buildtest Documentation

Release 0.9.0

Shahzeb Siddiqui

Dec 15, 2020

BACKGROUND

This documentation was last rebuild on Dec 15, 2020 and is intended for version 0.9.0.

Please refer to https://buildtest.readthedocs.io/en/latest/ for documentation on latest release. If you are working off *devel* branch then please to *devel* docs at https://buildtest.readthedocs.io/en/devel/.

CHAPTER

ONE

STATUS

CHAPTER
TWO

SOURCE CODE

- buildtest framework: https://github.com/buildtesters/buildtest

CHAPTER
THREE

TEST REPOSITORIES

- Cori @ NERSC: https://github.com/buildtesters/buildtest-cori
- Stampede2 @ TACC: https://github.com/buildtesters/buildtest-stampede2

CHAPTER
FOUR

USEFUL LINKS

- Documentation: http://buildtest.rtfd.io/
- Schema Docs: https://buildtesters.github.io/buildtest/
- ReadTheDocs: https://readthedocs.org/projects/buildtest/
- CodeCov: https://codecov.io/gh/buildtesters/buildtest
- Coveralls: https://coveralls.io/github/buildtesters/buildtest
- CodeFactor: https://www.codefactor.io/repository/github/buildtesters/buildtest
- Snyk: https://app.snyk.io/org/buildtesters/
- Slack Channel: http://hpcbuildtest.slack.com. Click Here to Join Slack

CHAPTER
FIVE

DESCRIPTION

buildtest is a HPC testing framework to help sites perform acceptance & regression testing of an HPC system. buildtest utilizes json schema to define structure of test written in YAML called a **Buildspec File**. The schema docs is a resource that hosts buildtest schemas and documents all field attributes for each schema, this will be useful when you are writting buildspecs.
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
To get started with buildtest, please review the *Terminology* and proceed to *Installing buildtest* followed by Getting Started.
For additional reference, you can read *Summary of buildtest* and *Conference and Publications*.

5.1 Summary of buildtest

Contents

5.1.1 Background

HPC computing environment is a tightly coupled system that includes a cluster of nodes and accelerators interconnected with a high-speed interconnect, a parallel filesystem, multiple storage tiers, a batch scheduler for users to submit jobs to the cluster and a software stack for users to run their workflows. A **software stack is a collection of compilers, MPI, libraries, system utilities and scientific packages** typically installed in a parallel file-system. A module tool like `environment-modules` or `Lmod` is generally used for loading the software environment into the users' shell environment.
Software are packaged in various forms that determine how they are installed. A few package formats are: `binary`, `Makefile`, `CMake`, `Autoconf`, `github`, `PyPi`, `Conda`, `RPM`, `tarball`, `rubygem`, `MakeCp`, `jar`, and many more. With many packaging formats, this creates a burden for HPC support team to learn how to build software since each one has a unique build process. Software build tools like EasyBuild and Spack can build up to 1000+ software packages by supporting many packaging formats to address all sorts of software builds. Easybuild and Spack provide end-end software build automation that helps HPC site to build a very large software stack with many combinatorial software configurations. During the installation, some packages will provide a test harness that can be executed via Easybuild or Spack which typically invokes a `make test` or `ctest` for packages that follow ConfigureMake, Autoconf, or CMake install process.

Many HPC sites rely on their users for testing the software stack, and some sites may develop in-house test scripts to run sanity check for popular scientific tools. Despite these efforts, there is little or no collaboration between HPC sites on sharing tests because they are site-specific and often provide no documentation. For many sites, the HPC support team don't have the time for conducting software stack testing because:

1. lack of domain expertise and understaffed
2. no standard test-suite and framework to automate test build and execution.

Frankly, HPC support teams are so busy with important day-day operation and engineering projects that software testing is either neglected or left to end-users. This demands for a concerted effort by HPC community to **build a strong open-source community** around software stack testing.
There are two points that need to be addressed. First, we need a **framework to do automatic testing** of installed software stack. Second, is to **build a test repository** for scientific software that is community driven and reusable amongst the HPC community. An automated test framework is a harness for *automating* the test creation process, but it requires a community contribution to accumulate this repository on per-package basis.
buildtest was designed to address both these points, it is a **framework** to perform automatic testing and it provides a repository of test-configurations that can be shared by HPC community.

5.1.2 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, a framework that automates test creation by using test configuration written in YAML syntax. YAML was picked given its simplicity and it lowers the barrier for new to start sharing test configuration in order to build a comprehensive test suite that will work with buildtest.

5.1.3 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.

5.1.4 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 :

1. Run your test suite during system maintenance
2. Perform daily tests for testing various system components. These tests should be short
3. Run weekly/biweekly test on medium/large workload including micro-benchmark

If you are interested in buildtest, please Join Slack Channel and your feedback will help improve buildtest.

5.1.5 Timeline

Date	Description
Feb 18th 2017	Start of project
Aug 20th 2017	In v0.1.5 buildtest was converted from bash to Python and project was moved into github https://github.com/HPC-buildtest/buildtest
Sep 11th 2018	In v0.4.0 buildtest was ported from Python 2 to 3
Mar 3rd 2020	A spin-off project called lmodule was formed based on buildtest module features

5.1.6 Related Projects and community efforts

- ReFrame: `Re` gression `FRAME` work for Software Testing. ReFrame is developed by CSCS
- Pavilion2: is a framework for running and analyzing tests targeting HPC systems. Pavilion2 is developed by LANL
- Automatic Testing of Installed Software (ATIS) - This project was presented by Xavier Besseron in FOSDEM14 however this project is no longer in development.
- hpcswtest - is a HPC Software Stack testing framework by Idaho National Lab however this project is no longer in development.

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.

5.2 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 by a `Buildspec Schema`.
Schema	is a JSON file that defines structure of a buildspec file and it is used for validating a buildspec
Global Schema	is a JSON schema that is validated for all schema types
Sub Schema	A buildspec is validated with one sub-schema defined by `type` field.
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` which defines if test is run locally or via a scheduler. The executors are defined in the `Settings` file.

5.3 Installing buildtest

5.3.1 Requirements

You need the following packages to get started.

- git
- Python >= 3.6
- pip

5.3.2 Cloning buildtest

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

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

If you prefer the SSH method, make sure your GitHub account is configured properly, for more details see Connecting to GitHub with SSH
Once your account is configured you can clone the repository as follows:

```
$ 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
```

5.3.3 Installing buildtest

To install buildtest run the following depending on your shell:

```
# BASH users
$ source setup.sh

# CSH users
$ source setup.csh
```

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

5.3.4 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
```

5.3.5 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 executingtests. Buildtest comes␣
↪with a set of json-schemas used to write test configuration (Buildspecs) in YAML to␣
↪generate test scripts.

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         show program's version number and exit
  -d {DEBUG,INFO,WARNING,ERROR,CRITICAL}, --debug {DEBUG,INFO,WARNING,ERROR,CRITICAL}
                        Enable debugging messages.

COMMANDS:

   build                          Options for building test scripts
   buildspec                      Command options for buildspecs
   report                         Show report for test results
   schema                         Commands for viewing buildtest schemas
   config                         Buildtest Configuration Menu
   inspect                        Inspect details for test from test report
   docs                           Open buildtest docs in browser
   schemadocs                     Open buildtest schema docs in browser

  {docs,schemadocs,build,buildspec,report,inspect,schema,config}

Documentation: https://buildtest.readthedocs.io/en/latest/index.html
```

buildtest commands make use of sub-commands (i.e `buildtest <subcommand>`). For more details on any subcommand run:

```
$ buildtest <subcommand> --help
```

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

5.4 Getting Started with buildtest

5.4.1 Interacting with the client

Once you install buildtest, you should find the client on your $PATH, you can run the following to see path to buildtest:

```
$ which buildtest
```

If you don't see buildtest go back and review section *Installing buildtest*.

5.4.2 Build Usage

The `buildtest build` command is used for building and running tests. Buildtest will read one or more Buildspecs (YAML) file that adheres to one of the buildtest schemas. For a complete list of build options, run `buildtest build --help`

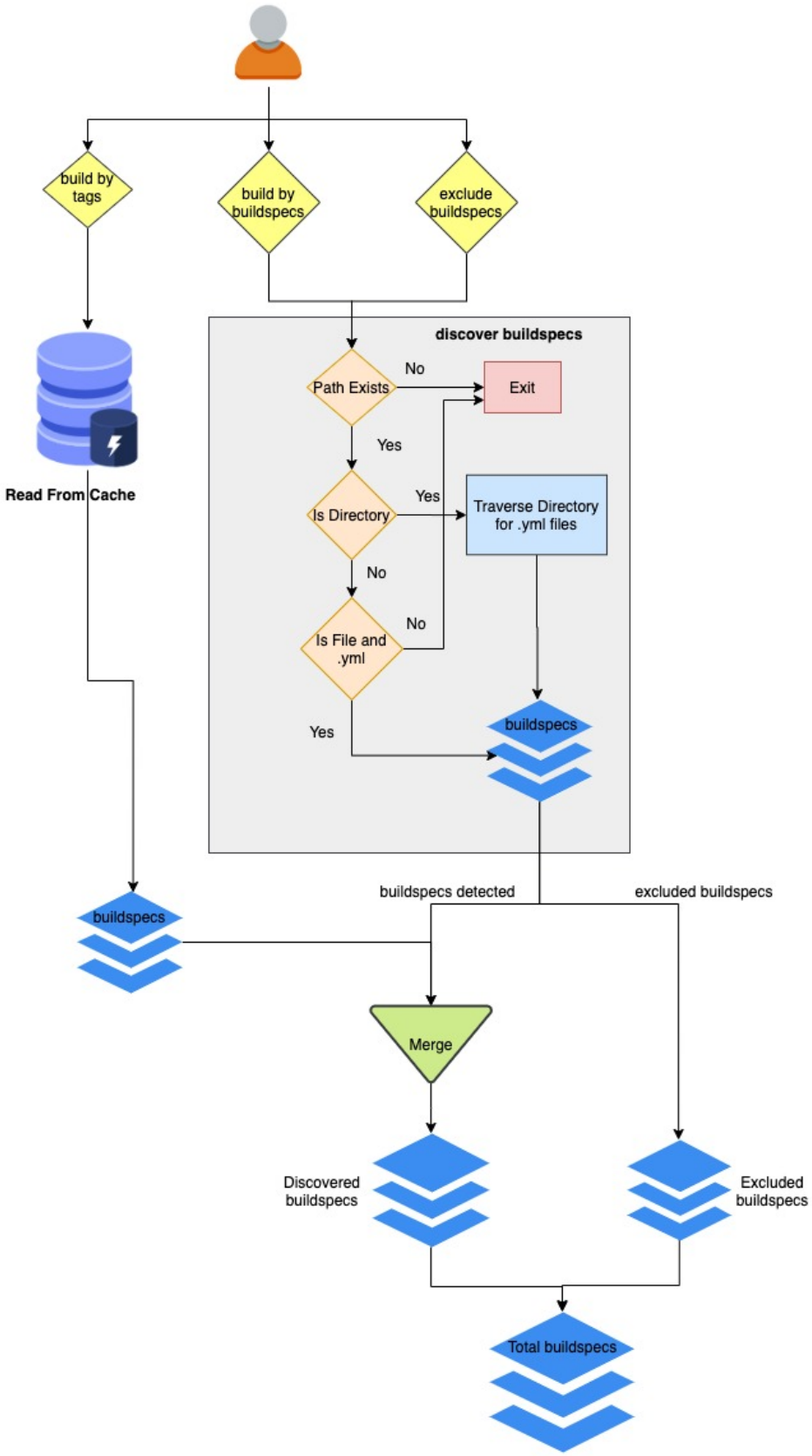
```
$ buildtest build --help
usage: buildtest [options] [COMMANDS] build [-h] [-b BUILDSPEC] [-x EXCLUDE] [--tags␣
↪TAGS] [-e EXECUTOR]
                                            [-s {parse,build}] [-t TESTDIR] [--
↪rebuild REBUILD]

optional arguments:
  -h, --help            show this help message and exit
  -b BUILDSPEC, --buildspec BUILDSPEC
                        Specify a Buildspec (YAML) file to build and run the test.
  -x EXCLUDE, --exclude EXCLUDE
                        Exclude one or more configs from processing. Configs can be␣
↪files or directories.
  --tags TAGS           Specify buildspecs by tags found in buildspec cache
  -e EXECUTOR, --executor EXECUTOR
                        Specify buildspecs by executor name found in buildspec cache
  -s {parse,build}, --stage {parse,build}
                        control behavior of buildtest build
  -t TESTDIR, --testdir TESTDIR
                        specify a custom test directory. By default, use .buildtest␣
↪in $PWD.
  --rebuild REBUILD     Rebuild test X number of times. Must be a positive number␣
↪between [1-50]
```

5.4.3 Discover Buildspecs

The buildspec search resolution is described as follows:

- If file doesn't exist, check for file in *buildspec roots* and break after first match
- If buildspec path is a directory, traverse directory recursively to find all `.yml` extensions
- If buildspec path is a file, check if file extension is not `.yml`, exit immediately

Shown below is a diagram on how buildtest discovers buildspecs. The user inputs a buildspec via `--buildspec` or tags (`--tags`) *Building By Tags* which will discover the buildspecs. User can *Excluding Buildspecs* using `--exclude` option which is processed after discovering buildspecs. The excluded buildspecs are removed from list if found and final list of buildspecs is processed.


build by tags
build by buildspecs
exclude buildspecs
discover buildspecs
Path Exists
No
Exit
Yes
Is Directory
Yes
Traverse Directory for .yml files
No
Is File and .yml
No
Yes
buildspecs
Read From Cache
buildspecs
buildspecs detected
excluded buildspecs
Merge
Discovered buildspecs
Excluded buildspecs
Total buildspecs

5.4.4 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

```
$ buildtest build -b /Users/siddiq90/Documents/buildtest/tutorials/systemd.yml

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/systemd.yml

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile              | validstate   | buildspec
-------------------------+--------------+-----------------------------------------
↪--------------
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/systemd.yml

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

 name                    | id       | type   | executor   | tags          | testpath
-------------------------+----------+--------+------------+---------------+------------
↪-----------------------------------------------------------------------------------
↪-------------
 systemd_default_target | 1770533b | script | local.bash | ['tutorials'] | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/systemd/systemd_default_target/0/
↪stage/generate.sh

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

 name                    | id       | executor   | status   |   returncode | testpath
-------------------------+----------+------------+----------+--------------+-----------
↪-----------------------------------------------------------------------------------
↪--------------
 systemd_default_target | 1770533b | local.bash | FAIL     |            1 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/systemd/systemd_default_target/0/
↪stage/generate.sh

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

Executed 1 tests
Passed Tests: 0/1 Percentage: 0.000%
Failed Tests: 1/1 Percentage: 100.000%
```

buildtest won't accept `.yaml` file extension for file, this can be demonstrated as follows:

```
$ buildtest build -b invalid_ext.yaml
invalid_ext.yaml does not end in file extension .yml
There are no config files to process.
```

buildtest can perform a directory build for instance let's build for directory `tests/examples/buildspecs` where buildtest will recursively search for all `.yml` files

```
$ buildtest build -b tests/examples/buildspecs/

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/python-shell.yml
/Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/environment.yml
/Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/shell_examples.yml
/Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/slurm.yml

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile              | validstate   | buildspec
-------------------------+--------------+-------------------------------------------
↪-------------------------------------
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/tests/
↪examples/buildspecs/python-shell.yml
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/tests/
↪examples/buildspecs/environment.yml
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/tests/
↪examples/buildspecs/shell_examples.yml
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/tests/
↪examples/buildspecs/slurm.yml

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

 name                        | id       | type   | executor     | tags   | testpath
-----------------------------+----------+--------+--------------+--------+-------------
↪-----------------------------------------------------------------------------------
↪--------------
 circle_area                 | 0bdaef77 | script | local.python |        | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.python/python-shell/circle_area/0/
↪stage/generate.sh
 hello_dinosaur              | dff7a691 | script | local.bash   |        | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/environment/hello_dinosaur/0/
↪stage/generate.sh
 _bin_sh_shell               | f31c3498 | script | local.sh     |        | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/_bin_sh_shell/0/
↪stage/generate.sh
 _bin_bash_shell             | 9700d000 | script | local.bash   |        | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/_bin_bash_shell/0/
↪stage/generate.sh
 bash_shell                  | c110d07a | script | local.bash   |        | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/bash_shell/0/stage/
↪generate.sh
```

(continues on next page)

(continued from previous page)

```
 sh_shell                    | 9cbe76d3 | script | local.sh     |         | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/sh_shell/0/stage/
↪generate.sh
 shell_options               | 47330a4a | script | local.sh     |         | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/shell_options/0/
↪stage/generate.sh
 slurm_down_nodes_reason     | 43858c19 | script | local.bash   |         | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/slurm/slurm_down_nodes_reason/0/
↪stage/generate.sh
 slurm_not_responding_nodes | 49a854e8 | script | local.bash   |         | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/slurm/slurm_not_responding_nodes/
↪0/stage/generate.sh

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

 name                         | id       | executor     | status   |   returncode |␣
↪testpath
------------------------------+----------+--------------+----------+--------------+-----
↪-------------------------------------------------------------------------------------
↪----------------------
 circle_area                  | 0bdaef77 | local.python | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.python/python-shell/circle_area/
↪0/stage/generate.sh
 hello_dinosaur               | dff7a691 | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/environment/hello_dinosaur/
↪0/stage/generate.sh
 _bin_sh_shell                | f31c3498 | local.sh     | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/_bin_sh_shell/
↪0/stage/generate.sh
 _bin_bash_shell              | 9700d000 | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/_bin_bash_
↪shell/0/stage/generate.sh
 bash_shell                   | c110d07a | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/bash_shell/0/
↪stage/generate.sh
 sh_shell                     | 9cbe76d3 | local.sh     | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/sh_shell/0/
↪stage/generate.sh
 shell_options                | 47330a4a | local.sh     | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/shell_options/
↪0/stage/generate.sh
 slurm_down_nodes_reason      | 43858c19 | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/slurm/slurm_down_nodes_
↪reason/0/stage/generate.sh
 slurm_not_responding_nodes | 49a854e8 | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/slurm/slurm_not_responding_
↪nodes/0/stage/generate.sh

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

Executed 9 tests
Passed Tests: 9/9 Percentage: 100.000%
Failed Tests: 0/9 Percentage: 0.000%
```

In next section, you will see, we can build multiple buildspecs and interchange file and directory with `-b` option.

Building Multiple Buildspecs

Buildtest supports building multiple buildspecs, just specify the `-b` option for every Buildspec you want to build. In this example, we specify a file and directory path. The search resolution is performed for every argument (`-b`) independently, and accumulated into list.

```
$ buildtest build -b tests/examples/buildspecs/ -b tutorials/systemd.yml

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/python-shell.yml
/Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/environment.yml
/Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/shell_examples.yml
/Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/slurm.yml
/Users/siddiq90/Documents/buildtest/tutorials/systemd.yml

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile               | validstate   | buildspec
--------------------------+--------------+-------------------------------------------
↪-------------------------------------
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/tests/
↪examples/buildspecs/python-shell.yml
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/tests/
↪examples/buildspecs/environment.yml
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/tests/
↪examples/buildspecs/shell_examples.yml
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/tests/
↪examples/buildspecs/slurm.yml
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/systemd.yml

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

 name                         | id       | type   | executor     | tags          |␣
↪testpath
-----------------------------+----------+--------+--------------+---------------+------
↪-------------------------------------------------------------------------------------
↪---------------------
 circle_area                 | b06f76c2 | script | local.python |               | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.python/python-shell/circle_area/
↪1/stage/generate.sh
 hello_dinosaur              | cebde392 | script | local.bash   |               | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/environment/hello_dinosaur/
↪1/stage/generate.sh
 _bin_sh_shell               | d554763a | script | local.sh     |               | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/_bin_sh_shell/
↪1/stage/generate.sh
```

(continues on next page)

(continued from previous page)

```
 _bin_bash_shell            | e412e216 | script | local.bash   |               | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/_bin_bash_
↪shell/1/stage/generate.sh
 bash_shell                 | 5c94695b | script | local.bash   |               | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/bash_shell/1/
↪stage/generate.sh
 sh_shell                   | c61e8164 | script | local.sh     |               | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/sh_shell/1/
↪stage/generate.sh
 shell_options              | d5d62f37 | script | local.sh     |               | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/shell_options/
↪1/stage/generate.sh
 slurm_down_nodes_reason    | 60b25553 | script | local.bash   |               | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/slurm/slurm_down_nodes_
↪reason/1/stage/generate.sh
 slurm_not_responding_nodes | 97bf2c4a | script | local.bash   |               | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/slurm/slurm_not_responding_
↪nodes/1/stage/generate.sh
 systemd_default_target     | 15a66b55 | script | local.bash   | ['tutorials'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/systemd/systemd_default_
↪target/1/stage/generate.sh

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

 name                       | id       | executor     | status   |   returncode |␣
↪testpath
----------------------------+----------+--------------+----------+--------------+-----
↪-------------------------------------------------------------------------------------
↪----------------------
 circle_area                | b06f76c2 | local.python | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.python/python-shell/circle_area/
↪1/stage/generate.sh
 hello_dinosaur             | cebde392 | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/environment/hello_dinosaur/
↪1/stage/generate.sh
 _bin_sh_shell              | d554763a | local.sh     | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/_bin_sh_shell/
↪1/stage/generate.sh
 _bin_bash_shell            | e412e216 | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/_bin_bash_
↪shell/1/stage/generate.sh
 bash_shell                 | 5c94695b | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/bash_shell/1/
↪stage/generate.sh
 sh_shell                   | c61e8164 | local.sh     | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/sh_shell/1/
↪stage/generate.sh
 shell_options              | d5d62f37 | local.sh     | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/shell_options/
↪1/stage/generate.sh
 slurm_down_nodes_reason    | 60b25553 | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/slurm/slurm_down_nodes_
↪reason/1/stage/generate.sh
 slurm_not_responding_nodes | 97bf2c4a | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/slurm/slurm_not_responding_
↪nodes/1/stage/generate.sh
```

(continues on next page)

(continued from previous page)

```
 systemd_default_target     | 15a66b55 | local.bash    | FAIL     |            1 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/systemd/systemd_default_
↪target/1/stage/generate.sh

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

Executed 10 tests
Passed Tests: 9/10 Percentage: 90.000%
Failed Tests: 1/10 Percentage: 10.000%
```

Excluding Buildspecs

Buildtest provides `--exclude` option or short option `-x` to exclude buildspecs which can be useful when you want to build all buildspecs in a directory but exclude a few buildspecs or exclude a sub-directory.
For example we can build all buildspecs in `examples` but exclude file `examples/systemd.yml` by running:

```
$ buildtest build -b examples -x examples/systemd.yml
```

buildtest will discover all Buildspecs and then exclude any buildspecs 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 examples/ -x examples/
There are no Buildspec files to process.
```

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

Building By Tags

buildtest can perform builds by tags by using `--tags` option. In order to use this feature, buildspecs must be in cache so you must run `buildtest buildspec find` or see *Finding Buildspecs*.
To build all tutorials tests you can perform `buildtest build --tags tutorials`. In the buildspec there is a field `tags: [tutorials]` to classify tests. buildtest will read the cache file `var/buildspec-cache.json` and see which buildspecs have a matching tag. You should run `buildtest buildspec find` atleast once, in order to detect cache file.

```
$ buildtest build --tags tutorials

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/environment.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/pre_post_build_run.yml
/Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml
/Users/siddiq90/Documents/buildtest/tutorials/vars.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/passing_args.yml
/Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
/Users/siddiq90/Documents/buildtest/tutorials/run_only_platform.yml
/Users/siddiq90/Documents/buildtest/tutorials/python-shell.yml
```

(continues on next page)

(continued from previous page)

```
/Users/siddiq90/Documents/buildtest/tutorials/root_user.yml
/Users/siddiq90/Documents/buildtest/tutorials/skip_tests.yml
/Users/siddiq90/Documents/buildtest/tutorials/sleep.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/vecadd.yml
/Users/siddiq90/Documents/buildtest/tutorials/shebang.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml
/Users/siddiq90/Documents/buildtest/tutorials/hello_world.yml
/Users/siddiq90/Documents/buildtest/tutorials/systemd.yml
/Users/siddiq90/Documents/buildtest/tutorials/selinux.yml
[run_only_platform_linux] test is skipped because ['run_only']['platform'] got value: Linux but detected platform: Darwin.
[run_only_as_root] test is skipped because ['run_only']['user'] got value: root but detected user: siddiq90.
[skip] test is skipped.

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile                | validstate   | buildspec
---------------------------+--------------+-----------------------------------------
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/tutorials/environment.yml
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/tutorials/compilers/pre_post_build_run.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/tutorials/vars.yml
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/tutorials/compilers/passing_args.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/tutorials/run_only_platform.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/tutorials/python-shell.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/tutorials/root_user.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/tutorials/skip_tests.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/tutorials/sleep.yml
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/tutorials/compilers/vecadd.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/tutorials/shebang.yml
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/tutorials/hello_world.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/tutorials/systemd.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/tutorials/selinux.yml
```

(continues on next page)

(continued from previous page)

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

 name                      | id       | type     | executor     | tags                ␣
↪    | testpath
--------------------------+----------+----------+--------------+---------------------
↪----+-------------------------------------------------------------------------------
↪-----------------------------------------
 environment_variables     | d0e51818 | script   | local.bash   | ['tutorials']       ␣
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/environment/
↪environment_variables/0/stage/generate.sh
 pre_post_build_run        | 1e81254f | compiler | local.bash   | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/pre_post_
↪build_run/pre_post_build_run/0/stage/generate.sh
 _bin_sh_shell             | 85c5e433 | script   | local.sh     | ['tutorials']       ␣
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/_bin_sh_
↪shell/2/stage/generate.sh
 _bin_bash_shell           | 06ef100e | script   | local.bash   | ['tutorials']       ␣
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/_bin_
↪bash_shell/2/stage/generate.sh
 bash_shell                | 13e306ff | script   | local.bash   | ['tutorials']       ␣
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/bash_
↪shell/2/stage/generate.sh
 sh_shell                  | a018fabc | script   | local.sh     | ['tutorials']       ␣
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/sh_
↪shell/2/stage/generate.sh
 shell_options             | a7a23ec8 | script   | local.sh     | ['tutorials']       ␣
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/shell_
↪options/2/stage/generate.sh
 variables                 | 3adfeb8b | script   | local.bash   | ['tutorials']       ␣
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/vars/variables/0/
↪stage/generate.sh
 executable_arguments      | 5d670438 | compiler | local.bash   | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/passing_args/
↪executable_arguments/0/stage/generate.sh
 exit1_fail                | d405aea9 | script   | local.sh     | ['tutorials', 'fail
↪']    | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/
↪exit1_fail/0/stage/generate.sh
 exit1_pass                | 992a08e0 | script   | local.sh     | ['tutorials', 'pass
↪']    | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/
↪exit1_pass/0/stage/generate.sh
 returncode_list_mismatch | 269abcb9 | script   | local.sh     | ['tutorials', 'fail
↪']    | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/
↪returncode_list_mismatch/0/stage/generate.sh
 returncode_int_match      | 146a0269 | script   | local.sh     | ['tutorials', 'pass
↪']    | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/
↪returncode_int_match/0/stage/generate.sh
 run_only_platform_darwin | 1d86e162 | script   | local.python | ['tutorials']       ␣
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.python/run_only_platform/
↪run_only_platform_darwin/0/stage/generate.sh
 circle_area               | 59d92815 | script   | local.python | ['tutorials', 'python
↪']  | /Users/siddiq90/Documents/buildtest/var/tests/local.python/python-shell/
↪circle_area/2/stage/generate.sh
 unskipped                 | 17706881 | script   | local.bash   | ['tutorials']       ␣
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/skip_tests/unskipped/
↪0/stage/generate.sh
```

(continues on next page)

(continued from previous page)

```
 sleep                    | eadfa0df | script   | local.bash   | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/sleep/sleep/0/stage/
↪generate.sh
 vecadd_gnu               | 62df938a | compiler | local.bash   | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/vecadd/vecadd_
↪gnu/0/stage/generate.sh
 bash_login_shebang       | a4196349 | script   | local.bash   | tutorials
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_login_
↪shebang/0/stage/generate.sh
 bash_nonlogin_shebang    | 72c038f0 | script   | local.bash   | tutorials
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_
↪nonlogin_shebang/0/stage/generate.sh
 hello_f                  | 34d5c6d8 | compiler | local.bash   | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/
↪hello_f/0/stage/generate.sh
 hello_c                  | 05ed531f | compiler | local.bash   | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/
↪hello_c/0/stage/generate.sh
 hello_cplusplus          | 2543a4e8 | compiler | local.bash   | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/
↪hello_cplusplus/0/stage/generate.sh
 cc_example               | 3143c39a | compiler | local.bash   | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cc_
↪example/0/stage/generate.sh
 fc_example               | b84feab0 | compiler | local.bash   | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/fc_
↪example/0/stage/generate.sh
 cxx_example              | 855fae37 | compiler | local.bash   | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cxx_
↪example/0/stage/generate.sh
 hello_world              | fc9f1058 | script   | local.bash   | tutorials
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/hello_world/hello_
↪world/0/stage/generate.sh
 systemd_default_target   | 8864cbc1 | script   | local.bash   | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/systemd/systemd_
↪default_target/2/stage/generate.sh
 selinux_disable          | 52b88227 | script   | local.bash   | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/selinux/selinux_
↪disable/0/stage/generate.sh

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

 name                     | id       | executor     | status   |   returncode |
↪testpath
--------------------------+----------+--------------+----------+--------------+-------
↪-------------------------------------------------------------------------------
↪---------------------------------
 environment_variables    | d0e51818 | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/environment/environment_
↪variables/0/stage/generate.sh
 pre_post_build_run       | 1e81254f | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/pre_post_build_run/pre_post_
↪build_run/0/stage/generate.sh
 _bin_sh_shell            | 85c5e433 | local.sh     | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/_bin_sh_shell/
↪2/stage/generate.sh
```

(continues on next page)

(continued from previous page)

```
 _bin_bash_shell          | 06ef100e | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/_bin_bash_
↪shell/2/stage/generate.sh
 bash_shell               | 13e306ff | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/bash_shell/2/
↪stage/generate.sh
 sh_shell                 | a018fabc | local.sh     | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/sh_shell/2/
↪stage/generate.sh
 shell_options            | a7a23ec8 | local.sh     | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/shell_options/
↪2/stage/generate.sh
 variables                | 3adfeb8b | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/vars/variables/0/stage/
↪generate.sh
 executable_arguments     | 5d670438 | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/passing_args/executable_
↪arguments/0/stage/generate.sh
 exit1_fail               | d405aea9 | local.sh     | FAIL     |            1 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_fail/0/
↪stage/generate.sh
 exit1_pass               | 992a08e0 | local.sh     | PASS     |            1 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_pass/0/
↪stage/generate.sh
 returncode_list_mismatch | 269abcb9 | local.sh     | FAIL     |            2 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_
↪list_mismatch/0/stage/generate.sh
 returncode_int_match     | 146a0269 | local.sh     | PASS     |          128 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_
↪int_match/0/stage/generate.sh
 run_only_platform_darwin | 1d86e162 | local.python | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.python/run_only_platform/run_
↪only_platform_darwin/0/stage/generate.sh
 circle_area              | 59d92815 | local.python | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.python/python-shell/circle_area/
↪2/stage/generate.sh
 unskipped                | 17706881 | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/skip_tests/unskipped/0/
↪stage/generate.sh
 sleep                    | eadfa0df | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/sleep/sleep/0/stage/
↪generate.sh
 vecadd_gnu               | 62df938a | local.bash   | FAIL     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/vecadd/vecadd_gnu/0/stage/
↪generate.sh
 bash_login_shebang       | a4196349 | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_login_shebang/
↪0/stage/generate.sh
 bash_nonlogin_shebang    | 72c038f0 | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_nonlogin_
↪shebang/0/stage/generate.sh
 hello_f                  | 34d5c6d8 | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_f/0/stage/
↪generate.sh
 hello_c                  | 05ed531f | local.bash   | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_c/0/stage/
↪generate.sh
```

(continues on next page)

(continued from previous page)

```
 hello_cplusplus          | 2543a4e8 | local.bash    | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_cplusplus/0/
↪stage/generate.sh
 cc_example               | 3143c39a | local.bash    | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cc_example/0/
↪stage/generate.sh
 fc_example               | b84feab0 | local.bash    | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/fc_example/0/
↪stage/generate.sh
 cxx_example              | 855fae37 | local.bash    | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cxx_example/0/
↪stage/generate.sh
 hello_world              | fc9f1058 | local.bash    | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/hello_world/hello_world/0/
↪stage/generate.sh
 systemd_default_target   | 8864cbc1 | local.bash    | FAIL     |            1 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/systemd/systemd_default_
↪target/2/stage/generate.sh
 selinux_disable          | 52b88227 | local.bash    | FAIL     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/selinux/selinux_disable/0/
↪stage/generate.sh

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

Executed 29 tests
Passed Tests: 24/29 Percentage: 82.759%
Failed Tests: 5/29 Percentage: 17.241%
```

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

```
$ buildtest build --tags compile --tags python

+-------------------------------+
| Stage: Discovering Buildspecs |
+-------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/python-hello.yml
/Users/siddiq90/Documents/buildtest/tutorials/python-shell.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/passing_args.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/pre_post_build_run.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/vecadd.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile                | validstate   | buildspec
---------------------------+--------------+-----------------------------------------
↪-------------------------------------
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/python-hello.yml
```

(continues on next page)

(continued from previous page)

```
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/python-shell.yml
 compiler-v1.0.schema.json | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/passing_args.yml
 compiler-v1.0.schema.json | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/pre_post_build_run.yml
 compiler-v1.0.schema.json | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/vecadd.yml
 compiler-v1.0.schema.json | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/gnu_hello.yml

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

 name                  | id       | type     | executor     | tags                ␣
↪| testpath
----------------------+----------+----------+--------------+---------------------
↪+-------------------------------------------------------------------------------
↪---------------------------------
 python_hello          | 48caf02e | script   | local.bash   | python              ␣
↪| /Users/siddiq90/Documents/buildtest/var/tests/local.bash/python-hello/python_
↪hello/0/stage/generate.sh
 circle_area           | 8a235c10 | script   | local.python | ['tutorials', 'python'] ␣
↪| /Users/siddiq90/Documents/buildtest/var/tests/local.python/python-shell/circle_
↪area/3/stage/generate.sh
 executable_arguments | 77c21de7 | compiler | local.bash   | ['tutorials', 'compile']␣
↪| /Users/siddiq90/Documents/buildtest/var/tests/local.bash/passing_args/executable_
↪arguments/1/stage/generate.sh
 pre_post_build_run    | d88a5039 | compiler | local.bash   | ['tutorials', 'compile']␣
↪| /Users/siddiq90/Documents/buildtest/var/tests/local.bash/pre_post_build_run/pre_
↪post_build_run/1/stage/generate.sh
 vecadd_gnu            | ce7bbe15 | compiler | local.bash   | ['tutorials', 'compile']␣
↪| /Users/siddiq90/Documents/buildtest/var/tests/local.bash/vecadd/vecadd_gnu/1/
↪stage/generate.sh
 hello_f               | e525904f | compiler | local.bash   | ['tutorials', 'compile']␣
↪| /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_f/1/
↪stage/generate.sh
 hello_c               | 83beacb7 | compiler | local.bash   | ['tutorials', 'compile']␣
↪| /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_c/1/
↪stage/generate.sh
 hello_cplusplus       | 6131d2d7 | compiler | local.bash   | ['tutorials', 'compile']␣
↪| /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_
↪cplusplus/1/stage/generate.sh
 cc_example            | 46f76cea | compiler | local.bash   | ['tutorials', 'compile']␣
↪| /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cc_example/1/
↪stage/generate.sh
 fc_example            | ca8b4485 | compiler | local.bash   | ['tutorials', 'compile']␣
↪| /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/fc_example/1/
↪stage/generate.sh
 cxx_example           | 3adf259c | compiler | local.bash   | ['tutorials', 'compile']␣
↪| /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cxx_example/1/
↪stage/generate.sh

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

(continues on next page)

(continued from previous page)

```
 name                  | id       | executor     | status   |   returncode | testpath
-----------------------+----------+--------------+----------+--------------+-----------
↪-------------------------------------------------------------------------------------
↪----------------------
 python_hello          | 48caf02e | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/python-hello/python_hello/0/stage/
↪generate.sh
 circle_area           | 8a235c10 | local.python | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.python/python-shell/circle_area/3/
↪stage/generate.sh
 executable_arguments | 77c21de7 | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/passing_args/executable_arguments/
↪1/stage/generate.sh
 pre_post_build_run   | d88a5039 | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/pre_post_build_run/pre_post_build_
↪run/1/stage/generate.sh
 vecadd_gnu            | ce7bbe15 | local.bash   | FAIL     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/vecadd/vecadd_gnu/1/stage/
↪generate.sh
 hello_f               | e525904f | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_f/1/stage/
↪generate.sh
 hello_c               | 83beacb7 | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_c/1/stage/
↪generate.sh
 hello_cplusplus       | 6131d2d7 | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_cplusplus/1/stage/
↪generate.sh
 cc_example            | 46f76cea | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cc_example/1/stage/
↪generate.sh
 fc_example            | ca8b4485 | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/fc_example/1/stage/
↪generate.sh
 cxx_example           | 3adf259c | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cxx_example/1/stage/
↪generate.sh

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

Executed 11 tests
Passed Tests: 10/11 Percentage: 90.909%
Failed Tests: 1/11 Percentage: 9.091%
```

When multiple tags are specified, we search each tag independently and if it is found in the buildspec cache we retrieve the test. To see a list of available tags in your buildspec cache see *Querying buildspec tags*.

Note: The `--tags` is used for discovering buildspecs and filtering tests during build phase. For example a buildspec file (`system.yml`) that contain three tests **hostname_check**, **timeout**, and **ping_test** are generally all run by default if you run as `buildtest build -b system.yml`, but if you specify `--tags` buildtest will exclude tests that don't have a matching tagname. It is possible `buildtest build --tags system` can discover buildspec file `system.yml` but only tests **timeout** and **ping_test** are built because they have a **system** tag while **hostname_check** is skipped because it's test doesn't have a **system** tag.

You can combine `--tags` with `--buildspec` and `--exclude` in a single command. buildtest will query tags and buildspecs independently and combine all discovered buildspecs, any duplicates are ignored and finally we apply the exclusion list to remove buildspecs.
In next example we combine all of these features together. This example builds all test with **python** tag, and build all buildspecs in directory - **tutorials/compilers** but we exclude **tutorials/compilers/vecadd.yml**.

```
$ buildtest build --tags python -b tutorials/compilers -x tutorials/compilers/vecadd.
↪yml

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/python-shell.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/passing_args.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/pre_post_build_run.yml
/Users/siddiq90/Documents/buildtest/tutorials/python-hello.yml

Excluded Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/compilers/vecadd.yml
[hello_f] test is skipped because it is not in tag filter list: ['python']
[hello_c] test is skipped because it is not in tag filter list: ['python']
[hello_cplusplus] test is skipped because it is not in tag filter list: ['python']
[cc_example] test is skipped because it is not in tag filter list: ['python']
[fc_example] test is skipped because it is not in tag filter list: ['python']
[cxx_example] test is skipped because it is not in tag filter list: ['python']
[executable_arguments] test is skipped because it is not in tag filter list: ['python
↪']
[pre_post_build_run] test is skipped because it is not in tag filter list: ['python']

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile                | validstate   | buildspec
---------------------------+--------------+-----------------------------------------
↪-------------------------------------
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/python-shell.yml
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/gnu_hello.yml
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/passing_args.yml
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/pre_post_build_run.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/python-hello.yml

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

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

(continues on next page)

(continued from previous page)

```
-------------+----------+--------+--------------+------------------------+----------
↪-------------------------------------------------------------------------------------
↪-----------
 circle_area  | 888d6562 | script | local.python | ['tutorials', 'python'] | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.python/python-shell/circle_area/4/
↪stage/generate.sh
 python_hello | bbc55590 | script | local.bash   | python                  | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/python-hello/python_hello/1/stage/
↪generate.sh

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

 name          | id       | executor     | status   |   returncode | testpath
-------------+----------+--------------+----------+--------------+--------------------
↪-------------------------------------------------------------------------------------
↪--
 circle_area  | 888d6562 | local.python | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.python/python-shell/circle_area/4/stage/
↪generate.sh
 python_hello | bbc55590 | local.bash   | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/python-hello/python_hello/1/stage/generate.
↪sh

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

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

5.4.5 Building by Executors

buildtest can build tests by executor name using the `--executor` option. If you to build all test associated to an executor such as `local.sh` you can run:

```
$ buildtest build --executor local.sh
```

buildtest will query buildspec cache for the executor name and retrieve a list of buildspecs with matching executor name. Later we process every buildspec and filter tests with executor name. In the first stage we retrieve the buildspec file which may contain one or more test and in second stage we process each test.
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 –executor is filtering by tests. This option behaves similar to tags, the **–executor** is used for discovering buildspecs and filtering tests with corresponding executor name.

In this example we run all tests that are associated to *local.sh* executor. Notice how buildtest skips tests that don't match executor **local.sh** even though they were discovered in buildspec file.

```
$ buildtest build --executor local.sh

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+
```

(continues on next page)

(continued from previous page)

```
Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
/Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml
[_bin_bash_shell] test is skipped because it is not in executor filter list: ['local.
↪sh']
[bash_shell] test is skipped because it is not in executor filter list: ['local.sh']

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile               | validstate   | buildspec
--------------------------+--------------+-----------------------------------------
↪----------------------
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/shell_examples.yml

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

 name                      | id       | type   | executor   | tags                  |␣
↪testpath
---------------------------+----------+--------+------------+-----------------------+--
↪-------------------------------------------------------------------------------------
↪-------------------------------
 exit1_fail                | 0a7c4951 | script | local.sh   | ['tutorials', 'fail'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_fail/1/
↪stage/generate.sh
 exit1_pass                | 943c3d32 | script | local.sh   | ['tutorials', 'pass'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_pass/1/
↪stage/generate.sh
 returncode_list_mismatch | e5905c73 | script | local.sh   | ['tutorials', 'fail'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_
↪list_mismatch/1/stage/generate.sh
 returncode_int_match      | ac11ac19 | script | local.sh   | ['tutorials', 'pass'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_
↪int_match/1/stage/generate.sh
 _bin_sh_shell             | 6ad1af21 | script | local.sh   | ['tutorials']         | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/_bin_sh_shell/
↪3/stage/generate.sh
 sh_shell                  | 26ec2ae6 | script | local.sh   | ['tutorials']         | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/sh_shell/3/
↪stage/generate.sh
 shell_options             | 9f7b7cd8 | script | local.sh   | ['tutorials']         | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/shell_options/
↪3/stage/generate.sh

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

(continues on next page)

(continued from previous page)

```
 name                      | id       | executor   | status   |   returncode | testpath
---------------------------+----------+------------+----------+--------------+---------
↪-------------------------------------------------------------------------------------
↪------------------------
 exit1_fail                | 0a7c4951 | local.sh   | FAIL     |            1 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_fail/1/stage/
↪generate.sh
 exit1_pass                | 943c3d32 | local.sh   | PASS     |            1 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_pass/1/stage/
↪generate.sh
 returncode_list_mismatch | e5905c73 | local.sh   | FAIL     |            2 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_list_
↪mismatch/1/stage/generate.sh
 returncode_int_match      | ac11ac19 | local.sh   | PASS     |          128 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_int_
↪match/1/stage/generate.sh
 _bin_sh_shell             | 6ad1af21 | local.sh   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/_bin_sh_shell/3/
↪stage/generate.sh
 sh_shell                  | 26ec2ae6 | local.sh   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/sh_shell/3/stage/
↪generate.sh
 shell_options             | 9f7b7cd8 | local.sh   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/shell_options/3/
↪stage/generate.sh

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

Executed 7 tests
Passed Tests: 5/7 Percentage: 71.429%
Failed Tests: 2/7 Percentage: 28.571%
```

We can append arguments to `--executor` to search for multiple executors by specifying `--executor <name1> --executor <name2>`. In next example we search all tests associated with `local.sh` and `local.bash` executor.

Note: If you specify multiple executors, buildtest will combine the executors into list, for example `--executor local.bash --executor local.sh` is converted into a list (executor filter) - `[local.bash, local.sh]`, and buildtest will skip any test whose `executor` field in testname doesn't belong to executor filter list are skipped.

```
$ buildtest build --executor local.sh --executor local.bash

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/selinux.yml
/Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
/Users/siddiq90/Documents/buildtest/general_tests/sched/slurm/sacctmgr.yml
```

(continues on next page)

(continued from previous page)

```
/Users/siddiq90/Documents/buildtest/tutorials/run_only_distro.yml
/Users/siddiq90/Documents/buildtest/general_tests/sched/lsf/bmgroups.yml
/Users/siddiq90/Documents/buildtest/tutorials/systemd.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/pre_post_build_run.yml
/Users/siddiq90/Documents/buildtest/general_tests/sched/lsf/bugroup.yml
/Users/siddiq90/Documents/buildtest/tutorials/hello_world.yml
/Users/siddiq90/Documents/buildtest/general_tests/sched/lsf/bhosts.yml
/Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml
/Users/siddiq90/Documents/buildtest/general_tests/sched/lsf/bqueues.yml
/Users/siddiq90/Documents/buildtest/tutorials/environment.yml
/Users/siddiq90/Documents/buildtest/tutorials/sleep.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/vecadd.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/passing_args.yml
/Users/siddiq90/Documents/buildtest/general_tests/sched/slurm/scontrol.yml
/Users/siddiq90/Documents/buildtest/general_tests/configuration/ssh_localhost.yml
/Users/siddiq90/Documents/buildtest/general_tests/sched/slurm/sinfo.yml
/Users/siddiq90/Documents/buildtest/general_tests/sched/lsf/lsinfo.yml
/Users/siddiq90/Documents/buildtest/general_tests/configuration/systemd-default-
↪target.yml
/Users/siddiq90/Documents/buildtest/tutorials/tags_example.yml
/Users/siddiq90/Documents/buildtest/tutorials/root_user.yml
/Users/siddiq90/Documents/buildtest/tutorials/vars.yml
/Users/siddiq90/Documents/buildtest/tutorials/shebang.yml
/Users/siddiq90/Documents/buildtest/general_tests/configuration/disk_usage.yml
/Users/siddiq90/Documents/buildtest/general_tests/sched/slurm/squeue.yml
/Users/siddiq90/Documents/buildtest/tutorials/python-hello.yml
/Users/siddiq90/Documents/buildtest/tutorials/skip_tests.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml
/Users/siddiq90/Documents/buildtest/general_tests/configuration/ulimits.yml
[show_accounts] test is skipped because ['run_only']['scheduler'] got value: slurm␣
↪but detected scheduler: [].
[show_users] test is skipped because ['run_only']['scheduler'] got value: slurm but␣
↪detected scheduler: [].
[show_qos] test is skipped because ['run_only']['scheduler'] got value: slurm but␣
↪detected scheduler: [].
[show_tres] test is skipped because ['run_only']['scheduler'] got value: slurm but␣
↪detected scheduler: [].
[show_host_groups] test is skipped because ['run_only']['scheduler'] got value: lsf␣
↪but detected scheduler: [].
[show_lsf_user_groups] test is skipped because ['run_only']['scheduler'] got value:␣
↪lsf but detected scheduler: [].
[display_lsf_hosts] test is skipped because ['run_only']['scheduler'] got value: lsf␣
↪but detected scheduler: [].
[display_hosts_format] test is skipped because ['run_only']['scheduler'] got value:␣
↪lsf but detected scheduler: [].
[bhosts_version] test is skipped because ['run_only']['scheduler'] got value: lsf but␣
↪detected scheduler: [].
[show_lsf_queues] test is skipped because ['run_only']['scheduler'] got value: lsf␣
↪but detected scheduler: [].
[show_lsf_queues_formatted] test is skipped because ['run_only']['scheduler'] got␣
↪value: lsf but detected scheduler: [].
[show_lsf_queues_current_user] test is skipped because ['run_only']['scheduler'] got␣
↪value: lsf but detected scheduler: [].
[slurm_config] test is skipped because ['run_only']['scheduler'] got value: slurm but␣
↪detected scheduler: [].
[show_partition] test is skipped because ['run_only']['scheduler'] got value: slurm␣
↪but detected scheduler: [].
```

(continues on next page)

(continued from previous page)

```
[ssh_localhost_remotecommand] test is skipped because ['run_only']['platform'] got␣
↪value: Linux but detected platform: Darwin.
[nodes_state_down] test is skipped because ['run_only']['scheduler'] got value: slurm␣
↪but detected scheduler: [].
[nodes_state_reboot] test is skipped because ['run_only']['scheduler'] got value:␣
↪slurm but detected scheduler: [].
[nodes_state_allocated] test is skipped because ['run_only']['scheduler'] got value:␣
↪slurm but detected scheduler: [].
[nodes_state_completing] test is skipped because ['run_only']['scheduler'] got value:␣
↪slurm but detected scheduler: [].
[nodes_state_idle] test is skipped because ['run_only']['scheduler'] got value: slurm␣
↪but detected scheduler: [].
[node_down_fail_list_reason] test is skipped because ['run_only']['scheduler'] got␣
↪value: slurm but detected scheduler: [].
[dead_nodes] test is skipped because ['run_only']['scheduler'] got value: slurm but␣
↪detected scheduler: [].
[get_partitions] test is skipped because ['run_only']['scheduler'] got value: slurm␣
↪but detected scheduler: [].
[sinfo_version] test is skipped because ['run_only']['scheduler'] got value: slurm␣
↪but detected scheduler: [].
[show_lsf_configuration] test is skipped because ['run_only']['scheduler'] got value:␣
↪lsf but detected scheduler: [].
[show_lsf_models] test is skipped because ['run_only']['scheduler'] got value: lsf␣
↪but detected scheduler: [].
[show_lsf_resources] test is skipped because ['run_only']['scheduler'] got value: lsf␣
↪but detected scheduler: [].
[lsf_version] test is skipped because ['run_only']['scheduler'] got value: lsf but␣
↪detected scheduler: [].
[run_only_as_root] test is skipped because ['run_only']['user'] got value: root but␣
↪detected user: siddiq90.
[current_user_queue] test is skipped because ['run_only']['scheduler'] got value:␣
↪slurm but detected scheduler: [].
[show_all_jobs] test is skipped because ['run_only']['scheduler'] got value: slurm␣
↪but detected scheduler: [].
[skip] test is skipped.

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile                | validstate   | buildspec
---------------------------+--------------+-----------------------------------------
↪-------------------------------------------------
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/selinux.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪general_tests/sched/slurm/sacctmgr.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/run_only_distro.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪general_tests/sched/lsf/bmgroups.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/systemd.yml
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/pre_post_build_run.yml
```

(continues on next page)

(continued from previous page)

```
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪general_tests/sched/lsf/bugroup.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/hello_world.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪general_tests/sched/lsf/bhosts.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/shell_examples.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪general_tests/sched/lsf/bqueues.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/environment.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/sleep.yml
 compiler-v1.0.schema.json | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/vecadd.yml
 compiler-v1.0.schema.json | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/passing_args.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪general_tests/sched/slurm/scontrol.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪general_tests/configuration/ssh_localhost.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪general_tests/sched/slurm/sinfo.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪general_tests/sched/lsf/lsinfo.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪general_tests/configuration/systemd-default-target.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/tags_example.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/root_user.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/vars.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/shebang.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪general_tests/configuration/disk_usage.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪general_tests/sched/slurm/squeue.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/python-hello.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/skip_tests.yml
 compiler-v1.0.schema.json | True          | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/gnu_hello.yml
 script-v1.0.schema.json    | True          | /Users/siddiq90/Documents/buildtest/
↪general_tests/configuration/ulimits.yml

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

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

(continues on next page)

(continued from previous page)

```
selinux_disable          | 88fb1b1c | script   | local.bash | ['tutorials']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/selinux/selinux_
↪disable/1/stage/generate.sh
exit1_fail               | b1b25a16 | script   | local.sh   | ['tutorials', 'fail']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_
↪fail/2/stage/generate.sh
exit1_pass               | 365fdd6e | script   | local.sh   | ['tutorials', 'pass']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_
↪pass/2/stage/generate.sh
returncode_list_mismatch | aeb6f626 | script   | local.sh   | ['tutorials', 'fail']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/
↪returncode_list_mismatch/2/stage/generate.sh
returncode_int_match     | d817c4fd | script   | local.sh   | ['tutorials', 'pass']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/
↪returncode_int_match/2/stage/generate.sh
run_only_macos_distro    | 87c4ddbf | script   | local.bash |
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/run_only_distro/run_
↪only_macos_distro/0/stage/generate.sh
systemd_default_target   | 6fc3c7b4 | script   | local.bash | ['tutorials']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/systemd/systemd_
↪default_target/3/stage/generate.sh
pre_post_build_run       | 878b19fc | compiler | local.bash | ['tutorials', 'compile
↪']  | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/pre_post_build_run/
↪pre_post_build_run/2/stage/generate.sh
hello_world              | 87211773 | script   | local.bash | tutorials
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/hello_world/hello_
↪world/1/stage/generate.sh
_bin_sh_shell            | f16e1275 | script   | local.sh   | ['tutorials']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/_bin_sh_
↪shell/4/stage/generate.sh
_bin_bash_shell          | 5786ac8b | script   | local.bash | ['tutorials']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/_bin_
↪bash_shell/3/stage/generate.sh
bash_shell               | ad7e4f41 | script   | local.bash | ['tutorials']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/bash_
↪shell/3/stage/generate.sh
sh_shell                 | b5e23cb1 | script   | local.sh   | ['tutorials']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/sh_shell/
↪4/stage/generate.sh
shell_options            | 265177ba | script   | local.sh   | ['tutorials']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/shell_
↪options/4/stage/generate.sh
environment_variables    | 72827962 | script   | local.bash | ['tutorials']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/environment/
↪environment_variables/1/stage/generate.sh
sleep                    | 1a04b18d | script   | local.bash | ['tutorials']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/sleep/sleep/1/stage/
↪generate.sh
vecadd_gnu               | 2f7420a3 | compiler | local.bash | ['tutorials', 'compile
↪']  | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/vecadd/vecadd_gnu/2/
↪stage/generate.sh
executable_arguments     | 03f1c6af | compiler | local.bash | ['tutorials', 'compile
↪']  | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/passing_args/
↪executable_arguments/2/stage/generate.sh
systemd_default_target   | cc2fdeab | script   | local.bash | ['system']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/systemd-default-
↪target/systemd_default_target/0/stage/generate.sh
```

(continues on next page)

(continued from previous page)

```
string_tag                | 66f45e29 | script   | local.bash | network
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/tags_example/string_
↪tag/0/stage/generate.sh
list_of_strings_tags      | 371520d5 | script   | local.bash | ['network', 'ping']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/tags_example/list_of_
↪strings_tags/0/stage/generate.sh
variables                 | b3c8fedf | script   | local.bash | ['tutorials']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/vars/variables/1/
↪stage/generate.sh
bash_login_shebang        | d54ed2f7 | script   | local.bash | tutorials
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_login_
↪shebang/1/stage/generate.sh
bash_nonlogin_shebang     | ae73cee9 | script   | local.bash | tutorials
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_nonlogin_
↪shebang/1/stage/generate.sh
root_disk_usage           | 9771523d | script   | local.bash | ['filesystem', 'storage
↪'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/disk_usage/root_disk_
↪usage/0/stage/generate.sh
python_hello              | adc8633f | script   | local.bash | python
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/python-hello/python_
↪hello/2/stage/generate.sh
unskipped                 | 33ea31ab | script   | local.bash | ['tutorials']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/skip_tests/unskipped/
↪1/stage/generate.sh
hello_f                   | 590ad365 | compiler | local.bash | ['tutorials', 'compile
↪']  | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_f/2/
↪stage/generate.sh
hello_c                   | cc592e31 | compiler | local.bash | ['tutorials', 'compile
↪']  | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_c/2/
↪stage/generate.sh
hello_cplusplus           | 8655c367 | compiler | local.bash | ['tutorials', 'compile
↪']  | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_
↪cplusplus/2/stage/generate.sh
cc_example                | 1edbc832 | compiler | local.bash | ['tutorials', 'compile
↪']  | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cc_example/
↪2/stage/generate.sh
fc_example                | 3e112e84 | compiler | local.bash | ['tutorials', 'compile
↪']  | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/fc_example/
↪2/stage/generate.sh
cxx_example               | a8eac662 | compiler | local.bash | ['tutorials', 'compile
↪']  | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cxx_
↪example/2/stage/generate.sh
ulimit_filelock           | b86d4a0e | script   | local.bash | ['system']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/ulimits/ulimit_
↪filelock/0/stage/generate.sh
ulimit_cputime            | b021f45d | script   | local.bash | ['system']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/ulimits/ulimit_
↪cputime/0/stage/generate.sh
ulimit_stacksize          | 178fd98a | script   | local.bash | ['system']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/ulimits/ulimit_
↪stacksize/0/stage/generate.sh
ulimit_vmsize             | aaea044c | script   | local.bash | ['system']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/ulimits/ulimit_vmsize/
↪0/stage/generate.sh
ulimit_filedescriptor     | f9968865 | script   | local.bash | ['system']
↪   | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/ulimits/ulimit_
↪filedescriptor/0/stage/generate.sh
```

(continues on next page)

(continued from previous page)

```
 ulimit_max_user_process  | f6787b9f | script   | local.bash | ['system']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/ulimits/ulimit_max_
↪user_process/0/stage/generate.sh

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

 name                       | id       | executor   | status   |   returncode | testpath
----------------------------+----------+------------+----------+--------------+---------
↪-------------------------------------------------------------------------------------
↪--------------------------------
 selinux_disable            | 88fb1b1c | local.bash | FAIL     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/selinux/selinux_disable/1/stage/
↪generate.sh
 exit1_fail                 | b1b25a16 | local.sh   | FAIL     |            1 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_fail/2/stage/
↪generate.sh
 exit1_pass                 | 365fdd6e | local.sh   | PASS     |            1 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_pass/2/stage/
↪generate.sh
 returncode_list_mismatch | aeb6f626 | local.sh   | FAIL     |            2 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_list_
↪mismatch/2/stage/generate.sh
 returncode_int_match       | d817c4fd | local.sh   | PASS     |          128 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_int_
↪match/2/stage/generate.sh
 run_only_macos_distro      | 87c4ddbf | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/run_only_distro/run_only_macos_
↪distro/0/stage/generate.sh
 systemd_default_target     | 6fc3c7b4 | local.bash | FAIL     |            1 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/systemd/systemd_default_target/3/
↪stage/generate.sh
 pre_post_build_run         | 878b19fc | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/pre_post_build_run/pre_post_build_
↪run/2/stage/generate.sh
 hello_world                | 87211773 | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/hello_world/hello_world/1/stage/
↪generate.sh
 _bin_sh_shell              | f16e1275 | local.sh   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/_bin_sh_shell/4/
↪stage/generate.sh
 _bin_bash_shell            | 5786ac8b | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/_bin_bash_shell/3/
↪stage/generate.sh
 bash_shell                 | ad7e4f41 | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/bash_shell/3/stage/
↪generate.sh
 sh_shell                   | b5e23cb1 | local.sh   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/sh_shell/4/stage/
↪generate.sh
 shell_options              | 265177ba | local.sh   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/shell_options/4/
↪stage/generate.sh
 environment_variables      | 72827962 | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/environment/environment_variables/
↪1/stage/generate.sh
```

(continues on next page)

(continued from previous page)

```
 sleep                    | 1a04b18d | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/sleep/sleep/1/stage/generate.sh
 vecadd_gnu               | 2f7420a3 | local.bash | FAIL     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/vecadd/vecadd_gnu/2/stage/
↪generate.sh
 executable_arguments     | 03f1c6af | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/passing_args/executable_arguments/
↪2/stage/generate.sh
 systemd_default_target   | cc2fdeab | local.bash | FAIL     |            1 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/systemd-default-target/systemd_
↪default_target/0/stage/generate.sh
 string_tag               | 66f45e29 | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/tags_example/string_tag/0/stage/
↪generate.sh
 list_of_strings_tags     | 371520d5 | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/tags_example/list_of_strings_tags/
↪0/stage/generate.sh
 variables                | b3c8fedf | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/vars/variables/1/stage/generate.sh
 bash_login_shebang       | d54ed2f7 | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_login_shebang/1/
↪stage/generate.sh
 bash_nonlogin_shebang    | ae73cee9 | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_nonlogin_shebang/1/
↪stage/generate.sh
 root_disk_usage          | 9771523d | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/disk_usage/root_disk_usage/0/
↪stage/generate.sh
 python_hello             | adc8633f | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/python-hello/python_hello/2/stage/
↪generate.sh
 unskipped                | 33ea31ab | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/skip_tests/unskipped/1/stage/
↪generate.sh
 hello_f                  | 590ad365 | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_f/2/stage/
↪generate.sh
 hello_c                  | cc592e31 | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_c/2/stage/
↪generate.sh
 hello_cplusplus          | 8655c367 | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_cplusplus/2/stage/
↪generate.sh
 cc_example               | 1edbc832 | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cc_example/2/stage/
↪generate.sh
 fc_example               | 3e112e84 | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/fc_example/2/stage/
↪generate.sh
 cxx_example              | a8eac662 | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cxx_example/2/stage/
↪generate.sh
 ulimit_filelock          | b86d4a0e | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/ulimits/ulimit_filelock/0/stage/
↪generate.sh
 ulimit_cputime           | b021f45d | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/ulimits/ulimit_cputime/0/stage/
↪generate.sh
```

(continues on next page)

(continued from previous page)

```
 ulimit_stacksize         | 178fd98a | local.bash | FAIL     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/ulimits/ulimit_stacksize/0/stage/
↪generate.sh
 ulimit_vmsize            | aaea044c | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/ulimits/ulimit_vmsize/0/stage/
↪generate.sh
 ulimit_filedescriptor    | f9968865 | local.bash | FAIL     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/ulimits/ulimit_filedescriptor/0/
↪stage/generate.sh
 ulimit_max_user_process  | f6787b9f | local.bash | FAIL     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/ulimits/ulimit_max_user_process/0/
↪stage/generate.sh

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

Executed 39 tests
Passed Tests: 30/39 Percentage: 76.923%
Failed Tests: 9/39 Percentage: 23.077%
```

5.4.6 Control builds by Stages

You can control behavior of `buildtest build` command to stop at certain point using `--stage` option. This takes two values `parse` or `build`, which will stop buildtest after parsing buildspecs or building the test content. If you want to know your buildspecs are valid you can use `--stage=parse` to stop after parsing the buildspec. Shown below is an example build where we stop after parse stage.

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

+-------------------------------+
| Stage: Discovering Buildspecs |
+-------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/systemd.yml

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile              | validstate   | buildspec
-------------------------+--------------+-------------------------------------------
↪--------------
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/systemd.yml
```

Likewise, if you want to troubleshoot your test script without running them you can use `--stage=build` which will stop after building your test script. This can be extremely useful when writing your buildspecs and not having to run your tests. In this next example, we stop our after the build stage using `--stage=build`.

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

+-------------------------------+
```

(continues on next page)

(continued from previous page)

```
| Stage: Discovering Buildspecs |
+-------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/systemd.yml

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile               | validstate   | buildspec
--------------------------+--------------+-------------------------------------------
↪--------------
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/systemd.yml

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

 name                    | id       | type   | executor   | tags          | testpath
-------------------------+----------+--------+------------+---------------+------------
↪-------------------------------------------------------------------------------------
↪-------------
 systemd_default_target | f1c076a7 | script | local.bash | ['tutorials'] | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/systemd/systemd_default_target/4/
↪stage/generate.sh
```

Invalid Buildspecs

buildtest will skip any buildspecs 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

+-------------------------------+
| Stage: Discovering Buildspecs |
+-------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/invalid_buildspec_section.yml



Error Messages from Stage: Parse
_____________________________________________________________________________
Skipping /Users/siddiq90/Documents/buildtest/tutorials/invalid_buildspec_section.yml␣
↪since it failed to validate
No buildspecs to process because there are no valid buildspecs
```

buildtest may skip tests from running if buildspec specifies an invalid executor name since buildtest needs to know this in order to delegate test to Executor class responsible for running the test. Here is an example where test failed to run since we provided invalid executor.

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

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/invalid_executor.yml



Error Messages from Stage: Parse
_____________________________________________________________________________
Skipping /Users/siddiq90/Documents/buildtest/tutorials/invalid_executor.yml since it␣
↪failed to validate
No buildspecs to process because there are no valid buildspecs
```

5.4.7 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 buildspecs 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

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/python-shell.yml

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile              | validstate   | buildspec
-------------------------+--------------+-------------------------------------------
↪-------------------
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/python-shell.yml

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

 name        | id       | type   | executor     | tags                     | testpath
-------------+----------+--------+--------------+--------------------------+-----------
↪-------------------------------------------------------------------------------------
↪----------
 circle_area | d20fd0ea | script | local.python | ['tutorials', 'python'] | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.python/python-shell/circle_area/5/
↪stage/generate.sh
```

(continues on next page)

(continued from previous page)

```
 circle_area | 6de553e7 | script | local.python | ['tutorials', 'python'] | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.python/python-shell/circle_area/6/
↪stage/generate.sh
 circle_area | 0fd67587 | script | local.python | ['tutorials', 'python'] | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.python/python-shell/circle_area/7/
↪stage/generate.sh

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

 name        | id       | executor     | status   |   returncode | testpath
-------------+----------+--------------+----------+--------------+--------------------
↪-----------------------------------------------------------------------------------
↪-
 circle_area | d20fd0ea | local.python | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.python/python-shell/circle_area/5/stage/
↪generate.sh
 circle_area | 6de553e7 | local.python | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.python/python-shell/circle_area/6/stage/
↪generate.sh
 circle_area | 0fd67587 | local.python | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.python/python-shell/circle_area/7/stage/
↪generate.sh

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

Executed 3 tests
Passed Tests: 3/3 Percentage: 100.000%
Failed Tests: 0/3 Percentage: 0.000%
```

The rebuild works with all options including: `--buildspec`, `--exclude`, `--tags` and `--executors`.
In the next example we rebuild tests by discovering all tags that contain **fail**.

```
$ buildtest build --tags fail --rebuild=2

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
[exit1_pass] test is skipped because it is not in tag filter list: ['fail']
[returncode_int_match] test is skipped because it is not in tag filter list: ['fail']
[exit1_pass] test is skipped because it is not in tag filter list: ['fail']
[returncode_int_match] test is skipped because it is not in tag filter list: ['fail']

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile              | validstate   | buildspec
-------------------------+--------------+-------------------------------------------
↪-----------------------
```

(continues on next page)

(continued from previous page)

```
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml

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

 name                      | id       | type   | executor   | tags                  |␣
↪testpath
---------------------------+----------+--------+------------+-----------------------+--
↪-------------------------------------------------------------------------------------
↪--------------------------------
 exit1_fail                | 46a14403 | script | local.sh   | ['tutorials', 'fail'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_fail/3/
↪stage/generate.sh
 returncode_list_mismatch | 78981e2b | script | local.sh   | ['tutorials', 'fail'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_
↪list_mismatch/3/stage/generate.sh
 exit1_fail                | f3a827e6 | script | local.sh   | ['tutorials', 'fail'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_fail/4/
↪stage/generate.sh
 returncode_list_mismatch | 1887648a | script | local.sh   | ['tutorials', 'fail'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_
↪list_mismatch/4/stage/generate.sh

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

 name                      | id       | executor   | status   |   returncode | testpath
---------------------------+----------+------------+----------+--------------+---------
↪-------------------------------------------------------------------------------------
↪-------------------------
 exit1_fail                | 46a14403 | local.sh   | FAIL     |            1 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_fail/3/stage/
↪generate.sh
 returncode_list_mismatch | 78981e2b | local.sh   | FAIL     |            2 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_list_
↪mismatch/3/stage/generate.sh
 exit1_fail                | f3a827e6 | local.sh   | FAIL     |            1 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_fail/4/stage/
↪generate.sh
 returncode_list_mismatch | 1887648a | local.sh   | FAIL     |            2 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_list_
↪mismatch/4/stage/generate.sh

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

Executed 4 tests
Passed Tests: 0/4 Percentage: 0.000%
Failed Tests: 4/4 Percentage: 100.000%
```

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 degredation due to high workload.
If you try to exceed this bound you will get an error such as:

```
$ buildtest build -b tutorials/pass_returncode.yml --rebuild 51
usage: buildtest [options] [COMMANDS] build [-h] [-b BUILDSPEC] [-x EXCLUDE] [--tags␣
↪TAGS] [-e EXECUTOR]
                                            [-s {parse,build}] [-t TESTDIR] [--
↪rebuild REBUILD] [--settings SETTINGS]
buildtest [options] [COMMANDS] build: error: argument --rebuild: 51 must be a␣
↪positive number between [1-50]
```

5.4.8 Buildspecs Interface

buildtest is able to find and validate all buildspecs in your repos. The command `buildtest buildspec` comes with the following options.

```
$ buildtest buildspec --help
usage: buildtest [options] [COMMANDS] buildspec [-h] {find,view,edit} ...

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

subcommands:
  Commands options for Buildspecs

  {find,view,edit}
    find            find all buildspecs
    view            view a buildspec
    edit            edit a buildspec
```

Finding Buildspecs

To find all buildspecs run `buildtest buildspec find` which will discover all buildspecs in all repos by recursively finding all *.yml* extensions.

```
$ buildtest buildspec find
Found 36 buildspecs
Validated 5/36 buildspecs
Validated 10/36 buildspecs
Validated 15/36 buildspecs
Validated 20/36 buildspecs
Validated 25/36 buildspecs
Validated 30/36 buildspecs
Validated 35/36 buildspecs
Validated 36/36 buildspecs

Detected 3 invalid buildspecs

Writing invalid buildspecs to file: /Users/siddiq90/Documents/buildtest/var/buildspec.
↪error



+------------------------------+----------+--------------+----------------------------
↪+-------------------------------------------------------------------+
| Name                         | Type     | Executor     | Tags                       ␣
↪| Description                                                       |
+==============================+==========+==============+============================+==============
| systemd_default_target       | script   | local.bash   | ['tutorials']              ␣
↪| check if default target is multi-user.target                      |
```

(continues on next page)

(continued from previous page)

```
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
| run_only_macos_distro       | script   | local.bash  | []                          ␣
↪| Run test only if linux distro is darwin.                    |
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
| _bin_sh_shell               | script   | local.sh    | ['tutorials']               ␣
↪| /bin/sh shell example                                       |
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
| _bin_bash_shell             | script   | local.bash  | ['tutorials']               ␣
↪| /bin/bash shell example                                     |
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
| bash_shell                  | script   | local.bash  | ['tutorials']               ␣
↪| bash shell example                                          |
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
| sh_shell                    | script   | local.sh    | ['tutorials']               ␣
↪| sh shell example                                            |
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
| shell_options               | script   | local.sh    | ['tutorials']               ␣
↪| shell options                                               |
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
| environment_variables       | script   | local.bash  | ['tutorials']               ␣
↪| Declare environment variables                               |
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
| python_hello                | script   | local.bash  | python                      ␣
↪| Hello World python                                          |
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
| variables                   | script   | local.bash  | ['tutorials']               ␣
↪| Declare shell variables                                     |
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
| selinux_disable             | script   | local.bash  | ['tutorials']               ␣
↪| Check if SELinux is Disabled                                |
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
| bash_login_shebang          | script   | local.bash  | tutorials                   ␣
↪| customize shebang line with bash login shell                |
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
| bash_nonlogin_shebang       | script   | local.bash  | tutorials                   ␣
↪| customize shebang line with default bash (nonlogin) shell   |
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
| exit1_fail                  | script   | local.sh    | ['tutorials', 'fail']       ␣
↪| exit 1 by default is FAIL                                   |
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
| exit1_pass                  | script   | local.sh    | ['tutorials', 'pass']       ␣
↪| report exit 1 as PASS                                       |
+-----------------------------+----------+-------------+-----------------------------
↪+-------------------------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
...
```

buildtest will validate each buildspec file with the appropriate schema type. buildspecs that pass validation will be displayed on screen. buildtest will report all invalid buildspecs in a text file for you to review.
buildtest will cache the results in **var/buildspec-cache.json** so subsequent runs to `buildtest buildspec find` will be much faster because it is read from cache. If you make changes to buildspec you may want to rebuild the buildspec cache then run:

```
$ buildtest buildspec find --clear
```

Shown below is a list of options for `buildtest buildspec find` command.

```
$ buildtest buildspec find --help
usage: buildtest [options] [COMMANDS] buildspec find [-h] [-c] [-t] [-bf] [-le] [--
↪filter FILTER] [--helpfilter]

optional arguments:
  -h, --help            show this help message and exit
  -c, --clear           Clear buildspec cache and find all buildspecs again
  -t, --tags            List all available tags
  -bf, --buildspec-files
                        Get all buildspec files from cache
  -le, --list-executors
                        get all unique executors from buildspecs
  --filter FILTER       Filter buildspec cache with filter fields in format --filter␣
↪key1=val1,key2=val2
  --helpfilter          Show Filter fields for --filter option for querying buildspec␣
↪cache
```

If you want to find all buildspec files in cache run `buildtest buildspec find --buildspec-files`

```
$ buildtest buildspec find --buildspec-files
+-------------------------------------------------------------------------------------
↪-------+
| buildspecs                                                                          ␣
↪       |
+=====================================================================================================+
| /Users/siddiq90/Documents/buildtest/tutorials/systemd.yml                           ␣
↪       |
+-------------------------------------------------------------------------------------
↪-------+
| /Users/siddiq90/Documents/buildtest/tutorials/run_only_distro.yml                   ␣
↪       |
+-------------------------------------------------------------------------------------
↪-------+
| /Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml                    ␣
↪       |
+-------------------------------------------------------------------------------------
↪-------+
| /Users/siddiq90/Documents/buildtest/tutorials/environment.yml                       ␣
↪       |
+-------------------------------------------------------------------------------------
↪-------+
| /Users/siddiq90/Documents/buildtest/tutorials/python-hello.yml                      ␣
↪       |
+-------------------------------------------------------------------------------------
↪-------+
| /Users/siddiq90/Documents/buildtest/tutorials/vars.yml                              ␣
↪       |
```

(continues on next page)

(continued from previous page)

```
+-------------------------------------------------------------------------------------
↪-------+
| /Users/siddiq90/Documents/buildtest/tutorials/selinux.yml                          ␣
↪       |
+-------------------------------------------------------------------------------------
↪-------+
| /Users/siddiq90/Documents/buildtest/tutorials/shebang.yml                          ␣
↪       |
+-------------------------------------------------------------------------------------
↪-------+
| /Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml                  ␣
↪       |
+-------------------------------------------------------------------------------------
↪-------+
| /Users/siddiq90/Documents/buildtest/tutorials/hello_world.yml                      ␣
↪       |
+-------------------------------------------------------------------------------------
↪-------+
| /Users/siddiq90/Documents/buildtest/tutorials/root_user.yml                        ␣
↪       |
+-------------------------------------------------------------------------------------
↪-------+
| /Users/siddiq90/Documents/buildtest/tutorials/tags_example.yml                     ␣
↪       |
+-------------------------------------------------------------------------------------
↪-------+
| /Users/siddiq90/Documents/buildtest/tutorials/run_only_platform.yml                ␣
↪       |
+-------------------------------------------------------------------------------------
↪-------+
...
```

Filtering buildspec

You can filter buildspec cache using the the `--filter` option. 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 accepts 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 | local.sh   | ['tutorials', 'fail'] | exit 1 by␣
↪default is FAIL                            |
```

(continues on next page)

(continued from previous page)

```
+-------------------------+--------+-----------+----------------------+------------
↪----------------------------------------+
| returncode_list_mismatch | script | local.sh   | ['tutorials', 'fail'] | exit 2␣
↪failed since it failed to match returncode 1 |
+-------------------------+--------+-----------+----------------------+------------
↪----------------------------------------+
```

In addition, we can query buildspecs by schema type, in next example we query all tests using the *script* schema

```
$ buildtest buildspec find --filter type=script
+------------------------------+--------+-------------+---------------------------+--
↪-------------------------------------------------------------+
| Name                         | Type   | Executor    | Tags                      |␣
↪Description                                                  |
+==============================+========+=============+===========================+==========
| systemd_default_target       | script | local.bash  | ['tutorials']             |␣
↪check if default target is multi-user.target                 |
+------------------------------+--------+-------------+---------------------------+--
↪-------------------------------------------------------------+
| run_only_macos_distro        | script | local.bash  | []                        |␣
↪Run test only if linux distro is darwin.                     |
+------------------------------+--------+-------------+---------------------------+--
↪-------------------------------------------------------------+
| _bin_sh_shell                | script | local.sh    | ['tutorials']             | /
↪bin/sh shell example                                         |
+------------------------------+--------+-------------+---------------------------+--
↪-------------------------------------------------------------+
| _bin_bash_shell              | script | local.bash  | ['tutorials']             | /
↪bin/bash shell example                                       |
+------------------------------+--------+-------------+---------------------------+--
↪-------------------------------------------------------------+
| bash_shell                   | script | local.bash  | ['tutorials']             |␣
↪bash shell example                                           |
+------------------------------+--------+-------------+---------------------------+--
↪-------------------------------------------------------------+
| sh_shell                     | script | local.sh    | ['tutorials']             |␣
↪sh shell example                                             |
+------------------------------+--------+-------------+---------------------------+--
↪-------------------------------------------------------------+
| shell_options                | script | local.sh    | ['tutorials']             |␣
↪shell options                                                |
+------------------------------+--------+-------------+---------------------------+--
↪-------------------------------------------------------------+
| environment_variables        | script | local.bash  | ['tutorials']             |␣
↪Declare environment variables                                |
+------------------------------+--------+-------------+---------------------------+--
↪-------------------------------------------------------------+
| python_hello                 | script | local.bash  | python                    |␣
↪Hello World python                                           |
+------------------------------+--------+-------------+---------------------------+--
↪-------------------------------------------------------------+
| variables                    | script | local.bash  | ['tutorials']             |␣
↪Declare shell variables                                      |
+------------------------------+--------+-------------+---------------------------+--
↪-------------------------------------------------------------+
| selinux_disable              | script | local.bash  | ['tutorials']             |␣
↪Check if SELinux is Disabled                                 |
```

(continues on next page)

(continued from previous page)

```
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
| bash_login_shebang          | script | local.bash  | tutorials                   |␣
↪customize shebang line with bash login shell                |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
| bash_nonlogin_shebang       | script | local.bash  | tutorials                   |␣
↪customize shebang line with default bash (nonlogin) shell   |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
| exit1_fail                  | script | local.sh    | ['tutorials', 'fail']       |␣
↪exit 1 by default is FAIL                                   |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
| exit1_pass                  | script | local.sh    | ['tutorials', 'pass']       |␣
↪report exit 1 as PASS                                       |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
| returncode_list_mismatch    | script | local.sh    | ['tutorials', 'fail']       |␣
↪exit 2 failed since it failed to match returncode 1         |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
| returncode_int_match        | script | local.sh    | ['tutorials', 'pass']       |␣
↪exit 128 matches returncode 128                             |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
| hello_world                 | script | local.bash  | tutorials                   |␣
↪hello world example                                         |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
| run_only_as_root            | script | local.bash  | ['tutorials']               |␣
↪This test will only run if current user is root             |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
| string_tag                  | script | local.bash  | network                     |␣
↪tags can be a string                                        |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
| list_of_strings_tags        | script | local.bash  | ['network', 'ping']         |␣
↪tags can be a list of strings                               |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
| run_only_platform_darwin    | script | local.python | ['tutorials']              |␣
↪This test will only run if target platform is Darwin        |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
| run_only_platform_linux     | script | local.python | ['tutorials']              |␣
↪This test will only run if target platform is Linux         |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
| circle_area                 | script | local.python | ['tutorials', 'python']    |␣
↪Calculate circle of area given a radius                     |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
| skip                        | script | local.bash  | ['tutorials']               | ␣
↪                                                            |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| unskipped                      | script | local.bash    | ['tutorials']              | ␣
↪                                                                    |
+------------------------------+--------+--------------+--------------------------+--
↪-----------------------------------------------------------------+
| sleep                          | script | local.bash    | ['tutorials']              |␣
↪sleep 2 seconds                                                     |
+------------------------------+--------+--------------+--------------------------+--
↪-----------------------------------------------------------------+
| root_disk_usage                | script | local.bash    | ['filesystem', 'storage'] |␣
↪Check root disk usage and report if it exceeds threshold            |
+------------------------------+--------+--------------+--------------------------+--
↪-----------------------------------------------------------------+
| ssh_localhost_remotecommand    | script | local.bash    | ['ssh']                    |␣
↪Test if ssh on localhost works and if we can run remote command.    |
+------------------------------+--------+--------------+--------------------------+--
↪-----------------------------------------------------------------+
| systemd_default_target         | script | local.bash    | ['system']                 |␣
↪check if default target is multi-user.target                        |
+------------------------------+--------+--------------+--------------------------+--
↪-----------------------------------------------------------------+
| ulimit_filelock                | script | local.bash    | ['system']                 |␣
↪Check if file lock is set to unlimited in ulimits                   |
+------------------------------+--------+--------------+--------------------------+--
↪-----------------------------------------------------------------+
| ulimit_cputime                 | script | local.bash    | ['system']                 |␣
↪Check if cputime is set to unlimited in ulimits                     |
+------------------------------+--------+--------------+--------------------------+--
↪-----------------------------------------------------------------+
| ulimit_stacksize               | script | local.bash    | ['system']                 |␣
↪Check if stack size is set to unlimited in ulimits                  |
+------------------------------+--------+--------------+--------------------------+--
↪-----------------------------------------------------------------+
| ulimit_vmsize                  | script | local.bash    | ['system']                 |␣
↪Check virtual memory size and check if its set to unlimited         |
+------------------------------+--------+--------------+--------------------------+--
↪-----------------------------------------------------------------+
| ulimit_filedescriptor          | script | local.bash    | ['system']                 |␣
↪Check if open file descriptors limit is set to 4096                 |
+------------------------------+--------+--------------+--------------------------+--
↪-----------------------------------------------------------------+
| ulimit_max_user_process        | script | local.bash    | ['system']                 |␣
↪Check max number of user process limit is set to 2048               |
+------------------------------+--------+--------------+--------------------------+--
↪-----------------------------------------------------------------+
| show_accounts                  | script | local.bash    | ['slurm']                  |␣
↪run sacctmgr list accounts                                          |
+------------------------------+--------+--------------+--------------------------+--
↪-----------------------------------------------------------------+
| show_users                     | script | local.bash    | ['slurm']                  |␣
↪run sacctmgr list users                                             |
+------------------------------+--------+--------------+--------------------------+--
↪-----------------------------------------------------------------+
| show_qos                       | script | local.bash    | ['slurm']                  |␣
↪run sacctmgr list qos                                               |
+------------------------------+--------+--------------+--------------------------+--
↪-----------------------------------------------------------------+
| show_tres                      | script | local.bash    | ['slurm']                  |␣
↪run sacctmgr list tres                                              |
```

(continues on next page)

(continued from previous page)

```
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
| slurm_config                | script | local.bash  | ['slurm']                   |␣
↪run scontrol show config                                         |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
| show_partition              | script | local.bash  | ['slurm']                   |␣
↪run scontrol show partition                                      |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
| current_user_queue          | script | local.bash  | ['slurm']                   |␣
↪show all current pending jobs for current user (squeue -u $USER) |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
| show_all_jobs               | script | local.bash  | ['slurm']                   |␣
↪show all pending + running jobs (squeue -a)                      |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
| nodes_state_down            | script | local.bash  | ['slurm']                   |␣
↪Show nodes in DOWN state                                         |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
| nodes_state_reboot          | script | local.bash  | ['slurm']                   |␣
↪Show nodes in REBOOT state                                       |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
| nodes_state_allocated       | script | local.bash  | ['slurm']                   |␣
↪Show nodes in ALLOCATED state                                    |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
| nodes_state_completing      | script | local.bash  | ['slurm']                   |␣
↪Show nodes in COMPLETING state                                   |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
| nodes_state_idle            | script | local.bash  | ['slurm']                   |␣
↪Show nodes in IDLE state                                         |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
| node_down_fail_list_reason  | script | local.bash  | ['slurm']                   |␣
↪Show nodes DOWN, DRAINED, FAIL or FAILING and list reason        |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
| dead_nodes                  | script | local.bash  | ['slurm']                   |␣
↪Show non-responding nodes                                        |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
| get_partitions              | script | local.bash  | ['slurm']                   |␣
↪Get all slurm partitions                                         |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
| sinfo_version               | script | local.bash  | ['slurm']                   |␣
↪get slurm version                                                |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
| show_host_groups            | script | local.bash  | lsf                         |␣
↪Show information about host groups using bmgroup                 |
+-----------------------------+--------+-------------+-----------------------------+--
↪-----------------------------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| show_lsf_configuration       | script | local.bash   | lsf                          |␣
↪Show LSF configuration using lsinfo                                  |
+------------------------------+--------+--------------+------------------------------+--
↪-----------------------------------------------------------------+
| show_lsf_models              | script | local.bash   | lsf                          |␣
↪Show information about host models in LSF cluster                    |
+------------------------------+--------+--------------+------------------------------+--
↪-----------------------------------------------------------------+
| show_lsf_resources           | script | local.bash   | lsf                          |␣
↪Show information about LSF resources                                 |
+------------------------------+--------+--------------+------------------------------+--
↪-----------------------------------------------------------------+
| lsf_version                  | script | local.bash   | lsf                          |␣
↪Display lsf version using lsinfo                                     |
+------------------------------+--------+--------------+------------------------------+--
↪-----------------------------------------------------------------+
| show_lsf_user_groups         | script | local.bash   | lsf                          |␣
↪Show information about all LSF user groups                           |
+------------------------------+--------+--------------+------------------------------+--
↪-----------------------------------------------------------------+
| show_lsf_queues              | script | local.bash   | lsf                          |␣
↪Show LSF queues                                                      |
+------------------------------+--------+--------------+------------------------------+--
↪-----------------------------------------------------------------+
| show_lsf_queues_formatted    | script | local.bash   | lsf                          |␣
↪Show LSF queues with formatted columns                               |
+------------------------------+--------+--------------+------------------------------+--
↪-----------------------------------------------------------------+
| show_lsf_queues_current_user | script | local.bash   | lsf                          |␣
↪Show LSF queues available for current user                           |
+------------------------------+--------+--------------+------------------------------+--
↪-----------------------------------------------------------------+
| display_lsf_hosts            | script | local.bash   | lsf                          |␣
↪Show all hosts in LSF cluster                                        |
+------------------------------+--------+--------------+------------------------------+--
↪-----------------------------------------------------------------+
| display_hosts_format         | script | local.bash   | lsf                          |␣
↪Show all hosts with column hostname and status                       |
+------------------------------+--------+--------------+------------------------------+--
↪-----------------------------------------------------------------+
| bhosts_version               | script | local.bash   | lsf                          |␣
↪Display bhosts version                                               |
+------------------------------+--------+--------------+------------------------------+--
↪-----------------------------------------------------------------+
```

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

```
$ buildtest buildspec find --filter tags=tutorials,executor=local.sh,type=script
+--------------------------+--------+------------+----------------------+------------
↪---------------------------------------+
| Name                     | Type   | Executor   | Tags                 |␣
↪Description                                 |
+==========================+========+============+======================+============
| _bin_sh_shell            | script | local.sh   | ['tutorials']        | /bin/sh␣
↪shell example                               |
+--------------------------+--------+------------+----------------------+------------
↪---------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| sh_shell                  | script | local.sh    | ['tutorials']          | sh shell␣
↪example                                  |
+---------------------------+--------+-------------+------------------------+------------
↪----------------------------------------+
| shell_options             | script | local.sh    | ['tutorials']          | shell␣
↪options                                    |
+---------------------------+--------+-------------+------------------------+------------
↪----------------------------------------+
| exit1_fail                | script | local.sh    | ['tutorials', 'fail'] | exit 1 by␣
↪default is FAIL                           |
+---------------------------+--------+-------------+------------------------+------------
↪----------------------------------------+
| exit1_pass                | script | local.sh    | ['tutorials', 'pass'] | report␣
↪exit 1 as PASS                            |
+---------------------------+--------+-------------+------------------------+------------
↪----------------------------------------+
| returncode_list_mismatch | script | local.sh    | ['tutorials', 'fail'] | exit 2␣
↪failed since it failed to match returncode 1 |
+---------------------------+--------+-------------+------------------------+------------
↪----------------------------------------+
| returncode_int_match      | script | local.sh    | ['tutorials', 'pass'] | exit 128␣
↪matches returncode 128                    |
+---------------------------+--------+-------------+------------------------+------------
↪----------------------------------------+
```

Querying buildspec tags

If you want to retrieve all unique tags from all buildspecs you can run `buildtest buildspec find --tags`

```
$ buildtest buildspec find --tags
+------------+
| Tags       |
+============+
| tutorials  |
+------------+
| compile    |
+------------+
| ssh        |
+------------+
| filesystem |
+------------+
| slurm      |
+------------+
| storage    |
+------------+
| pass       |
+------------+
| ping       |
+------------+
| lsf        |
+------------+
| python     |
+------------+
| system     |
+------------+
| network    |
```

(continues on next page)

(continued from previous page)

```
+------------+
| fail       |
+------------+
```

Querying buildspec executor

To find all executors from cache you can run `buildtest buildspec find --list-executors`. This will retrieve the *'executor'* field from all buildspec and any duplicates will be ignored.

```
$ buildtest buildspec find --list-executors
+--------------+
| executors    |
+==============+
| local.bash   |
+--------------+
| local.python |
+--------------+
| local.sh     |
+--------------+
```

Viewing Buildspecs

If you want to view or edit a buildspec you can type the name of test. Since we can have more than one test in a buildspec, opening any of the *name* entry that map to same file will result in same operation.
For example, we can view `systemd_default_target` as follows

```
$ buildtest buildspec view systemd_default_target
version: "1.0"
buildspecs:
  systemd_default_target:
    executor: local.bash
    type: script
    description: check if default target is multi-user.target
    tags: [tutorials]
    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
    status:
      returncode: 0
```

Editing Buildspecs

To edit a buildspec you can run `buildtest buildspec edit <name>` which will open file in editor. Once you make change, buildtest will validate the buildspec upon closure, if there is an issue buildtest will report an error during validation and you will be prompted to fix issue until it is resolved.
For example we can see an output message after editing file, user will be prompted to press a key which will open the file in editor:

```
$ buildtest buildspec edit systemd_default_target
version 1.1 is not known for type {'1.0': 'script-v1.0.schema.json', 'latest':
↪'script-v1.0.schema.json'}. Try using latest.
```

(continues on next page)

(continued from previous page)

```
Press any key to continue
```

5.4.9 Test Reports (`buildtest report`)

buildtest keeps track of all test results which can be retrieved via **buildtest report**. Shown below is command usage.

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

optional arguments:
  -h, --help       show this help message and exit
  --helpformat     List of available format fields
  --format FORMAT  format field for printing purposes. For more details see --
↪helpformat for list of available fields.
                   Fields must be separated by comma (--format <field1>,<field2>,...)
  --filter FILTER  Filter report by filter fields. The filter fields must be set in␣
↪format: --filter
                   key1=val1,key2=val2,...
  --helpfilter     Report a list of filter fields to be used with --filter option
```

You may run `buildtest report` and buildtest will display report with default format fields.

```
$ buildtest report
+---------------------------+----------+---------+-------------+---------------------
↪-+---------------------+-----------+--------------------+---------------------------
↪-------------------------------------------------------------+
| name                      | id       | state   |   returncode | starttime          ␣
↪ | endtime             |   runtime | tags               | buildspec                ␣
↪                                                                |
+===========================+==========+=========+=============+=====================+============
| systemd_default_target    | 1770533b | FAIL    |            1 | 2020/10/19␣
↪15:54:23 | 2020/10/19 15:54:23 | 0.182725  | tutorials          | /Users/siddiq90/
↪Documents/buildtest/tutorials/systemd.yml                                    |
+---------------------------+----------+---------+-------------+---------------------
↪-+---------------------+-----------+--------------------+---------------------------
↪-------------------------------------------------------------+
| systemd_default_target    | 15a66b55 | FAIL    |            1 | 2020/10/19␣
↪15:54:25 | 2020/10/19 15:54:25 | 0.0458719 | tutorials          | /Users/siddiq90/
↪Documents/buildtest/tutorials/systemd.yml                                    |
+---------------------------+----------+---------+-------------+---------------------
↪-+---------------------+-----------+--------------------+---------------------------
↪-------------------------------------------------------------+
| systemd_default_target    | 8864cbc1 | FAIL    |            1 | 2020/10/19␣
↪15:54:31 | 2020/10/19 15:54:31 | 0.0545546 | tutorials          | /Users/siddiq90/
↪Documents/buildtest/tutorials/systemd.yml                                    |
+---------------------------+----------+---------+-------------+---------------------
↪-+---------------------+-----------+--------------------+---------------------------
↪-------------------------------------------------------------+
| systemd_default_target    | 6fc3c7b4 | FAIL    |            1 | 2020/10/19␣
↪15:54:36 | 2020/10/19 15:54:36 | 0.0453    | tutorials          | /Users/siddiq90/
↪Documents/buildtest/tutorials/systemd.yml                                    |
+---------------------------+----------+---------+-------------+---------------------
↪-+---------------------+-----------+--------------------+---------------------------
↪-------------------------------------------------------------+
| circle_area               | 0bdaef77 | PASS    |            0 | 2020/10/19␣
↪15:54:24 | 2020/10/19 15:54:24 | 0.147359  |                    | /Users/siddiq90/
↪Documents/buildtest/tests/examples/buildspecs/python-shell.yml
```

(continues on next page)

(continued from previous page)

```
+---------------------------+----------+---------+--------------+--------------------
↪-+--------------------+-----------+--------------------+--------------------------
↪-------------------------------------------------------------+
| circle_area               | b06f76c2 | PASS    |            0 | 2020/10/19␣
↪15:54:25 | 2020/10/19 15:54:25 | 0.149374  |                    | /Users/siddiq90/
↪Documents/buildtest/tests/examples/buildspecs/python-shell.yml             |
+---------------------------+----------+---------+--------------+--------------------
↪-+--------------------+-----------+--------------------+--------------------------
↪-------------------------------------------------------------+
| hello_dinosaur            | dff7a691 | PASS    |            0 | 2020/10/19␣
↪15:54:24 | 2020/10/19 15:54:24 | 0.0483172 |                    | /Users/siddiq90/
↪Documents/buildtest/tests/examples/buildspecs/environment.yml              |
+---------------------------+----------+---------+--------------+--------------------
↪-+--------------------+-----------+--------------------+--------------------------
↪-------------------------------------------------------------+
| hello_dinosaur            | cebde392 | PASS    |            0 | 2020/10/19␣
↪15:54:25 | 2020/10/19 15:54:25 | 0.0721789 |                    | /Users/siddiq90/
↪Documents/buildtest/tests/examples/buildspecs/environment.yml              |
+---------------------------+----------+---------+--------------+--------------------
↪-+--------------------+-----------+--------------------+--------------------------
↪-------------------------------------------------------------+
...
```

Format Reports

There are more fields captured in the report, so if you want to see a list of available format fields run `buildtest report --helpformat`.

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

You can format report using `--format` field which expects field name separated by comma (i.e **–format <field1>,<field2>**). In this example we format by fields `--format id,executor,state,returncode`

```
$ buildtest report --format name,id,executor,state,returncode
+---------------------------+----------+--------------+---------+--------------+
| name                      | id       | executor     | state   |   returncode |
+===========================+==========+==============+=========+==============+
| systemd_default_target    | 1770533b | local.bash   | FAIL    |            1 |
```

(continues on next page)

(continued from previous page)

```
+---------------------------+----------+--------------+---------+--------------+
| systemd_default_target    | 15a66b55 | local.bash   | FAIL    |            1 |
+---------------------------+----------+--------------+---------+--------------+
| systemd_default_target    | 8864cbc1 | local.bash   | FAIL    |            1 |
+---------------------------+----------+--------------+---------+--------------+
| systemd_default_target    | 6fc3c7b4 | local.bash   | FAIL    |            1 |
+---------------------------+----------+--------------+---------+--------------+
| circle_area               | 0bdaef77 | local.python | PASS    |            0 |
+---------------------------+----------+--------------+---------+--------------+
| circle_area               | b06f76c2 | local.python | PASS    |            0 |
+---------------------------+----------+--------------+---------+--------------+
| hello_dinosaur            | dff7a691 | local.bash   | PASS    |            0 |
+---------------------------+----------+--------------+---------+--------------+
| hello_dinosaur            | cebde392 | local.bash   | PASS    |            0 |
+---------------------------+----------+--------------+---------+--------------+
...
```

Filter Reports

You can also filter reports using the `--filter` option, but first let's check the available filter fields. In order to see available filter fields run `buildtest report --helpfilter`.

```
$ 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` expects arguments in **key=value** format, you can specify multiple filter fields by a 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.
To see all tests with returncode of 2 we set `--filter returncode=2`.

```
$ buildtest report --filter returncode=2 --format=name,id,returncode
+--------------------------+----------+--------------+
| name                     | id       |   returncode |
+==========================+==========+==============+
| returncode_list_mismatch | 269abcb9 |            2 |
+--------------------------+----------+--------------+
| returncode_list_mismatch | e5905c73 |            2 |
+--------------------------+----------+--------------+
| returncode_list_mismatch | aeb6f626 |            2 |
+--------------------------+----------+--------------+
| returncode_list_mismatch | 78981e2b |            2 |
+--------------------------+----------+--------------+
| returncode_list_mismatch | 1887648a |            2 |
+--------------------------+----------+--------------+
| returncode_list_mismatch | 49d45f72 |            2 |
+--------------------------+----------+--------------+
```

Note: buildtest automatically converts returncode to integer when matching returncode, so `--filter`

`returncode="2"` will work too

If you want to filter by test name `exit1_pass` you can use the `name=exit1_pass` field as shown below

```
$ buildtest report --filter name=exit1_pass --format=name,id,returncode,state
+------------+----------+--------------+---------+
| name       | id       |   returncode | state   |
+============+==========+==============+=========+
| exit1_pass | 992a08e0 |            1 | PASS    |
+------------+----------+--------------+---------+
| exit1_pass | 943c3d32 |            1 | PASS    |
+------------+----------+--------------+---------+
| exit1_pass | 365fdd6e |            1 | PASS    |
+------------+----------+--------------+---------+
| exit1_pass | 59997545 |            1 | PASS    |
+------------+----------+--------------+---------+
```

Likewise, we can filter tests by buildspec file using the `--filter buildspec=<file>`. In example below we set `buildspec=tutorials/pass_returncode.yml`. In this example, buildtest will resolve path and find the buildspec. If file doesn't exist or is not found in cache it will raise an error

```
$ buildtest report --filter buildspec=tutorials/pass_returncode.yml --format=name,id,
↪state,buildspec
+-------------------------+----------+---------+-----------------------------------
↪--------------------------------+
| name                    | id       | state   | buildspec
↪                               |
+=========================+==========+=========+===================================================
| exit1_fail              | d405aea9 | FAIL    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪--------------------------------+
| exit1_fail              | 0a7c4951 | FAIL    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪--------------------------------+
| exit1_fail              | b1b25a16 | FAIL    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪--------------------------------+
| exit1_fail              | 46a14403 | FAIL    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪--------------------------------+
| exit1_fail              | f3a827e6 | FAIL    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪--------------------------------+
| exit1_fail              | b046ccd8 | FAIL    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪--------------------------------+
| exit1_pass              | 992a08e0 | PASS    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪--------------------------------+
| exit1_pass              | 943c3d32 | PASS    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
```

(continues on next page)

(continued from previous page)

```
+-------------------------+----------+---------+-----------------------------------
↪------------------------------+
| exit1_pass              | 365fdd6e | PASS    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪------------------------------+
| exit1_pass              | 59997545 | PASS    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪------------------------------+
| returncode_list_mismatch | 269abcb9 | FAIL    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪------------------------------+
| returncode_list_mismatch | e5905c73 | FAIL    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪------------------------------+
| returncode_list_mismatch | aeb6f626 | FAIL    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪------------------------------+
| returncode_list_mismatch | 78981e2b | FAIL    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪------------------------------+
| returncode_list_mismatch | 1887648a | FAIL    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪------------------------------+
| returncode_list_mismatch | 49d45f72 | FAIL    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪------------------------------+
| returncode_int_match     | 146a0269 | PASS    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪------------------------------+
| returncode_int_match     | ac11ac19 | PASS    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪------------------------------+
| returncode_int_match     | d817c4fd | PASS    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪------------------------------+
| returncode_int_match     | 502be830 | PASS    | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml |
+-------------------------+----------+---------+-----------------------------------
↪------------------------------+
```

We can also pass multiple filter fields for instance if we want to find all **FAIL** tests for executor **local.sh** we can do the following

```
$ buildtest report --filter state=FAIL,executor=local.sh --format=name,id,state,
↪executor
+-------------------------+----------+---------+------------+
```

(continues on next page)

(continued from previous page)

```
| name                      | id       | state   | executor   |
+===========================+==========+=========+============+
| exit1_fail                | d405aea9 | FAIL    | local.sh   |
+---------------------------+----------+---------+------------+
| exit1_fail                | 0a7c4951 | FAIL    | local.sh   |
+---------------------------+----------+---------+------------+
| exit1_fail                | b1b25a16 | FAIL    | local.sh   |
+---------------------------+----------+---------+------------+
| exit1_fail                | 46a14403 | FAIL    | local.sh   |
+---------------------------+----------+---------+------------+
| exit1_fail                | f3a827e6 | FAIL    | local.sh   |
+---------------------------+----------+---------+------------+
| exit1_fail                | b046ccd8 | FAIL    | local.sh   |
+---------------------------+----------+---------+------------+
| returncode_list_mismatch  | 269abcb9 | FAIL    | local.sh   |
+---------------------------+----------+---------+------------+
| returncode_list_mismatch  | e5905c73 | FAIL    | local.sh   |
+---------------------------+----------+---------+------------+
| returncode_list_mismatch  | aeb6f626 | FAIL    | local.sh   |
+---------------------------+----------+---------+------------+
| returncode_list_mismatch  | 78981e2b | FAIL    | local.sh   |
+---------------------------+----------+---------+------------+
| returncode_list_mismatch  | 1887648a | FAIL    | local.sh   |
+---------------------------+----------+---------+------------+
| returncode_list_mismatch  | 49d45f72 | FAIL    | 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 "/Users/siddiq90/Documents/buildtest/bin/buildtest", line 17, in <module>
    buildtest.main.main()
  File "/Users/siddiq90/Documents/buildtest/buildtest/main.py", line 45, in main
    args.func(args)
  File "/Users/siddiq90/Documents/buildtest/buildtest/menu/report.py", line 128, in␣
↪func_report
    raise BuildTestError(f"Invalid returncode:{filter_args[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 `buildspec` 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 buildspec=/path/to/invalid.yml
Invalid File Path for filter field 'buildspec': /path/to/invalid.yml
```

You may have a valid filepath for buildspec filter field such as `tutorials/invalid_executor.yml`, but there is no record in the report cache because this test can't be run. In this case you will get the following message:

```
$ buildtest report --filter buildspec=tutorials/invalid_executor.yml
buildspec file: /Users/siddiq90/Documents/buildtest/tutorials/invalid_executor.yml␣
↪not found in cache
```

5.4.10 Test Inspection

buildtest provides an interface via `buildtest inspect` to query test details once test is recorded in `var/report.json`. The command usage is the following.

```
$ buildtest inspect --help
usage: buildtest [options] [COMMANDS] inspect [-h] test

positional arguments:
  test        select unique test

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

The `buildtest inspect` expects a **unique** test id this can be retrieve using the `full_id` format field if you are not sure:

```
$ buildtest report --format name, full_id
```

For example, let's assume we have the following tests in our report:

```
$ buildtest report --format name,full_id
+-------------------------+--------------------------------------+
| name                    | full_id                              |
+=========================+======================================+
| bash_login_shebang      | eb6e26b2-938b-4913-8b98-e21528c82778 |
+-------------------------+--------------------------------------+
| bash_login_shebang      | d7937a9a-d3fb-4d3f-95e1-465488757820 |
+-------------------------+--------------------------------------+
| bash_login_shebang      | dea6c6fd-b9a6-4b07-a3fc-b483d02d7ff9 |
+-------------------------+--------------------------------------+
| bash_nonlogin_shebang   | bbf94b94-949d-4f97-987a-9a93309f1dc2 |
+-------------------------+--------------------------------------+
| bash_nonlogin_shebang   | 7ca9db2f-1e2b-4739-b9a2-71c8cc00249e |
+-------------------------+--------------------------------------+
| bash_nonlogin_shebang   | 4c5caf85-6ba0-4ca0-90b0-c769a2fcf501 |
+-------------------------+--------------------------------------+
| root_disk_usage         | e78071ef-6444-4228-b7f9-b4eb39071fdd |
+-------------------------+--------------------------------------+
| ulimit_filelock         | c6294cfa-c559-493b-b44f-b17b54ec276d |
+-------------------------+--------------------------------------+
| ulimit_cputime          | aa5530e2-be09-4d49-b8c0-0e818f855a40 |
+-------------------------+--------------------------------------+
| ulimit_stacksize        | 3591925d-7dfa-4bc7-a3b1-fb9dfadf956e |
+-------------------------+--------------------------------------+
| ulimit_vmsize           | 4a01f26b-9c8a-4870-8e33-51923c8c46ad |
+-------------------------+--------------------------------------+
| ulimit_filedescriptor   | 565b85ac-e51f-46f9-8c6f-c2899a370609 |
+-------------------------+--------------------------------------+
| ulimit_max_user_process | 0486c11c-5733-4d8e-822e-c0adddbb2af7 |
+-------------------------+--------------------------------------+
| systemd_default_target  | 7cfc9057-6338-403c-a7af-b1301d04d817 |
+-------------------------+--------------------------------------+
```

Let's assume we are interested in viewing test `bash_login_shebang`, since we have multiple instance for same

test we must specify a unique id. In example below we query the the test id **eb6e26b2-938b-4913-8b98-e21528c82778**:

```
$ buildtest inspect eb6e26b2-938b-4913-8b98-e21528c82778
{
  "id": "eb6e26b2",
  "full_id": "eb6e26b2-938b-4913-8b98-e21528c82778",
  "testroot": "/Users/siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_
↪login_shebang/0",
  "testpath": "/Users/siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_
↪login_shebang/0/stage/generate.sh",
  "command": "/Users/siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_
↪login_shebang/0/stage/generate.sh",
  "outfile": "/Users/siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_
↪login_shebang/0/run/bash_login_shebang.out",
  "errfile": "/Users/siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_
↪login_shebang/0/run/bash_login_shebang.err",
  "schemafile": "script-v1.0.schema.json",
  "executor": "local.bash",
  "tags": "tutorials",
  "starttime": "2020/10/21 16:27:18",
  "endtime": "2020/10/21 16:27:18",
  "runtime": 0.26172968399999996,
  "state": "PASS",
  "returncode": 0
}



Output File
_____________________________________
Login Shell




Error File
_____________________________________




Test Content
_____________________________________
#!/bin/bash -l
source /Users/siddiq90/Documents/buildtest/var/executors/local.bash/before_script.sh
shopt -q login_shell && echo 'Login Shell' || echo 'Not Login Shell'
source /Users/siddiq90/Documents/buildtest/var/executors/local.bash/after_script.sh



buildspec:  /Users/siddiq90/Documents/buildtest/tutorials/shebang.yml
_____________________________________
version: "1.0"
buildspecs:
  bash_login_shebang:
    type: script
    executor: local.bash
```

(continues on next page)

(continued from previous page)

```
    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: 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:
        exp: "^Not Login Shell$"
        stream: stdout
```

buildtest will present the test record from JSON record including contents of output file, error file, testscript and buildspec file.
User can can specify first few characters of the id and buildtest will detect if its a unique test id. If buildtest discovers more than one test id, then buildtest will report all the ids where there is a conflict. In example below we find two tests with id **7c**:

```
$ buildtest inspect 7c
Detected 2 test records, please specify a unique test id
7ca9db2f-1e2b-4739-b9a2-71c8cc00249e
7cfc9057-6338-403c-a7af-b1301d04d817
```

Note: This feature is in development and may change in future

5.4.11 buildtest schemas

The `buildtest schema` command can show you list of available schemas just run the command with no options and it will show all the json schemas supported by buildtest.

```
$ buildtest schema
global.schema.json
definitions.schema.json
settings.schema.json
compiler-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] [-v]

optional arguments:
  -h, --help            show this help message and exit
  -n Schema Name, --name Schema Name
                        show schema by name (e.g., script)
```

(continues on next page)

(continued from previous page)

```
  -e, --example         Show schema examples
  -j, --json            Display json schema file
  -v, --validate        Validate all schema examples with corresponding schemafile
```

The json schemas are hosted on the web at https://buildtesters.github.io/schemas/. buildtest provides a means to display the json schema from the buildtest interface. Note that buildtest will show the schemas provided in buildtest repo and not ones provided by schemas repo. This is because, we let development of schema run independent of the framework.
To select a JSON schema use the `--name` option to select a schema, for example to view a JSON Schema for **script-v1.0.schema.json** run the following:

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

Similarly, if you want to view example buildspecs 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
```

5.4.12 Debug Mode

buildtest can stream logs to `stdout` stream for debugging. You can use `buildtest -d <DEBUGLEVEL>` or long option `--debug` with any buildtest commands. The DEBUGLEVEL are the following:

- DEBUG
- INFO
- WARNING
- ERROR
- CRITICAL

buildtest is using logging.setLevel to control log level. The content is logged in file **buildtest.log** in your current directory with default log level of `DEBUG`. If you want to get all logs use `-d DEBUG` with your buildtest command:

```
buildtest -d DEBUG <command>
```

The debug mode can be useful when troubleshooting builds, in this example we set debug level to `DEBUG` for an invalid buildspec.

```
$ buildtest -d DEBUG build -b tutorials/invalid_executor.yml
2020-10-19 15:54:47,065 [config.py:42 - check_settings() ] - [DEBUG] Loading default␣
↪settings schema: /Users/siddiq90/Documents/buildtest/buildtest/schemas/settings.
↪schema.json
2020-10-19 15:54:47,065 [utils.py:34 - load_schema() ] - [DEBUG] Successfully loaded␣
↪schema file: /Users/siddiq90/Documents/buildtest/buildtest/schemas/settings.schema.
↪json
2020-10-19 15:54:47,065 [config.py:45 - check_settings() ] - [DEBUG] Validating user␣
↪schema with schema: /Users/siddiq90/Documents/buildtest/buildtest/schemas/settings.
↪schema.json
2020-10-19 15:54:47,067 [config.py:47 - check_settings() ] - [DEBUG] Validation was␣
↪successful
2020-10-19 15:54:47,067 [build.py:139 - discover_by_buildspecs() ] - [DEBUG]␣
↪BuildSpec: tutorials/invalid_executor.yml is a file
2020-10-19 15:54:47,067 [build.py:151 - discover_by_buildspecs() ] - [INFO] Found the␣
↪following config files: ['/Users/siddiq90/Documents/buildtest/tutorials/invalid_
↪executor.yml']
2020-10-19 15:54:47,068 [build.py:212 - discover_buildspecs() ] - [DEBUG] Based on␣
↪input argument: --buildspec ['tutorials/invalid_executor.yml'] buildtest discovered␣
↪the following Buildspecs: ['/Users/siddiq90/Documents/buildtest/tutorials/invalid_
↪executor.yml']
```

(continues on next page)

(continued from previous page)

```
+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/invalid_executor.yml
2020-10-19 15:54:47,068 [setup.py:40 - __init__() ] - [DEBUG] Getting Executors from␣
↪buildtest settings
2020-10-19 15:54:47,070 [parser.py:94 - _validate_global() ] - [INFO] Validating /
↪Users/siddiq90/Documents/buildtest/tutorials/invalid_executor.yml with schema: /
↪Users/siddiq90/Documents/buildtest/buildtest/schemas/global.schema.json
2020-10-19 15:54:47,070 [parser.py:115 - _validate() ] - [INFO] Validating test -
↪'wrongexecutor' in recipe: /Users/siddiq90/Documents/buildtest/tutorials/invalid_
↪executor.yml
2020-10-19 15:54:47,070 [parser.py:123 - _validate() ] - [INFO] wrongexecutor is a␣
↪dictionary
2020-10-19 15:54:47,070 [parser.py:133 - _validate() ] - [INFO] Detected field 'type:␣
↪script'
2020-10-19 15:54:47,070 [parser.py:140 - _validate() ] - [INFO] Checking script in␣
↪supported type schemas: ['script', 'compiler']
2020-10-19 15:54:47,071 [buildspec.py:406 - parse_buildspecs() ] - [ERROR] executor:␣
↪badexecutor not found in executor list: ['local.bash', 'local.sh', 'local.python']



Error Messages from Stage: Parse
_____________________________________________________________________________
Skipping /Users/siddiq90/Documents/buildtest/tutorials/invalid_executor.yml since it␣
↪failed to validate
No buildspecs to process because there are no valid buildspecs
```

5.4.13 Accessing buildtest documentation

We provide two command line options to access buildtest and schema docs. To access buildtest docs you can run:

```
$ buildtest docs
```

To access schema docs run:

```
$ buildtest schemadocs
```

5.4.14 Logfile

Currently, buildtest will write the log file for any `buildtest build` command in `buildtest.log` of the current directory. The logfile will be overwritten if you run repeative commands from same directory. A permanent log file location will be implemented (TBD).

5.5 Configuring buildtest

The buildtest configuration file is used for configuring buildtest. This is defined by JSON schemafile named settings.schema.json. For more details on all properties see Settings Schema Documentation.

5.5.1 Default Configuration

The default buildtest configuration is located at buildtest/settings/config.yml relative to root of repo. User may override the default configuration by creating their own buildtest configuration at `$HOME/.buildtest/config.yml` and buildtest will read the user configuration instead.
Shown below is the default configuration provided by buildtest.

```
editor: vi
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

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

5.5.2 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.

5.5.3 Executor Declaration

executors is a JSON object, the structure looks as follows:

```
executors:
  local:
    <local1>:
    <local2>:
    <local3>:
  slurm:
    <slurm1>:
    <slurm2>:
    <slurm3>:
 lsf:
   <lsf1>:
   <lsf2>:
   <lsf3>:
```

The **LocalExecutors** are defined in section *local* where each executor must be unique name:

```
executors:
  local:
```

The *LocalExecutors* can be `bash`, `sh` and `python` shell and they are referenced in buildspec using `executor` field as follows:

```
executor: local.bash
```

The executor is referenced in buildspec in the format: `<type>.<name>` where **type** is **local**, **slurm**, **lsf** defined in the **executors** section and **name** is the executor name. In example above *local.bash* refers to the LocalExecutor using bash shell. Similarly, **SlurmExecutors** and **LSFExecutors** are defined in similar structure.
In this example below we define a local executor named *bash* that is referenced in buildspec as `executor: local.bash`:

```
executors:
  local:
    bash:
      shell: bash
```

The local executors requires the `shell` key which takes the pattern `^(/bin/bash|/bin/sh|sh|bash|python).*` Any buildspec that references `local.bash` 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 bash login executor you can do the following:

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

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

5.5.4 buildtest configuration for Cori @ NERSC

Let's take a look at Cori buildtest configuration:

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

executors:

  defaults:
    pollinterval: 10
    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
```

(continues on next page)

(continued from previous page)

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

    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/spackecp/luke-wyatt-testing/spack/share/spack/
↪setup-env.sh
        source setup.sh

  slurm:
    debug:
      description: jobs for debug qos
      qos: debug
      cluster: cori

    shared:
      description: jobs for shared qos
      qos: shared
      max_pend_time: 180

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

    xfer:
      description: xfer qos jobs
      qos: xfer
      cluster: escori

    gpu:
      description: submit jobs to GPU partition
      options: ["-C gpu"]
      cluster: escori
      max_pend_time: 300

    premium:
      description: submit jobs to premium queue
      qos: premium

    e4s:
      description: "E4S runner"
      qos: debug
      cluster: cori
      options:
        - "-C haswell"
      before_script: |
        source /global/common/software/spackecp/luke-wyatt-testing/spack/share/spack/
↪setup-env.sh
        source $HOME/buildtest-cori/e4s/setup.sh
```

In this setting, we define the following executors

- LocalExecutors: `local.bash`, `local.sh`, `local.python`, `local.e4s`
- SlurmExecutors: `slurm.debug`, `slurm.shared`, `slurm.bigmem`, `slurm.xfer`, `slurm.gpu`, `slurm.premium`, `slurm.e4s`

We introduce section `defaults` which defines configuration for all executors as follows:

```
defaults:
  pollinterval: 10
  launcher: sbatch
  max_pend_time: 90
```

The *launcher* field is applicable for **SlurmExecutor** and **LSFExecutor** in this case, `launcher: sbatch` inherits **sbatch** as the job launcher for all executors. The `pollinterval` field is used to poll jobs at set interval in seconds when job is active in queue. The `max_pend_time` is **maximum** time job can be pending within an executor, if it exceeds the limit buildtest will cancel the job. buildtest will invoke `scancel` or `bkill` to cancel Slurm or LSF job. The *pollinterval*, *launcher* and *max_pend_time* have no effect on **LocalExecutors**.

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. The `description` key is a brief description of the executor only served for documentation purpose. 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 specification. 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 `slurm.gpu` executor, we use the `options: -C gpu` in order 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*

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, not if it is in run state. 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 in **PENDING** (SLURM) or **PEND** (LSF) state. If job pendtime exceeds *max_pend_time* limit, buildtest will cancel job using `scancel` or `bkill` depending on the scheduler. Buildtest will remove cancelled jobs from poll queue, in addition cancelled jobs won't be reported in test report.

5.5.5 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 using `--clear` option it will detect all buildspecs in defined in all directories specified by **buildspec_roots**. buildtest will recursively find all **.yml** extension and validate each buildspec with appropriate schema. By default buildtest will add the `$BUILDTEST_ROOT/tutorials` and `$BUILDTEST_ROOT/general_tests` to search path, where

$BUILDTEST_ROOT is root of repo.

5.5.6 before_script and after_script for executors

Often times, you may want to run a set of commands before or after tests for more than one test. For this reason, we support `before_script` and `after_script` section per executor which is of string type where you can specify multi-line commands.
This can be demonstrated with an executor name **local.e4s** responsible for building E4S Testsuite:

```
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/spackecp/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` and `after_script.sh` for every executor. This can be found in `var/executors` directory as shown below:

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


4 directories, 8 files
```

The `before_script` and `after_script` field is available for all executors and if its not specified the file will be empty. Every test will source the before and after script for the given executor.
The `editor: vi` is used to open buildspecs in *vi* editor, this is used by commands like `buildtest buildspec edit`. For more details see *Editing Buildspecs*. The *editor* field can be *vi*, *vim*, *nano*, or *emacs* depending on your editor preference.

5.5.7 buildtest configuration for 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:

```
editor: vi
executors:
  defaults:
    launcher: bsub
    pollinterval: 10
    max_pend_time: 45

  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

    python:
      description: submit jobs on local machine using python shell
      shell: python
  lsf:
    batch:
      queue: batch
      description: Submit job to batch queue

    test:
      queue: test
      description: Submit job to test queue
```

5.5.8 CLI to buildtest configuration

The `buildtest config` command provides access to buildtest configuration, shown below is the command usage.

```
$ buildtest config --help
usage: buildtest [options] [COMMANDS] config [-h] {view,validate,summary} ...

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

subcommands:
  buildtest configuration

  {view,validate,summary}
    view                View Buildtest Configuration File
    validate            Validate buildtest settings file with schema.
    summary             Provide summary of buildtest settings.
```

View buildtest configuration

If you want to view buildtest configuration you can run the following

```
$ buildtest config view
editor: vi
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

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

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

Validate buildtest configuration

To check if your buildtest settings is valid, run `buildtest config validate`. This will validate your configuration with the schema **settings.schema.json**. The output will be the following.

```
$ buildtest config validate
/Users/siddiq90/Documents/buildtest/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 there was an error on `editor` key in **config** object which expects the editor to be one of the enum types [`vi`, `vim`, `nano`, `emacs`]:

```
$ buildtest config validate
Traceback (most recent call last):
  File "/Users/siddiq90/.local/share/virtualenvs/buildtest-1gHVG2Pd/bin/buildtest",␣
↪line 11, in <module>
    load_entry_point('buildtest', 'console_scripts', 'buildtest')()
  File "/Users/siddiq90/Documents/buildtest/buildtest/main.py", line 32, in main
    check_settings()
  File "/Users/siddiq90/Documents/buildtest/buildtest/config.py", line 71, in check_
↪settings
    validate(instance=user_schema, schema=config_schema)
  File "/Users/siddiq90/.local/share/virtualenvs/buildtest-1gHVG2Pd/lib/python3.7/
↪site-packages/jsonschema/validators.py", line 899, in validate
    raise error
jsonschema.exceptions.ValidationError: 'gedit' is not one of ['vi', 'vim', 'nano',
↪'emacs']

Failed validating 'enum' in schema['properties']['config']['properties']['editor']:
    {'default': 'vim',
```

(continues on next page)

(continued from previous page)

```
    'enum': ['vi', 'vim', 'nano', 'emacs'],
    'type': 'string'}

On instance['config']['editor']:
    'gedit'
```

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:  0.8.1
buildtest Path: /Users/siddiq90/Documents/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-1gHVG2Pd/bin/python
Python Version: 3.7.3
User: siddiq90


Buildtest Settings
_____________________________________________________________________________
Buildtest Settings: /Users/siddiq90/.buildtest/config.yml
Buildtest Settings is  VALID
Executors:  ['local.bash', 'local.sh', 'local.python']
Buildspec Cache File: /Users/siddiq90/Documents/buildtest/var/buildspec-cache.json
Number of buildspecs:  2
Number of Tests: 33
Tests:  ['/Users/siddiq90/Documents/buildtest/tutorials/systemd.yml', '/Users/
↪siddiq90/Documents/buildtest/tutorials/run_only_distro.yml', '/Users/siddiq90/
↪Documents/buildtest/tutorials/shell_examples.yml', '/Users/siddiq90/Documents/
↪buildtest/tutorials/environment.yml', '/Users/siddiq90/Documents/buildtest/
↪tutorials/python-hello.yml', '/Users/siddiq90/Documents/buildtest/tutorials/vars.yml
↪', '/Users/siddiq90/Documents/buildtest/tutorials/selinux.yml', '/Users/siddiq90/
↪Documents/buildtest/tutorials/shebang.yml', '/Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml', '/Users/siddiq90/Documents/buildtest/tutorials/
↪hello_world.yml', '/Users/siddiq90/Documents/buildtest/tutorials/root_user.yml', '/
↪Users/siddiq90/Documents/buildtest/tutorials/tags_example.yml', '/Users/siddiq90/
↪Documents/buildtest/tutorials/run_only_platform.yml', '/Users/siddiq90/Documents/
↪buildtest/tutorials/python-shell.yml', '/Users/siddiq90/Documents/buildtest/
↪tutorials/skip_tests.yml', '/Users/siddiq90/Documents/buildtest/tutorials/sleep.yml
↪', '/Users/siddiq90/Documents/buildtest/tutorials/compilers/vecadd.yml', '/Users/
↪siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml', '/Users/siddiq90/
↪Documents/buildtest/tutorials/compilers/pre_post_build_run.yml', '/Users/siddiq90/
↪Documents/buildtest/tutorials/compilers/passing_args.yml', '/Users/siddiq90/
↪Documents/buildtest/general_tests/configuration/disk_usage.yml', '/Users/siddiq90/
↪Documents/buildtest/general_tests/configuration/ssh_localhost.yml', '/Users/
↪siddiq90/Documents/buildtest/general_tests/configuration/systemd-default-target.yml
↪', '/Users/siddiq90/Documents/buildtest/general_tests/configuration/ulimits.yml', '/
↪Users/siddiq90/Documents/buildtest/general_tests/sched/slurm/sacctmgr.yml', '/Users/
↪siddiq90/Documents/buildtest/general_tests/sched/slurm/scontrol.yml', '/Users/
↪siddiq90/Documents/buildtest/general_tests/sched/slurm/squeue.yml', '/Users/
↪siddiq90/Documents/buildtest/general_tests/sched/slurm/sinfo.yml', '/Users/siddiq90/
↪Documents/buildtest/general_tests/sched/lsf/bmgroups.yml', '/Users/siddiq90/
↪Documents/buildtest/general_tests/sched/lsf/lsinfo.yml', '/Users/siddiq90/Documents/
↪buildtest/general_tests/sched/lsf/bugroup.yml', '/Users/siddiq90/Documents/
↪buildtest/general_tests/sched/lsf/bqueues.yml', '/Users/siddiq90/Documents/
```

(continues on next page)

(continued from previous page)

```
Buildtest Schemas
_____________________________________________________________________________________
Available Schemas: ['script-v1.0.schema.json', 'compiler-v1.0.schema.json', 'global.
↪schema.json', 'settings.schema.json']
Supported Sub-Schemas
_____________________________________________________________________________________
script-v1.0.schema.json : /Users/siddiq90/Documents/buildtest/buildtest/schemas/
↪script-v1.0.schema.json
Examples Directory for schema:  /Users/siddiq90/Documents/buildtest/buildtest/schemas/
↪examples
compiler-v1.0.schema.json : /Users/siddiq90/Documents/buildtest/buildtest/schemas/
↪compiler-v1.0.schema.json
Examples Directory for schema:  /Users/siddiq90/Documents/buildtest/buildtest/schemas/
↪examples
```

5.5.9 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: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/settings.schema.
↪json/valid/local-executor.yml
_____________________________________________________________________________________
editor: vi
executors:
  local:
    bash:
      description: submit jobs on local machine using bash shell
      shell: bash
      before_script: |
        time
        echo "commands run before job"
      after_script: |
        time
        echo "commands run after job"

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

    python:
      description: submit jobs on local machine using python shell
      shell: python
File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/settings.schema.
↪json/valid/slurm-example.yml
_____________________________________________________________________________________
editor: emacs
buildspec_roots:
  - $HOME/buildtest-cori
testdir: /tmp/buildtest
executors:
  defaults:
```

(continues on next page)

(continued from previous page)

```
    pollinterval: 20
    launcher: sbatch
    max_pend_time: 30
  slurm:
    normal:
      options: ["-C haswell"]
      qos: normal
      before_script: |
        time
        echo "commands run before job"
      after_script: |
        time
        echo "commands run after job"
File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/settings.schema.
↪json/valid/combined_executor.yml
--------------------------------------------------------------------------------
editor: vi
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
      options:
        - "-q batch"
        - "-t 00:10"
File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/settings.schema.
↪json/valid/lsf-example.yml
--------------------------------------------------------------------------------
editor: vi
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
      options: ["-W 20"]
      before_script: |
        time
        echo "commands run before job"
      after_script: |
        time
        echo "commands run after job"
    test:
```

(continues on next page)

(continued from previous page)

```
    description: "LSF Executor name 'test' that submits jobs to 'test' queue"
    launcher: bsub
    queue: test
    options: ["-W 20"]
```

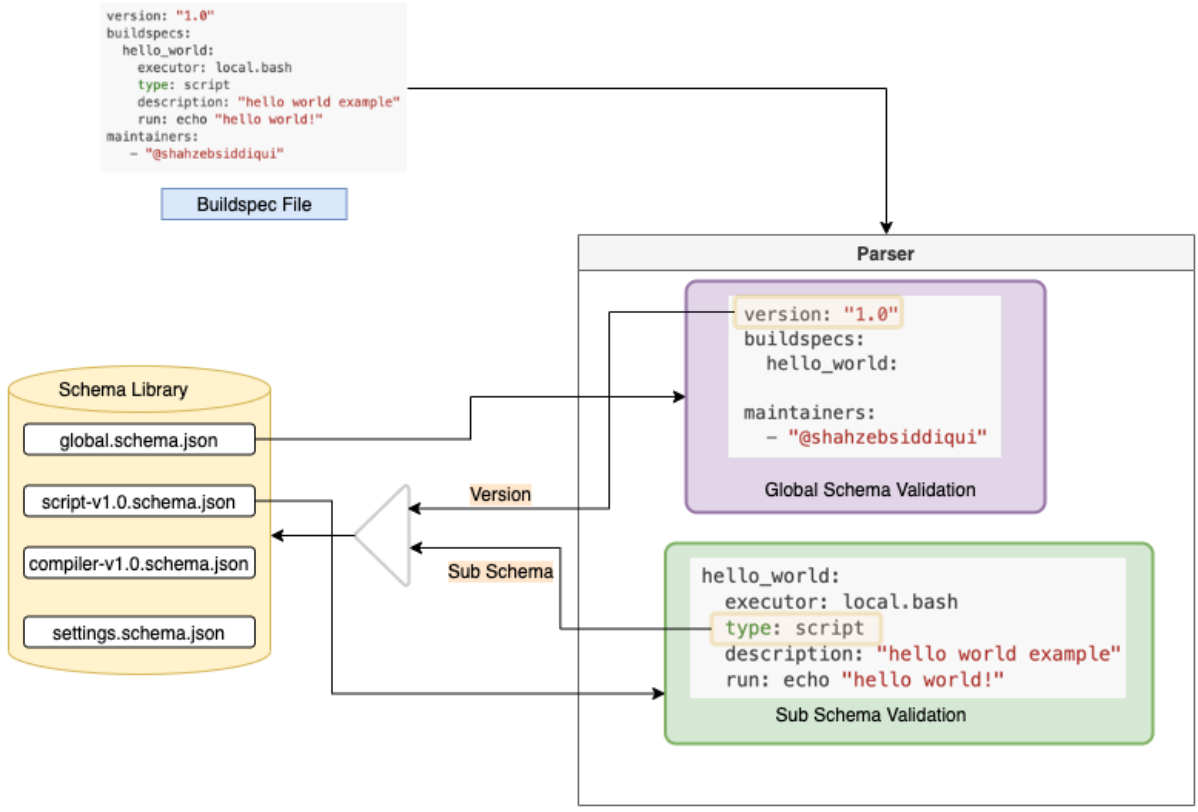
If you want to retrieve full json schema file for buildtest configuration you can run `buildtest schema -n settings.schema.json --json` or short option `-j`.

5.6 Builder Process

buildtest will process all buildspecs that are discovered see diagram *Discover Buildspecs*. The **BuildspecParser** class is responsible for validating the buildspec. The validation is performed using jsonschema.validate. The parser will validate every buildspec using the global.schema.json which validates the top-level structure. This is performed using `BuildspecParser._validate_global` method.
The build pipeline is comprised of 5 stages shown below. Every buildspec goes through this pipeline, if one stage fails, buildtest will skip the test. For instance, a buildspec that fails `Parse` stage will not be built. It is possible a buildspec passes `Parse` stage but fails to build because we have an *Invalid Buildspecs* for example an invalid executor name.

5.6.1 Parse Stage

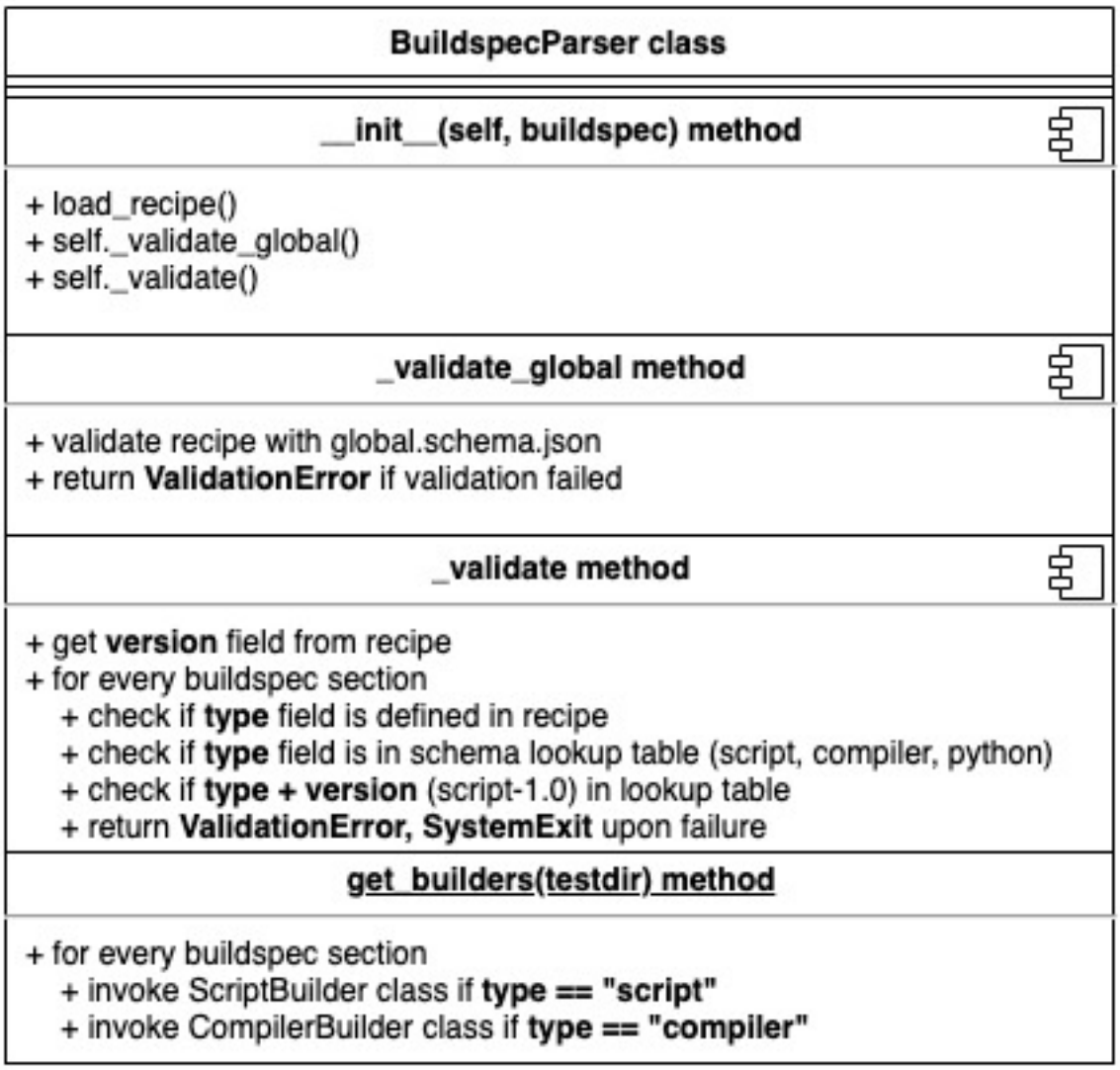
A buildspec file may contain one or more test sections specified via `buildspec` field. Each test is validated by a sub-schema specified by `type` field. The `BuildspecParser._validate` method will validate buildspec test section with the sub-schema.
In this diagram, a buildspec file is passed to the Parser and validated with global schema and a sub-schema.



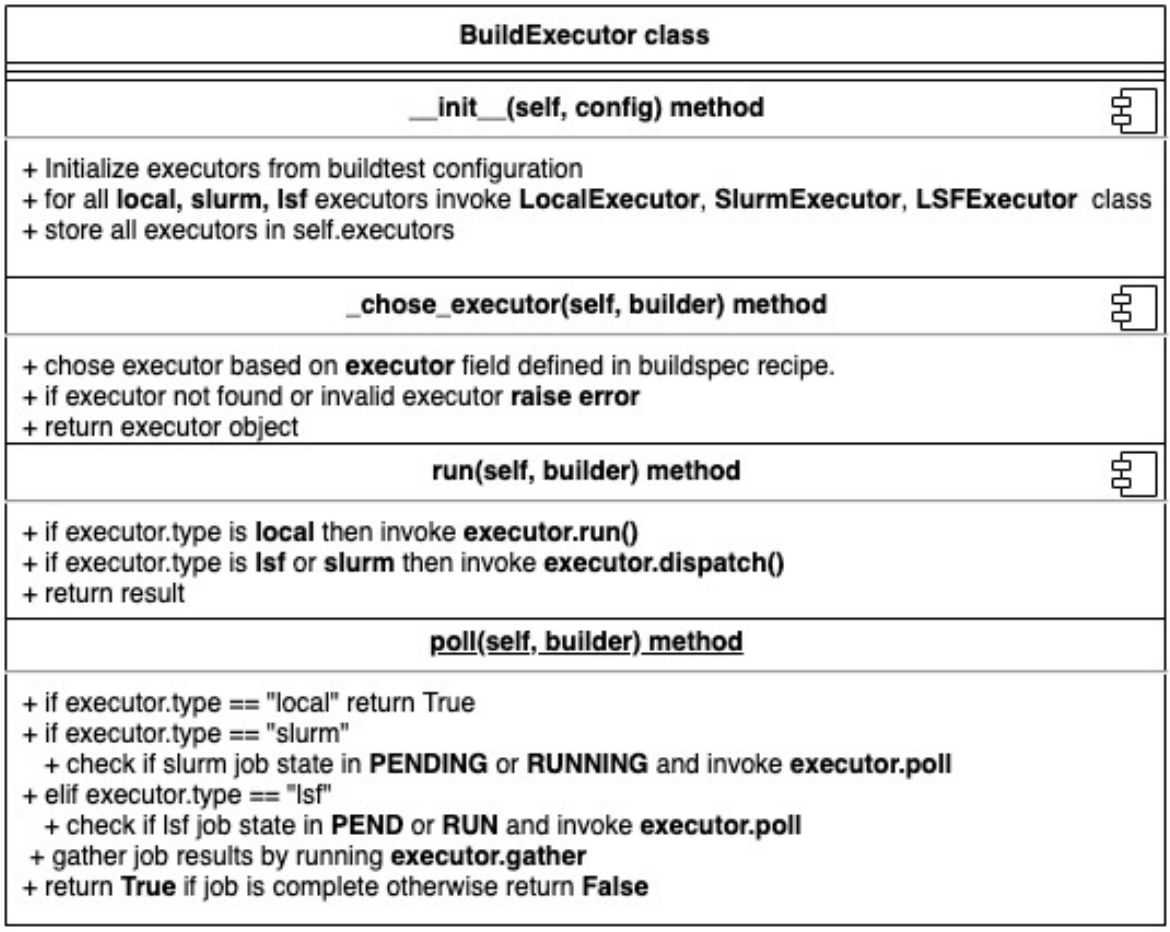
buildtest will invoke BuildspecParser against all discovered buildspecs and catch exceptions **ValidationError** and **SystemExit** and ignore those buildspecs. Next buildtest will build each test, this is implemented using base class *BuilderBase*. The subclass for **BuilderBase** such as **ScriptBuilder** and **CompilerBuilder** are responsible for generating the test script based on the `type` field.

- `type: script` will invoke ScriptBuilder class
- `type: compiler` will invoke CompilerBuilder class

This allows buildtest to extend **BuilderBase** class and each subclass is responsible for one schema type.



The **BuildExecutor** class is responsible for initializing the executors defined in your *Configuring buildtest*. The BuildExecutor class is invoked once and buildtest configuration is passed to this class. buildtest will process all executors defined in *executors* field by invoking the appropriate sub-class. The *local*, *slurm*, *lsf* executors are implemented in class **LocalExecutor**, **SlurmExecutor**, **LSFExecutor** that are sub-class of of **BaseExecutor**. The BaseExecutor class is responsible for implementing common methods for all executors. Each executor class is responsible for running test that is performed in the **Run** stage of the general pipeline.



5.7 Writing buildspecs

5.7.1 Global Schema

The global schema is validated with for all schema types and is the top-level schema when defining a buildspec file. For more details see Global Schema Documentation.

Global Keys in buildspec

Shown below is the start of the 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 buildspecs. The global␣
↪schema defines top-level structure of buildspec and defintions that are inherited␣
↪for sub-schemas",
"type": "object",
"required": ["version","buildspecs"],
```

The global keys required for any buildspec are `version` and `buildspecs`. The version key is required to lookup an a sub-schema using the `type` field. The `buildspecs` is the start of test declaration. The `maintainers` is an optional field that is an array test maintainers. To understand how buildtest validates the buildspec see *Parse Stage*.

Shown below is an example buildspec.

```
version: "1.0"
buildspecs:
  hello_world:
    executor: local.bash
    type: script
    tags: tutorials
    description: "hello world example"
    run: echo "hello world!"
maintainers:
- "@shahzebsiddiqui"
```

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: 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 structure. In this test we define two test named `hello_f` and `environment_variables`. Recall that tests are defined within the `buildspecs` scope and each test is validated with a sub-schema defined by `type` field. Every buildspec must be tied to an executor which defines how test is to be executed.

```
version: "1.0"                                          Schema Version
buildspecs:                                             Declaration of tests
  hello_f:                                              Name of Test
    type: compiler                                      Schema Type
    description: "Hello World Fortran Compilation"      Description of Test
    executor: local.bash                                Name of Executor
    module:                                             Specify Modules to Load
      - "module purge && module load gcc"
    build:                                              Compilation
      source: "src/hello.f90"
      name: gnu
      fflags: -Wall

  environment_variables:                                Name of Test
    executor: local.bash                                Name of Executor
    type: script                                        Schema Type
    env:                                                Declare environment variables
      FIRST_NAME: avocado
      LAST_NAME: dinosaur
    run: |                                              Content of script
      hostname
      whoami
      echo $USER
      printf "${FIRST_NAME} ${LAST_NAME}\n"
```

Proceed to *Buildspecs Overview* to learn more about buildspecs.

5.7.2 Buildspecs Overview

buildspec is your test recipe that buildtest processes to generate a test script. A buildspec can be composed of several test sections. The buildspec file is validated with the *Global Schema* and each test section is validated with a sub-schema defined by the `type` field.
Let's start off with an example:

```
version: "1.0"
buildspecs:
  variables:
    type: script
    executor: local.bash
    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 **buildspec** in order to validate file. The **buildspec** is where you define each test. In this example their is one test called **variables**. 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.
Shown below is schema definition for script-v1.0.schema.json

```
{
  "$id": "script-v1.0.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
```

(continues on next page)

(continued from previous page)

```
  "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 sub-schemas are of type `object` and have a list of required fields that must be provided when using the schema. The `"required"` field specifies a list of fields that must be specified in order to validate the Buildspec. 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 this 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.
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: local.bash
    type: script
    description: check if default target is multi-user.target
    run: |
      if [ "multi-user.target" == `systemctl get-default` ]; then
        echo "multi-user is the default target";
        exit 0
      fi
      echo "multi-user is not the default target";
      exit 1
    status:
      returncode: 0
```

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 **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:

```
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 `<type>.<name>` in this case **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 Buildspec. To use this schema you must set `type: script`. The `run` field is responsible for writing the content of test.
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: local.sh
    type: script
    description: exit 1 by default is FAIL
    tags: [tutorials, fail]
    run: exit 1

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

  returncode_list_mismatch:
    executor: 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: 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

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+
```

(continues on next page)

(continued from previous page)

```
Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml

+--------------------------+
| Stage: Parsing Buildspecs |
+--------------------------+

 schemafile               | validstate   | buildspec
--------------------------+--------------+-----------------------------------------
↪----------------------
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml

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

 name                     | id       | type   | executor   | tags                 |␣
↪testpath
--------------------------+----------+--------+------------+----------------------+--
↪-----------------------------------------------------------------------------------
↪--------------------------------
 exit1_fail               | b046ccd8 | script | local.sh   | ['tutorials', 'fail'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_fail/5/
↪stage/generate.sh
 exit1_pass               | 59997545 | script | local.sh   | ['tutorials', 'pass'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_pass/3/
↪stage/generate.sh
 returncode_list_mismatch | 49d45f72 | script | local.sh   | ['tutorials', 'fail'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_
↪list_mismatch/5/stage/generate.sh
 returncode_int_match     | 502be830 | script | local.sh   | ['tutorials', 'pass'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_
↪int_match/3/stage/generate.sh

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

 name                     | id       | executor   | status   |   returncode | testpath
--------------------------+----------+------------+----------+--------------+---------
↪-----------------------------------------------------------------------------------
↪------------------------
 exit1_fail               | b046ccd8 | local.sh   | FAIL     |            1 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_fail/5/stage/
↪generate.sh
 exit1_pass               | 59997545 | local.sh   | PASS     |            1 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_pass/3/stage/
↪generate.sh
 returncode_list_mismatch | 49d45f72 | local.sh   | FAIL     |            2 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_list_
↪mismatch/5/stage/generate.sh
 returncode_int_match     | 502be830 | local.sh   | PASS     |          128 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_int_
↪match/3/stage/generate.sh
```

(continues on next page)

(continued from previous page)

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

Executed 4 tests
Passed Tests: 2/4 Percentage: 50.000%
Failed Tests: 2/4 Percentage: 50.000%
```

The `returncode` field can be an integer or list of integers. If you specify a list of exit codes, buildtest will `PASS` test if actual exit code is found in list.
A floating point exit-code is invalid:

```
returncode: 1.5
```

If **returncode** is a list, all items must be integers and unique items. The list must contain **atleast** one item. The following examples are invalid values for returncode:

```
returncode: []

returncode: [1, 1.5]

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 test `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: local.bash
    description: tags can be a string
    tags: network
    run: hostname

  list_of_strings_tags:
    type: script
    executor: 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: local.bash
    description: duplicate strings in tags list is not allowed
    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.

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|sh|bash|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 Line in the test-script. In this example, we illustrate a few tests using different shell field.

```
version: "1.0"
buildspecs:
  _bin_sh_shell:
    executor: local.sh
    type: script
    description: "/bin/sh shell example"
    shell: /bin/sh
    tags: [tutorials]
    run: "bzip2 --help"

  _bin_bash_shell:
    executor: local.bash
    type: script
    description: "/bin/bash shell example"
    shell: /bin/bash
    tags: [tutorials]
    run: "bzip2 -h"

  bash_shell:
    executor: local.bash
    type: script
    description: "bash shell example"
    shell: bash
    tags: [tutorials]
    run: "echo $SHELL"

  sh_shell:
```

(continues on next page)

(continued from previous page)

```
    executor: local.sh
    type: script
    description: "sh shell example"
    shell: sh
    tags: [tutorials]
    run: "echo $SHELL"

  shell_options:
    executor: 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
source /Users/siddiq90/Documents/buildtest/var/executors/local.sh/before_script.sh
bzip2 --help
source /Users/siddiq90/Documents/buildtest/var/executors/local.sh/after_script.sh
```

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 shebang line. The generated test for this script is the following:

```
#!/bin/sh -x
source /Users/siddiq90/Documents/buildtest/var/executors/local.sh/before_script.sh
echo $SHELL
hostname
source /Users/siddiq90/Documents/buildtest/var/executors/local.sh/after_script.sh
```

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: 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:
```

(continues on next page)

(continued from previous page)

```
        exp: "^Login Shell$"
        stream: stdout

  bash_nonlogin_shebang:
    type: script
    executor: 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:
        exp: "^Not Login Shell$"
        stream: stdout
```

Now let's run this test as we see the following.

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

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/shebang.yml

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile              | validstate   | buildspec
-------------------------+--------------+-------------------------------------------
↪--------------
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/shebang.yml

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

 name                   | id       | type    | executor    | tags       | testpath
------------------------+----------+---------+-------------+------------+-----------------
↪-----------------------------------------------------------------------------------------
↪-------
 bash_login_shebang     | c99deb8a | script | local.bash | tutorials | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/shebang/bash_login_shebang/2/stage/
↪generate.sh
 bash_nonlogin_shebang | 512a55d5 | script | local.bash | tutorials | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/shebang/bash_nonlogin_shebang/2/stage/
↪generate.sh

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

(continues on next page)

(continued from previous page)

```
 name                   | id       | executor    | status    |   returncode | testpath
------------------------+----------+-------------+-----------+--------------+------------
↪-------------------------------------------------------------------------------------
↪------------
 bash_login_shebang     | c99deb8a | local.bash | PASS      |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_login_shebang/2/
↪stage/generate.sh
 bash_nonlogin_shebang | 512a55d5 | local.bash | PASS      |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_nonlogin_shebang/2/
↪stage/generate.sh

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

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

If we look at the generated test for **bash_login_shebang** we see the shebang line is passed into the script:

```
#!/bin/bash -l
source /Users/siddiq90/Documents/buildtest/var/executors/local.bash/before_script.sh
shopt -q login_shell && echo 'Login Shell' || echo 'Not Login Shell'
source /Users/siddiq90/Documents/buildtest/var/executors/local.bash/after_script.sh
```

Python Shell

You can use *script* schema to write python scripts using the run section. This can be achieved if you use the `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: 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: local.bash
    skip: Yes
    tags: [tutorials]
    run: hostname

  unskipped:
    type: script
    executor: local.bash
    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: Ommitting line `skip: No` from test **unskipped** will result in same behavior

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

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/skip_tests.yml
[skip] test is skipped.

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile              | validstate   | buildspec
-------------------------+--------------+-------------------------------------------
↪-----------------
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/skip_tests.yml

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

(continues on next page)

(continued from previous page)

```
+----------------------+

 name      | id       | type   | executor   | tags          | testpath
-----------+----------+--------+------------+---------------+-------------------------
↪-------------------------------------------------------------------------
 unskipped | a9e0ff3d | script | local.bash | ['tutorials'] | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/skip_tests/unskipped/2/stage/generate.sh

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

 name      | id       | executor   | status   |   returncode | testpath
-----------+----------+------------+----------+--------------+-----------------------
↪-------------------------------------------------------------------------
 unskipped | a9e0ff3d | local.bash | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/skip_tests/unskipped/2/stage/generate.sh

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

Executed 1 tests
Passed Tests: 1/1 Percentage: 100.000%
Failed Tests: 0/1 Percentage: 0.000%
```

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 buildspec 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: 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

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+
```

(continues on next page)

(continued from previous page)

```
Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/root_user.yml
[run_only_as_root] test is skipped because ['run_only']['user'] got value: root but␣
↪detected user: siddiq90.
No buildspecs to process because there are no valid buildspecs
```

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: 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: local.python
    type: script
    tags: ["tutorials"]
    run_only:
      platform: Linux
    shell: python
    run: |
      import platform
      print(platform.system())
    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

```

(continues on next page)

(continued from previous page)

```
+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/run_only_platform.yml
[run_only_platform_linux] test is skipped because ['run_only']['platform'] got value:␣
↪Linux but detected platform: Darwin.

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile               | validstate   | buildspec
--------------------------+--------------+-----------------------------------------
↪-------------------------
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/run_only_platform.yml

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

 name                       | id       | type    | executor     | tags          |␣
↪testpath
---------------------------+----------+--------+--------------+---------------+--------
↪-------------------------------------------------------------------------------------
↪--------------------------------
 run_only_platform_darwin | f337083e | script | local.python | ['tutorials'] | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.python/run_only_platform/run_only_
↪platform_darwin/1/stage/generate.sh

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

 name                       | id       | executor     | status   |   returncode |␣
↪testpath
---------------------------+----------+--------------+----------+--------------+-------
↪-------------------------------------------------------------------------------------
↪--------------------------------
 run_only_platform_darwin | f337083e | local.python | PASS     |            0 | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.python/run_only_platform/run_
↪only_platform_darwin/1/stage/generate.sh

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

Executed 1 tests
Passed Tests: 1/1 Percentage: 100.000%
Failed Tests: 0/1 Percentage: 0.000%
```

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 `lsf` or `slurm` as valid values and buildtest will check if target system supports 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: 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/bmgroups.yml

+-------------------------------+
| Stage: Discovering Buildspecs |
+-------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/general_tests/sched/lsf/bmgroups.yml
[show_host_groups] test is skipped because ['run_only']['scheduler'] got value: lsf␣
↪but detected scheduler: [].
No buildspecs to process because there are no valid buildspecs
```

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: local.bash
    description: "Run test only if linux distro is darwin."
    run_only:
      linux_distro:
        - darwin
    run: uname
    status:
      regex:
        stream: stdout
        exp: "^Darwin$"
```

This test will run successfully because this was ran on a Mac OS (darwin) system.

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

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/run_only_distro.yml

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile              | validstate   | buildspec
-------------------------+--------------+-----------------------------------------------
↪-----------------------
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/run_only_distro.yml

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

 name                   | id       | type   | executor   | tags   | testpath
------------------------+----------+--------+------------+--------+--------------------
↪-------------------------------------------------------------------------------------
↪------------
 run_only_macos_distro | 48c8ebae | script | local.bash |        | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/run_only_distro/run_only_macos_distro/1/
↪stage/generate.sh

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

 name                   | id       | executor   | status   |   returncode | testpath
------------------------+----------+------------+----------+--------------+------------
↪-------------------------------------------------------------------------------------
↪--------------------
 run_only_macos_distro | 48c8ebae | local.bash | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/run_only_distro/run_only_macos_
↪distro/1/stage/generate.sh

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

Executed 1 tests
Passed Tests: 1/1 Percentage: 100.000%
Failed Tests: 0/1 Percentage: 0.000%
```

5.7.3 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 definition for compiler schema:

```
"$id": "compiler-v1.0.schema.json",
"$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", "build", "executor"],
"additionalProperties": false,
```

The required fields for compiler schema are **type**, **build**, and **executor**. The compiler schema is a JSON object defined by `"type": "object"` which is similar to the *script* schema.
Shown below are 6 test examples performing Hello World compilation with C, C++, and Fortran using GNU compiler

```
version: "1.0"
buildspecs:
  hello_f:
    type: compiler
    description: "Hello World Fortran Compilation"
    executor: local.bash
    tags: [tutorials, compile]
    build:
      source: "src/hello.f90"
      name: gnu
      fflags: -Wall

  hello_c:
    type: compiler
    description: "Hello World C Compilation"
    executor: local.bash
    tags: [tutorials, compile]
    build:
      source: "src/hello.c"
      name: gnu
      cflags: -Wall

  hello_cplusplus:
    type: compiler
    description: "Hello World C++ Compilation"
    executor: local.bash
    tags: [tutorials, compile]
    build:
      source: "src/hello.cpp"
      name: gnu
      cxxflags: -Wall

  cc_example:
    type: compiler
    description: Example by using cc to set C compiler
    executor: local.bash
```

(continues on next page)

(continued from previous page)

```
    tags: [tutorials, compile]
    build:
      source: "src/hello.c"
      name: gnu
      cc: gcc

  fc_example:
    type: compiler
    description: Example by using fc to set Fortran compiler
    executor: local.bash
    tags: [tutorials, compile]
    build:
      source: "src/hello.f90"
      name: gnu
      fc: gfortran

  cxx_example:
    type: compiler
    description: Example by using cxx to set C++ compiler
    executor: local.bash
    tags: [tutorials, compile]
    build:
      source: "src/hello.cpp"
      name: gnu
      cxx: g++
```

The tests `hello_f`, `hello_c` and `hello_cplusplus` rely on buildtest to detect compiler wrappers while tests `cc_example`, `fc_example`, `cxx_example` rely on user to specify compiler wrappers manually.
The `compiler` object is start of compilation section, the required keys are `source` and `name`. The **source** key requires an input program for compilation, this can be a file relative to buildspec file or an absolute path. In this example our source examples are in `src` directory. The `name` field informs buildtest to auto-detect compiler wrappers (`cc`, `fc`, `cxx`).
The compilation pattern buildtest utilizes is the following:

```
# C example
$cc $cppflags $cflags -o <executable> $SOURCE $ldflags

# Fortran example
$cxx $cppflags $cxxflags -o <executable> $SOURCE $ldflags

# Fortran example
$fc $cppflags $fflags -o <executable> $SOURCE $ldflags
```

If you specify `cc`, `fc` and `cxx` field attributes you are responsible for selecting the correct compiler wrapper. You can use `cflags`, `cxxflags` and `fflags` field to pass compiler options to C, C++ and Fortran compilers.
Shown below is an example build for the buildspec example

```
$ buildtest build -b tutorials/compilers/gnu_hello.yml

+--------------------------------+
| Stage: Discovering Buildspecs |
+--------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml
```

(continues on next page)

(continued from previous page)

```
+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile                | validstate   | buildspec
---------------------------+--------------+-----------------------------------------
↪---------------------------
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/gnu_hello.yml

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

 name            | id       | type     | executor   | tags                      |␣
↪testpath
-----------------+----------+----------+------------+---------------------------+------
↪-------------------------------------------------------------------------------------
↪--------------
 hello_f         | 3e4017b8 | compiler | local.bash | ['tutorials', 'compile'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_f/3/stage/
↪generate.sh
 hello_c         | 33ba91d2 | compiler | local.bash | ['tutorials', 'compile'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_c/3/stage/
↪generate.sh
 hello_cplusplus | b7ffc06c | compiler | local.bash | ['tutorials', 'compile'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/hello_cplusplus/3/
↪stage/generate.sh
 cc_example      | e565abb3 | compiler | local.bash | ['tutorials', 'compile'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cc_example/3/
↪stage/generate.sh
 fc_example      | cf7c3505 | compiler | local.bash | ['tutorials', 'compile'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/fc_example/3/
↪stage/generate.sh
 cxx_example     | 6dcf90b8 | compiler | local.bash | ['tutorials', 'compile'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cxx_example/3/
↪stage/generate.sh

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

 name            | id       | executor   | status   |   returncode | testpath
-----------------+----------+------------+----------+--------------+------------------
↪-------------------------------------------------------------------------------------
↪--
 hello_f         | 3e4017b8 | local.bash | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/gnu_hello/hello_f/3/stage/generate.sh
 hello_c         | 33ba91d2 | local.bash | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/gnu_hello/hello_c/3/stage/generate.sh
 hello_cplusplus | b7ffc06c | local.bash | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/gnu_hello/hello_cplusplus/3/stage/generate.
↪sh
 cc_example      | e565abb3 | local.bash | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/gnu_hello/cc_example/3/stage/generate.sh
 fc_example      | cf7c3505 | local.bash | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/gnu_hello/fc_example/3/stage/generate.sh P
```

(continues on next page)

(continued from previous page)

```
 cxx_example     | 6dcf90b8 | local.bash | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/gnu_hello/cxx_example/3/stage/generate.sh

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

Executed 6 tests
Passed Tests: 6/6 Percentage: 100.000%
Failed Tests: 0/6 Percentage: 0.000%
```

The generated test for test name **hello_f** is the following:

```
#!/bin/bash
source /Users/siddiq90/Documents/buildtest/var/executors/local.bash/before_script.sh
gfortran -Wall -o hello.f90.exe src/hello.f90
./hello.f90.exe
source /Users/siddiq90/Documents/buildtest/var/executors/local.bash/after_script.sh
```

buildtest will fill in the compilation line based on compilation pattern. buildtest, will detect the file extensions and perform a lookup to find the programming language, and finally generate the appropriate C, C++, or Fortran compilation based on language detected.
buildtest detects the programming language and it finds **.f90** file extension and infers it must be Fortran program, hence `gfortran` was selected. The executable name is generated by adding `.exe` to end of source file name so we get `hello.f90.exe`. Finally, we run the executable.

File Extension Language Table

Shown below is the file extension table for your reference

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

Passing Arguments

If you want to pass options to executable command use the `args` key. Shown below is an example test

```
version: "1.0"
buildspecs:
  executable_arguments:
    type: compiler
    description: Passing arguments example
    executor: local.bash
    tags: [tutorials, compile]
    build:
      source: "src/argc.c"
      name: gnu
      cflags: -Wall
    run:
      args: "1 2 3"
```

The exec_args will pass options to the executable, use this if your binary requires input arguments. Shown below is a generated test:

```
#!/bin/bash
gcc -Wall -o argc.c.exe /global/u1/s/siddiq90/tutorials/examples/serial/src/argc.c
./argc.c.exe 1 2 3
```

OpenMP Example

Here is an example OpenMP reduction test that runs on 1 node using 32 tasks on a haswell node:

```
version: "1.0"
buildspecs:
  reduction:
    type: compiler
    executor: slurm.debug
    sbatch: ["-N 1", "--ntasks-per-node 32", "-C haswell", "-t 1"]
    module:
      - "module load PrgEnv-gnu"
    env:
      OMP_NUM_THREADS: 32
      OMP_PROC_BIND: spread
      OMP_PLACES: cores
    build:
      source: src/reduction.c
      name: gnu
      cflags: -fopenmp
    tags: [openmp]
```

In this example, we use the SlurmExecutor `slurm.debug`, the source file is `src/reduction.c` that is relative to buildspec file. The environment variables are defined using `env` section. To enable openmp flag, for GNU compilers we pass `-fopenmp` to C compiler. Finally we classify this test using `tags` key which is set to *openmp*. The generated test looks as follows:

```
#!/bin/bash
#SBATCH -N 1
#SBATCH --ntasks-per-node 32
#SBATCH -C haswell
#SBATCH -t 1
export OMP_NUM_THREADS=32
export OMP_PROC_BIND=spread
export OMP_PLACES=cores
module load PrgEnv-gnu
gcc -fopenmp -o reduction.c.exe /global/u1/s/siddiq90/buildtest-cori/apps/openmp/src/
↪reduction.c
./reduction.c.exe
```

MPI Example

In this example we run a MPI Laplace code using 4 process on a KNL node using the module `PrgEnv-intel`. The executable is launched using `srun`, that is set via `launcher` field. The source code `src/laplace_mpi.c` must be run with 4 process, for this test we allocate 1 node with 4 tasks.
The `name` field is a required field, buildtest uses this field to select the appropriate subclass, when you set `name: intel` buildtest will select the IntelCompiler subclass which sets the `cc`, `fc` and `cxx` variables automatically. If you want to specify your compiler variables you can use `cc`, `fc` and `cxx` fields and buildtest will honor your options.

```
version: "1.0"
buildspecs:
  laplace_mpi:
```

(continues on next page)

(continued from previous page)

```
    type: compiler
    description: Laplace MPI code in C
    sbatch: ["-C knl", "-N 1", "-n 4"]
    executor: slurm.debug
    tags: ["mpi"]
    module:
      - "module load PrgEnv-intel"
    build:
      name: intel
      source: src/laplace_mpi.c
      cflags: -O3
    run:
      launcher: srun -n 4
```

The generated test is as follows:

```
#!/bin/bash
#SBATCH -C knl
#SBATCH -N 1
#SBATCH -n 4
module load PrgEnv-intel
icc -O3 -o laplace_mpi.c.exe /global/u1/s/siddiq90/buildtest-cori/apps/mpi/src/
↪laplace_mpi.c
srun -n 4 ./laplace_mpi.c.exe
```

Shown below is a sample build for this buildspec:

```
$ buildtest build -b mpi/laplace_mpi.yml
Paths:
___________
Prefix: /global/u1/s/siddiq90/cache
Buildspec Search Path: ['/global/u1/s/siddiq90/buildtest/tutorials']
Test Directory: /global/u1/s/siddiq90/cache/tests

+--------------------------------+
| Stage: Discovered Buildspecs   |
+--------------------------------+

/global/u1/s/siddiq90/buildtest-cori/apps/mpi/laplace_mpi.yml

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

 Name         | Schema File                | Test Path
↪                   | Buildspec
-------------+----------------------------+-----------------------------------------
↪-----------------+-------------------------------------------------------------------
 laplace_mpi | compiler-v1.0.schema.json | /global/u1/s/siddiq90/cache/tests/laplace_
↪mpi/laplace_mpi.sh | /global/u1/s/siddiq90/buildtest-cori/apps/mpi/laplace_mpi.yml

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

[laplace_mpi] job dispatched to scheduler
[laplace_mpi] acquiring job id in 2 seconds
 name         | executor     | status    |   returncode | testpath
```

(continues on next page)

(continued from previous page)

```
------------+------------+----------+--------------+-------------------------------
↪------------------------------
 laplace_mpi | slurm.debug | N/A      |            0 | /global/u1/s/siddiq90/cache/
↪tests/laplace_mpi/laplace_mpi.sh


Polling Jobs in 10 seconds
_____________________________________________
[laplace_mpi]: JobID 33306420 in COMPLETED state


Polling Jobs in 10 seconds
_____________________________________________

+---------------------------------------------+
| Stage: Final Results after Polling all Jobs |
+---------------------------------------------+

 name         | executor    | status   |   returncode | testpath
------------+------------+----------+--------------+-------------------------------
↪------------------------------
 laplace_mpi | slurm.debug | PASS     |            0 | /global/u1/s/siddiq90/cache/
↪tests/laplace_mpi/laplace_mpi.sh

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

Executed 1 tests
Passed Tests: 1/1 Percentage: 100.000%
Failed Tests: 0/1 Percentage: 0.000%
```

OpenACC Examples

Next, we will make use of an OpenACC vector addition example shown below is an example test

```
version: "1.0"
buildspecs:
  vecadd_gnu:
    type: compiler
    description: Vector Addition example with GNU compiler
    tags: [tutorials, compile]
    executor: local.bash
    build:
      name: gnu
      source: src/vecAdd.c
      cflags: -fopenacc
      ldflags: -lm
    status:
      regex:
        stream: stdout
        exp: "^final result: 1.000000$"
```

To compile OpenACC program with gnu compiler we must use `-fopenacc` flag, this program requires linking with math library so we can specify linker flags (ldflags) using `ldflags: -lm`.
The output of this test will generate a single line output as follows:

```
final result: 1.000000
```

The `status` field with `regex` is used for checking output stream using `stream: stdout` and `exp` key to specify regular expression to use. If we are to build this test, you will notice the run section will have a Status of `PASS`

```
$ buildtest build -b tutorials/compilers/vecadd.yml

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/compilers/vecadd.yml

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile                | validstate   | buildspec
---------------------------+--------------+-----------------------------------------
↪--------------------------
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/vecadd.yml

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

 name       | id       | type     | executor   | tags                      | testpath
------------+----------+----------+------------+---------------------------+-----------
↪-------------------------------------------------------------------------------------
↪-
 vecadd_gnu | 1a0f6265 | compiler | local.bash | ['tutorials', 'compile'] | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/vecadd/vecadd_gnu/3/stage/
↪generate.sh

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

 name       | id       | executor   | status   |   returncode | testpath
------------+----------+------------+----------+--------------+----------------------
↪-------------------------------------------------------------------------
 vecadd_gnu | 1a0f6265 | local.bash | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/vecadd/vecadd_gnu/3/stage/generate.sh

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

Executed 1 tests
Passed Tests: 0/1 Percentage: 0.000%
Failed Tests: 1/1 Percentage: 100.000%
```

The regular expression is performed using re.search, for example if we can change the `exp` field as follows:

```
exp: "^final result: 0.99$"
```

Next if we re-run test we will notice the Status is `FAIL` even though we have a Return Code of **0**:

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

 name        | executor    | status    |   returncode | testpath
-------------+-------------+-----------+--------------+-----------------------------------
↪-------------------------------------------------------
 vecadd_gnu | local.bash | FAIL      |            0 | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/vecadd/vecadd_gnu/run_script.sh
```

In the next example, we extend the previous buildspec test to run at Cori GPU machine using Slurm scheduler. We use the executor `slurm.gpu` where our executor is defined as follows:

```
gpu:
  description: submit jobs to GPU partition
  options: ["-C gpu"]
  cluster: escori
```

In order to submit job to the Cori GPU cluster we must use `sbatch -C gpu -M escori` which is what `slurm.gpu` executor is doing.
In this example we make use of `module` field to load modules into the test, for this test we load the modules `cuda` and `gcc/8.1.1-openacc-gcc-8-branch-20190215`. This test will launch job via `srun` and check job state code is `COMPLETED`.

```
version: "1.0"
buildspecs:
  vecadd_openacc_gnu:
    type: compiler
    description: Vector Addition example with GNU compiler
    executor: slurm.gpu
    sbatch: ["-G 1", "-t 5", "-N 1"]
    module:
      - "module load cuda"
      - "module load gcc/8.1.1-openacc-gcc-8-branch-20190215"
    build:
      name: gnu
      source: src/vecAdd.c
      cflags: -fopenacc
      ldflags: -lm
    run:
      launcher: srun
    status:
      slurm_job_state: COMPLETED
```

buildtest will generate the following test, buildtest will add the #SBATCH directives followed by module commands. The executable is run via `srun` because we specify the `launcher` field.

```
#!/bin/bash
#SBATCH -G 1
#SBATCH -t 5
#SBATCH -N 1
module load cuda
module load gcc/8.1.1-openacc-gcc-8-branch-20190215
gcc -fopenacc -o vecAdd.c.exe /global/u1/s/siddiq90/buildtest-cori/apps/openacc/src/
↪vecAdd.c -lm
```

(continues on next page)

(continued from previous page)

```
srun ./vecAdd.c.exe
```

In this next example, we build same test using hpcsdk compiler by NVIDIA that recently acquired PGI compiler. At Cori, we must load `hpcsdk` and `cuda` module in order to use the hpcsdk compiler. The `name` is a required field however buildtest will ignore since we specify `cc` field. NVIDIA changed their compiler names instead of `pgcc` we must use `nvc` with flag `-acc` to offload to GPU. For CoriGPU we must use `srun` to acquire GPU access hence `launcher` field is set to srun.

```
version: "1.0"
buildspecs:
  vecadd_hpcsdk_gnu:
    type: compiler
    description: Vector Addition example with hpcsdk (pgi) compiler
    executor: slurm.gpu
    sbatch: ["-G 1", "-t 5", "-N 1"]
    module:
      - "module load hpcsdk"
      - "module load cuda"
    build:
      name: pgi
      cc: nvc
      source: src/vecAdd.c
      cflags: -acc
      ldflags: -lm
    run:
      launcher: srun
```

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: local.bash
    tags: [tutorials, compile]
    pre_build: |
      echo "This is a pre-build section"
      gcc --version
    build:
      source: "src/hello.c"
      name: gnu
      cflags: -Wall
    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 the following:

```
#!{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 buildspec is the following:

```
#!/bin/bash
echo "This is a pre-build section"
gcc --version

gcc -Wall -o hello.c.exe /Users/siddiq90/Documents/buildtest/tutorials/compilers/src/
↪hello.c
echo "This is post-build section"

echo "This is pre-run section"
export FOO=BAR

./hello.c.exe
echo "This is post-run section"
```

5.7.4 Batch Scheduler Support

buildtest batch scheduler support is an experimental feature, currently buildtest supports Slurm and LSF Executor. In order for buildtest to submit jobs to scheduler, you must define a slurm or lsf executor.

Slurm Executor

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 all jobs and poll all jobs in a `while (True)` until all jobs are complete. If job is in [**PENDING** | **RUNNING**] then buildtest will keep polling at a set interval. Once job is not in **PENDING** or **RUNNING** stage, buildtest will gather job results and wait until all jobs have finished.
In order to use a slurm scheduler, you must define some slurm executors and reference them via `executor` property. In this example we have a slurm executor `slurm.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 buildspec:

```
version: "1.0"
buildspecs:
  slurm_metadata:
    description: Get metadata from compute node when submitting job
    type: script
```

(continues on next page)

(continued from previous page)

```
    executor: slurm.debug
    sbatch:
      - "-t 00:05"
      - "-C haswell"
      - "-N 1"
    run: |
      export SLURM_JOB_NAME="firstjob"
      echo "jobname:" $SLURM_JOB_NAME
      echo "slurmdb host:" $SLURMD_NODENAME
      echo "pid:" $SLURM_TASK_PID
      echo "submit host:" $SLURM_SUBMIT_HOST
      echo "nodeid:" $SLURM_NODEID
      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

```
#!/bin/bash
#SBATCH -t 00:05
#SBATCH -C haswell
#SBATCH -N 1
export SLURM_JOB_NAME="firstjob"
echo "jobname:" $SLURM_JOB_NAME
echo "slurmdb host:" $SLURMD_NODENAME
echo "pid:" $SLURM_TASK_PID
echo "submit host:" $SLURM_SUBMIT_HOST
echo "nodeid:" $SLURM_NODEID
echo "partition:" $SLURM_JOB_PARTITION
```

The slurm.debug executor in our `settings.yml` is defined as follows:

```
slurm:
  debug:
    description: jobs for debug qos
    qos: debug
    cluster: cori
```

With this setting, any buildspec test that use `slurm.debug` executor will result in the following launch option: `sbatch --qos debug --clusters=cori </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 buildspecs, 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 metadata.yml
Paths:
___________
Prefix: /global/u1/s/siddiq90/cache
Buildspec Search Path: ['/global/homes/s/siddiq90/.buildtest/site']
Test Directory: /global/u1/s/siddiq90/cache/tests

+------------------------------+
| Stage: Discovered Buildspecs  |
```

(continues on next page)

(continued from previous page)

```
+-------------------------------+

/global/u1/s/siddiq90/buildtest-cori/slurm/valid_jobs/metadata.yml

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

 Name           | Schema File            | Test Path                                   ␣
↪                    | Buildspec
----------------+------------------------+---------------------------------------------
↪-------------------+-----------------------------------------------------------------
↪----
 slurm_metadata | script-v1.0.schema.json | /global/u1/s/siddiq90/cache/tests/
↪metadata/slurm_metadata.sh | /global/u1/s/siddiq90/buildtest-cori/slurm/valid_jobs/
↪metadata.yml

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

[slurm_metadata] job dispatched to scheduler
[slurm_metadata] acquiring job id in 2 seconds
 name           | executor    | status    |   returncode | testpath
----------------+-------------+-----------+--------------+-----------------------------
↪---------------------------------
 slurm_metadata | slurm.debug | N/A       |            0 | /global/u1/s/siddiq90/cache/
↪tests/metadata/slurm_metadata.sh


Polling Jobs in 10 seconds
________________________________________
[slurm_metadata]: JobID 32740760 in PENDING state


Polling Jobs in 10 seconds
________________________________________
[slurm_metadata]: JobID 32740760 in COMPLETED state


Polling Jobs in 10 seconds
________________________________________

+---------------------------------------------+
| Stage: Final Results after Polling all Jobs |
+---------------------------------------------+

 name           | executor    | status    |   returncode | testpath
----------------+-------------+-----------+--------------+-----------------------------
↪---------------------------------
 slurm_metadata | slurm.debug | PASS      |            0 | /global/u1/s/siddiq90/cache/
↪tests/metadata/slurm_metadata.sh

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

(continues on next page)

(continued from previous page)

```
Executed 1 tests
Passed Tests: 1/1 Percentage: 100.000%
Failed Tests: 0/1 Percentage: 0.000%
```

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
- 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`.

```
version: "1.0"
buildspecs:
  wall_timeout:
    type: script
    executor: slurm.debug
    sbatch: [ "-t 2", "-C haswell", "-n 1"]
    run: exit 1
    status:
      slurm_job_state: "FAILED"
```

If we run this test, buildtest will mark this test as `PASS` because the slurm job state matches with expected result even though returncode is 1.

```
+-------------------------------------------+
| Stage: Final Results after Polling all Jobs |
+-------------------------------------------+

 name          | executor     | status    |   returncode | testpath
---------------+--------------+-----------+--------------+--------------------------------
 ↪--------------------------
```

(continues on next page)

(continued from previous page)

```
 wall_timeout | slurm.debug | PASS     |            1 | /global/u1/s/siddiq90/cache/
↪tests/exit1/wall_timeout.sh
```

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:

```
2020-07-22 18:20:48,170 [base.py:587 - gather() ] - [DEBUG] Gather slurm job data by␣
↪running: sacct -j 32741040 -X -n -P -o Account,AllocNodes,AllocTRES,
↪ConsumedEnergyRaw,CPUTimeRaw,End,ExitCode,JobID,JobName,NCPUS,NNodes,QOS,ReqGRES,
↪ReqMem,ReqNodes,ReqTRES,Start,State,Submit,UID,User,WorkDir -M cori
...
2020-07-22 18:20:48,405 [base.py:598 - gather() ] - [DEBUG] field: State   value:␣
↪FAILED
```

LSF Executor (Experimental)

The **LSFExecutor** is responsible for submitting jobs to LSF scheduler. The LSFExecutor behaves similar to SlurmExecutor with the five stages implemented as class methods:

- Check: check lsf binaries (`bsub`, `bjobs`)
- Load: load lsf executor from buildtest configuration `config.yml`
- Dispatch: Dispatch job using bsub and retrieve JobID
- Poll: Poll job using `bjobs` to retrieve job state
- Gather: Retrieve job results once job is finished

The `bsub` key works similar to `sbatch` key which allows one to specify **#BSUB** directive into job script. This example will use the `lsf.batch` executor with executor name `batch` defined in buildtest configuration.

```
version: "1.0"
buildspecs:
  hostname:
    type: script
    executor: lsf.batch
    bsub: [ "-W 10",  "-nnodes 1"]

    run: jsrun hostname
```

The LSFExecutor `poll` method will 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 proceed to `gather` stage and acquire job results.
The LSFExecutor `gather` method 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

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
account	–account	-P
begin	–begin	-b
cpucount	–ntasks	-n
email-address	–mail-user	-u
exclusive	–exclusive=user	-x
memory	–mem	-M
network	–network	-network
nodecount	–nodes	-nnodes
qos	–qos	N/A
queue	–partition	-q
tasks-per-core	–ntasks-per-core	N/A
tasks-per-node	–ntasks-per-node	N/A
tasks-per-socket	–ntasks-per-socket	N/A
timelimit	–time	-W

In this example, we rewrite the LSF buildspec to use `batch` instead of `bsub` field:

```
version: "1.0"
buildspecs:
  hostname:
    type: script
    executor: lsf.batch
    batch:
      timelimit: "10"
      nodecount: "1"
    run: jsrun hostname
```

buildtest will translate the batch field into #BSUB directive as you can see in the generated test:

```
#!/usr/bin/bash
#BSUB -W 10
#BSUB -nnodes 1
source /autofs/nccsopen-svm1_home/shahzebsiddiqui/buildtest/var/executors/lsf.batch/
↪before_script.sh
jsrun hostname
```

In next example we use `batch` field with on a Slurm cluster that submits a sleep job as follows:

```
version: "1.0"
buildspecs:
  sleep:
    type: script
    executor: slurm.normal
    description: sleep 2 seconds
    tags: [tutorials]
    batch:
```

(continues on next page)

(continued from previous page)

```
      nodecount: "1"
      cpucount: "1"
      timelimit: "5"
      memory: "5MB"
      exclusive: true

    vars:
      SLEEP_TIME: 2
    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
source /home1/06908/sms1990/buildtest/var/executors/slurm.normal/before_script.sh
SLEEP_TIME=2
sleep $SLEEP_TIME
```

You may leverage `batch` with `sbatch` or `bsub` field to specify your job directives. If a particular field is not available in `batch` property then utilize `sbatch` or `bsub` 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, if job is pending then job will be cancelled. To demonstrate, here is an example of two buildspecs submitted to scheduler and notice job `shared_qos_haswell_hostname` was cancelled during after *max_pend_time* of 10 sec. 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`).

```
$ buildtest build -b queues/shared.yml -b queues/xfer.yml

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/global/u1/s/siddiq90/buildtest-cori/queues/xfer.yml
/global/u1/s/siddiq90/buildtest-cori/queues/shared.yml

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile               | validstate   | buildspec
--------------------------+--------------+-------------------------------------------
 script-v1.0.schema.json | True         | /global/u1/s/siddiq90/buildtest-cori/queues/
↪xfer.yml
 script-v1.0.schema.json | True         | /global/u1/s/siddiq90/buildtest-cori/queues/
↪shared.yml
```

(continues on next page)

(continued from previous page)

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

 name                          | id       | type    | executor     | tags                ␣
↪   | testpath
------------------------------+----------+---------+--------------+---------------------
↪--+-------------------------------------------------------------------------------------
↪-------------------------------
 xfer_qos_hostname             | d0043be3 | script | slurm.xfer    | ['queues']          ␣
↪   | /global/u1/s/siddiq90/buildtest/var/tests/slurm.xfer/xfer/xfer_qos_hostname/1/
↪stage/generate.sh
 shared_qos_haswell_hostname | 9d3723ac | script | slurm.shared | ['queues', 'reframe
↪'] | /global/u1/s/siddiq90/buildtest/var/tests/slurm.shared/shared/shared_qos_
↪haswell_hostname/1/stage/generate.sh

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

[xfer_qos_hostname] JobID: 1089664 dispatched to scheduler
[shared_qos_haswell_hostname] JobID: 35189528 dispatched to scheduler
 name                          | id       | executor     | status    |   returncode |␣
↪testpath
------------------------------+----------+--------------+-----------+--------------+----
↪-------------------------------------------------------------------------------------
↪------------------------
 xfer_qos_hostname             | d0043be3 | slurm.xfer    | N/A       |            0 | /
↪global/u1/s/siddiq90/buildtest/var/tests/slurm.xfer/xfer/xfer_qos_hostname/1/stage/
↪generate.sh
 shared_qos_haswell_hostname | 9d3723ac | slurm.shared | N/A       |            0 | /
↪global/u1/s/siddiq90/buildtest/var/tests/slurm.shared/shared/shared_qos_haswell_
↪hostname/1/stage/generate.sh


Polling Jobs in 10 seconds
_____________________________________________
[xfer_qos_hostname]: JobID 1089664 in COMPLETED state
[shared_qos_haswell_hostname]: JobID 35189528 in PENDING state

Polling Jobs in 10 seconds
_____________________________________________
[shared_qos_haswell_hostname]: JobID 35189528 in PENDING state
Cancelling Job: shared_qos_haswell_hostname running command: scancel 35189528
Cancelling Job because duration time: 20.573901 sec exceeds max pend time: 10 sec


Polling Jobs in 10 seconds
_____________________________________________
Cancelled Tests:
shared_qos_haswell_hostname

+-----------------------------------------------+
| Stage: Final Results after Polling all Jobs |
+-----------------------------------------------+

```

(continues on next page)

(continued from previous page)

```
 name                | id       | executor    | status    |   returncode | testpath
--------------------+----------+-------------+-----------+--------------+----------------
↪-------------------------------------------------------------------------------
 xfer_qos_hostname | d0043be3 | slurm.xfer | PASS      |            0 | /global/u1/s/
↪siddiq90/buildtest/var/tests/slurm.xfer/xfer/xfer_qos_hostname/1/stage/generate.sh

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

Executed 1 tests
Passed Tests: 1/1 Percentage: 100.000%
Failed Tests: 0/1 Percentage: 0.000%
```

5.7.5 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. All sub-schemas must be versioned, with the exception of `global.schema.json`.

Schema Examples

The schema examples are great way to help write your buildspecs 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
```

If you want to validate the schema examples you can run:

```
buildtest schema -n <schema> --validate
```

You may combine `--examples` and `--validate` option if you want to view and validate schema examples.

Schema - definitions.schema.json

```
$ 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": [
```

(continues on next page)

(continued from previous page)

```
        {
          "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",
        "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 buildspec␣
↪file, if ``skip: True`` is set it will skip the buildspec."
    },
    "executor": {
      "type": "string",
```

(continues on next page)

(continued from previous page)

```
      "description": "Select one of the executor name defined in your configuration␣
↪file (``config.yml``). Every buildspec must have an executor which is responsible␣
↪for running job. "
    },
    "sbatch": {
      "type": "array",
      "description": "This field is used for specifying #SBATCH options in test␣
↪script. buildtest will insert #SBATCH in front of each value",
      "items": {
        "type": "string"
      }
    },
    "bsub": {
      "type": "array",
      "description": "This field is used for specifying #BSUB options in test script.␣
↪buildtest will insert #BSUB in front of each value",
      "items": {
        "type": "string"
      }
    },
    "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"
          ]
        },
        "user": {
          "type": "string",
          "description": "Test will run only if current user matches this field,␣
↪otherwise test will be skipped"
        },
        "platform": {
          "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",
```

(continues on next page)

(continued from previous page)

```
            "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": {
          "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",
```

(continues on next page)

(continued from previous page)

```
          "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": {
        "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": [
```

(continues on next page)

(continued from previous page)

```
                "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"
          ]
        }
      }
    }
  }
}
```

Schema - global.schema.json

```
$ 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 defintions 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)."
    },
    "maintainers": {
      "type": "array",
      "description": "One or more maintainers or aliases",
      "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",
```

(continues on next page)

(continued from previous page)

```
      "propertyNames": {
        "pattern": "^[A-Za-z_][A-Za-z0-9_]*$",
        "maxLength": 32
      }
    }
  }
}
```

Schema Examples - global.schema.json

```
$ buildtest schema -n global.schema.json --example
File: /Users/siddiq90/Documents/buildtest/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"

  # testing '_' at end of word
  abcdefghijklmnopqrstuvwxyz_:
    type: script
    run: "hostname"
File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/global.schema.
↪json/invalid/maintainers_type_mismatch.yml
--------------------------------------------------------------------------------
version: "1.0"
# wrong type for maintainers key, expects a string
maintainers: 1
buildspecs:
  hostname:
    type: script
    run: "hostname"
File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/global.schema.
↪json/invalid/invalid_pattern.yml
--------------------------------------------------------------------------------
version: "1.0"
```

(continues on next page)

(continued from previous page)

```
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"
File: /Users/siddiq90/Documents/buildtest/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"
```

Schema - script-v1.0.schema.json

```
$ 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": {
      "$ref": "definitions.schema.json#/definitions/sbatch"
    },
    "bsub": {
      "$ref": "definitions.schema.json#/definitions/bsub"
    },
    "batch": {
      "$ref": "definitions.schema.json#/definitions/batch"
    },
    "env": {
      "$ref": "definitions.schema.json#/definitions/env"
    },
    "vars": {
      "$ref": "definitions.schema.json#/definitions/env"
```

(continues on next page)

(continued from previous page)

```
    },
    "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|sh|bash|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",
      "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 - script-v1.0.schema.json

```
$ buildtest schema -n script-v1.0.schema.json --example
File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/script-v1.0.
↪schema.json/valid/examples.yml
--------------------------------------------------------------------------------
version: "1.0"
buildspecs:
  multiline_run:
    executor: local.bash
    type: script
    description: multiline run command
    run: |
      echo "1"
      echo "2"

  single_command_run:
    executor: local.bash
    type: script
    description: single command as a string for run command
```

(continues on next page)

(continued from previous page)

```
  run: "hostname"

declare_env:
  executor: local.bash
  type: script
  description: declaring environment variables
  env:
    FOO: BAR
    X: 1
  run: |
    echo $FOO
    echo $X

declare_vars:
  executor: local.bash
  type: script
  description: declaring variables
  vars:
    First: Bob
    Last:  Bill
  run: |
    echo "First:" $First
    echo "Last:" $Last


declare_shell_sh:
  executor: local.sh
  type: script
  description: declare shell name to sh
  shell: sh
  run: hostname

declare_shell_bash:
  executor: local.bash
  type: script
  description: declare shell name to bash
  shell: bash
  run: hostname

declare_shell_python:
  executor: local.python
  type: script
  description: declare shell name to python
  shell: python
  run: |
    print("Hello World")

declare_shell_bin_bash:
  executor: local.bash
  type: script
  description: declare shell name to /bin/bash
  shell: "/bin/bash -e"
  run: hostname

declare_shell_name_bin_sh:
  executor: local.sh
  type: script
```

(continues on next page)

(continued from previous page)

```
  description: declare shell name to /bin/sh
  shell: "/bin/sh -e"
  run: hostname


declare_shell_opts:
  executor: local.sh
  type: script
  description: declare shell name to sh
  shell: "sh -e"
  run: hostname

declare_shebang:
  executor: local.bash
  type: script
  description: declare shell name to sh
  shebang: "#!/usr/bin/env bash"
  run: hostname

status_returncode_list:
  executor: local.bash
  type: script
  description: The returncode can be a list of integers
  run: exit 0
  status:
    returncode: [0]

status_returncode_int:
  executor: 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: 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: local.bash
  type: script
  description: This test fails because returncode and regex specified
  run: hostname
  status:
    returncode: [0]
    regex:
      stream: stdout
      exp: "^hello"

sbatch_example:
```

(continues on next page)

(continued from previous page)

```
    type: script
    executor: local.bash
    description: This test pass sbatch options in test.
    sbatch:
      - "-t 10:00:00"
      - "-p normal"
      - "-N 1"
      - "-n 8"
    run: hostname

  bsub_example:
    type: script
    executor: local.bash
    description: This test pass bsub options in test.
    bsub:
      - "-W 00:30"
      - "-N 1"
    run: hostname

  skip_example:
    type: script
    executor: local.bash
    description: this test is skip
    skip: true
    run: hostname

  tag_str_example:
    type: script
    executor: local.bash
    description: tags can be defined as string
    tags: network
    run: hostname

  tag_list_example:
    type: script
    executor: 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: 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
```

(continues on next page)

(continued from previous page)

```
    run: |
      uname -av
      lsinfo
File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/script-v1.0.
↪schema.json/invalid/examples.yml
_____________________________________________________________________________
version: "1.0"
buildspecs:
  invalid_test_name_&!@#$%:
    type: script
    executor: local.bash
    description: "invalid test name"

  invalid_bash:
    type: script
    executor: local.bash
    shell: "bash-missing-run"

  missing_run_key:
    type: script
    executor: 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: local.bash
    description: env key should be a dictionary
    env:
      - FOO=BAR
    run: echo $FOO

  invalid_vars_type:
    type: script
    executor: local.bash
    description: var key should be a dictionary
    vars:
      - FOO=BAR
    run: echo $FOO


  invalid_description:
    type: script
    executor: local.bash
    description:
      - "Multi Line description"
      - "is not accepted"

  invalid_regex_stream:
    type: script
    executor: local.bash
    description: This test fails because of invalid regex stream
    run: hostname
    status:
      regex:
```

(continues on next page)

(continued from previous page)

```
        stream: file
        exp: "world$"

  missing_regex_exp:
    type: script
    executor: 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: local.bash
    description: This test fails because of invalid return code type
    run: hostname
    status:
      returncode: ["1"]

  empty_returncode_list:
    type: script
    executor: local.bash
    description: An empty returncode list will cause an error
    run: hostname
    status:
      returncode: []

  non_int_returncodes:
    type: script
    executor: 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: 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: local.bash
    description: invalid shell name, since we only support 'sh', 'bash', 'python' '/
↪bin/bash' /bin/sh
    shell: /usr/bin/bash
    run: hostname

  invalid_shell_type:
    type: script
    executor: local.bash
    description: invalid shell type must be a string
    shell: ["/bin/bash"]
```

(continues on next page)

(continued from previous page)

```
    run: hostname

  invalid_type_shell_shebang:
    type: script
    executor: local.bash
    description: invalid type for shell shebang, must be a string
    shebang: ["#!/bin/bash"]
    run: hostname

  invalid_skip_value:
    type: script
    executor: local.bash
    description: invalid value for skip, must be boolean
    skip: 1
    run: hostname

  empty_tags:
    type: script
    executor: 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: local.bash
    description: invalid tag value must be all string items
    tags: ["network", 400 ]
    run: hostname

  additionalProperties_test:
    type: script
    executor: 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: slurm.debug
    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_slurm_job_state:
    type: script
```

(continues on next page)

(continued from previous page)

```
    executor: slurm.debug
    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: 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: 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: local.bash
    description: additional Properties not allowed in run_only field. Invalid field␣
↪python
    run_only:
      user: root
      python: 3.5
    run: hostname
```

Schema - compiler-v1.0.schema.json

```
$ 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",
    "build",
    "executor"
  ],
  "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"
    },
    "module": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "description": "A list of modules to load into test script"
    },
    "executor": {
      "$ref": "definitions.schema.json#/definitions/executor"
    },
    "sbatch": {
      "$ref": "definitions.schema.json#/definitions/sbatch"
    },
    "bsub": {
      "$ref": "definitions.schema.json#/definitions/bsub"
    },
    "batch": {
      "$ref": "definitions.schema.json#/definitions/batch"
    },
    "env": {
      "$ref": "definitions.schema.json#/definitions/env"
    },
    "vars": {
      "$ref": "definitions.schema.json#/definitions/env"
    },
    "run_only": {
      "$ref": "definitions.schema.json#/definitions/run_only"
    },
    "status": {
      "$ref": "definitions.schema.json#/definitions/status"
    },
    "skip": {
      "$ref": "definitions.schema.json#/definitions/skip"
    },
    "tags": {
      "$ref": "definitions.schema.json#/definitions/tags"
    },
    "pre_build": {
      "type": "string",
      "description": "Run commands before building program"
    },
    "post_build": {
      "type": "string",
      "description": "Run commands after building program"
    },
    "build": {
      "type": "object",
```

(continues on next page)

(continued from previous page)

```
      "description": "The ``build`` section is used for compiling a single program,␣
↪this section specifies fields for setting C, C++, Fortran compiler and flags␣
↪including CPP flags and linker flags",
      "properties": {
        "name": {
          "type": "string",
          "enum": [
            "gnu",
            "intel",
            "pgi",
            "cray"
          ],
          "description": "Select the compiler class to use, buildtest will set cc,␣
↪cxx, and fc compiler wrapper based on compiler name"
        },
        "cc": {
          "type": "string",
          "description": "Set C compiler. Use this field to override buildtest␣
↪selection for **cc**"
        },
        "fc": {
          "type": "string",
          "description": "Set Fortran compiler. Use this field to override buildtest␣
↪selection for **fc**"
        },
        "cxx": {
          "type": "string",
          "description": "Set C++ compiler. Use this field to override buildtest␣
↪selection for **cxx**"
        },
        "source": {
          "type": "string",
          "description": "Specify a source file for compilation, the file can be␣
↪relative path to buildspec or an absolute path"
        },
        "cflags": {
          "type": "string",
          "description": "Set C compiler flags (**cflags**)"
        },
        "cxxflags": {
          "type": "string",
          "description": "Set C++ compiler flags (**cxxflags**)"
        },
        "fflags": {
          "type": "string",
          "description": "Set Fortran compiler flags (**fflags**)"
        },
        "cppflags": {
          "type": "string",
          "description": "Set Pre Processor Flags (**cppflags**)"
        },
        "ldflags": {
          "type": "string",
          "description": "Set linker flags (**ldflags**)"
        }
      },
      "required": [
```

(continues on next page)

(continued from previous page)

```
        "source",
        "name"
      ],
      "additionalProperties": false
    },
    "pre_run": {
      "type": "string",
      "description": "Run commands before running program"
    },
    "post_run": {
      "type": "string",
      "description": "Run commands after running program"
    },
    "run": {
      "type": "object",
      "description": "The ``run`` section is used for specifying launch configuration␣
↪of executable",
      "properties": {
        "launcher": {
          "type": "string",
          "description": "The ``launcher`` field is inserted before the executable.␣
↪This can be used when running programs with ``mpirun``, ``mpiexec``, ``srun``, etc..
↪. You may specify launcher options with this field"
        },
        "args": {
          "type": "string",
          "description": "The ``args`` field is used to specify arguments to␣
↪executable."
        }
      },
      "additionalProperties": false
    }
  }
}
```

Schema Examples - compiler-v1.0.schema.json

```
$ buildtest schema -n compiler-v1.0.schema.json --example
File: /Users/siddiq90/Documents/buildtest/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"
    module:
       - "module purge && module load gcc/4.0"
       - "module purge && module load gcc/6.0"
    build:
      name: gnu
      source: src/hello.c
      cflags: "-O1"
```

(continues on next page)

(continued from previous page)

```
intel_example:
  executor: local.bash
  type: compiler
  description: "intel example using cflags"
  module:
    -  "module purge &&  module load intel/17"
    -  "module purge &&  module load intel/18"
  build:
    name: intel
    source: src/hello.c
    cflags: "-O1"

pgi_example:
  executor: local.bash
  type: compiler
  description: "pgi example using cxxflags, ldflags key"
  module:
    -  "module purge &&  module load pgi"
  build:
    source: src/hello.cpp
    name: pgi
    cxxflags: "-O1"
    ldflags: "-lm"


cray_example:
  executor: local.bash
  type: compiler
  description: "cray example using fflags and cppflags"
  sbatch: ["-C knl", "-q normal", "-t 01:00"]
  build:
    name: cray
    source: src/hello.f90
    fflags: "-O1"
    cppflags: "-DFOO"

cc_example:
  type: compiler
  description: Example by using cc to set C compiler
  executor: local.bash
  build:
    source: "src/hello.c"
    name: gnu
    cc: gcc

fc_example:
  type: compiler
  description: Example by using fc to set Fortran compiler
  executor: local.bash
  build:
    source: "src/hello.f90"
    name: gnu
    fc: gfortran

cxx_example:
  type: compiler
  description: Example by using cxx to set C++ compiler
```

(continues on next page)

(continued from previous page)

```
    executor: local.bash
    build:
      source: "src/hello.cpp"
      name: gnu
      cxx: g++

  sbatch_example:
    type: compiler
    description: sbatch example to configure #SBATCH options
    executor: local.bash
    sbatch:
      - "-t 10"
      - "-n 2"
    build:
      source: "src/hello.cpp"
      name: gnu
      cxx: g++
    status:
      slurm_job_state: "COMPLETED"

  bsub_example:
    type: compiler
    description: bsub example to configure #BSUB options
    executor: local.bash
    bsub:
      - "-W 00:30"
      - "-N 2"
    build:
      source: "src/hello.cpp"
      name: gnu
      cxx: g++

  batch_example:
    type: compiler
    description: example using batch field
    executor: local.bash
    batch:
      "timelimit": "30"
      "nodecount": "2"
      "queue": "batch"
      "account": "biology"
    build:
      source: "src/hello.cpp"
      name: gnu
      cxx: g++


  batch_bsub_example:
    type: compiler
    description: example using batch with bsub key
    executor: local.bash
    batch:
      "timelimit": "30"
      "nodecount": "2"
      "queue": "batch"
      "account": "biology"
    bsub: ["-n 4"]
```

(continues on next page)

(continued from previous page)

```
  build:
    source: "src/hello.cpp"
    name: gnu
    cxx: g++


batch_sbatch_example:
  type: compiler
  description: example using batch with sbatch key
  executor: local.bash
  batch:
    "timelimit": "30"
    "nodecount": "2"
    "queue": "batch"
    "account": "biology"
  sbatch: ["--ntasks=4"]
  build:
    source: "src/hello.cpp"
    name: gnu
    cxx: g++


args_example:
  type: compiler
  description: Launcher example
  executor: local.bash
  build:
    source: "src/hello.cpp"
    name: gnu
  run:
    args: "1 2 4"

mpi_launcher_example:
  type: compiler
  description: Launcher example
  executor: local.bash
  build:
    source: "src/hello.cpp"
    name: gnu
    cxx: mpicxx
    cxxflags: "-O3"
  run:
    launcher: mpirun -np 2

pre_post_build_run_sections:
  type: compiler
  description: Run commands pre and post build and run section
  executor: local.bash
  pre_build: echo "pre-build section"

  build:
    source: "src/hello.cpp"
    name: gnu
    cxx: mpicxx
    cxxflags: "-O3"

  post_build: echo "post-build section"
```

(continues on next page)

(continued from previous page)

```
    pre_run: echo "pre-run section"
    run:
      launcher: mpirun -np 2
    post_run: echo "post-run section"

File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/compiler-v1.0.
↪schema.json/invalid/examples.yml
_____________________________________________________________________________
version: "1.0"
buildspecs:
  missing_type:
    executor: local.bash
    description: "type key is missing, this is a required field"
    module:
     -  "module purge &&  module load intel/17"
     -  "module purge &&  module load intel/18"
    build:
      source: src/hello.c
      name: intel
      cflags: "-O1"

  missing_build:
    executor: local.bash
    type: compiler
    description: "build key is missing, this is a required field"
    module:
     -  "module purge &&  module load intel/17"
     -  "module purge &&  module load intel/18"

  invalid_type_value:
    executor: local.bash
    type: script
    description: "invalid value for type field must be 'compiler' "
    module:
       - "module purge && module load gcc/4.0"
       - "module purge && module load gcc/6.0"
    build:
      source: src/hello.c
      name: gnu
      cflags: "-O1"

  invalid_description_value:
    executor: local.bash
    type: compiler
    description: 1
    module:
       - "module purge && module load gcc/4.0"
       - "module purge && module load gcc/6.0"
    build:
      source: src/hello.c
      name: gnu
      cflags: "-O1"

  invalid_type_module:
    executor: local.bash
    type: compiler
    description: "type for 'module' key, expecting type 'array' but received 'string'
↪"
```

(continues on next page)

(continued from previous page)

```
    module: "module purge && module load gcc/4.0"
    build:
      source: src/hello.c
      name: gnu
      cflags: "-O1"

  module_mismatch_array_items:
    executor: local.bash
    type: compiler
    description: "The module is an array of string items, this test as a mix of␣
↪numbers and string"
    module:
      - 1
      - "module purge && module load intel"
    build:
      source: src/hello.c
      name: intel
      cflags: "-O1"

  missing_source_in_compiler:
    executor: local.bash
    type: compiler
    description: "missing source key in compiler object"
    module:
    - "module purge && module load gcc/4.0"
    build:
      name: gnu
      cflags: "-O1"

  missing_name_in_build:
    executor: local.bash
    type: compiler
    description: "missing name key in build object"
    module:
    - "module purge && module load gcc/4.0"
    build:
      source: src/hello.c

  name_type_mismatch:
    executor: local.bash
    type: compiler
    description: "compiler 'name' expects a string but received a list"
    module:
    - "module purge && module load gcc/4.0"
    build:
      source: src/hello.c
      name: ["gnu", "intel"]
      cflags: "-O1"
      ldflags: "-lm"


  additionalProperties_build:
    executor: local.bash
    type: compiler
    description: "test additionalProperties in build object. Schema does not allow␣
↪for additional keys"
    module:
```

(continues on next page)

(continued from previous page)

```
  - "module purge && module load gcc/4.0"
  build:
    source: src/hello.c
    foo: bar
    name: gnu
    cflags: "-O1"
    ldflags: "-lm"

additionalProperties_main:
  executor: local.bash
  type: compiler
  description: "test additionalProperties in main schema"
  foo: bar
  module:
  - "module purge && module load gcc/4.0"
  build:
    source: src/hello.c
    name: gnu
    cflags: "-O1"
    ldflags: "-lm"

additionalProperties_batch:
  executor: local.bash
  type: compiler
  description: "test additionalProperties in batch field"
  module:
  - "module purge && module load gcc/4.0"
  batch:
    "nodecount": "2"
    "EXPORT": "ALL"
  build:
    source: src/hello.c
    name: gnu
    cflags: "-O1"
    ldflags: "-lm"

type_mismatch_args:
  executor: local.bash
  type: compiler
  description: "type mismatch on args key"
  module:
  - "module purge && module load gcc/4.0"
  build:
    source: src/hello.c
    name: gnu
    cflags: "-O1"
    ldflags: "-lm"

  run:
    args: 1
```

5.8 Scripting in buildtest

This guide will walk you through on how to script with buildtest.

5.8.1 Discovering Buildspecs

Let's take this first example where we discover all buildspecs found in top-level `tutorials` directory.

```
import os
from buildtest.defaults import BUILDTEST_ROOT
from buildtest.menu.build import discover_buildspecs

included_bp, excluded_bp = discover_buildspecs(
    buildspec=[os.path.join(BUILDTEST_ROOT, "tutorials")]
)
print("\n Discovered buildspecs: \n")
[print(f) for f in included_bp]

print("\n Excluded buildspecs: \n")
[print(f) for f in excluded_bp]
```

The variable `BUILDTEST_ROOT` is the root of buildtest and typically setup once you install buildtest. The `discover_buildspecs` method can be invoked to retrieve a list of buildspecs discovered. The method will return two list, one for discovered and excluded buildspecs.
Now let's run this example and note we see all buildspecs in **tutorials** directory were retrieved. This is equivalent to running `buildtest build --buildspec tutorials`.

```
 Discovered buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/python-hello.yml
/Users/siddiq90/Documents/buildtest/tutorials/run_only_platform.yml
/Users/siddiq90/Documents/buildtest/tutorials/systemd.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/pre_post_build_run.yml
/Users/siddiq90/Documents/buildtest/tutorials/skip_tests.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/vecadd.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/passing_args.yml
/Users/siddiq90/Documents/buildtest/tutorials/invalid_executor.yml
/Users/siddiq90/Documents/buildtest/tutorials/shebang.yml
/Users/siddiq90/Documents/buildtest/tutorials/environment.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml
/Users/siddiq90/Documents/buildtest/tutorials/root_user.yml
/Users/siddiq90/Documents/buildtest/tutorials/run_only_distro.yml
/Users/siddiq90/Documents/buildtest/tutorials/python-shell.yml
/Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
/Users/siddiq90/Documents/buildtest/tutorials/selinux.yml
/Users/siddiq90/Documents/buildtest/tutorials/invalid_tags.yml
/Users/siddiq90/Documents/buildtest/tutorials/hello_world.yml
/Users/siddiq90/Documents/buildtest/tutorials/sleep.yml
/Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml
/Users/siddiq90/Documents/buildtest/tutorials/tags_example.yml
/Users/siddiq90/Documents/buildtest/tutorials/invalid_buildspec_section.yml
/Users/siddiq90/Documents/buildtest/tutorials/vars.yml

 Excluded buildspecs:
```

We can also discover buildspecs by tags, in next example we discover all buildspecs by **tutorials** tag. This can be done by passing a tagname for argument **tags** in *discover_buildspecs* method.
This is equivalent to running `buildtest build --tags tutorials`.

```
from buildtest.menu.build import discover_buildspecs

tagname = ["tutorials"]
print(f"Searching by tagname: {tagname}")
included_bp, excluded_bp = discover_buildspecs(tags=tagname)
print("\n Discovered buildspecs: \n")
[print(f) for f in included_bp]
```

Note: You must have a buildspec cache in order to discover tags (`buildtest buildspec find`)

Now let's run this test

```
Searching by tagname: ['tutorials']

 Discovered buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/hello_world.yml
/Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml
/Users/siddiq90/Documents/buildtest/tutorials/run_only_platform.yml
/Users/siddiq90/Documents/buildtest/tutorials/vars.yml
/Users/siddiq90/Documents/buildtest/tutorials/root_user.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/vecadd.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/pre_post_build_run.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/passing_args.yml
/Users/siddiq90/Documents/buildtest/tutorials/skip_tests.yml
/Users/siddiq90/Documents/buildtest/tutorials/shebang.yml
/Users/siddiq90/Documents/buildtest/tutorials/selinux.yml
/Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
/Users/siddiq90/Documents/buildtest/tutorials/python-shell.yml
/Users/siddiq90/Documents/buildtest/tutorials/environment.yml
/Users/siddiq90/Documents/buildtest/tutorials/sleep.yml
/Users/siddiq90/Documents/buildtest/tutorials/systemd.yml
```

You can combine tags and buildspecs with *discover_buildspecs* method and buildtest will combine the results.

5.8.2 Build Phase

Now that we can find buildspecs, let's try to parse and build the tests. In next example we will *discover*, *parse*, and *build* all tests with tag name **tutorials**.

```
from buildtest.config import load_settings
from buildtest.menu.build import discover_buildspecs, resolve_testdirectory, build_
↪phase
from buildtest.menu.buildspec import parse_buildspecs

tagname = ["tutorials"]
print(f"Searching by tagname: {tagname}")
included_bp, excluded_bp = discover_buildspecs(tags=tagname, debug=True)

configuration = load_settings()
testdir = resolve_testdirectory(configuration)
builders = parse_buildspecs(included_bp, testdir, rebuild=1, printTable=True)

build_phase(builders, printTable=True)
```

We retrieve all buildspecs by tag *tutorials* as mentioned in previous example. Next we load buildtest configuration using **load_settings** which returns a dictionary containing buildtest configuration. During this process, we validate

the buildtest configuration.
Next, we need to figure out our test directory in order to write tests. This can be achieved by passing the loaded configuration to method **resolve_testdirectory**. The return will be path to test directory. The test directory can be specified on command line `buildtest build --testdir` or path in configuration. If its not set in configuration we default to `$BUILDTEST_ROOT/var/tests`
Next we invoke `parse_buildspecs` which expects a list of buildspecs, test directory and executor. The `printTable=True` will print parse table of buildspecs that are validated. The *parse_buildspecs* will validate each buildspec, and skip any buildspecs that fail validation. The parser is implemented in class `BuildspecParser`. For all valid buildspecs we return a list of builders that is a list of tests for each buildspec that is an instance of `BuilderBase` class that is responsible for building the test.
Next we pass all builders to `build_phase` method which will generate testscript for each builder. The `printTable=True` will print table for builder phase.

Note: Each builder corresponds to a single test name.

Now let's run this script and notice the output resembles similar to running `buildtest build --tags tutorials` but we stop right after build. In other words this is equivalent to `buildtest build --tags tutorials --stage=build`.

```
Searching by tagname: ['tutorials']

+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
/Users/siddiq90/Documents/buildtest/tutorials/root_user.yml
/Users/siddiq90/Documents/buildtest/tutorials/run_only_platform.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/pre_post_build_run.yml
/Users/siddiq90/Documents/buildtest/tutorials/environment.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/vecadd.yml
/Users/siddiq90/Documents/buildtest/tutorials/shebang.yml
/Users/siddiq90/Documents/buildtest/tutorials/skip_tests.yml
/Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/passing_args.yml
/Users/siddiq90/Documents/buildtest/tutorials/systemd.yml
/Users/siddiq90/Documents/buildtest/tutorials/selinux.yml
/Users/siddiq90/Documents/buildtest/tutorials/python-shell.yml
/Users/siddiq90/Documents/buildtest/tutorials/vars.yml
/Users/siddiq90/Documents/buildtest/tutorials/sleep.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml
/Users/siddiq90/Documents/buildtest/tutorials/hello_world.yml
[run_only_as_root] test is skipped because ['run_only']['user'] got value: root but␣
↪detected user: siddiq90.
[run_only_platform_linux] test is skipped because ['run_only']['platform'] got value:␣
↪Linux but detected platform: Darwin.
[skip] test is skipped.

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile                | validstate   | buildspec
```

(continues on next page)

(continued from previous page)

```
--------------------------+--------------+----------------------------------------
↪-------------------------------------
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/root_user.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/run_only_platform.yml
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/pre_post_build_run.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/environment.yml
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/vecadd.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/shebang.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/skip_tests.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/shell_examples.yml
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/passing_args.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/systemd.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/selinux.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/python-shell.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/vars.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/sleep.yml
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/gnu_hello.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/hello_world.yml

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

 name                      | id       | type     | executor     | tags
↪    | testpath
--------------------------+----------+----------+--------------+----------------------
↪----+-------------------------------------------------------------------------------
↪---------------------------------------------
 exit1_fail                | caeb1cd5 | script   | local.sh     | ['tutorials', 'fail
↪']    | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/
↪exit1_fail/8/stage/generate.sh
 exit1_pass                | 0b2fac5b | script   | local.sh     | ['tutorials', 'pass
↪']    | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/
↪exit1_pass/6/stage/generate.sh
 returncode_list_mismatch | 653f6fae | script   | local.sh     | ['tutorials', 'fail
↪']    | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/
↪returncode_list_mismatch/8/stage/generate.sh
 returncode_int_match      | 13d7cc98 | script   | local.sh     | ['tutorials', 'pass
↪']    | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/
↪returncode_int_match/6/stage/generate.sh
```

(continues on next page)

(continued from previous page)

```
run_only_platform_darwin | b27688fd | script   | local.python | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.python/run_only_platform/
↪run_only_platform_darwin/3/stage/generate.sh
pre_post_build_run       | 143a9bb4 | compiler | local.bash   | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/pre_post_
↪build_run/pre_post_build_run/4/stage/generate.sh
environment_variables    | 27625a4e | script   | local.bash   | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/environment/
↪environment_variables/3/stage/generate.sh
vecadd_gnu               | b048e564 | compiler | local.bash   | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/vecadd/vecadd_
↪gnu/5/stage/generate.sh
bash_login_shebang       | 7d4303d1 | script   | local.bash   | tutorials
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_login_
↪shebang/4/stage/generate.sh
bash_nonlogin_shebang    | d87c79c1 | script   | local.bash   | tutorials
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/shebang/bash_
↪nonlogin_shebang/4/stage/generate.sh
unskipped                | 013f04a2 | script   | local.bash   | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/skip_tests/unskipped/
↪4/stage/generate.sh
_bin_sh_shell            | 764fc41a | script   | local.sh     | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/_bin_sh_
↪shell/6/stage/generate.sh
_bin_bash_shell          | be2673cd | script   | local.bash   | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/_bin_
↪bash_shell/5/stage/generate.sh
bash_shell               | cb2805d5 | script   | local.bash   | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/bash_
↪shell/5/stage/generate.sh
sh_shell                 | 5966a1c2 | script   | local.sh     | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/sh_
↪shell/6/stage/generate.sh
shell_options            | 05ffa6cb | script   | local.sh     | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/shell_
↪options/6/stage/generate.sh
executable_arguments     | c6619fcc | compiler | local.bash   | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/passing_args/
↪executable_arguments/4/stage/generate.sh
systemd_default_target   | 5d717ba8 | script   | local.bash   | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/systemd/systemd_
↪default_target/6/stage/generate.sh
selinux_disable          | 1df1ac5f | script   | local.bash   | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/selinux/selinux_
↪disable/3/stage/generate.sh
circle_area              | f0d12e1e | script   | local.python | ['tutorials', 'python
↪']  | /Users/siddiq90/Documents/buildtest/var/tests/local.python/python-shell/
↪circle_area/9/stage/generate.sh
variables                | 3bd1a67a | script   | local.bash   | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/vars/variables/3/
↪stage/generate.sh
sleep                    | c39f3421 | script   | local.bash   | ['tutorials']
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/sleep/sleep/3/stage/
↪generate.sh
hello_f                  | 16de9e5d | compiler | local.bash   | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/
↪hello_f/5/stage/generate.sh
```

(continues on next page)

(continued from previous page)

```
 hello_c                  | c53af412 | compiler | local.bash    | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/
↪hello_c/5/stage/generate.sh
 hello_cplusplus          | 5a599d47 | compiler | local.bash    | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/
↪hello_cplusplus/5/stage/generate.sh
 cc_example               | 0818a978 | compiler | local.bash    | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cc_
↪example/5/stage/generate.sh
 fc_example               | 9b502366 | compiler | local.bash    | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/fc_
↪example/5/stage/generate.sh
 cxx_example              | 79590f19 | compiler | local.bash    | ['tutorials',
↪'compile'] | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/gnu_hello/cxx_
↪example/5/stage/generate.sh
 hello_world              | 342ea38e | script   | local.bash    | tutorials
↪    | /Users/siddiq90/Documents/buildtest/var/tests/local.bash/hello_world/hello_
↪world/3/stage/generate.sh
```

Now you may have guessed it, `--stage=parse` will stop right after the parse stage, in this case we won't invoke *build_phase* method.

5.8.3 Run Phase

In the *Build Phase* example, we discovered and validated buildspecs and built all the tests, but tests were not run. In this example, we will build off this example to run the test. In this example, we demonstrate a script that is emulating the command `buildtest build --buildspec tutorials/pass_returncode.yml`

```
import os
from buildtest.config import load_settings
from buildtest.defaults import BUILDTEST_ROOT
from buildtest.executors.setup import BuildExecutor
from buildtest.menu.build import (
    discover_buildspecs,
    resolve_testdirectory,
    build_phase,
    run_phase,
)
from buildtest.menu.buildspec import parse_buildspecs


input_buildspecs = [os.path.join(BUILDTEST_ROOT, "tutorials", "pass_returncode.yml")]
included_bp, excluded_bp = discover_buildspecs(buildspec=input_buildspecs, debug=True)

configuration = load_settings()
testdir = resolve_testdirectory(configuration)
executor = BuildExecutor(configuration)

print("List of executors: ", executor.executors)

builders = parse_buildspecs(included_bp, testdir, rebuild=1, printTable=True)

build_phase(builders, printTable=True)
run_phase(builders, executor, configuration, printTable=True)
```

In-order to run the tests, we need to initialize the executors defined in buildtest settings see *What is an executor?*. This action is performed in line:

```
executor = BuildExecutor(configuration)
```

The *BuildExecutor* takes an input buildtest settings, and builds a list of executors objects that is responsible for running tests. Next, we parse and build buildspecs by invoking `parse_buildspecs` and `build_phase` as discussed previously. Finally, we invoke `run_phase` which runs the test.

```
+------------------------------+
| Stage: Discovering Buildspecs |
+------------------------------+


Discovered Buildspecs:

/Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
List of executors:  {'local.bash': local.bash, 'local.sh': local.sh, 'local.python':␣
↪local.python}

+---------------------------+
| Stage: Parsing Buildspecs |
+---------------------------+

 schemafile               | validstate   | buildspec
--------------------------+--------------+-----------------------------------------
↪----------------------
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml

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

 name                      | id       | type   | executor   | tags                  |␣
↪testpath
--------------------------+----------+--------+------------+-----------------------+--
↪-----------------------------------------------------------------------------------
↪-------------------------------
 exit1_fail                | 98a2e55c | script | local.sh   | ['tutorials', 'fail'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_fail/9/
↪stage/generate.sh
 exit1_pass                | 73b4fd50 | script | local.sh   | ['tutorials', 'pass'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_pass/7/
↪stage/generate.sh
 returncode_list_mismatch | 87285388 | script | local.sh   | ['tutorials', 'fail'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_
↪list_mismatch/9/stage/generate.sh
 returncode_int_match      | 88197672 | script | local.sh   | ['tutorials', 'pass'] | /
↪Users/siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_
↪int_match/7/stage/generate.sh

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

 name                      | id       | executor   | status    |   returncode | testpath
--------------------------+----------+------------+-----------+--------------+---------
↪-----------------------------------------------------------------------------------
↪------------------------
 exit1_fail                | 98a2e55c | local.sh   | FAIL      |            1 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_fail/9/stage/
↪generate.sh
```

(continues on next page)

(continued from previous page)

```
 exit1_pass               | 73b4fd50 | local.sh    | PASS     |            1 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_pass/7/stage/
↪generate.sh
 returncode_list_mismatch | 87285388 | local.sh    | FAIL     |            2 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_list_
↪mismatch/9/stage/generate.sh
 returncode_int_match     | 88197672 | local.sh    | PASS     |          128 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_int_
↪match/7/stage/generate.sh

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

Executed 4 tests
Passed Tests: 2/4 Percentage: 50.000%
Failed Tests: 2/4 Percentage: 50.000%
```

5.9 Conference and Publications

5.9.1 Talks

Date	Title	Link
Feb 2nd 2020	**buildtest: HPC Software Stack Testing Framework** @ FOSDEM20	PDF, VIDEO
Jan 30th 2020	**buildtest: HPC Software Stack Testing Framework** @ 5thEasybuildUserMeeting	PDF, VIDEO
Nov 18th 2019	**buildtest: A Software Testing Framework with Module Operations for HPC systems** @ SC19 in HUST workshop	PDF
June 22th 2018	**Software Stack Testing with buildtest** @ HPCKP18	PDF
June 15th 2017	**HPC Application Testing Framework - buildtest** @ HPCKP17	PDF

5.9.2 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

5.9.3 Article

- https://www.hpcwire.com/2019/01/17/pfizer-hpc-engineer-aims-to-automate-software-stack-testing/

5.10 Contributing Guide

There are many ways you can help contribute to buildtest that may include

- File an issue with the framework
- Proofread documentation and report or fix issues
- Participate in discussions and join the slack channel
- Increase test coverage of buildtest regression tests.
- Provide feedback on buildtest options.

5.10.1 Maintainers

If you need to get hold of a buildtest maintainer, please contact one of the maintainers.
Maintainers are listed in alphabetical order by last name:

- Shahzeb Siddiqui (@**shahzebsiddiqui**)
- Vanessa Sochat (@**vsoch**)

5.10.2 Contributing Topics

Getting Started

Contribution is not easy, so we created this document to describe how to get you setup so you can contribute back and make everyone's life easier.

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:

```
cd buildtest
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

Feature Branch

Please make sure to create a new branch when adding and new feature. Do not push to `master` or `devel` branch on your fork or upstream.
Create a new branch from `devel` as follows:

```
cd buildtest
git checkout devel
git checkout -b featureX
```

Once you are ready to push to your fork repo do the following:

```
git push origin featureX
```

Once the branch is created in your fork, you can create a PR for the `devel` branch for `upstream` repo (https://github.com/buildtesters/buildtest)

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.

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 Pre-Commit Hook* 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>
```

GitHub Integrations

buildtest has several github integration, including automated checks during PR that maintainers will check during the PR review. You should check results from the buildtest actions that are also typically linked as part of the pull request testing suite.
You will want to make sure code is formatted via black as we have automated checks for python formatting. If you have not setup the black hook check out *Configuring Black Pre-Commit Hook*
If you notice the black linter step in GitHub Actions is failing, make sure you have the right version of black installation.

GitHub Apps

The following apps are configured with buildtest.

- CodeCov - Codecov provides highly integrated tools to group, merge, archive and compare coverage reports
- Coveralls - Coveralls is a web service to help you track your code coverage over time, and ensure that all your new code is fully covered.
- CodeFactor - CodeFactor instantly performs Code Review with every GitHub Commit or PR. Zero setup time. Get actionable feedback within seconds. Customize rules, get refactoring tips and ignore irrelevant issues.
- Snyk - Snyk tracks vulnerabilities in over 800,000 open source packages, and helps protect over 25,000 applications.

Coverage

We use coverage to measure code coverage of buildtest when running regression test. We use CodeCov and Coveralls for displaying 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 and Coveralls

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.
Coveralls is a web service that tracks code coverage similar to Codecov. We use Coveralls GitHub Action to publish coverage report to coveralls from GitHub workflows.

GitHub Actions

buildtest runs a few automated checks via GitHub Actions that can be found in `.github/workflows`

- Black - Black auto-formats Python code, so we let **black** take care of formatting the entire project so you can focus more time in development. The workflow is defined in black.yml.
- urlchecker-action - is a GitHub action to collect and check URLs in project code and documentation. There is an automated check for every issued PR and the workflow is defined in urlchecker.yml

Configuring Black Pre-Commit Hook

To configure pre-commit hook, make sure `pre-commit` is available if not `pip install pre-commit`. The pre-commit is available if you install buildtest dependencies.
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
- 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.

pyflakes

There is an automated test to check for unused imports 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 buildtest source by running:

```
pyflakes buildtest
```

If you see errors, please fix them and wait for CI checks to pass.

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.
- Trafico - is a bot that automatically labels Pull Request depending on their status, during code reviews. The configuration is found in `.github/trafico.yml`.

Building 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 directory `docs`. 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` or `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 get familiar with the coverage report gather in codecov and coveralls . 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 Test with pytest

To run all the tests you can run the following:

```
pytest tests/
```

Some other options can be useful for troubleshooting such as:

```
# print passed test with output
pytest -rP tests

# print all failed tests
pytest -rf tests

# print all test with verbose
pytest -v tests

# print all except Pass tests
pytest -ra tests
```

If you want to run all schema tests you can run via `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

You may want to run coverage report against your test, this can be done by running:

```
coverage run -m pytest tests
```

This can be used with `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                                 2      0      0      0   100%
buildtest/buildsystem/__init__.py                     0      0      0      0   100%
buildtest/buildsystem/base.py                       222     19     76     19    85%
buildtest/buildsystem/schemas/__init__.py             0      0      0      0   100%
buildtest/buildsystem/schemas/utils.py               53      8     26      8    77%
buildtest/config.py                                  65     29     28      5    48%
buildtest/defaults.py                                18      0      0      0   100%
buildtest/exceptions.py                               5      1      2      1    71%
buildtest/log.py                                     18      0      0      0   100%
buildtest/main.py                                    11     11      0      0     0%
buildtest/menu/__init__.py                           62     47      4      0    23%
buildtest/menu/build.py                              62     52     28      0    11%
buildtest/menu/config.py                             35      1     18      0    98%
buildtest/menu/get.py                                31     23     10      0    20%
buildtest/menu/show.py                               17      3      6      3    74%
buildtest/menu/status.py                             11      8      2      0    23%
buildtest/system.py                                  37     37     10      0     0%
buildtest/utils/__init__.py                           0      0      0      0   100%
buildtest/utils/command.py                           49      2     12      3    92%
buildtest/utils/file.py                              46      0     14      2    97%
-----------------------------------------------------------------------------------
TOTAL                                               744    241    236     41    63%
```

You may want to run `coverage report -m` which will show missing line numbers in report. For more details on coverage 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 envrionment 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 Buildspecs 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 defintions.schema.json which are fields need to be reused in other schemas. A JSON definition is defined under `defintions` 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. All incoming PRs should be label by maintainer to help sort through PRs. Trafico and Pull Request bot will label PRs with additional labels, the maintainers are responsible for labeling PRs based on functionality.
6. 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.

7. 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
```

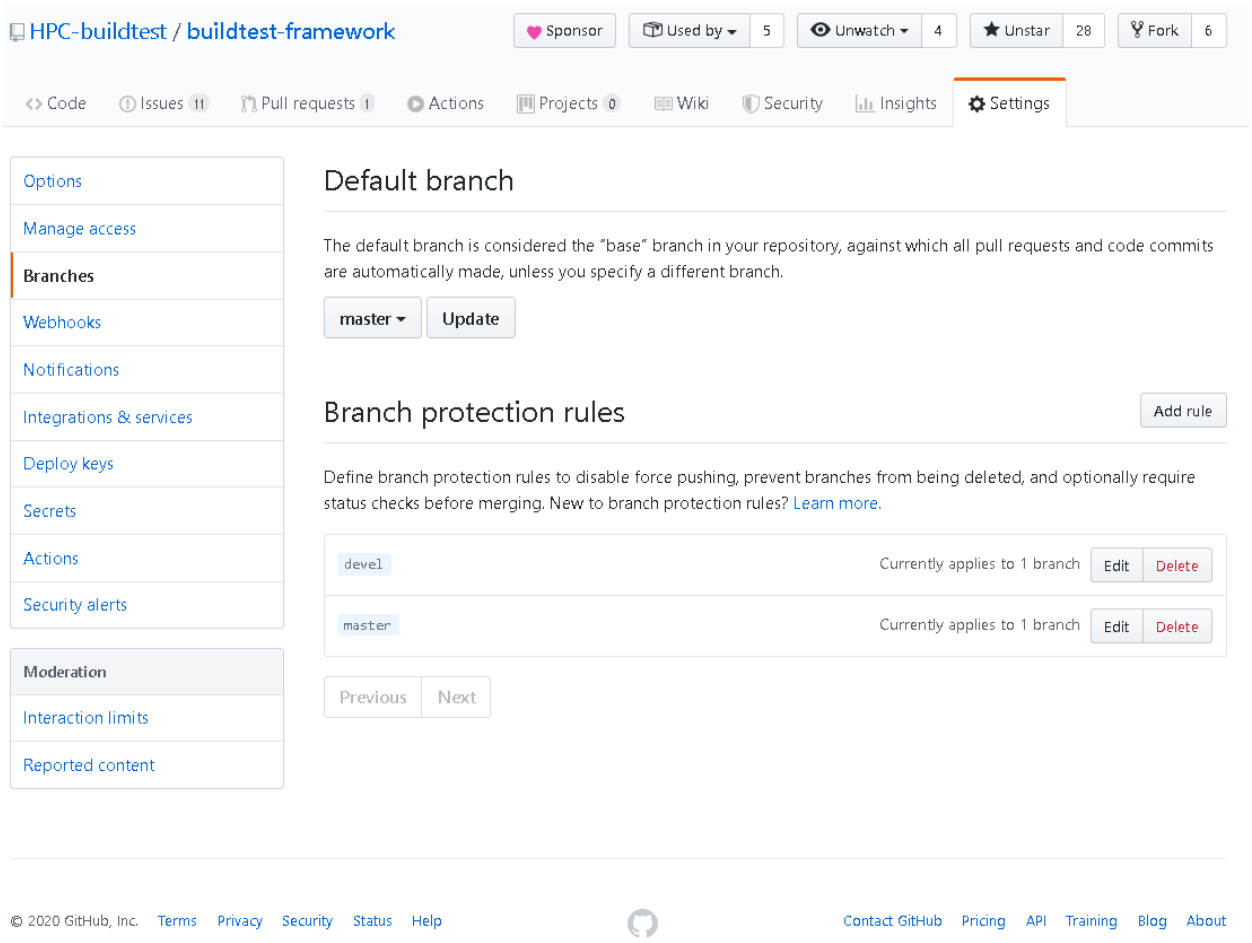
Every release must have a release note that is maintained in file CHANGELOG.rst
Under buildtest releases a new release can be created that corresponds to the git tag. In the release summary, just direct with a message stating **refer to CHANGELOG.rst for more details**
Once the release is published, 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 `master` branch should be setup as the default branch.

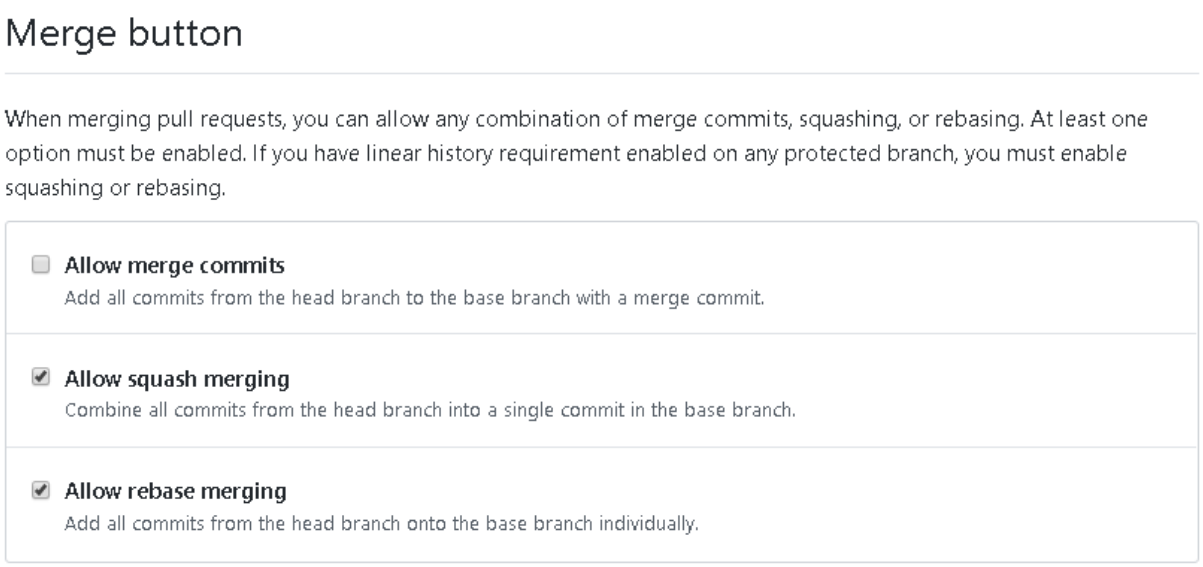
Branch Settings

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.



Merge Settings

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 accidently merge a PR with `Merge Commit` which adds an extra commit.
Shown below is the recommended configuration.



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

5.11 API Reference

This page contains auto-generated API reference documentation[1].

5.11.1 `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'

[1] Created with sphinx-autoapi

Module Contents

Classes

BuilderBase(name, recipe, buildspec, testdir=None)	The BuilderBase is an abstract class that implements common functions for
CompilerBuilder(name, recipe, buildspec, testdir=None)	The BuilderBase is an abstract class that implements common functions for
CrayCompiler(name, recipe, buildspec, testdir=None)	The BuilderBase is an abstract class that implements common functions for
GNUCompiler(name, recipe, buildspec, testdir=None)	The BuilderBase is an abstract class that implements common functions for
IntelCompiler(name, recipe, buildspec, testdir=None)	The BuilderBase is an abstract class that implements common functions for
PGICompiler(name, recipe, buildspec, testdir=None)	The BuilderBase is an abstract class that implements common functions for
ScriptBuilder(name, recipe, buildspec, testdir=None)	The BuilderBase is an abstract class that implements common functions for

class buildtest.buildsystem.base.**BuilderBase**(*name*, *recipe*, *buildspec*, *testdir=None*)

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.

_create_symlinks(*self*)

This method will retrieve all files relative to buildspec file and create symlinks in destination directory

_generate_unique_id(*self*)

Generate a build id based on the Buildspec name, and datetime.

_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()`.

detect_executor(*self*)

generate_script(*self*)

Build the testscript content implemented in each subclass

get_environment(*self*)

Retrieve a list of environment variables defined in buildspec 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_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

get_variables(*self*)

Retrieve a list of variables defined in buildspec and return them as list with the shell equivalent command.

Returns list of variables variable lines to add to test script.

Return type list

start(*self*)

Keep internal time for start of test

stop(*self*)

class buildtest.buildsystem.base.**CompilerBuilder**(*name*, *recipe*, *buildspec*, *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

executable

fc

fflags

lang_ext_table

ldflags

type = compiler

build_run_cmd(*self*)

This method builds the run command which refers to how to run the generated binary after compilation.

detect_lang(*self*, *sourcefile*)

This method will return the Programming Language based by looking up file extension of source file.

generate_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`.

generate_script(*self*)

This method will build the test content from a Buildspec that uses compiler schema. We need a 'compiler' and 'source' key which specifies the source files to compile. We resolve the source file path which can be an absolute value or relative path with respect to Buildspec. The file extension of sourcefile is used to detect the Programming Language which is used to lookup the compiler wrapper based on Language + Compiler. During compiler detection, we set class variables `self.cc`, `self.cxx`. `self.fc`, `self.cflags`, `self.cxxflags`, `self.fflags`, `self.cppflags`. `self.ldflags`. Finally we generate the compile command and add each instruction to `lines` which contains content of test. Upon completion, we return a list that contains content of the test.

get_cc(*self*)

get_cflags(*self*)

get_cppfilags(*self*)

get_cxx(*self*)

get_cxxflags(*self*)

get_fc(*self*)

get_fflags(*self*)

get_ldflags(*self*)

get_modules(*self*, *modules*)

Return a list of modules as a list

get_path(*self*)

This method returns the full path for GNU Compilers: `gcc`, `g++`, `gfortran`

resolve_source(*self*, *source*)

This method resolves full path to source file, it checks for absolute path first before checking relative path that is relative to Buildspec recipe.

set_cc(*self*, *cc*)
set_cflags(*self*, *cflags*)
set_cppflags(*self*, *cppflags*)
set_cxx(*self*, *cxx*)
set_cxxflags(*self*, *cxxflags*)
set_executable_name(*self*, *name=None*)
This method set the executable name. One may specify a custom name to executable via `name` argument. Otherwise the executable is using the filename of `self.sourcefile` and adding `.exe` extension at end.
set_fc(*self*, *fc*)
set_fflags(*self*, *fflags*)
set_ldflags(*self*, *ldflags*)
setup(*self*)

class buildtest.buildsystem.base.**CrayCompiler**(*name*, *recipe*, *buildspec*, *testdir=None*)
Bases: *buildtest.buildsystem.base.CompilerBuilder*
The BuilderBase is an abstract class that implements common functions for any kind of builder.
cc = cc
cxx = CC
fc = ftn

class buildtest.buildsystem.base.**GNUCompiler**(*name*, *recipe*, *buildspec*, *testdir=None*)
Bases: *buildtest.buildsystem.base.CompilerBuilder*
The BuilderBase is an abstract class that implements common functions for any kind of builder.
cc = gcc
cxx = g++
fc = gfortran

class buildtest.buildsystem.base.**IntelCompiler**(*name*, *recipe*, *buildspec*, *testdir=None*)
Bases: *buildtest.buildsystem.base.CompilerBuilder*
The BuilderBase is an abstract class that implements common functions for any kind of builder.
cc = icc
cxx = icpc
fc = ifort

class buildtest.buildsystem.base.**PGICompiler**(*name*, *recipe*, *buildspec*, *testdir=None*)
Bases: *buildtest.buildsystem.base.CompilerBuilder*
The BuilderBase is an abstract class that implements common functions for any kind of builder.
cc = pgcc
cxx = pgc++
fc = pgfortran

class buildtest.buildsystem.base.**ScriptBuilder**(*name*, *recipe*, *buildspec*, *testdir=None*)
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. Environment variables are declared first followed by run section
Returns return content of test script
Return type list

buildtest.buildsystem.batch

Module Contents

Classes

BatchScript()	
LSFBatchScript(batch=None, bsub=None)	
SlurmBatchScript(batch=None, sbatch=None)	

class buildtest.buildsystem.batch.**BatchScript**

get_headers(*self*)

class buildtest.buildsystem.batch.**LSFBatchScript**(*batch=None*, *bsub=None*)

Bases: *buildtest.buildsystem.batch.BatchScript*

batch_translation

build_header(*self*)

Generate BSUB directive that will be part of the script

class buildtest.buildsystem.batch.**SlurmBatchScript**(*batch=None*, *sbatch=None*)

Bases: *buildtest.buildsystem.batch.BatchScript*

batch_translation

build_header(*self*)

Generate SBATCH directive that will be part of the script

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(buildspec)	A BuildspecParser is a base class for loading and validating a Buildspec file.

class buildtest.buildsystem.parser.**BuildspecParser**(*buildspec*)

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, each of which is developed at https://github.com/buildtesters/schemas and added to subfolders named accordingly under buildtest/buildsystem/schemas. The schema object can load in a general Buildspec file to validate it, and then match it to a Buildspec Schema available. If the version of a schema is not specified, we use the latest. If the schema fails validation check, then we stop immediately.

__repr__(*self*)

Return repr(self).

__str__(*self*)

Return str(self).

_validate(*self*)

Given a loaded recipe, validate that the type is known in the lookup to buildtest. We check that `type` is found in schema lookup table, and `executor` field is a valid executor. Finally we validate loaded recipe with sub-schema

_validate_global(*self*)

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. The function also allows a custom Buildspec to extend the usage of the class.

get (*self*, *name*)
Given the name of a section (typically a build configuration name) return the loaded section from self.recipe. If you need to parse through just section names, use self.keys() to filter out metadata.

get_builders (*self*, *testdir*, *rebuild=1*, *tag_filter=None*, *executor_filter=None*)
Based on a loaded Buildspec file, return the correct builder for each based on the type. Each type is associated with a known Builder class.

Parameters
- **testdir** (*str, optional*) – Test Destination directory, specified by –testdir
- **tag_filter** (*list, optional*) – A list of input tags (`buildtest build --tags` option) to filter builders
- **executor_filter** – A list of input executors (`buildtest build --executor` option) to filter builders

keys (*self*)
Return the list of keys for the loaded Buildspec recipe, not including the metadata keys defined for any global recipe.

buildtest.buildsystem.parser.**configuration**
buildtest.buildsystem.parser.**executors**
buildtest.buildsystem.parser.**master_executors**

buildtest.executors

Submodules

buildtest.executors.base

BuildExecutor: manager for test executors

Module Contents

Classes

BaseExecutor(name, settings, config_opts)	The BaseExecutor is an abstract base class for all executors. All

class buildtest.executors.base.**BaseExecutor** (*name*, *settings*, *config_opts*)
The BaseExecutor is an abstract base class for all executors. All executors must have a listing of steps and dryrun_steps

steps = ['setup', 'run']
type = base
__repr__ (*self*)
Return repr(self).

__str__ (*self*)
Return str(self).

check_regex (*self*, *regex*)
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.

Parameters **regex** (*str*) – Regular expression object defined in Buildspec file
Returns A boolean return True/False based on if re.search is successful or not
Return type bool

check_test_state (*self*)

This method is responsible for detecting state of test (PASS/FAIL) based on returncode or regular expression.

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.

write_testresults(*self*, *out*, *err*)
This method writes test results into output and error file.

Parameters
- **out** (*list*) – content of output stream
- **err** (*list*) – content of error stream

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

LocalExecutor(name, settings, config_opts)	The LocalExecutor class is responsible for running tests locally for

class buildtest.executors.local.**LocalExecutor**(*name*, *settings*, *config_opts*)
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*)
This method is responsible for running test for LocalExecutor which runs test locally. We keep track of metadata in `self.builder.metadata` and `self.result` keeps track of run result. The output and error file are written to filesystem.

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

LSFExecutor(name, settings, config_opts)	The LSFExecutor class is responsible for submitting jobs to LSF Scheduler.

class buildtest.executors.lsf.**LSFExecutor**(*name*, *settings*, *config_opts*)

Bases: *buildtest.executors.base.BaseExecutor*

The LSFExecutor class is responsible for submitting jobs to LSF Scheduler. The LSFExecutor performs the following steps

check: check if lsf queue is available for accepting jobs. 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

format_fields = ['job_name', 'stat', 'user', 'user_group', 'queue', 'proj_name', 'pids'
job_state
poll_cmd = bjobs
steps = ['check', 'dispatch', 'poll', 'gather', 'close']
type = lsf

cancel(*self*)

Cancel LSF job, this is required if job exceeds max pending time in queue

check(*self*)

Checking binary for lsf launcher and poll command. If not found we raise error

dispatch(*self*)

This method is responsible for dispatching job to scheduler.

gather(*self*)

Gather Job detail after completion of job. This method will retrieve output fields defined for `self.format_fields`. buildtest will run `bjobs -o '<field1> ... <fieldN>' <JOBID> -json`.

load(*self*)

Load the a LSF executor configuration from buildtest settings.

poll(*self*)

This method will poll for job by using bjobs and return state of job. The command to be run is `bjobs -noheader -o 'stat' <JOBID>` which returns job state.

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) to call depending on executor name.

Module Contents

Classes

BuildExecutor(config_opts)	A BuildExecutor is a base class some type of executor, for example,

class buildtest.executors.setup.**BuildExecutor**(*config_opts*)

A BuildExecutor is a base class some type of executor, for example, the types "local", "lsf", "slurm" would map to `LocalExecutor`, `LSFExecutor` and `SlurmExecutor` here, each expecting a particular set of variables under the config options. If options are required and not provided, we exit on error. If they are optional and not provided, we use reasonable defaults.

__repr__(*self*)

Return repr(self).

__str__(*self*)

Return str(self).

_choose_executor(*self*, *builder*)

Choose executor is called at the onset of a run or dryrun. We look at the builder metadata to determine if a default is set for the executor, and fall back to the default.

Parameters **builder** (*buildtest.buildsystem.BuilderBase (or subclass)*) – the builder with the loaded Buildspec.

get(*self*, *name*)

Given the name of an executor return the executor for running a buildtest build, or get the default.

poll(*self*, *builder*)

Poll all jobs for batch executors (LSF, Slurm). For slurm we poll until job is in `PENDING` or `RUNNING` state. If it is not in these states, we assume job is complete and gather results. For LSF jobs we poll job if its in job-state `PEND` and `RUN`. The method returns `True` or `False` depending on the input builder.

Parameters **builder** (*BuilderBase (subclass)*) – an instance of BuilderBase (subclass)

Returns Return a boolean to indicate if builder needs further polling

Return type bool

run(*self*, *builder*)

Given a BuilderBase (subclass) go through the steps defined for the executor to run the build. This should be instantiated by the subclass. For a simple script run, we expect a setup, build, and finish.

Parameters **builder** (*buildtest.buildsystem.BuilderBase (or subclass)*) – 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 and after_script.sh if the fields `before_script` and `after_script` are specified in executor section. This method is called after executors are initialized in the class **__init__** method

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

SlurmExecutor(name, settings, config_opts)	The SlurmExecutor class is responsible for submitting jobs to Slurm Scheduler.

class buildtest.executors.slurm.**SlurmExecutor**(*name*, *settings*, *config_opts*)

Bases: *buildtest.executors.base.BaseExecutor*

The SlurmExecutor class is responsible for submitting jobs to Slurm Scheduler. The SlurmExecutor performs the following steps

check: check if slurm partition is available for accepting jobs. load: load slurm configuration from buildtest configuration file dispatch: dispatch job to scheduler and acquire job ID poll: wait for Slurm jobs to finish gather: Once job is complete, gather job data

job_state

poll_cmd = sacct

sacct_fields = ['Account', 'AllocNodes', 'AllocTRES', 'ConsumedEnergyRaw', 'CPUTimeRaw'

steps = ['dispatch', 'poll', 'gather', 'close']

type = slurm

cancel(*self*)

Cancel slurm job, this operation is performed if job exceeds pending or runtime.

check(*self*)

Check slurm binary is available before running tests. This will check the launcher (sbatch) and sacct are available. If qos, partition, and cluster key defined we check if its a valid entity in slurm configuration. For partition, we also check if its in the `up` state before dispatching jobs. This method will raise an exception of type SystemExit if any checks fail.

dispatch(*self*)

This method is responsible for dispatching job to slurm scheduler.

gather(*self*)

Gather Slurm detail after job completion

load(*self*)

Load the a slurm executor configuration from buildtest settings.

poll(*self*)

This method will poll for job each interval specified by time interval until job finishes. We use *sacct* to poll for job id and sleep for given time interval until trying again. The command to be run is `sacct -j <jobid> -o State -n -X -P`

buildtest.menu

buildtest menu: include functions to build, get test configurations, and interact with a global configuration for buildtest.

Submodules

buildtest.menu.build

This module contains all the methods related to "buildtest build" which is used for building test scripts from a Buildspec

Module Contents

Functions

build_phase(builders, printTable=False)	This method will build all tests by invoking class method `build` for
discover_buildspecs(tags=None, executorname=None, buildspec=None, exclude_buildspec=None, debug=False)	This method discovers all buildspecs and returns a list of discovered
discover_buildspecs_by_executor_name(executorname)	This method discovers buildspecs by executor name, using `--executor-name`
discover_buildspecs_by_tags(input_tag)	This method discovers buildspecs by tags, using `--tags` option
discover_by_buildspecs(buildspec)	Given a buildspec file specified by the user with `buildtest build --buildspec`,
func_build_subcmd(args, config_opts)	Entry point for `buildtest build` sub-command. This method will discover
resolve_testdirectory(config_opts, input_testdir=None)	This method resolves which test directory to select. For example, one
run_phase(builders, executor, config_dict, printTable=False)	This method will run all builders with the appropriate executor.

buildtest.menu.build.**build_phase**(*builders*, *printTable=False*)

This method will build all tests by invoking class method `build` for each builder that generates testscript in the test directory.

Parameters

- **builders** (*list*) – A list of builders
- **printTable** (*boolean*) – Print builder table

buildtest.menu.build.**discover_buildspecs**(*tags=None*, *executorname=None*, *buildspec=None*, *exclude_buildspec=None*, *debug=False*)

This method discovers all buildspecs and returns a list of discovered excluded buildspecs. The input arguments `tags`, `buildspec`, `exclude_buildspec` map to `--tags` `--buildspec` and `--exclude` option in buildtest build.

Parameters

- **tags** (*str*) – Input argument from `buildtest build --tags`
- **executorname** (*list*) – Input argument from `buildtest build --executor-name`
- **buildspec** (*str*) – Input argument from `buildtest build --buildspec`
- **exclude_buildspec** – Input argument from `buildtest build --exclude`
- **debug** (*boolean*) – Boolean to control print messages to stdout

Returns two lists of discovered and excluded buildspecs

Return type list, list

buildtest.menu.build.**discover_buildspecs_by_executor_name**(*executor_name*)

This method discovers buildspecs by executor name, using `--executor-name` option from `buildtest build` 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 **executor_name** (*string*) – Input executor name from command line argument `buildtest build --executor-name <name>`

Returns a list of buildspec files that match tag name

Return type list

buildtest.menu.build.**discover_buildspecs_by_tags**(*input_tag*)

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** (*string*) – Input tags from command line argument `buildtest build --tags <tags>`

Returns a list of buildspec files that match tag name

Return type list

buildtest.menu.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 .yml extension. If filepath doesn't exist or file extension is not .yml we return None and log as an error.
file path buildtest build –buildspec tutorials/hello.sh.yml
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.menu.build.**func_build_subcmd**(*args*, *config_opts*)

Entry point for `buildtest build` sub-command. This method will discover Buildspecs in method `discover_buildspecs`. If there is an exclusion list this will be checked, once buildtest knows all Buildspecs to process it will begin validation by calling `BuildspecParser` and followed by an executor instance by invoking BuildExecutor that is responsible for executing the test based on the executor type. A report of all builds, along with test summary will be displayed to screen.

Parameters **args** (*dict, required*) – arguments passed from command line

Return type None

buildtest.menu.build.**logger**

buildtest.menu.build.**resolve_testdirectory**(*config_opts*, *input_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 $BUILDTEST_ROOT/var/tests

Parameters

- **config_opts** (*dict*) – loaded buildtest configuration as a dict.
- **input_testdir** (*str*) – Input test directory from command line `buildtest build --testdir`

Returns Path to test directory to use

Return type str

buildtest.menu.build.**run_phase**(*builders*, *executor*, *config_dict*, *printTable=False*)

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 to tally number of pass and fail test which is printed at end in Test Summary. For tests that need to run via scheduler (Slurm, LSF) 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 calculate 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.

Parameters

- **builders** – A list of builders that need to be run. These correspond to test names
- **executor** (BuildExecutor) – The master executor class responsible for invoking appropriate executor class corresponding to builder.
- **config_dict** (*dict*) – loaded buildtest configuration
- **printTable** (*bool*) – boolean to control print statement for run phase

Type builders: list of objects of type *BuilderBase*

Returns A list of valid builders

Return type list

buildtest.menu.buildspec

Module Contents

Functions

func_buildspec_edit(args)	This method implement `buildtest buildspec edit` which
func_buildspec_find(args)	Entry point for `buildtest buildspec find`. This method
func_buildspec_view(args)	This method implements `buildtest buildspec view` which shows
func_buildspec_view_edit(buildspec, view=False, edit=False)	This is a shared method for `buildtest buildspec view` and
get_all_tags(cache)	This method implements `buildtest buildspec find --tags` which
get_buildspecfiles(cache)	This method implements `buildtest buildspec find --buildspec-files` which
get_executors(cache)	This method implements `buildtest buildspec find --list-executors` which
parse_buildspecs(buildspecs, test_directory, rebuild, tags=None, executors=None, printTable=False)	Parse all buildspecs by invoking class `BuildspecParser`. If buildspec
rebuild_buildspec_cache(paths)	This method will rebuild the buildspec cache file by recursively searching

buildtest.menu.buildspec.**func_buildspec_edit**(*args*)

This method implement `buildtest buildspec edit` which allows one to edit a Buildspec file with one of the editors set in buildtest settings.

buildtest.menu.buildspec.**func_buildspec_find**(*args*)

Entry point for `buildtest buildspec find`. This method will attempt to read for buildspec cache file (BUILDSPEC_CACHE_FILE) if found and print a list of all buildspecs. Otherwise, it will read the repo file (REPO_FILE) and find all buildspecs and validate them via BuildspecParser. BuildspecParser will raise SystemError or ValidationError if a buildspec is invalid which will be added to list of invalid buildspecs. Finally we print a list of all valid buildspecs and any invalid buildspecs are written to file along with error message.

Parameters **args** – Input argument from command line passed from argparse

Returns A list of valid buildspecs found in all repositories.

buildtest.menu.buildspec.**func_buildspec_view**(*args*)

This method implements `buildtest buildspec view` which shows content of a buildspec file

buildtest.menu.buildspec.**func_buildspec_view_edit**(*buildspec*, *view=False*, *edit=False*)

This is a shared method for `buildtest buildspec view` and `buildtest buildspec edit`.

Parameters

- **buildspec** (*str (filepath)*) – buildspec file section to view or edit.
- **view** (*bool*) – boolean to determine if we want to view buildspec file
- **edit** (*bool*) – boolean to control if we want to edit buildspec file in editor.

Returns Shows the content of buildspec or let's user interactively edit buildspec. An exception can be raised if it's unable to find buildspec

buildtest.menu.buildspec.**get_all_tags**(*cache*)

This method implements `buildtest buildspec find --tags` which reports a list of unique tags from all buildspecs in cache file.

Parameters **cache** (*dict*) – content of cache as dictionary

buildtest.menu.buildspec.**get_buildspecfiles**(*cache*)

This method implements `buildtest buildspec find --buildspec-files` which reports all buildspec files in cache.

Parameters **cache** (*dict*) – content of cache as dictionary

buildtest.menu.buildspec.**get_executors**(*cache*)

This method implements `buildtest buildspec find --list-executors` which reports all executors from cache.

Parameters **cache** (*dict*) – content of cache as dictionary

buildtest.menu.buildspec.**logger**

buildtest.menu.buildspec.**parse_buildspecs**(*buildspecs*, *test_directory*, *rebuild*, *tags=None*, *executors=None*, *printTable=False*)

Parse all buildspecs by invoking class `BuildspecParser`. If buildspec fails validation we add it to `skipped_tests` list and print all skipped tests at end. If buildspec passes validation we get all builders by invoking `get_builders` method in BuildspecParser class which gets all tests from buildspec file.

Parameters

- **buildspecs** (*list of filepaths*) – A list of input buildspecs to parse
- **test_directory** (*str (directory path)*) – Test directory where buildspecs will be written
- **tags** (*list*) – A list of input tags to filter tests
- **executors** (*list*) – A list of input executors to filter tests
- **printTable** (*bool, optional*) – a boolean to control if parse table is printed

Returns A list of builder objects which are instances of `BuilderBase` class

Return type list

buildtest.menu.buildspec.**rebuild_buildspec_cache**(*paths*)

This method will rebuild the buildspec cache file by recursively searching all .yml 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

buildtest.menu.config

Module Contents

Functions

func_config_summary(args=None)	This method implements `buildtest config summary` option. In this method
func_config_validate(args=None)	This method implements `buildtest config validate` which attempts to

continues on next page

Table 13 – continued from previous page

func_config_view(args=None)	View buildtest configuration file. This implements `buildtest config view`

buildtest.menu.config.**func_config_summary**(*args=None*)
This method implements `buildtest config summary` option. In this method we will display a summary of System Details, Buildtest settings, Schemas, Repository details, Buildspecs files and test names.

buildtest.menu.config.**func_config_validate**(*args=None*)
This method implements `buildtest config validate` which attempts to validate buildtest settings with schema. If it not validate an exception an exception of type SystemError is raised. We invoke `check_settings` method which will validate the configuration, if it fails we except an exception of type ValidationError which we catch and print message.

buildtest.menu.config.**func_config_view**(*args=None*)
View buildtest configuration file. This implements `buildtest config view`

buildtest.menu.inspect

Module Contents

Functions

func_inspect(args)	Entry point for `buildtest inspect` command
get_all_ids()	Return all unique test ids from report cache

buildtest.menu.inspect.**func_inspect**(*args*)
Entry point for `buildtest inspect` command

buildtest.menu.inspect.**get_all_ids**()
Return all unique test ids from report cache :return: list of unique ids :rtype: list

buildtest.menu.report

Module Contents

Functions

func_report(args=None)	
is_int(val)	
update_report(valid_builders)	This method will update BUILD_REPORT after every test run performed

buildtest.menu.report.**func_report**(*args=None*)

buildtest.menu.report.**is_int**(*val*)

buildtest.menu.report.**update_report**(*valid_builders*)
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

buildtest.menu.schema

Module Contents

Functions

func_schema(args)	This method implements command `buildtest schema` which shows a list

buildtest.menu.schema.**func_schema**(*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

Classes

BuildTestParser()	

Functions

buildtestdocs(args=None)	Open buildtest docs in web browser. This implements `buildtest docs`
func_buildspec_edit(args)	This method implement `buildtest buildspec edit` which
func_buildspec_find(args)	Entry point for `buildtest buildspec find`. This method
func_buildspec_view(args)	This method implements `buildtest buildspec view` which shows
func_config_summary(args=None)	This method implements `buildtest config summary` option. In this method
func_config_validate(args=None)	This method implements `buildtest config validate` which attempts to
func_config_view(args=None)	View buildtest configuration file. This implements `buildtest config view`
func_inspect(args)	Entry point for `buildtest inspect` command
func_report(args=None)	
func_schema(args)	This method implements command `buildtest schema` which shows a list
handle_kv_string(val)	This method is used as type field in –filter argument in `buildtest buildspec find`.
positive_number(value)	
schemadocs(args=None)	Open buildtest schema docs in web browser. This implements `buildtest schemadocs`

buildtest.menu.**BUILDTEST_VERSION = 0.9.0**

class buildtest.menu.**BuildTestParser**

build_menu(*self*)
This method implements the `buildtest build` command

buildspec_menu(*self*)
This method implements `buildtest buildspec` command

config_menu(*self*)
This method adds argparse argument for `buildtest config`

inspect_menu(*self*)
This method builds menu for *buildtest inspect*

main_menu(*self*)
This method adds argument to ArgumentParser to main menu of buildtest

parse_options(*self*)
This method parses the argument from ArgumentParser class and returns the arguments. We store extra (non parsed arguments) with the class if they are needed.

Returns return a parsed dictionary returned by ArgumentParser

Return type dict

report_menu(*self*)
This method implements the `buildtest report` command options

schema_menu(*self*)
This method adds argparse argument for `buildtest show`

buildtest.menu.**buildtestdocs**(*args=None*)
Open buildtest docs in web browser. This implements `buildtest docs`

buildtest.menu.**func_buildspec_edit**(*args*)
This method implement `buildtest buildspec edit` which allows one to edit a Buildspec file with one of the editors set in buildtest settings.

buildtest.menu.**func_buildspec_find**(*args*)
Entry point for `buildtest buildspec find`. This method will attempt to read for buildspec cache file (BUILDSPEC_CACHE_FILE) if found and print a list of all buildspecs. Otherwise, it will read the repo file (REPO_FILE) and find all buildspecs and validate them via BuildspecParser. BuildspecParser will raise SystemError or ValidationError if a buildspec is invalid which will be added to list of invalid buildspecs. Finally we print a list of all valid buildspecs and any invalid buildspecs are written to file along with error message.

Parameters **args** – Input argument from command line passed from argparse

Returns A list of valid buildspecs found in all repositories.

buildtest.menu.**func_buildspec_view**(*args*)
This method implements `buildtest buildspec view` which shows content of a buildspec file

buildtest.menu.**func_config_summary**(*args=None*)
This method implements `buildtest config summary` option. In this method we will display a summary of System Details, Buildtest settings, Schemas, Repository details, Buildspecs files and test names.

buildtest.menu.**func_config_validate**(*args=None*)
This method implements `buildtest config validate` which attempts to validate buildtest settings with schema. If it not validate an exception an exception of type SystemError is raised. We invoke `check_settings` method which will validate the configuration, if it fails we except an exception of type ValidationError which we catch and print message.

buildtest.menu.**func_config_view**(*args=None*)
View buildtest configuration file. This implements `buildtest config view`

buildtest.menu.**func_inspect**(*args*)
Entry point for `buildtest inspect` command

buildtest.menu.**func_report**(*args=None*)

buildtest.menu.**func_schema**(*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

buildtest.menu.**handle_kv_string**(*val*)

This method is used as type field in –filter argument in `buildtest buildspec 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.menu.**positive_number**(*value*)

buildtest.menu.**schema_table**

buildtest.menu.**schemadocs**(*args=None*)

Open buildtest schema docs in web browser. This implements `buildtest schemadocs`

buildtest.schemas

Submodules

buildtest.schemas.defaults

Module Contents

Functions

custom_validator(recipe, schema)	This is a custom validator for validating JSON documents. We implement a

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

get_schema_fullpath(schema_file, name=None)	Return the full path of a schema file
load_recipe(path)	Load a yaml recipe file. The file must be in .yml extension
load_schema(path)	Load a json schema file, the file extension must be '.schema.json'

buildtest.schemas.utils.**get_schema_fullpath**(*schema_file*, *name=None*)

Return the full path of a schema file

Parameters

- **schema_file** (*str*) – the path to the schema file.
- **name** (*str, optional*) – the schema type. If not provided, derived from filename.

buildtest.schemas.utils.**here**

buildtest.schemas.utils.**load_recipe**(*path*)

Load a yaml recipe file. The file must be in .yml 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

BuildTestCommand(cmd=None)	Class method to invoke shell commands and retrieve output and error.
Capturing()	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_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

create_dir(dirname)	Create directory if it doesn't exist. Runs a "try" block
is_dir(dirname)	This method will check if a directory exist and if not found throws an exception.
is_file(fname)	This method will check if file exist and if not found throws an exception.
read_file(filepath)	This method is used to read a file specified by argument `filepath`. If filepath is not a string we raise
resolve_path(path, exist=True)	This method will resolve a file path to account for shell expansion and resolve paths in
walk_tree(root_dir, ext=None)	This method will traverse a directory tree and return list of files
write_file(filepath, content)	This method is used to write an input `content` to a file specified by ``filepath. Both filepath

buildtest.utils.file.**create_dir**(*dirname*)
Create directory if it doesn't exist. Runs a "try" block to run os.makedirs() which creates all sub-directories if they dont exist. Catches exception of type OSError and prints message
Parameters:
Parameters **dirname** (*string, 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 and if not found throws an exception.

Parameters:

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 and if not found throws an exception.

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.**logger**

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:

Parameters **filepath** (*str, required*) – file name to read

Raises SystemError: If filepath is not a string SystemError: If filepath is not valid file

Returns return content of file as a string

Return type str

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.

Parameters:

Parameters **path** (*str, required*) – file path to resolve

Returns return realpath to file if found otherwise return None

Return type 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:

Parameters

- **filepath** (*str, required*) – file name to write
- **content** (*str, required*) – content to write to file

Raises SystemError: System error if filepath is not string SystemError: 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(shell='bash')

class buildtest.utils.shell.**Shell**(*shell='bash'*)

__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 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

Submodules

buildtest.config

Module Contents

Functions

check_settings(settings_path=None, executor_check=True, retrieve_settings=False)	Checks all keys in configuration file (settings/config.yml) are valid
load_settings(settings_path=None)	Load the default settings file if no argument is specified.
resolve_settings_file()	Returns path to buildtest settings file that should be used. If there
validate_lsf_executors(lsf_executors)	This method validates all LSF executors. We check if queue is available
validate_slurm_executors(slurm_executor)	This method will validate slurm executors, we check if partition, qos,

buildtest.config.**check_settings**(*settings_path=None*, *executor_check=True*, *retrieve_settings=False*)

Checks all keys in configuration file (settings/config.yml) are valid keys and ensure value of each key matches expected type. For some keys special logic is taken to ensure values are correct and directory path exists. If any error is found buildtest will terminate immediately.

Parameters

- **settings_path** (*str, optional*) – Path to buildtest settings file
- **executor_check** (*bool*) – boolean to control if executor checks are performed
- **retrieve_settings** (*bool*) – return loaded buildtest settings that is validated by schema. By default, this method doesn't return anything other than validating buildtest settings

Returns returns gracefully if all checks passes otherwise terminate immediately

Return type exit code 1 if checks failed

buildtest.config.**load_settings**(*settings_path=None*)

Load the default settings file if no argument is specified.

Parameters **settings_path** (*str, optional*) – Path to buildtest settings file

buildtest.config.**logger**

buildtest.config.**resolve_settings_file**()

Returns path to buildtest settings file that should be used. If there is a user defined buildtest settings ($HOME/.buildtest/config.yml) it will be honored, otherwise default settings from buildtest will be used.

buildtest.config.**validate_lsf_executors**(*lsf_executors*)

This method validates all LSF executors. We check if queue is available and in `Open:Active` state. :param lsf_executors: A list of LSF executors to validate :type lsf_executors: dict

buildtest.config.**validate_slurm_executors**(*slurm_executor*)

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.

Parameters **slurm_executor** (*dict*) – list of slurm executors defined in loaded buildtest configuration

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.**BUILDTEST_ROOT**
buildtest.defaults.**BUILDTEST_SETTINGS_FILE**
buildtest.defaults.**BUILDTEST_USER_HOME**
buildtest.defaults.**BUILD_REPORT**
buildtest.defaults.**DEFAULT_SETTINGS_FILE**
buildtest.defaults.**DEFAULT_SETTINGS_SCHEMA**
buildtest.defaults.**SCHEMA_ROOT**
buildtest.defaults.**executor_root**
buildtest.defaults.**logID = buildtest**
buildtest.defaults.**supported_schemas**
buildtest.defaults.**supported_type_schemas = ['script-v1.0.schema.json', 'compiler-v1.0.sche**
buildtest.defaults.**userhome**
buildtest.defaults.**var_root**

buildtest.docs

This file provides method to access buildtest and schema docs when requested from command line.

Module Contents

Functions

buildtestdocs(args=None)	Open buildtest docs in web browser. This implements `buildtest docs`
schemadocs(args=None)	Open buildtest schema docs in web browser. This implements `buildtest schemadocs`

buildtest.docs.**buildtestdocs**(*args=None*)
Open buildtest docs in web browser. This implements `buildtest docs`
buildtest.docs.**schemadocs**(*args=None*)
Open buildtest schema docs in web browser. This implements `buildtest schemadocs`

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).

buildtest.log

Methods related to buildtest logging

Module Contents

Functions

init_logfile(logfile)	Initialize a log file intended for a builder. This requires
streamlog(debuglevel)	

buildtest.log.**init_logfile**(*logfile*)
Initialize a log file intended for a builder. This requires passing the filename intended for the log (from the builder) and returns the logger.
Parameters **logfile** (*str*) – logfile name

buildtest.log.**streamlog**(*debuglevel*)

buildtest.main

Module Contents

Functions

main()	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

BuildTestSystem()	BuildTestSystem is a class that detects system configuration and outputs the result

Functions

get_lsf_queues()	Return json dictionary of available LSF Queues and their queue states
get_slurm_clusters()	
get_slurm_partitions()	Get slurm partitions
get_slurm_qos()	Return all slurm qos

class buildtest.system.**BuildTestSystem**
BuildTestSystem is a class that detects system configuration and outputs the result in .run file which are generated upon test execution. This module also keeps track of what is supported (or not supported) for a system.
system

check_scheduler(*self*)
Check for batch scheduler. Currently checks for LSF or SLURM by running `bhosts` and `sinfo` command. It must be present in $PATH when running buildtest. Since it's unlikely for a single host to have more than one scheduler, we check for multiple and return the first found.
Returns return string **LSF** or **SLURM**. If neither found returns **None**
Return type str or None

init_system(*self*)
Based on the module "distro" get system details like linux distro, processor, hostname, machine name.

buildtest.system.**get_lsf_queues**()
Return json dictionary of available LSF Queues and their queue states

buildtest.system.**get_slurm_clusters**()

buildtest.system.**get_slurm_partitions**()
Get slurm partitions

buildtest.system.**get_slurm_qos**()
Return all slurm qos

Package Contents

buildtest.**BUILDTEST_VERSION = 0.9.0**

buildtest.**__version__**

CHAPTER
SIX

LICENSE

buildtest is released under the MIT license

CHAPTER
SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

INDEX

Symbols

B

C

D

E

F

O

P

R

S

T

U

V

W