll Cobalt job via qstat and retrieve job state
- **gather**: gather job record including output, error, exit code

type = cobalt

dispatch(*self, builder*)

This method is responsible for dispatching job to Cobalt Scheduler by invoking `builder.run()` which runs the build script. If job is submitted to scheduler, we get the JobID and pass this to CobaltJob class. At job submission, cobalt will report the output and error file which can be retrieved using **qstat**. We retrieve the cobalt job record using `builder.job.gather()`.

Parameters **builder** ([BuilderBase](#), *required*) – builder object

gather(*self, builder*)

This method is responsible for moving output and error file in the run directory. We need to read <JOBID>.cobaltlog file which contains output of exit code by performing a regular expression (`exit code of.)(\d+)(\;)`). The cobalt log file will contain a line: **task completed normally with an exit code of 0; initiating job cleanup and removal**

Parameters **builder** ([BuilderBase](#), *required*) – builder object

launcher_command(*self*)

load(*self*)

Load the a Cobalt executor configuration from buildtest settings.

poll(*self, builder*)

This method is responsible for polling Cobalt job by invoking the builder method `builder.job.poll()`. We check the job state and existence of output file. If file exists or job is complete, we gather the results and return from function. If job is pending we check if job time exceeds `max_pend_time` time limit and cancel job.

Parameters **builder** ([BuilderBase](#), *required*) – builder object

class buildtest.executors.cobalt.**CobaltJob**(*jobID*)

Bases: [buildtest.executors.job.Job](#)

The CobaltJob class performs operation on cobalt job upon job submission such as polling job, gather job record, cancel job. We also retrieve job state and determine if job is pending, running, complete, suspended.

cancel(*self*)

Cancel job by running `qdel <jobid>`. This method is called if job timer exceeds `max_pend_time` if job is pending.

cobalt_log(*self*)
Return job cobalt.log file

error_file(*self*)
Return job error file

exitcode(*self*)
Return job exit code

gather(*self*)

Gather Job state by running `qstat -lf <jobid>` which retrieves all fields. The output is in text format which is parsed into key/value pair and stored in a dictionary. This method will return a dict containing the job record

```
$ qstat -lf 347106
JobID: 347106
JobName           : hold_job
User              : shahzebsiddiqui
WallTime          : 00:10:00
QueuedTime        : 00:13:14
RunTime           : N/A
TimeRemaining     : N/A
```

is_cancelled(*self*)
Return True if job is cancelled otherwise returns False. Job state is cancelled which is set by class cancel method

is_complete(*self*)
Return True if job is complete otherwise returns False. Cobalt job state for completed job job is marked as exiting

is_pending(*self*)
Return True if job is pending otherwise returns False. When cobalt receives job it is in starting followed by queued state. We check if job is in either state.

is_running(*self*)
Return True if job is running otherwise returns False. Cobalt job state for running job is marked as running

is_suspended(*self*)
Return True if job is suspended otherwise returns False. Cobalt job state for suspended is marked as user_hold

output_file(*self*)
Return job output file

poll(*self*)
Poll job by running `qstat -l --header State <jobid>` which retrieves job state.

buildtest.executors.cobalt.logger

buildtest.executors.job

Module Contents

Classes

<i>Job</i>	This is a base class for holding job level data and common methods for used
------------	---

class buildtest.executors.job.**Job**(*jobID*)

This is a base class for holding job level data and common methods for used for batch job submission.

abstract cancel(*self*)

Cancel job

get(*self*)

Return Job ID

abstract is_pending(*self*)

Check if job is in pending state

abstract is_running(*self*)

Check if job is in running state

abstract is_suspended(*self*)

Check if job is in suspended state

abstract poll(*self*)

Poll job and update job state.

state(*self*)

buildtest.executors.local

This module implements the LocalExecutor class responsible for submitting jobs to localhost. This class is called in class BuildExecutor when initializing the executors.

Module Contents

Classes

<i>LocalExecutor</i>	The LocalExecutor class is responsible for running tests locally for
----------------------	--

class buildtest.executors.local.**LocalExecutor**(*name, settings, site_configs*)

Bases: *buildtest.executors.base.BaseExecutor*

The LocalExecutor class is responsible for running tests locally for bash, sh and python shell. The LocalExecutor runs the tests and gathers the output and error results and writes to file. This class implements load, check and run method.

type = local

check(*self*)

Check if shell binary is available

load(*self*)

Load a particular configuration based on the name. This method should set defaults for the executor, and will vary based on the class.

run(*self, builder*)

This method is responsible for running test for LocalExecutor which runs test locally. We keep track of metadata in `builder.metadata` that keeps track of run result. The output and error file are written to filesystem.

Parameters **builder** (*BuilderBase*, *required*) – builder object

write_testresults(*self, builder*)

Upon execution of tests we write stdout and stderr to output and error file.

Parameters **builder** (*BuilderBase*, *required*) – builder object

buildtest.executors.lsf

This module implements the LSFExecutor class responsible for submitting jobs to LSF Scheduler. This class is called in class BuildExecutor when initializing the executors.

Module Contents

Classes

<i>LSFExecutor</i>	The LSFExecutor class is responsible for submitting jobs to LSF Scheduler.
<i>LSFJob</i>	This is a base class for holding job level data and common methods for used

Attributes

<i>logger</i>

class buildtest.executors.lsf.**LSFExecutor**(*name, settings, site_configs, max_pend_time=None*)

Bases: *buildtest.executors.base.BaseExecutor*

The LSFExecutor class is responsible for submitting jobs to LSF Scheduler. The LSFExecutor performs the following steps

- **load**: load lsf configuration from buildtest configuration file
- **dispatch**: dispatch job to scheduler and acquire job ID
- **poll**: wait for LSF jobs to finish
- **gather**: Once job is complete, gather job data

type = **lsf**

dispatch(*self*, *builder*)

This method is responsible for dispatching job to scheduler and extracting job ID by applying a `re.search` against output at onset of job submission. If job id is not retrieved due to job failure or unable to match regular expression we mark job incomplete by invoking `builder.incomplete()` method and return from method.

If we have a valid job ID we invoke `LSFJob` class given the job id to poll job and store this into `builder.job` attribute.

Parameters `builder` (`BuilderBase`, *required*) – builder object

gather(*self*, *builder*)

Gather Job detail after completion of job by invoking the builder method `builder.job.gather()`. We retrieve exit code, output file, error file and update builder metadata.

Parameters `builder` (`BuilderBase`, *required*) – builder object

launcher_command(*self*)

This command returns the launcher command and any options specified in configuration file. This is useful when generating the build script in the `BuilderBase` class

load(*self*)

Load the a LSF executor configuration from buildtest settings.

poll(*self*, *builder*)

Given a builder object we poll the job by invoking builder method `builder.job.poll()` return state of job. If job is suspended or pending we stop timer and check if timer exceeds `max_pend_time` value which could be defined in configuration file or passed via command line `--max-pend-time`

Parameters `builder` (`BuilderBase`, *required*) – builder object

class `buildtest.executors.lsf.LSFJob`(*jobID*)

Bases: `buildtest.executors.job.Job`

This is a base class for holding job level data and common methods for used for batch job submission.

cancel(*self*)

Cancel LSF Job by running `bkill <jobid>`. This is called if job has exceeded `max_pend_time` limit during poll stage.

error_file(*self*)

Return job error file

exitcode(*self*)

Return job exit code

gather(*self*)

Gather Job record at onset of job completion by running `bjobs -o '<format1> <format2>' <jobid> -json`. The format fields extracted from job are the following:

- “job_name”
- “stat”
- “user”
- “user_group”
- “queue”
- “proj_name”
- “pids”
- “exit_code”

- “from_host”
- “exec_host”
- “submit_time”
- “start_time”
- “finish_time”
- “nthreads”
- “exec_home”
- “exec_cwd”
- “output_file”
- “error_file”

Shown below is the output format and we retrieve the job records defined in **RECORDS** property

```
$ bjobs -o 'job_name stat user user_group queue proj_name pids exit_code from_
↪host exec_host submit_time start_time finish_time nthreads exec_home exec_cwd_
↪output_file error_file' 58652 -json
{
  "COMMAND": "bjobs",
  "JOBS": 1,
  "RECORDS": [
    {
      "JOB_NAME": "hold_job",
      "STAT": "PSUSP",
      "USER": "shahzebsiddiqui",
      "USER_GROUP": "GEN014ECPCI",
      "QUEUE": "batch",
      "PROJ_NAME": "GEN014ECPCI",
      "PIDS": "",
      "EXIT_CODE": "",
      "FROM_HOST": "login1",
      "EXEC_HOST": "",
      "SUBMIT_TIME": "May 28 12:45",
      "START_TIME": "",
      "FINISH_TIME": "",
      "NTHREADS": "",
      "EXEC_HOME": "",
      "EXEC_CWD": "",
      "OUTPUT_FILE": "hold_job.out",
      "ERROR_FILE": "hold_job.err"
    }
  ]
}
```

is_complete(self)

Check if Job is complete which is in DONE state. Return True if there is a match otherwise return False

is_failed(self)

Check if Job failed. We return True if job is in EXIT state otherwise return False

is_pending(self)

Check if Job is pending which is reported by LSF as PEND. Return True if there is a match otherwise returns False

is_running(self)

Check if Job is running which is reported by LSF as RUN. Return True if there is a match otherwise returns False

is_suspended(self)

Check if Job is in suspended state which could be in any of the following states: [PSUSP, USUSP, SSUSP]. We return True if job is in one of the states otherwise return False

output_file(self)

Return job output file

poll(self)

Given a job id we poll the LSF Job by retrieving its job state, output file, error file and exit code. We run the following commands to retrieve following states

- Job State: `bjobs -noheader -o 'stat' <JOBID>`
- Output File: `bjobs -noheader -o 'output_file' <JOBID>`
- Error File: `bjobs -noheader -o 'error_file' <JOBID>`
- Exit Code File: `bjobs -noheader -o 'EXIT_CODE' <JOBID>`

`buildtest.executors.lsf.logger`

buildtest.executors.pbs

This module implements PBSExecutor class that defines how executors submit job to PBS Scheduler

Module Contents

Classes

<i>PBSExecutor</i>	The PBSExecutor class is responsible for submitting jobs to PBS Scheduler.
<i>PBSJob</i>	See https://www.altair.com/pdfs/pbsworks/PBSReferenceGuide2021.1.pdf section 8.1 for Job State Codes

Attributes

<i>logger</i>

class `buildtest.executors.pbs.PBSExecutor(name, settings, site_configs, max_pend_time=None)`

Bases: `buildtest.executors.base.BaseExecutor`

The PBSExecutor class is responsible for submitting jobs to PBS Scheduler. The class implements the following methods:

load: load PBS executors from configuration file
 dispatch: submit PBS job to scheduler
 poll: poll PBS job via qstat and retrieve job state
 gather: gather job result
 cancel: cancel job if it exceeds max pending time

poll_cmd = qstat

type = pbs

dispatch(*self*, *builder*)

This method is responsible for dispatching PBS job, get JobID and start record metadata in builder object. If job failed to submit we check returncode and exit with failure. After we submit job, we start timer and record when job was submitted and poll job once to get job details and store them in builder object.

Parameters **builder** (*BuilderBase*, *required*) – builder object

gather(*self*, *builder*)

This method is responsible for getting output of job using `qstat -x -f -F json <jobID>` and storing the result in builder object. We retrieve specific fields such as exit status, start time, end time, runtime and store them in builder object. We read output and error file and store the content in builder object.

Parameters **builder** (*BuilderBase*, *required*) – builder object

launcher_command(*self*)

load(*self*)

Load the a Cobalt executor configuration from buildtest settings.

poll(*self*, *builder*)

This method is responsible for polling Cobalt job, we check the job state and existence of output file. If file exists or job is in 'exiting' stage we set job to 'done' stage and gather results. If job is in 'pending' stage we check if job exceeds 'max_pend_time' time limit by checking with builder timer attribute using `start` and `stop` method. If job exceeds the time limit job is cancelled.

Parameters **builder** (*BuilderBase*, *required*) – builder object

class buildtest.executors.pbs.PBSJob(*jobID*)

Bases: *buildtest.executors.job.Job*

See <https://www.altair.com/pdfs/pbsworks/PBSReferenceGuide2021.1.pdf> section 8.1 for Job State Codes

cancel(*self*)

Cancel job

error_file(*self*)

exitcode(*self*)

fail(*self*)

gather(*self*)

is_complete(*self*)

is_pending(*self*)

Check if job is in pending state

is_running(*self*)

Check if job is in running state

is_suspended(*self*)

Check if job is in suspended state

output_file(*self*)

poll(*self*)

Poll job and update job state.

success(*self*)

This method determines if job was completed successfully. According to <https://www.altair.com/pdfs/pbsworks/PBSAdminGuide2021.1.pdf> section 14.9 Job Exit Status Codes we have the following:

Exit Code: $X < 0$ - Job could not be executed
Exit Code: $0 \leq X < 128$ - Exit value of Shell or top-level process
Exit Code: $X \geq 128$ - Job was killed by signal

Exit Code 0 is a success

`buildtest.executors.pbs.logger`

buildtest.executors.setup

This module is responsible for setup of executors defined in buildtest configuration. The BuildExecutor class initializes the executors and chooses the executor class (LocalExecutor, LSFExecutor, SlurmExecutor, CobaltExecutor) to call depending on executor name.

Module Contents

Classes

<i>BuildExecutor</i>	A BuildExecutor is responsible for initialing executors from buildtest configuration
----------------------	--

Attributes

<i>logger</i>

class `buildtest.executors.setup.BuildExecutor`(*site_config*, *max_pend_time=None*)

A BuildExecutor is responsible for initialing executors from buildtest configuration file which provides a list of executors. This class keeps track of all executors and provides the following methods:

setup: This method will write executor's `before_script.sh` that is sourced in each test upon calling executor.

run: Responsible for invoking executor's `run` method based on builder object which is of type `BuilderBase`. **poll**: This is responsible for invoking `poll` method for corresponding executor from the builder object by checking job state

__repr__(*self*)

Return repr(self).

__str__(*self*)

Return str(self).

_choose_executor(*self*, *builder*)

Choose executor is called at the onset of a run and poll stage. Given a builder object we retrieve the executor property `builder.executor` of the builder and check if there is an executor object and of type `BaseExecutor`.

Parameters `builder` (`BuilderBase` (subclass), *required.*) – the builder with the loaded Builds spec.

get(*self*, *name*)

Given the name of an executor return the executor object which is of subclass of *BaseExecutor*

is_cobalt(*self*, *executor_type*)

is_local(*self*, *executor_type*)

is_lsf(*self*, *executor_type*)

is_pbs(*self*, *executor_type*)

is_slurm(*self*, *executor_type*)

list_executors(*self*)

poll(*self*, *builders*)

The poll stage is called after the *run* stage for builders that require job submission through a batch executor. Given a set of builders object which are instance of *BuilderBase*, we select the executor object and invoke the *poll* method for the executor.

1. If job is pending, running, suspended we poll job
2. If job is complete we gather job results and mark job complete
3. Otherwise we mark job incomplete and it will be ignored by buildtest in reporting

Poll all jobs for batch executors (LSF, Slurm, Cobalt, PBS). For slurm we poll until job is in PENDING or RUNNING state. If Slurm job is in FAILED or COMPLETED state we assume job is finished and we gather results. If its in any other state we ignore job and return out of method.

For LSF jobs we poll job if it's in PEND or RUN state, if its in DONE state we gather results, otherwise we assume job is incomplete and return with *ignore_job* set to True. This informs buildtest to ignore job when showing report.

For Cobalt jobs, we poll if its in *starting*, *queued*, or *running* state. For Cobalt jobs we cannot query job after its complete since JobID is no longer present in queuing system. Therefore, for when job is complete which is *done* or *exiting* state, we mark job is complete.

For PBS jobs we poll job if its in *queued* or *running* stage which corresponds to Q and R in job stage. If job is finished (F) we gather results. If job is in H stage we automatically cancel job otherwise we ignore job and mark job complete.

Parameters *builder* (*list* , *required*) – a list of builder objects for polling. Each element is an instance of *BuilderBase* (subclass)

Returns Return a list of builders

Return type list

run(*self*, *builder*)

This method implements the executor run implementation. Given a builder object we first detect the correct executor object to use and invoke its *run* method. The executor object is a sub-class of *BaseExecutor* (i.e *LocalExecutor*, *SlurmExecutor*, *LSFExecutor*,...).

Parameters *builder* (*BuilderBase* (subclass), *required*.) – the builder with the loaded test configuration.

setup(*self*)

This method creates directory *var/executors/<executor-name>* for every executor defined in buildtest configuration and write scripts *before_script.sh* if the field *before_script* is specified in executor section. This method is called after executors are initialized in the class *__init__* method.

`buildtest.executors.setup.logger`

buildtest.executors.slurm

This module implements the `SlurmExecutor` class responsible for submitting jobs to Slurm Scheduler. This class is called in class `BuildExecutor` when initializing the executors.

Module Contents**Classes**

<i><code>SlurmExecutor</code></i>	The <code>SlurmExecutor</code> class is responsible for submitting jobs to Slurm Scheduler.
<i><code>SlurmJob</code></i>	This is a base class for holding job level data and common methods for used

Attributes

<i><code>logger</code></i>

class `buildtest.executors.slurm.SlurmExecutor`(*name, settings, site_configs, max_pend_time=None*)

Bases: `buildtest.executors.base.BaseExecutor`

The `SlurmExecutor` class is responsible for submitting jobs to Slurm Scheduler. The `SlurmExecutor` performs the following steps:

- **load**: load slurm configuration from buildtest configuration file
- **dispatch**: dispatch job to scheduler and acquire job ID
- **poll**: wait for Slurm jobs to finish, if job is pending and exceeds *max_pend_time* then cancel job
- **gather**: Once job is complete, gather job data

type = `slurm`

dispatch(*self, builder*)

This method is responsible for dispatching job to slurm scheduler and extracting job id. If job id is valid we pass the job to `SlurmJob` class and store object in `builder.job`.

Parameters **builder** (`BuilderBase`, *required*) – builder object

gather(*self, builder*)

Gather Slurm job data after job completion. In this step we call `builder.job.gather()`, and update builder metadata such as returncode, output and error file.

Parameters **builder** (`BuilderBase` (*subclass*), *required*) – instance of `BuilderBase`

launcher_command(*self*)

Return sbatch launcher command with options used to submit job

load(*self*)

Load the a slurm executor configuration from buildtest settings.

poll(*self, builder*)

This method is called during poll stage where we invoke `builder.job.poll()` to get updated job state.

If job is pending or suspended we stop timer and check if job needs to be cancelled if time exceeds *max_pend_time* value.

Parameters **builder** (*BuilderBase*, *required*) – builder object

class `buildtest.executors.slurm.SlurmJob(jobID, cluster=None)`

Bases: `buildtest.executors.job.Job`

This is a base class for holding job level data and common methods for used for batch job submission.

cancel(*self*)

Cancel job by running `scancel <jobid>`. If job is specified to a slurm cluster we cancel job using `scancel <jobid> --clusters=<cluster>`. This method is called if job exceeds *max_pend_time*.

complete(*self*)

This method is used for gathering job result we assume job is complete if it's in any of the following state: COMPLETED, FAILED, OUT_OF_MEMORY, TIMEOUT

exitcode(*self*)

Return job exit code

gather(*self*)

Gather job record which is called after job completion. We use *sacct* to gather job record and return the job record as a dictionary. The command we run is `sacct -j <jobid> -X -n -P -o <field1>, <field2>, ..., <fieldN>`. We retrieve the following format fields from job record:

- “Account”
- “AllocNodes”
- “AllocTRES”
- “ConsumedEnergyRaw”
- “CPUTimeRaw”
- “Elapsed”
- “End”
- “ExitCode”
- “JobID”
- “JobName”
- “NCPUS”
- “NNodes”
- “QOS”
- “ReqGRES”
- “ReqMem”
- “ReqNodes”
- “ReqTRES”
- “Start”
- “State”
- “Submit”
- “UID”

- “User”
- “WorkDir”

The output of `sacct` is parseable using the pipe symbol (`|`) and stored into a dict

```
$ sacct -j 42909266 -X -n -P -o Account,AllocNodes,AllocTRES,ConsumedEnergyRaw,
CPUTimeRaw,Elapsed,End,ExitCode,JobID,JobName,NCPUS,NNodes,QOS,ReqGRES,ReqMem,
ReqNodes,ReqTRES,Start,State,Submit,UID,User,WorkDir --clusters=cori
nstaff|1|billing=272,cpu=272,energy=262,mem=87G,node=1|262|2176|00:00:08|2021-
05-27T18:47:49|0:0|42909266|slurm_metadata|272|1|debug_knl|PER_
NODE:craynetwork:1|87Gn|1|billing=1,cpu=1,node=1|2021-05-
27T18:47:41|COMPLETED|2021-05-27T18:44:07|92503|siddiq90|/global/u1/s/
siddiq90/.buildtest/tests/cori.slurm.knl_debug/metadata/slurm_metadata/0/stage
```

is_cancelled(*self*)

If job is cancelled return True otherwise return False. Slurm will report CANCELLED for job state.

is_complete(*self*)

If job is complete return True otherwise return False. Slurm will report COMPLETED for job state.

is_failed(*self*)

If job failed return True otherwise return False. Slurm will report FAILED for job state.

is_out_of_memory(*self*)

If job is out of memory return True otherwise return False. Slurm will report OUT_OF_MEMORY for job state.

is_pending(*self*)

If job is pending return True otherwise return False. Slurm Job state for pending is PENDING.

is_running(*self*)

If job is running return True otherwise return False. Slurm will report RUNNING for job state.

is_suspended(*self*)

If job is suspended return True otherwise return False. Slurm will report SUSPENDED for job state.

is_timeout(*self*)

If job timed out return True otherwise return False. Slurm will report TIMEOUT for job state.

poll(*self*)

Poll job to extract job state and exit code. We also retrieve job work directory. We run the following commands to retrieve the following properties.

- Job State: `sacct -j <jobid> -o State -n -X -P`
- ExitCode and Workdir: `sacct -j <jobid> -X -n -P -o ExitCode,Workdir`

state(*self*)

Return job state

workdir(*self*)

Return job work directory

`buildtest.executors.slurm.logger`

`buildtest.schemas`

Submodules

`buildtest.schemas.defaults`

Module Contents

Functions

<code>custom_validator(recipe, schema)</code>	This is a custom validator for validating JSON documents. We implement a
---	--

Attributes

<code>here</code>
<code>resolver</code>
<code>schema_store</code>
<code>schema_table</code>

`buildtest.schemas.defaults.custom_validator(recipe, schema)`

This is a custom validator for validating JSON documents. We implement a custom resolver for finding json schemas locally by implementing a schema store. The input arguments `recipe` and `schema` is your input JSON recipe and schema content for validating the recipe. This method uses Draft7Validator for validating schemas.

Parameters

- **recipe** (*dict*) – Input recipe as JSON document
- **schema** (*dict*) – Input JSON Schema content to validate JSON document

`buildtest.schemas.defaults.here`

`buildtest.schemas.defaults.resolver`

`buildtest.schemas.defaults.schema_store`

`buildtest.schemas.defaults.schema_table`

buildtest.schemas.utils

Utility and helper functions for schemas.

Module Contents

Functions

<code>load_recipe(path)</code>	Load a yaml recipe file. The file must be in .yaml extension
<code>load_schema(path)</code>	Load a json schema file, the file extension must be '.schema.json'

Attributes

<code>here</code>

buildtest.schemas.utils.**here**

buildtest.schemas.utils.**load_recipe**(*path*)

Load a yaml recipe file. The file must be in .yaml extension for buildtest to load.

Parameters *path* (*str*) – the path to the recipe file.

buildtest.schemas.utils.**load_schema**(*path*)

Load a json schema file, the file extension must be '.schema.json'

Parameters *path* (*str*) – the path to the schema file.

buildtest.utils

Submodules

buildtest.utils.command

Module Contents

Classes

<code>BuildTestCommand</code>	Class method to invoke shell commands and retrieve output and error.
<code>Capturing</code>	capture output from stdout and stderr into capture object.

class buildtest.utils.command.**BuildTestCommand**(*cmd=None*)

Class method to invoke shell commands and retrieve output and error. This class is inspired and derived from utils functions in <https://github.com/vsoch/scif>

decode(*self*, *line*)

Given a line of output (error or regular) decode using the system default, if appropriate

execute(*self*)

Execute a system command and return output and error. :param cmd: shell command to execute :type cmd: str, required :return: Output and Error from shell command :rtype: two str objects

get_command(*self*)

Returns the executed command :rtype: str

get_error(*self*)

Returns the error from shell command :rtype: str

get_output(*self*)

Returns the output from shell command :rtype: str

returncode(*self*)

Returns the return code from shell command :rtype: int

set_command(*self*, *cmd*)

parse is called when a new command is provided to ensure we have a list. We don't check that the executable is on the path, as the initialization might not occur in the runtime environment.

class buildtest.utils.command.Capturing

capture output from stdout and stderr into capture object. This is based off of github.com/vsoch/gridtest but modified to write files. The stderr and stdout are set to temporary files at the init of the capture, and then they are closed when we exit. This means expected usage looks like:

with Capturing() **as** capture: process = subprocess.Popen(...)

And then the output and error are retrieved from reading the files: and exposed as properties to the client:

capture.out capture.err

And cleanup means deleting these files, if they exist.

__enter__(*self*)

__exit__(*self*, *args)

cleanup(*self*)

property err(*self*)

Return error stream. Returns empty string if empty or doesn't exist. Returns (str) : error stream written to file

property out(*self*)

Return output stream. Returns empty string if empty or doesn't exist. Returns (str) : output stream written to file

set_stderr(*self*)

set_stdout(*self*)

buildtest.utils.file

This module provides some generic file and directory level operation that include the following: 1. Check if path is a File or Directory via `is_file()`, `is_dir()` 2. Create a directory via `create_dir()` 3. Walk a directory tree based on single extension using `walk_tree()` 4. Resolve path including shell and user expansion along with getting realpath to file using `resolve_path()` 5. Read and write a file via `read_file()`, `write_file()`

Module Contents**Functions**

<code>create_dir(dirname)</code>	Create a directory if it doesn't exist. If directory contains variable
<code>is_dir(dirname)</code>	This method will check if a directory exist. If directory found we return
<code>is_file(fname)</code>	This method will check if file exist, if so returns True otherwise returns False
<code>load_json(fname)</code>	Given a filename, resolves full path to file and loads json file. This method will
<code>read_file(filepath)</code>	This method is used to read a file specified by argument <code>filepath</code> .
<code>remove_file(fpath)</code>	This method is responsible for removing a file. The input path is an absolute path
<code>resolve_path(path, exist=True)</code>	This method will resolve a file path to account for shell expansion and resolve paths in
<code>walk_tree(root_dir, ext=None)</code>	This method will traverse a directory tree and return list of files
<code>write_file(filepath, content)</code>	This method is used to write an input content to a file specified by

buildtest.utils.file.create_dir(*dirname*)

Create a directory if it doesn't exist. If directory contains variable expansion (\$HOME), user expansion (~) we resolve this before creating directory. If there is an error creating directory we raise an exception

Parameters `dirname` (*str*, *required*) – directory path to create

Returns creates the directory or print an exception message upon failure

Return type Catches exception of type `OSError`

buildtest.utils.file.is_dir(*dirname*)

This method will check if a directory exist. If directory found we return True otherwise False.

Parameters `dir` (*str*, *required*) – directory path

Returns returns a boolean True/False depending on if input is a valid directory.

Return type bool

buildtest.utils.file.is_file(*fname*)

This method will check if file exist, if so returns True otherwise returns False

Parameters `file` (*str*, *required*) – file path

Returns returns a boolean True/False depending on if input is a valid file.

Return type bool

`buildtest.utils.file.load_json(fname)`

Given a filename, resolves full path to file and loads json file. This method will catch exception `json.JSONDecodeError` and raise an exception with useful message. If there is no error we return content of json file

`buildtest.utils.file.read_file(filepath)`

This method is used to read a file specified by argument `filepath`. If `filepath` is not a string we raise an error. We also run `resolve_path` to get realpath to file and account for shell or user expansion. The return from `resolve_path` will be a valid file or `None` so we check if input is an invalid file. Finally we read the file and return the content of the file as a string.

Parameters `filepath` (*str*, *required*) – file name to read

Raises `BuildTestError`: If `filepath` is not a string `BuildTestError`: If `filepath` is not valid file

Returns return content of file as a string

Return type *str*

`buildtest.utils.file.remove_file(fpath)`

This method is responsible for removing a file. The input path is an absolute path to file. We check for exceptions first, and return immediately before removing file.

Parameters `fpath` (*str*, *required*) – full path to file

`buildtest.utils.file.resolve_path(path, exist=True)`

This method will resolve a file path to account for shell expansion and resolve paths in when a symlink is provided in the file. This method assumes file already exists.

param path file path to resolve

type path *str*, *required*

param exist expects a boolean to determine if `filepath` should be returned or `None`. By default, *exist* is *True*

and file is checked using `os.path.exists` to return full path.

```
>>> a = resolve_path("$HOME/.bashrc")
>>> assert a
>>> b = resolve_path("$HOME/.bashrc1", exist=False)
>>> assert b
>>> c = resolve_path("$HOME/.bashrc1", exist=True)
>>> assert not c
```

file is checked `os.path.exists` after getting realpath using `os.path.realpath()`. `resolve` :return: return realpath to file if found otherwise return `None` :rtype: *str* or `None`

`buildtest.utils.file.walk_tree(root_dir, ext=None)`

This method will traverse a directory tree and return list of files based on extension type. This method invokes `is_dir()` to check if directory exists before traversal.

Parameters:

Parameters

- **root_dir** (*str*, *required*) – directory path to traverse
- **ext** (*str*, *optional*) – file extensions to search in traversal

Returns returns a list of file paths

Return type list

`buildtest.utils.file.write_file(filepath, content)`

This method is used to write an input `content` to a file specified by `filepath`. Both `filepath` and `content` must be a `str`. An error is raised if `filepath` is not a string or a directory. If `content` is not a `str`, we return `None` since we can't process the content for writing. Finally, we write the content to file and return. A successful write will return nothing otherwise an exception will occur during the write process.

Parameters

- **filepath** (*str, required*) – file name to write
- **content** (*str, required*) – content to write to file

Raises `BuildTestError`: System error if `filepath` is not string `BuildTestError`: System error if `filepath` is a directory

Returns Return nothing if write is successful. A system error if `filepath` is not `str` or directory. If argument `content` is not `str` we return `None`

`buildtest.utils.shell`

Module Contents

Classes

Shell

`class buildtest.utils.shell.Shell(shell='bash')`

`valid_shells = ['bash', 'sh', 'zsh', 'csh', 'tcsh', '/bin/bash', '/bin/csh',
'/bin/sh', '/bin/tcsh', ...]`

`__repr__(self)`
Return `repr(self)`.

`__str__(self)`
Return `str(self)`.

`get(self)`
Return shell attributes as a dictionary

`property opts(self)`
retrieve the shell opts that are set on init, and updated with setter

`property path(self)`
This method returns the full path to shell program using `shutil.which()` If shell program is not found we raise an exception. The shebang is updated assuming path is valid which is just adding character `'#!'` in front of path. The return is full path to shell program. This method automatically updates the shell path when there is a change in attribute `self.name`

```
>>> shell = Shell("bash")
>>> shell.path
'/usr/bin/bash'
>>> shell.name="sh"
>>> shell.path
'/usr/bin/sh'
```

buildtest.utils.timer

Module Contents

Classes

Timer

class buildtest.utils.timer.Timer

start(self)

Start a new timer

stop(self)

Stop the timer, and report the elapsed time

exception buildtest.utils.timer.TimerError

Bases: Exception

A custom exception used to report errors in use of Timer class

buildtest.utils.tools

Module Contents

Classes

Hasher

dict() -> new empty dictionary

Functions

deep_get(dictionary, *keys)

class buildtest.utils.tools.Hasher

Bases: dict

dict() -> new empty dictionary dict(mapping) -> new dictionary initialized from a mapping object's

(key, value) pairs

dict(iterable) -> new dictionary initialized as if via: `d = { } for k, v in iterable:`

`d[k] = v`

dict(kwargs)** -> new dictionary initialized with the name=value pairs in the keyword argument list. For example: `dict(one=1, two=2)`

__missing__(self, key)

__str__(self)

Return str(self).

get(self, path, sep='.', default=None)

`D.get(k[,d]) -> D[k] if k in D, else d. d defaults to None.`

`buildtest.utils.tools.deep_get(dictionary, *keys)`

Submodules

`buildtest.config`

Module Contents

Classes

SiteConfiguration

This class is an interface to buildtest configuration

Attributes

logger

class `buildtest.config.SiteConfiguration(settings_file=None)`

This class is an interface to buildtest configuration

_executor_check(self)

_validate_cobalt_executors(self)

Validate cobalt queue property by running `qstat -Ql <queue>`. If its a non-zero exit code then queue doesn't exist otherwise it is a valid queue.

_validate_lsf_executors(self)

This method validates all LSF executors. We check if queue is available and in `Open:Active` state.

_validate_pbs_executors(self)

Validate pbs queue property by running by checking if queue is found and queue is 'enabled' and 'started' which are two properties found in pbs queue configuration that can be retrieved using `qstat -Q -f -F json`. The output is in the following format

```
$ qstat -Q -f -F json
{
```

(continues on next page)

(continued from previous page)

```
"timestamp":1615924938,
"pbs_version":"19.0.0",
"pbs_server":"pbs",
"Queue":{
  "workq":{
    "queue_type":"Execution",
    "total_jobs":0,
    "state_count":"Transit:0 Queued:0 Held:0 Waiting:0 Running:0_
↳Exiting:0 Begun:0 ",
    "resources_assigned":{
      "mem":"0kb",
      "ncpus":0,
      "nodect":0
    },
    "hasnodes":"True",
    "enabled":"True",
    "started":"True"
  }
}
```

_validate_slurm_executors(*self*)

This method will validate slurm executors, we check if partition, qos, and cluster fields are valid values by retrieving details from slurm configuration. These checks are performed on fields `partition`, `qos` or `cluster` if specified in executor section.

detect_system(*self*)

This method gets current system by setting `self.target` by matching `hostnames` entry in each system list with actual system. We retrieve target hostname and determine which system configuration to use. If no system is found we raise an error.

property file(*self*)**load(*self*)**

Loads configuration file

name(*self*)

Return name of matched system from configuration file

resolve(*self*)

This method will resolve path to configuration file. The order of precedence is as follows:

1. command line argument - Must be valid path
2. User Configuration: `$HOME/.buildtest/config.yml`
3. Default Configuration: `$BUILDTEST_ROOT/buildtest/settings/config.yml`

validate(*self*, *validate_executors=True*)

This method validates the site configuration with schema

`buildtest.config.logger`

buildtest.defaults

Buildtest defaults, including environment variables and paths, are defined or derived here.

Module Contents

```
buildtest.defaults.BUILDSPEC_CACHE_FILE
buildtest.defaults.BUILDSPEC_DEFAULT_PATH
buildtest.defaults.BUILDSPEC_ERROR_FILE
buildtest.defaults.BUILDTEST_BUILDSPEC_DIR
buildtest.defaults.BUILDTEST_DEFAULT_TESTDIR
buildtest.defaults.BUILDTEST_EXECUTOR_DIR
buildtest.defaults.BUILDTEST_REPORT_SUMMARY
buildtest.defaults.BUILDTEST_ROOT
buildtest.defaults.BUILDTEST_USER_HOME
buildtest.defaults.BUILD_HISTORY_DIR
buildtest.defaults.BUILD_REPORT
buildtest.defaults.DEFAULT_SETTINGS_FILE
buildtest.defaults.DEFAULT_SETTINGS_SCHEMA
buildtest.defaults.SCHEMA_ROOT
buildtest.defaults.USER_SETTINGS_FILE
buildtest.defaults.VAR_DIR
buildtest.defaults.supported_schemas
buildtest.defaults.supported_type_schemas = ['script-v1.0.schema.json',
'compiler-v1.0.schema.json']
buildtest.defaults.userhome
```

buildtest.exceptions

Module Contents

```
exception buildtest.exceptions.BuildTestError(msg, *args)
    Bases: Exception
    Class responsible for error handling in buildtest. This is a sub-class of Exception class.
    __str__(self)
        Return str(self).
exception buildtest.exceptions.BuildspecError(buildspec, msg)
    Bases: Exception
    raise exception if there is an issue with Buildspec in parsing or building test
```

```
__str__(self)
    Return str(self).
```

exception buildtest.exceptions.**ConfigurationError**(*config, settings_file, msg*)

Bases: Exception

This will raise an error related with buildtest configuration file

```
__str__(self)
    Return str(self).
```

exception buildtest.exceptions.**ExecutorError**

Bases: Exception

This class raises an error with Executor class and its operation

buildtest.log

Methods related to buildtest logging

Module Contents

Functions

<code>init_logfile(logfile=FILE_LOG, debug=None)</code>	Initialize a log file intended for a builder. This requires
---	---

Attributes

`FILE_LOG`

`LOG_FORMATTER`

`LOG_NAME`

buildtest.log.**FILE_LOG**

buildtest.log.**LOG_FORMATTER** = `%(asctime)s [%(filename)s:%(lineno)s - %(funcName)5s()] - [%(levelname)s] %(message)s`

buildtest.log.**LOG_NAME** = **buildtest**

buildtest.log.**init_logfile**(*logfile=FILE_LOG, debug=None*)

Initialize a log file intended for a builder. This requires passing the filename intended for the log (from the builder) and returns the logger. :param logfile: logfile name :type logfile: str

buildtest.main

Entry point for buildtest

Module Contents**Functions**

<i>main()</i>	Entry point to buildtest.
---------------	---------------------------

buildtest.main.main()
Entry point to buildtest.

buildtest.system

This module detects System changes defined in class BuildTestSystem.

Module Contents**Classes**

<i>BuildTestSystem</i>	BuildTestSystem is a class that detects system configuration
<i>Cobalt</i>	The Cobalt class checks for Cobalt binaries and gets a list of Cobalt queues
<i>LSF</i>	The LSF class checks for LSF binaries and returns a list of LSF queues
<i>PBS</i>	The PBS class checks for Cobalt binaries and gets a list of Cobalt queues
<i>Scheduler</i>	This is a base Scheduler class used for implementing common methods for
<i>Slurm</i>	The Slurm class implements common functions to query Slurm cluster

Attributes

<i>system</i>

class buildtest.system.BuildTestSystem

BuildTestSystem is a class that detects system configuration

system**check(self)**

Based on the module “distro” get system details like linux distro, processor, hostname, etc...

check_scheduler(*self*)

Check existence of batch scheduler and if so determine which scheduler it is. Currently we support Slurm, LSF, and Cobalt we invoke each class and see if its valid state. The checks determine if scheduler binaries exist in \$PATH.

detect_module_tool(*self*)

Check if module tool exists, we check for Lmod or environment-modules by checking if environment variable LMOD_VERSION, MODULE_VERSION or MODULES_CMD exist. We check this with input specification in buildtest configuration. If user specifies lmod as the module tool but detected environment-modules, buildtest should pick this up and report this as part of configuration check

get(*self*)

class buildtest.system.Cobalt

Bases: [Scheduler](#)

The Cobalt class checks for Cobalt binaries and gets a list of Cobalt queues

binaries = ['qsub', 'qstat', 'qdel', 'nodelist', 'showres', 'partlist']

_get_queues(*self*)

Get all Cobalt queues by running qstat -Ql and parsing output

class buildtest.system.LSF

Bases: [Scheduler](#)

The LSF class checks for LSF binaries and returns a list of LSF queues

binaries = ['bsub', 'bqueues', 'bkill', 'bjobs']

_get_queues(*self*)

Return json dictionary of available LSF Queues and their queue states. The command we run is the following: bqueues -o 'queue_name status' -json which returns a JSON record of all queue details.

```
$ bqueues -o 'queue_name status' -json
{
  "COMMAND": "bqueues",
  "QUEUES": 2,
  "RECORDS": [
    {
      "QUEUE_NAME": "batch",
      "STATUS": "Open:Active"
    },
    {
      "QUEUE_NAME": "test",
      "STATUS": "Open:Active"
    }
  ]
}
```

class buildtest.system.PBS

Bases: [Scheduler](#)

The PBS class checks for Cobalt binaries and gets a list of Cobalt queues

binaries = ['qsub', 'qstat', 'qdel', 'qstart', 'qhold', 'qmgr']

_get_queues(*self*)

Get queue configuration using qstat -Q -f -F json and retrieve a list of queues.

class buildtest.system.Scheduler

This is a base Scheduler class used for implementing common methods for detecting Scheduler details. The subclass implement specific queries that are scheduler specific. The Slurm, LSF, PBS and Cobalt class inherit from Base Class Scheduler.

logger

check(*self*)

Check if binaries exist binary exist in \$PATH

class buildtest.system.Slurm

Bases: [Scheduler](#)

The Slurm class implements common functions to query Slurm cluster including partitions, qos, cluster. We check existence of slurm binaries in \$PATH and return if slurm cluster is in valid state.

binaries = ['sbatch', 'sacct', 'sacctmgr', 'sinfo', 'scancel']

_get_clusters(*self*)

Get list of slurm clusters by running `sacctmgr list cluster -P -n format=Cluster`. The output is a list of slurm clusters something as follows

```
$ sacctmgr list cluster -P -n format=Cluster
cori
escori
```

_get_partitions(*self*)

Get list of all partitions slurm partitions using `sinfo -a -h -O partitionname`. The output is a list of queue names

```
$ sinfo -a -h -O partitionname
system
system_shared
debug_hsw
debug_knl
jupyter
```

_get_qos(*self*)

Retrieve a list of all slurm qos by running `sacctmgr list qos -P -n format=Name`. The output is a list of qos. Shown below is an example output

```
$ sacctmgr list qos -P -n format=Name
normal
premium
low
serialize
scavenger
```

buildtest.system.system

Package Contents

buildtest.BUILDTEST_COPYRIGHT = Copyright (c) 2021, The Regents of the University of California, through Lawrence Berkeley...

buildtest.BUILDTEST_VERSION = 0.10.0

buildtest.__version__

3.12 Buildtest Command Reference

buildtest is a HPC testing framework for building and running tests.

usage: buildtest [options] [COMMANDS]

3.12.1 Named Arguments

-V, --version	show program's version number and exit
-c, --config	Specify Path to Configuration File
-d, --debug	Print debug messages to screen
	Default: False

3.12.2 COMMANDS

Possible choices: build, history, buildspec, config, report, inspect, schema, cdash, docs, schemadocs, help

3.12.3 Sub-commands:

build

Build and Run test

buildtest build [-h] [-b BUILDSPEC] [-x EXCLUDE] [-t TAGS] [-ft FILTER_TAGS]
[-e EXECUTOR] [-s {parse,build}] [--testdir TESTDIR]
[--rebuild REBUILD] [-r REPORT]
[--max-pend-time MAX_PEND_TIME]
[--poll-interval POLL_INTERVAL] [-k]

Named Arguments

-b, --buildspec	Specify a buildspec (file or directory) to build. A buildspec must end in '.yaml' extension.
-x, --exclude	Exclude one or more buildspecs (file or directory) from processing. A buildspec must end in '.yaml' extension.
-t, --tags	Discover buildspecs by tags found in buildspec cache
-ft, --filter-tags	Filter buildspecs by tags when building tests.
-e, --executor	Discover buildspecs by executor name found in buildspec cache
-s, --stage	Possible choices: parse, build control behavior of buildtest build
--testdir	Specify a custom test directory where to write tests. This overrides configuration file and default location.
--rebuild	Rebuild test X number of times. Must be a positive number between [1-50]
-r, --report	Specify a report file where tests will be written.
--max-pend-time	Specify Maximum Pending Time (sec) for job before cancelling job. This only applies for batch job submission.
--poll-interval	Specify Poll Interval (sec) for polling batch jobs
-k, --keep-stage-dir	Keep stage directory after job completion. Default: False

history

Query build history

```
buildtest history [-h] ...
```

subcommands

Query build history file

Possible choices: list, query

Sub-commands:

list

List a summary of all builds

```
buildtest history list [-h]
```

query

Query information for a particular build

```
buildtest history query [-h] [-l] id
```

Positional Arguments

id	Select a build ID
-----------	-------------------

Named Arguments

-l, --log	Display logfile for corresponding build id
	Default: False

buildspec

Options for querying buildspec cache

```
buildtest buildspec [-h] ...
```

subcommands

Find buildspec from cache file

Possible choices: find, validate

Sub-commands:

find

find all buildspecs

```
buildtest buildspec find [-h] [--root ROOT] [-r] [-t] [-b] [-e] [-p]
                        [--group-by-tags] [--group-by-executor] [-m] [-mb]
                        [--filter FILTER] [--format FORMAT] [--helpfilter]
                        [--helpformat] [--terse]
```

Named Arguments

--root	Specify root buildsspecs (directory) path to load buildsspecs into buildspec cache.
-r, --rebuild	Rebuild buildspec cache and find all buildsspecs again Default: False
-t, --tags	List all available tags Default: False
-b, --buildspec	Get all buildspec files from cache Default: False
-e, --executors	get all unique executors from buildsspecs Default: False
-p, --paths	print all root buildspec paths Default: False
--group-by-tags	Group tests by tag name Default: False
--group-by-executor	Group tests by executor name Default: False
-m, --maintainers	Get all maintainers for all buildsspecs Default: False
-mb, --maintainers-by-buildspecs	Show maintainers breakdown by buildsspecs Default: False
--filter	Filter buildspec cache with filter fields in format <code>--filter key1=val1,key2=val2</code>
--format	Format buildspec cache with format fields in format <code>--format field1,field2,...</code>
--helpfilter	Show Filter fields for <code>--filter</code> option for filtering buildspec cache output Default: False
--helpformat	Show Format fields for <code>--format</code> option for formatting buildspec cache output Default: False
--terse	Print output in machine readable format Default: False

validate

Validate buildsspecs

```
buildtest buildspec validate [-h] [-b BUILDSPEC] [-x EXCLUDE] [-t TAG]
                             [-e EXECUTOR]
```

Named Arguments

-b, --buildspec	Specify path to buildspec (file, or directory) to validate
-x, --exclude	Specify path to buildspec to exclude (file or directory) during validation
-t, --tag	Specify buildspecs by tag name to validate
-e, --executor	Specify buildspecs by executor name to validate

config

Query buildtest configuration

```
buildtest config [-h] ...
```

subcommands

Query information from buildtest configuration file

Possible choices: executors, view, validate, summary, systems, compilers

Sub-commands:

executors

Query executors from buildtest configuration

```
buildtest config executors [-h] [-j] [-y]
```

Named Arguments

-j, --json	View executor in JSON format Default: False
-y, --yaml	View executors in YAML format Default: False

view

View Buildtest Configuration File

```
buildtest config view [-h]
```


validate

Validate buildtest settings file with schema.

```
buildtest config validate [-h]
```

summary

Provide summary of buildtest settings.

```
buildtest config summary [-h]
```

systems

List all available systems

```
buildtest config systems [-h]
```

compilers

search or find compilers

```
buildtest config compilers [-h] [-j] [-y] ...
```

Named Arguments

-j, --json	List compiler details in JSON format Default: False
-y, --yaml	List compiler details in YAML format Default: False

subcommands

Find new compilers and add them to detected compiler section

Possible choices: find

Sub-commands:

find

Find compilers

```
buildtest config compilers find [-h] [-d]
```

Named Arguments

-d, --debug Display Debugging output when finding compilers
Default: False

report

Query test report

```
buildtest report [-h] [--helpformat] [--helpfilter] [--format FORMAT]
                  [--filter FILTER] [--latest] [--oldest] [-r REPORT]
                  . . .
```

Named Arguments

--helpformat List of available format fields
Default: False

--helpfilter List available filter fields to be used with `-filter` option
Default: False

--format format field for printing purposes. For more details see `-helpformat` for list of available fields. Fields must be separated by comma (usage: `-format <field1>,<field2>,...`)

--filter Filter report by filter fields. The filter fields must be a key=value pair and multiple fields can be comma separated in the following format: `-filter key1=val1,key2=val2` . For list of filter fields run: `-helpfilter`.

--latest Retrieve latest record of particular test
Default: False

--oldest Retrieve oldest record of particular test
Default: False

-r, --report Specify a report file to read
Default: `"/home/docs/checkouts/readthedocs.org/user_builds/buildtest/checkouts/v0.10.0/var/report.json"`

subcommands

Fetch test results from report file and print them in table format

Possible choices: clear, list

Sub-commands:

clear

delete report file

```
buildtest report clear [-h]
```

list

List all report files

```
buildtest report list [-h]
```

inspect

Inspect a test based on NAME or ID

```
buildtest inspect [-h] [-r REPORT] ...
```

Named Arguments

-r, --report Specify a report file to load when inspecting test

subcommands

Inspect Test result based on Test ID or Test Name

Possible choices: name, id, list

Sub-commands:

name

Specify name of test

```
buildtest inspect name [-h] [name [name ...]]
```

Positional Arguments

name Name of test

id

Specify a Test ID

```
buildtest inspect id [-h] [id [id ...]]
```

Positional Arguments

id Test ID

list

List all test ids

```
buildtest inspect list [-h] [-p]
```

Named Arguments

-p, --parse Print output in parseable format
Default: False

schema

List schema contents and examples

```
buildtest schema [-h] [-n Schema Name] [-e] [-j]
```

Named Arguments

-n, --name Possible choices: global.schema.json, definitions.schema.json, settings.schema.json, compiler-v1.0.schema.json, spack-v1.0.schema.json, script-v1.0.schema.json
show schema by name (e.g., script)

-e, --example Show schema examples
Default: False

-j, --json Display json schema file
Default: False

cdash

Upload test to CDASH server

```
buildtest cdash [-h] ...
```

subcommands

buildtest CDASH integration

Possible choices: view, upload

Sub-commands:

view

Open CDASH project in webbrowser

```
buildtest cdash view [-h] [--url URL]
```

Named Arguments

--url Specify a url to CDASH project

upload

Upload Test to CDASH server

```
buildtest cdash upload [-h] [--site SITE] [-r REPORT] buildname
```

Positional Arguments

buildname Specify Build Name reported in CDASH

Named Arguments

--site Specify site name reported in CDASH

-r, --report Path to report file to upload test results

docs

Open buildtest docs in browser

```
buildtest docs [-h]
```

schemadocs

Open buildtest schema docs in browser

```
buildtest schemadocs [-h]
```

help

buildtest command guide

```
buildtest help [-h]
```

GitHub: <https://github.com/buildtesters/buildtest> Documentation: <https://buildtest.readthedocs.io/en/latest/index.html> Schema Documentation: <https://buildtesters.github.io/buildtest/> Slack: <http://hpcbuildtest.slack.com/>

Please report issues at <https://github.com/buildtesters/buildtest/issues>

Copyright (c) 2021, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from the U.S. Dept. of Energy), Shahzeb Siddiqui, and Vanessa Sochat. All rights reserved.

LICENSE

buildtest is released under the [MIT license](#)

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

b

- `buildtest`, 340
- `buildtest.buildsystem`, 340
 - `buildtest.buildsystem.base`, 340
 - `buildtest.buildsystem.batch`, 343
 - `buildtest.buildsystem.builders`, 344
 - `buildtest.buildsystem.compilerbuilder`, 345
 - `buildtest.buildsystem.parser`, 347
 - `buildtest.buildsystem.scriptbuilder`, 348
 - `buildtest.buildsystem.spack`, 348
- `buildtest.cli`, 349
 - `buildtest.cli.build`, 349
 - `buildtest.cli.buildspec`, 352
 - `buildtest.cli.cdash`, 355
 - `buildtest.cli.compilers`, 356
 - `buildtest.cli.config`, 357
 - `buildtest.cli.help`, 358
 - `buildtest.cli.history`, 358
 - `buildtest.cli.inspect`, 359
 - `buildtest.cli.report`, 360
 - `buildtest.cli.schema`, 361
- `buildtest.config`, 385
- `buildtest.defaults`, 387
- `buildtest.exceptions`, 387
- `buildtest.executors`, 363
 - `buildtest.executors.base`, 363
 - `buildtest.executors.cobalt`, 364
 - `buildtest.executors.job`, 367
 - `buildtest.executors.local`, 367
 - `buildtest.executors.lsf`, 368
 - `buildtest.executors.pbs`, 371
 - `buildtest.executors.setup`, 373
 - `buildtest.executors.slurm`, 375
- `buildtest.log`, 388
- `buildtest.main`, 389
- `buildtest.schemas`, 378
 - `buildtest.schemas.defaults`, 378
 - `buildtest.schemas.utils`, 379
- `buildtest.system`, 389
- `buildtest.utils`, 379
 - `buildtest.utils.command`, 379
 - `buildtest.utils.file`, 381
 - `buildtest.utils.shell`, 383
 - `buildtest.utils.timer`, 384
 - `buildtest.utils.tools`, 384

Symbols

`__enter__()` (*buildtest.utils.command.Capturing method*), 380
`__exit__()` (*buildtest.utils.command.Capturing method*), 380
`__missing__()` (*buildtest.utils.tools.Hasher method*), 385
`__repr__()` (*buildtest.buildsystem.base.BuilderBase method*), 340
`__repr__()` (*buildtest.buildsystem.parser.BuildspecParser method*), 347
`__repr__()` (*buildtest.executors.base.BaseExecutor method*), 363
`__repr__()` (*buildtest.executors.setup.BuildExecutor method*), 373
`__repr__()` (*buildtest.utils.shell.Shell method*), 383
`__str__()` (*buildtest.buildsystem.base.BuilderBase method*), 340
`__str__()` (*buildtest.buildsystem.parser.BuildspecParser method*), 347
`__str__()` (*buildtest.exceptions.BuildTestError method*), 387
`__str__()` (*buildtest.exceptions.BuildspecError method*), 387
`__str__()` (*buildtest.exceptions.ConfigurationError method*), 388
`__str__()` (*buildtest.executors.base.BaseExecutor method*), 363
`__str__()` (*buildtest.executors.setup.BuildExecutor method*), 373
`__str__()` (*buildtest.utils.shell.Shell method*), 383
`__str__()` (*buildtest.utils.tools.Hasher method*), 385
`__version__` (in module *buildtest*), 392
`_build_compilers()` (*buildtest.buildsystem.builders.Builder method*), 344
`_build_setup()` (*buildtest.buildsystem.base.BuilderBase method*), 340
`_check_executor()` (*buildtest.buildsystem.parser.BuildspecParser method*), 347
`_check_filter_fields()` (*buildtest.cli.report.Report method*), 360
`_check_format_fields()` (*buildtest.cli.report.Report method*), 360
`_check_regex()` (*buildtest.executors.base.BaseExecutor method*), 363
`_check_runtime()` (*buildtest.executors.base.BaseExecutor method*), 364
`_check_schema_type()` (*buildtest.buildsystem.parser.BuildspecParser method*), 347
`_choose_executor()` (*buildtest.executors.setup.BuildExecutor method*), 373
`_compile_cmd()` (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 345
`_default_test_variables()` (*buildtest.buildsystem.base.BuilderBase method*), 340
`_detect_lang()` (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 345
`_discover_buildspecs()` (*buildtest.cli.buildspec.BuildspecCache method*), 353
`_emit_command()` (*buildtest.buildsystem.base.BuilderBase method*), 340
`_executor_check()` (*buildtest.config.SiteConfiguration method*), 385
`_filter_buildspecs()` (*buildtest.cli.buildspec.BuildspecCache method*), 353
`_filter_by_executor()` (*buildtest.cli.report.Report method*), 360
`_filter_by_names()` (*buildtest.cli.report.Report method*), 360
`_filter_by_returncode()` (*buildtest.cli.report.Report method*), 360
`_filter_by_state()` (*buildtest.cli.report.Report method*), 360
`_filter_by_tags()` (*buildtest.cli.report.Report method*), 360
`_generate_builders()` (*buildtest.buildsystem.builders.Builder method*), 344
`_generate_unique_id()` (*buildtest.buildsystem.base.BuilderBase method*), 360

- `method`), 340
 - `_get_burst_buffer()` (*buildtest.buildsystem.base.BuilderBase method*), 341
 - `_get_clusters()` (*buildtest.system.Slurm method*), 391
 - `_get_data_warp()` (*buildtest.buildsystem.base.BuilderBase method*), 341
 - `_get_environment()` (*buildtest.buildsystem.base.BuilderBase method*), 341
 - `_get_modules()` (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 345
 - `_get_partitions()` (*buildtest.system.Slurm method*), 391
 - `_get_qos()` (*buildtest.system.Slurm method*), 391
 - `_get_queues()` (*buildtest.system.Cobalt method*), 390
 - `_get_queues()` (*buildtest.system.LSF method*), 390
 - `_get_queues()` (*buildtest.system.PBS method*), 390
 - `_get_scheduler_directives()` (*buildtest.buildsystem.base.BuilderBase method*), 341
 - `_get_variables()` (*buildtest.buildsystem.base.BuilderBase method*), 341
 - `_print_build_phase()` (*buildtest.cli.build.BuildTest method*), 350
 - `_print_jobs_after_poll()` (*buildtest.cli.build.BuildTest method*), 350
 - `_print_test_summary()` (*buildtest.cli.build.BuildTest method*), 350
 - `_process_compiler_config()` (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 346
 - `_resolve_source()` (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 346
 - `_resolve_spack_root()` (*buildtest.buildsystem.spack.SpackBuilder method*), 348
 - `_returncode_check()` (*buildtest.executors.base.BaseExecutor method*), 364
 - `_run_cmd()` (*buildtest.buildsystem.compilerbuilder.CompilerBuilder attribute*), 343
 - `_set_execute_perm()` (*buildtest.buildsystem.base.BuilderBase method*), 341
 - `_set_metadata_values()` (*buildtest.buildsystem.base.BuilderBase method*), 341
 - `_skip_tests_by_tags()` (*buildtest.buildsystem.builders.Builder method*), 344
 - `_skip_tests_run_only()` (*buildtest.buildsystem.builders.Builder method*), 344
 - `_spack_environment()` (*buildtest.buildsystem.spack.SpackBuilder method*), 349
 - `_update_build_history()` (*buildtest.cli.build.BuildTest method*), 350
 - `_update_compiler_section()` (*buildtest.cli.compilers.BuildtestCompilers method*), 356
 - `_validate()` (*buildtest.buildsystem.parser.BuildspecParser method*), 347
 - `_validate_buildspecs()` (*buildtest.cli.buildspec.BuildspecCache method*), 353
 - `_validate_cobalt_executors()` (*buildtest.config.SiteConfiguration method*), 385
 - `_validate_lsf_executors()` (*buildtest.config.SiteConfiguration method*), 385
 - `_validate_modules()` (*buildtest.cli.compilers.BuildtestCompilers method*), 356
 - `_validate_pbs_executors()` (*buildtest.config.SiteConfiguration method*), 385
 - `_validate_slurm_executors()` (*buildtest.config.SiteConfiguration method*), 386
 - `_write_build_script()` (*buildtest.buildsystem.base.BuilderBase method*), 341
 - `_write_buildspec_cache()` (*buildtest.cli.buildspec.BuildspecCache method*), 353
 - `_write_test()` (*buildtest.buildsystem.base.BuilderBase method*), 341
- ## B
- `BaseExecutor` (class in *buildtest.executors.base*), 363
 - `batch_translation` (*buildtest.buildsystem.batch.CobaltBatchScript attribute*), 343
 - `batch_translation` (*buildtest.buildsystem.batch.LSFBatchScript attribute*), 343
 - `batch_translation` (*buildtest.buildsystem.batch.PBSBatchScript attribute*), 343
 - `batch_translation` (*buildtest.buildsystem.batch.SlurmBatchScript attribute*), 343
 - `BatchScript` (class in *buildtest.buildsystem.batch*), 343
 - `binaries` (*buildtest.system.Cobalt attribute*), 390
 - `binaries` (*buildtest.system.LSF attribute*), 390
 - `binaries` (*buildtest.system.PBS attribute*), 390
 - `binaries` (*buildtest.system.Slurm attribute*), 391
 - `build()` (*buildtest.buildsystem.base.BuilderBase method*), 342
 - `build()` (*buildtest.cli.build.BuildTest method*), 350

build() (*buildtest.cli.buildspec.BuildspecCache* method), 353
 build_cache() (*buildtest.cli.buildspec.BuildspecCache* method), 353
 build_header() (*buildtest.buildsystem.batch.CobaltBatchScheduler* method), 343
 build_header() (*buildtest.buildsystem.batch.LSFBatchScheduler* method), 343
 build_header() (*buildtest.buildsystem.batch.PBSBatchScheduler* method), 343
 build_header() (*buildtest.buildsystem.batch.SlurmBatchScheduler* method), 343
 build_history() (*in module buildtest.cli.history*), 359
 BUILD_HISTORY_DIR (*in module buildtest.defaults*), 387
 build_menu() (*in module buildtest.cli*), 362
 build_phase() (*buildtest.cli.build.BuildTest* method), 350
 BUILD_REPORT (*in module buildtest.cli*), 362
 BUILD_REPORT (*in module buildtest.defaults*), 387
 Builder (*class in buildtest.buildsystem.builders*), 344
 BuilderBase (*class in buildtest.buildsystem.base*), 340
 BuildExecutor (*class in buildtest.executors.setup*), 373
 BUILDSPEC_CACHE_FILE (*in module buildtest.defaults*), 387
 BUILDSPEC_DEFAULT_PATH (*in module buildtest.defaults*), 387
 BUILDSPEC_ERROR_FILE (*in module buildtest.defaults*), 387
 buildspec_find() (*in module buildtest.cli.buildspec*), 355
 buildspec_menu() (*in module buildtest.cli*), 362
 buildspec_validate() (*in module buildtest.cli.buildspec*), 355
 BuildspecCache (*class in buildtest.cli.buildspec*), 353
 BuildspecError, 387
 BuildspecParser (*class in buildtest.buildsystem.parser*), 347
 buildtest module, 340
 BuildTest (*class in buildtest.cli.build*), 350
 buildtest.buildsystem module, 340
 buildtest.buildsystem.base module, 340
 buildtest.buildsystem.batch module, 343
 buildtest.buildsystem.builders module, 344
 buildtest.buildsystem.compilerbuilder module, 345
 buildtest.buildsystem.parser module, 347
 buildtest.buildsystem.scriptbuilder module, 348
 buildtest.buildsystem.spack module, 348
 buildtest.cli module, 349
 buildtest.cli.build module, 349
 buildtest.cli.buildspec module, 352
 buildtest.cli.cdash module, 355
 buildtest.cli.compilers module, 356
 buildtest.cli.config module, 357
 buildtest.cli.help module, 358
 buildtest.cli.history module, 358
 buildtest.cli.inspect module, 359
 buildtest.cli.report module, 360
 buildtest.cli.schema module, 361
 buildtest.config module, 385
 buildtest.defaults module, 387
 buildtest.exceptions module, 387
 buildtest.executors module, 363
 buildtest.executors.base module, 363
 buildtest.executors.cobalt module, 364
 buildtest.executors.job module, 367
 buildtest.executors.local module, 367
 buildtest.executors.lsf module, 368
 buildtest.executors.pbs module, 371
 buildtest.executors.setup module, 373
 buildtest.executors.slurm module, 375
 buildtest.log module, 388
 buildtest.main module, 389
 buildtest.schemas module, 378

buildtest.schemas.defaults
 module, 378
 buildtest.schemas.utils
 module, 379
 buildtest.system
 module, 389
 buildtest.utils
 module, 379
 buildtest.utils.command
 module, 379
 buildtest.utils.file
 module, 381
 buildtest.utils.shell
 module, 383
 buildtest.utils.timer
 module, 384
 buildtest.utils.tools
 module, 384
 BUILDTEST_BUILDSPEC_DIR (in module buildtest.defaults), 387
 BUILDTEST_COPYRIGHT (in module buildtest), 392
 BUILDTEST_COPYRIGHT (in module buildtest.cli), 362
 BUILDTEST_DEFAULT_TESTDIR (in module buildtest.defaults), 387
 BUILDTEST_EXECUTOR_DIR (in module buildtest.defaults), 387
 buildtest_help() (in module buildtest.cli.help), 358
 BUILDTEST_REPORT_SUMMARY (in module buildtest.defaults), 387
 BUILDTEST_ROOT (in module buildtest.defaults), 387
 BUILDTEST_USER_HOME (in module buildtest.defaults), 387
 BUILDTEST_VERSION (in module buildtest), 392
 BUILDTEST_VERSION (in module buildtest.cli), 362
 BuildTestCommand (class in buildtest.utils.command), 379
 BuildtestCompilers (class in buildtest.cli.compilers), 356
 BuildTestError, 387
 BuildTestSystem (class in buildtest.system), 389

C

cancel() (buildtest.executors.cobalt.CobaltJob method), 365
 cancel() (buildtest.executors.job.Job method), 367
 cancel() (buildtest.executors.lsf.LSFJob method), 369
 cancel() (buildtest.executors.pbs.PBSJob method), 372
 cancel() (buildtest.executors.slurm.SlurmJob method), 376
 Capturing (class in buildtest.utils.command), 380
 cc (buildtest.buildsystem.compilerbuilder.CompilerBuilder attribute), 345
 cdash_cmd() (in module buildtest.cli.cdash), 355
 cdash_menu() (in module buildtest.cli), 362
 cflags (buildtest.buildsystem.compilerbuilder.CompilerBuilder attribute), 345
 check() (buildtest.executors.local.LocalExecutor method), 367
 check() (buildtest.system.BuildTestSystem method), 389
 check() (buildtest.system.Scheduler method), 391
 check_filter_fields() (buildtest.cli.buildspec.BuildspecCache method), 353
 check_format_fields() (buildtest.cli.buildspec.BuildspecCache method), 353
 check_scheduler() (buildtest.system.BuildTestSystem method), 389
 check_test_state() (buildtest.executors.base.BaseExecutor method), 364
 cleanup() (buildtest.utils.command.Capturing method), 380
 Cobalt (class in buildtest.system), 390
 cobalt_log() (buildtest.executors.cobalt.CobaltJob method), 365
 CobaltBatchScript (class in buildtest.buildsystem.batch), 343
 CobaltExecutor (class in buildtest.executors.cobalt), 365
 CobaltJob (class in buildtest.executors.cobalt), 365
 compiler_cmd() (in module buildtest.cli.compilers), 357
 compiler_find() (in module buildtest.cli.compilers), 357
 compiler_table (buildtest.cli.compilers.BuildtestCompilers attribute), 356
 CompilerBuilder (class in buildtest.buildsystem.compilerbuilder), 345
 complete() (buildtest.buildsystem.base.BuilderBase method), 342
 complete() (buildtest.executors.slurm.SlurmJob method), 376
 config_cmd() (in module buildtest.cli.config), 357
 config_menu() (in module buildtest.cli), 362
 ConfigurationError, 388
 copy_stage_files() (buildtest.buildsystem.base.BuilderBase method), 342
 cppflags (buildtest.buildsystem.compilerbuilder.CompilerBuilder attribute), 345
 create_dir() (in module buildtest.utils.file), 381
 custom_validator() (in module buildtest.schemas.defaults), 378
 cxx (buildtest.buildsystem.compilerbuilder.CompilerBuilder attribute), 345
 cxxflags (buildtest.buildsystem.compilerbuilder.CompilerBuilder attribute), 345

D

`decode()` (*buildtest.utils.command.BuildTestCommand method*), 379

`deep_get()` (*in module buildtest.utils.tools*), 385

`default_format_fields` (*buildtest.cli.buildspec.BuildspecCache attribute*), 353

`DEFAULT_SETTINGS_FILE` (*in module buildtest.defaults*), 387

`DEFAULT_SETTINGS_SCHEMA` (*in module buildtest.defaults*), 387

`detect_module_tool()` (*buildtest.system.BuildTestSystem method*), 390

`detect_system()` (*buildtest.config.SiteConfiguration method*), 386

`discover_buildspecs()` (*in module buildtest.cli.build*), 351

`discover_buildspecs_by_executor()` (*in module buildtest.cli.build*), 351

`discover_buildspecs_by_tags()` (*in module buildtest.cli.build*), 351

`discover_by_buildspecs()` (*in module buildtest.cli.build*), 352

`dispatch()` (*buildtest.executors.cobalt.CobaltExecutor method*), 365

`dispatch()` (*buildtest.executors.lsf.LSFExecutor method*), 368

`dispatch()` (*buildtest.executors.pbs.PBSExecutor method*), 372

`dispatch()` (*buildtest.executors.slurm.SlurmExecutor method*), 375

`display_table` (*buildtest.cli.report.Report attribute*), 360

E

`endtime()` (*buildtest.buildsystem.base.BuilderBase method*), 342

`err()` (*buildtest.utils.command.Capturing property*), 380

`error_file()` (*buildtest.executors.cobalt.CobaltJob method*), 366

`error_file()` (*buildtest.executors.lsf.LSFJob method*), 369

`error_file()` (*buildtest.executors.pbs.PBSJob method*), 372

`execute()` (*buildtest.utils.command.BuildTestCommand method*), 380

`ExecutorError`, 388

`exitcode()` (*buildtest.executors.cobalt.CobaltJob method*), 366

`exitcode()` (*buildtest.executors.lsf.LSFJob method*), 369

`exitcode()` (*buildtest.executors.pbs.PBSJob method*), 372

`exitcode()` (*buildtest.executors.slurm.SlurmJob method*), 376

F

`fail()` (*buildtest.executors.pbs.PBSJob method*), 372

`fc` (*buildtest.buildsystem.compilerbuilder.CompilerBuilder attribute*), 345

`fflags` (*buildtest.buildsystem.compilerbuilder.CompilerBuilder attribute*), 345

`file()` (*buildtest.config.SiteConfiguration property*), 386

`FILE_LOG` (*in module buildtest.log*), 388

`filter_buildspecs_from_report()` (*buildtest.cli.report.Report method*), 361

`filter_fields` (*buildtest.cli.buildspec.BuildspecCache attribute*), 353

`filter_fields` (*buildtest.cli.report.Report attribute*), 360

`find_buildspecs()` (*buildtest.cli.buildspec.BuildspecCache method*), 353

`find_compilers()` (*buildtest.cli.compilers.BuildtestCompilers method*), 356

`format_fields` (*buildtest.cli.buildspec.BuildspecCache attribute*), 353

`format_fields` (*buildtest.cli.report.Report attribute*), 360

G

`gather()` (*buildtest.executors.cobalt.CobaltExecutor method*), 365

`gather()` (*buildtest.executors.cobalt.CobaltJob method*), 366

`gather()` (*buildtest.executors.lsf.LSFExecutor method*), 369

`gather()` (*buildtest.executors.lsf.LSFJob method*), 369

`gather()` (*buildtest.executors.pbs.PBSExecutor method*), 372

`gather()` (*buildtest.executors.pbs.PBSJob method*), 372

`gather()` (*buildtest.executors.slurm.SlurmExecutor method*), 375

`gather()` (*buildtest.executors.slurm.SlurmJob method*), 376

`generate_script()` (*buildtest.buildsystem.base.BuilderBase method*), 342

`generate_script()` (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 346

`generate_script()` (*buildtest.buildsystem.scriptbuilder.ScriptBuilder method*), 348

`generate_script()` (*buildtest.buildsystem.spack.SpackBuilder method*), 349

`get()` (*buildtest.executors.job.Job method*), 367

`get()` (*buildtest.executors.setup.BuildExecutor method*), 373

`get()` (*buildtest.system.BuildTestSystem method*), 390

`get()` (*buildtest.utils.shell.Shell method*), 383

[get\(\)](#) (*buildtest.utils.tools.Hasher method*), 385
[get_all_ids\(\)](#) (*in module buildtest.cli.inspect*), 359
[get_builders\(\)](#) (*buildtest.buildsystem.builders.Builder method*), 345
[get_buildspecfiles\(\)](#) (*buildtest.cli.buildspec.BuildspecCache method*), 354
[get_cache\(\)](#) (*buildtest.cli.buildspec.BuildspecCache method*), 354
[get_cc\(\)](#) (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 346
[get_cflags\(\)](#) (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 346
[get_command\(\)](#) (*buildtest.utils.command.BuildTestCommand method*), 380
[get_cppfilags\(\)](#) (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 346
[get_cxx\(\)](#) (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 346
[get_cxxflags\(\)](#) (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 346
[get_error\(\)](#) (*buildtest.utils.command.BuildTestCommand method*), 380
[get_executors\(\)](#) (*buildtest.cli.buildspec.BuildspecCache method*), 354
[get_fc\(\)](#) (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 346
[get_fflags\(\)](#) (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 346
[get_headers\(\)](#) (*buildtest.buildsystem.batch.BatchScript method*), 343
[get_ldflags\(\)](#) (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 346
[get_output\(\)](#) (*buildtest.utils.command.BuildTestCommand method*), 380
[get_parser\(\)](#) (*in module buildtest.cli*), 363
[get_path\(\)](#) (*buildtest.buildsystem.compilerbuilder.CompilerBuilder method*), 346
[get_report_file\(\)](#) (*buildtest.cli.report.Report method*), 361
[get_tags\(\)](#) (*buildtest.cli.buildspec.BuildspecCache method*), 354
[get_test_extension\(\)](#) (*buildtest.buildsystem.base.BuilderBase method*), 342
[get_test_names\(\)](#) (*buildtest.buildsystem.builders.Builder method*), 345

H

[handle_kv_string\(\)](#) (*in module buildtest.cli*), 363
[Hasher](#) (*class in buildtest.utils.tools*), 384
[here](#) (*in module buildtest.schemas.defaults*), 378
[here](#) (*in module buildtest.schemas.utils*), 379
[history_menu\(\)](#) (*in module buildtest.cli*), 363

[incomplete\(\)](#) (*buildtest.buildsystem.base.BuilderBase method*), 342
[init_logfile\(\)](#) (*in module buildtest.log*), 388
[inspect_by_id\(\)](#) (*in module buildtest.cli.inspect*), 359
[inspect_by_name\(\)](#) (*in module buildtest.cli.inspect*), 359
[inspect_cmd\(\)](#) (*in module buildtest.cli.inspect*), 359
[inspect_list\(\)](#) (*in module buildtest.cli.inspect*), 359
[inspect_menu\(\)](#) (*in module buildtest.cli*), 363
[is_cancelled\(\)](#) (*buildtest.executors.cobalt.CobaltJob method*), 366
[is_cancelled\(\)](#) (*buildtest.executors.slurm.SlurmJob method*), 377
[is_cobalt\(\)](#) (*buildtest.executors.setup.BuildExecutor method*), 374
[is_complete\(\)](#) (*buildtest.executors.cobalt.CobaltJob method*), 366
[is_complete\(\)](#) (*buildtest.executors.lsf.LSFJob method*), 370
[is_complete\(\)](#) (*buildtest.executors.pbs.PBSJob method*), 372
[is_complete\(\)](#) (*buildtest.executors.slurm.SlurmJob method*), 377
[is_dir\(\)](#) (*in module buildtest.utils.file*), 381
[is_failed\(\)](#) (*buildtest.executors.lsf.LSFJob method*), 370
[is_failed\(\)](#) (*buildtest.executors.slurm.SlurmJob method*), 377
[is_file\(\)](#) (*in module buildtest.utils.file*), 381
[is_int\(\)](#) (*in module buildtest.cli.report*), 361
[is_local\(\)](#) (*buildtest.executors.setup.BuildExecutor method*), 374
[is_lsf\(\)](#) (*buildtest.executors.setup.BuildExecutor method*), 374
[is_out_of_memory\(\)](#) (*buildtest.executors.slurm.SlurmJob method*), 377
[is_pbs\(\)](#) (*buildtest.executors.setup.BuildExecutor method*), 374
[is_pending\(\)](#) (*buildtest.executors.cobalt.CobaltJob method*), 366
[is_pending\(\)](#) (*buildtest.executors.job.Job method*), 367
[is_pending\(\)](#) (*buildtest.executors.lsf.LSFJob method*), 370
[is_pending\(\)](#) (*buildtest.executors.pbs.PBSJob method*), 372
[is_pending\(\)](#) (*buildtest.executors.slurm.SlurmJob method*), 377
[is_running\(\)](#) (*buildtest.executors.cobalt.CobaltJob method*), 366
[is_running\(\)](#) (*buildtest.executors.job.Job method*), 367
[is_running\(\)](#) (*buildtest.executors.lsf.LSFJob method*), 371

- `is_running()` (*buildtest.executors.pbs.PBSJob method*), 372
- `is_running()` (*buildtest.executors.slurm.SlurmJob method*), 377
- `is_slurm()` (*buildtest.executors.setup.BuildExecutor method*), 374
- `is_suspended()` (*buildtest.executors.cobalt.CobaltJob method*), 366
- `is_suspended()` (*buildtest.executors.job.Job method*), 367
- `is_suspended()` (*buildtest.executors.lsf.LSFJob method*), 371
- `is_suspended()` (*buildtest.executors.pbs.PBSJob method*), 372
- `is_suspended()` (*buildtest.executors.slurm.SlurmJob method*), 377
- `is_timeout()` (*buildtest.executors.slurm.SlurmJob method*), 377
- J**
- `Job` (*class in buildtest.executors.job*), 367
- L**
- `lang_ext_table` (*buildtest.buildsystem.compilerbuilder.CompilerBuilder attribute*), 345
- `launcher_command()` (*buildtest.executors.cobalt.CobaltExecutor method*), 365
- `launcher_command()` (*buildtest.executors.lsf.LSFExecutor method*), 369
- `launcher_command()` (*buildtest.executors.pbs.PBSExecutor method*), 372
- `launcher_command()` (*buildtest.executors.slurm.SlurmExecutor method*), 375
- `ldflags` (*buildtest.buildsystem.compilerbuilder.CompilerBuilder attribute*), 345
- `list()` (*buildtest.cli.compilers.BuildtestCompilers method*), 357
- `list_builds()` (*in module buildtest.cli.history*), 359
- `list_executors()` (*buildtest.executors.setup.BuildExecutor method*), 374
- `load()` (*buildtest.cli.report.Report method*), 361
- `load()` (*buildtest.config.SiteConfiguration method*), 386
- `load()` (*buildtest.executors.base.BaseExecutor method*), 364
- `load()` (*buildtest.executors.cobalt.CobaltExecutor method*), 365
- `load()` (*buildtest.executors.local.LocalExecutor method*), 368
- `load()` (*buildtest.executors.lsf.LSFExecutor method*), 369
- `load()` (*buildtest.executors.pbs.PBSExecutor method*), 372
- `load()` (*buildtest.executors.slurm.SlurmExecutor method*), 375
- `load_json()` (*in module buildtest.utils.file*), 382
- `load_paths()` (*buildtest.cli.buildspec.BuildspecCache method*), 354
- `load_recipe()` (*in module buildtest.schemas.utils*), 379
- `load_schema()` (*in module buildtest.schemas.utils*), 379
- `LocalExecutor` (*class in buildtest.executors.local*), 367
- `LOG_FORMATTER` (*in module buildtest.log*), 388
- `LOG_NAME` (*in module buildtest.log*), 388
- `logger` (*buildtest.system.Scheduler attribute*), 391
- `logger` (*in module buildtest.cli.build*), 352
- `logger` (*in module buildtest.cli.buildspec*), 355
- `logger` (*in module buildtest.cli.history*), 359
- `logger` (*in module buildtest.cli.report*), 361
- `logger` (*in module buildtest.config*), 386
- `logger` (*in module buildtest.executors.cobalt*), 366
- `logger` (*in module buildtest.executors.lsf*), 371
- `logger` (*in module buildtest.executors.pbs*), 373
- `logger` (*in module buildtest.executors.setup*), 374
- `logger` (*in module buildtest.executors.slurm*), 377
- `LSF` (*class in buildtest.system*), 390
- `LSFBatchScript` (*class in buildtest.buildsystem.batch*), 343
- `LSFExecutor` (*class in buildtest.executors.lsf*), 368
- `LSFJob` (*class in buildtest.executors.lsf*), 369
- M**
- `main()` (*in module buildtest.main*), 389
- module**
- `buildtest`, 340
- `buildtest.buildsystem`, 340
- `buildtest.buildsystem.base`, 340
- `buildtest.buildsystem.batch`, 343
- `buildtest.buildsystem.builders`, 344
- `buildtest.buildsystem.compilerbuilder`, 345
- `buildtest.buildsystem.parser`, 347
- `buildtest.buildsystem.scripbuilder`, 348
- `buildtest.buildsystem.spack`, 348
- `buildtest.cli`, 349
- `buildtest.cli.build`, 349
- `buildtest.cli.buildspec`, 352
- `buildtest.cli.cdash`, 355
- `buildtest.cli.compilers`, 356
- `buildtest.cli.config`, 357
- `buildtest.cli.help`, 358
- `buildtest.cli.history`, 358
- `buildtest.cli.inspect`, 359
- `buildtest.cli.report`, 360
- `buildtest.cli.schema`, 361
- `buildtest.config`, 385
- `buildtest.defaults`, 387
- `buildtest.exceptions`, 387
- `buildtest.executors`, 363
- `buildtest.executors.base`, 363

[buildtest.executors.cobalt](#), 364
[buildtest.executors.job](#), 367
[buildtest.executors.local](#), 367
[buildtest.executors.lsf](#), 368
[buildtest.executors.pbs](#), 371
[buildtest.executors.setup](#), 373
[buildtest.executors.slurm](#), 375
[buildtest.log](#), 388
[buildtest.main](#), 389
[buildtest.schemas](#), 378
[buildtest.schemas.defaults](#), 378
[buildtest.schemas.utils](#), 379
[buildtest.system](#), 389
[buildtest.utils](#), 379
[buildtest.utils.command](#), 379
[buildtest.utils.file](#), 381
[buildtest.utils.shell](#), 383
[buildtest.utils.timer](#), 384
[buildtest.utils.tools](#), 384

N

[name\(\)](#) (*buildtest.config.SiteConfiguration* method), 386

O

[opts\(\)](#) (*buildtest.utils.shell.Shell* property), 383
[out\(\)](#) (*buildtest.utils.command.Capturing* property), 380
[output_file\(\)](#) (*buildtest.executors.cobalt.CobaltJob* method), 366
[output_file\(\)](#) (*buildtest.executors.lsf.LSFJob* method), 371
[output_file\(\)](#) (*buildtest.executors.pbs.PBSJob* method), 372

P

[parse_buildspecs\(\)](#) (*buildtest.cli.build.BuildTest* method), 350
[path\(\)](#) (*buildtest.utils.shell.Shell* property), 383
[PBS](#) (class in *buildtest.system*), 390
[PBSBatchScript](#) (class in *buildtest.buildsystem.batch*), 343
[PBSExecutor](#) (class in *buildtest.executors.pbs*), 371
[PBSJob](#) (class in *buildtest.executors.pbs*), 372
[poll\(\)](#) (*buildtest.executors.cobalt.CobaltExecutor* method), 365
[poll\(\)](#) (*buildtest.executors.cobalt.CobaltJob* method), 366
[poll\(\)](#) (*buildtest.executors.job.Job* method), 367
[poll\(\)](#) (*buildtest.executors.lsf.LSFExecutor* method), 369
[poll\(\)](#) (*buildtest.executors.lsf.LSFJob* method), 371
[poll\(\)](#) (*buildtest.executors.pbs.PBSExecutor* method), 372
[poll\(\)](#) (*buildtest.executors.pbs.PBSJob* method), 372

[poll\(\)](#) (*buildtest.executors.setup.BuildExecutor* method), 374
[poll\(\)](#) (*buildtest.executors.slurm.SlurmExecutor* method), 375
[poll\(\)](#) (*buildtest.executors.slurm.SlurmJob* method), 377
[poll_cmd](#) (*buildtest.executors.pbs.PBSExecutor* attribute), 372
[poll_jobs\(\)](#) (*buildtest.cli.build.BuildTest* method), 350
[positive_number\(\)](#) (in module *buildtest.cli*), 363
[print_buildspecs\(\)](#) (*buildtest.cli.buildspec.BuildspecCache* method), 354
[print_by_executors\(\)](#) (*buildtest.cli.buildspec.BuildspecCache* method), 354
[print_by_tags\(\)](#) (*buildtest.cli.buildspec.BuildspecCache* method), 354
[print_compilers\(\)](#) (*buildtest.cli.compilers.BuildtestCompilers* method), 357
[print_discovered_buildspecs\(\)](#) (in module *buildtest.cli.build*), 352
[print_display_table\(\)](#) (*buildtest.cli.report.Report* method), 361
[print_filter_fields\(\)](#) (*buildtest.cli.buildspec.BuildspecCache* static method), 354
[print_filter_fields\(\)](#) (*buildtest.cli.report.Report* method), 361
[print_format_fields\(\)](#) (*buildtest.cli.buildspec.BuildspecCache* static method), 354
[print_format_fields\(\)](#) (*buildtest.cli.report.Report* method), 361
[print_json\(\)](#) (*buildtest.cli.compilers.BuildtestCompilers* method), 357
[print_maintainer\(\)](#) (*buildtest.cli.buildspec.BuildspecCache* method), 354
[print_maintainers_by_buildspecs\(\)](#) (*buildtest.cli.buildspec.BuildspecCache* method), 355
[print_paths\(\)](#) (*buildtest.cli.buildspec.BuildspecCache* method), 355
[print_yaml\(\)](#) (*buildtest.cli.compilers.BuildtestCompilers* method), 357
[process_report\(\)](#) (*buildtest.cli.report.Report* method), 361

Q

[query_builds\(\)](#) (in module *buildtest.cli.history*), 359

R

[read_file\(\)](#) (in module *buildtest.utils.file*), 382
[remove_file\(\)](#) (in module *buildtest.utils.file*), 382
[Report](#) (class in *buildtest.cli.report*), 360

- `report_cmd()` (in module `buildtest.cli.report`), 361
- `report_menu()` (in module `buildtest.cli`), 363
- `resolve()` (`buildtest.config.SiteConfiguration` method), 386
- `resolve_path()` (in module `buildtest.utils.file`), 382
- `resolve_testdirectory()` (`buildtest.cli.build.BuildTest` method), 350
- `resolver` (in module `buildtest.schemas.defaults`), 378
- `returncode()` (`buildtest.utils.command.BuildTestCommand` method), 380
- `run()` (`buildtest.buildsystem.base.BuilderBase` method), 342
- `run()` (`buildtest.executors.base.BaseExecutor` method), 364
- `run()` (`buildtest.executors.local.LocalExecutor` method), 368
- `run()` (`buildtest.executors.setup.BuildExecutor` method), 374
- `run_command()` (`buildtest.buildsystem.base.BuilderBase` method), 342
- `run_phase()` (`buildtest.cli.build.BuildTest` method), 351
- `runtime()` (`buildtest.buildsystem.base.BuilderBase` method), 342
- S**
- `Scheduler` (class in `buildtest.system`), 390
- `schema_cmd()` (in module `buildtest.cli.schema`), 361
- `schema_menu()` (in module `buildtest.cli`), 363
- `SCHEMA_ROOT` (in module `buildtest.defaults`), 387
- `schema_store` (in module `buildtest.schemas.defaults`), 378
- `schema_table` (in module `buildtest.cli`), 363
- `schema_table` (in module `buildtest.schemas.defaults`), 378
- `ScriptBuilder` (class in `buildtest.buildsystem.scriptbuilder`), 348
- `set_cc()` (`buildtest.buildsystem.compilerbuilder.CompilerBuilder` method), 346
- `set_cflags()` (`buildtest.buildsystem.compilerbuilder.CompilerBuilder` method), 346
- `set_command()` (`buildtest.utils.command.BuildTestCommand` method), 380
- `set_cppflags()` (`buildtest.buildsystem.compilerbuilder.CompilerBuilder` method), 346
- `set_cxx()` (`buildtest.buildsystem.compilerbuilder.CompilerBuilder` attribute), 348
- `set_cxxflags()` (`buildtest.buildsystem.compilerbuilder.CompilerBuilder` method), 346
- `set_fc()` (`buildtest.buildsystem.compilerbuilder.CompilerBuilder` method), 346
- `set_fflags()` (`buildtest.buildsystem.compilerbuilder.CompilerBuilder` method), 346
- `set_ldflags()` (`buildtest.buildsystem.compilerbuilder.CompilerBuilder` method), 346
- `set_stderr()` (`buildtest.utils.command.Capturing` method), 380
- `set_stdout()` (`buildtest.utils.command.Capturing` method), 380
- `setup()` (`buildtest.buildsystem.compilerbuilder.CompilerBuilder` method), 346
- `setup()` (`buildtest.executors.setup.BuildExecutor` method), 374
- `Shell` (class in `buildtest.utils.shell`), 383
- `SiteConfiguration` (class in `buildtest.config`), 385
- `Slurm` (class in `buildtest.system`), 391
- `SlurmBatchScript` (class in `buildtest.buildsystem.batch`), 343
- `SlurmExecutor` (class in `buildtest.executors.slurm`), 375
- `SlurmJob` (class in `buildtest.executors.slurm`), 376
- `SpackBuilder` (class in `buildtest.buildsystem.spack`), 348
- `start()` (`buildtest.buildsystem.base.BuilderBase` method), 342
- `start()` (`buildtest.utils.timer.Timer` method), 384
- `starttime()` (`buildtest.buildsystem.base.BuilderBase` method), 342
- `state()` (`buildtest.executors.job.Job` method), 367
- `state()` (`buildtest.executors.slurm.SlurmJob` method), 377
- `stop()` (`buildtest.buildsystem.base.BuilderBase` method), 342
- `stop()` (`buildtest.utils.timer.Timer` method), 384
- `success()` (`buildtest.executors.pbs.PBSJob` method), 372
- `supported_schemas` (in module `buildtest.defaults`), 387
- `supported_type_schemas` (in module `buildtest.defaults`), 387
- `system` (`buildtest.system.BuildTestSystem` attribute), 389
- `system` (in module `buildtest.system`), 391
- T**
- `table` (`buildtest.cli.buildspec.BuildspecCache` attribute), 363
- `Timer` (class in `buildtest.utils.timer`), 384
- `TimerError`, 384
- `type` (`buildtest.buildsystem.compilerbuilder.CompilerBuilder` attribute), 345
- `type` (`buildtest.buildsystem.scriptbuilder.ScriptBuilder` attribute), 348
- `type` (`buildtest.buildsystem.spack.SpackBuilder` attribute), 348
- `type` (`buildtest.executors.base.BaseExecutor` attribute), 363
- `type` (`buildtest.executors.cobalt.CobaltExecutor` attribute), 365
- `type` (`buildtest.executors.local.LocalExecutor` attribute), 367
- `type` (`buildtest.executors.lsf.LSFExecutor` attribute), 368

type (*buildtest.executors.pbs.PBSExecutor* attribute),
372
type (*buildtest.executors.slurm.SlurmExecutor* at-
tribute), 375

U

update_report() (*in module buildtest.cli.build*), 352
upload_test_cdash() (*in module buildtest.cli.cdash*),
356
USER_SETTINGS_FILE (*in module buildtest.defaults*),
387
userhome (*in module buildtest.defaults*), 387

V

valid_shells (*buildtest.utils.shell.Shell* attribute), 383
validate() (*buildtest.config.SiteConfiguration* method),
386
validate_config() (*in module buildtest.cli.config*),
357
VAR_DIR (*in module buildtest.defaults*), 387
view_configuration() (*in module buildtest.cli.config*), 357
view_executors() (*in module buildtest.cli.config*), 357
view_summary() (*in module buildtest.cli.config*), 358
view_system() (*in module buildtest.cli.config*), 358

W

walk_tree() (*in module buildtest.utils.file*), 382
workdir() (*buildtest.executors.slurm.SlurmJob*
method), 377
write_file() (*in module buildtest.utils.file*), 383
write_python_script()
(*buildtest.buildsystem.scriptbuilder.ScriptBuilder*
method), 348
write_testresults()
(*buildtest.executors.local.LocalExecutor*
method), 368