buildtest Documentation

Release 2019.04.02

Shahzeb Siddiqui

Dec 15, 2020

BACKGROUND

This documentation was last rebuild on Dec 15, 2020 and is intended for version 0.8.0.

CHAPTER

ONE

STATUS

CHAPTER

TWO

SOURCE CODE

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

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/schemas/
- ReadTheDocs: https://readthedocs.org/projects/buildtest/
- Travis: https://travis-ci.com/buildtesters/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 *Writing buildspecs*.

A spin-off project called lmodule which is a Python API for Lmod. The buildtest module features were deprecated and moved to lmodule with the main object of automating 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 with buildtest*.

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 the following:

- *HPC Staff*: that wants to perform acceptance & regression testing of their HPC system.
- *Research Software Engineers*: that want to test software installed in 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.

- Python >= 3.6

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 code release, please pull the **devel** branch:

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

Or you may switch to the **devel** branch if you already cloned it:

```
$ git checkout devel
```

5.3.3 Installing buildtest

To install buildtest run the following:

```
# for site installed python
$ pip install --user .

# for virtual environment, local install
$ pip install .
```

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

  {build,buildspec,report,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 sub-command run:

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

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

5.4 Getting Started with buildtest

5.4.1 Interacting with the client

After you install buildtest, you should find the client on your path:

```
$ which buildtest
~/.local/bin/buildtest
```

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

5.4.2 Buildspecs

Finding Buildspecs

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
```

To find all buildspecs run `buildtest buildspec find` which will discover all buildspecs in all repos. buildtest will recursively traverse all repos and seek out all *.yml* extensions so make sure your buildspecs conform to the file extension.

```
$ buildtest buildspec find
Searching buildspec in following directories:  /Users/siddiq90/Documents/buildtest/
↪tutorials
Found 14 buildspecs
Validated 5/14 buildspecs
Validated 10/14 buildspecs
Validated 14/14 buildspecs

Detected 1 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                   |
+-----------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| _bin_sh_shell         | script   | local.sh     | ['tutorials'] | /bin/sh shell␣
↪example                                       |
+-----------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| _bin_bash_shell       | script   | local.bash   | ['tutorials'] | /bin/bash shell␣
↪example                                        |
+-----------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| 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                                                        |
+-------------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| variables               | script   | local.bash   | ['tutorials'] | Declare shell␣
↪variables                                                                  |
+-------------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| selinux_disable         | script   | local.bash   | ['tutorials'] | Check if SELinux␣
↪is Disabled                                                                |
+-------------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| exit1_fail              | script   | local.sh     | ['tutorials'] | exit 1 by␣
↪default is FAIL                                                             |
+-------------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| exit1_pass              | script   | local.sh     | ['tutorials'] | report exit 1 as␣
↪PASS                                                                       |
+-------------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| returncode_mismatch     | script   | local.sh     | ['tutorials'] | exit 2 failed␣
↪since it failed to match returncode 1                                      |
+-------------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| wrongexecutor           | script   | badexecutor  | ['tutorials'] | valid test but␣
↪invalid executor name so test won't run                                  |
+-------------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| circle_area             | script   | local.python | ['tutorials'] | Calculate circle␣
↪of area given a radius                                                     |
+-------------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| skip                    | script   | local.bash   | ['tutorials'] |                 ␣
↪                                                                           |
+-------------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| unskipped               | script   | local.bash   | ['tutorials'] |                 ␣
↪                                                                           |
+-------------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| vecadd_gnu              | compiler | local.bash   | ['tutorials'] | Vector Addition␣
↪example with GNU compiler                                                  |
+-------------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| hello_f                 | compiler | local.bash   | ['tutorials'] | Hello World␣
↪Fortran Compilation                                                        |
```

(continues on next page)

(continued from previous page)

```
+-----------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| hello_c               | compiler | local.bash   | ['tutorials'] | Hello World C␣
↪Compilation                                     |
+-----------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| hello_cplusplus       | compiler | local.bash   | ['tutorials'] | Hello World C++␣
↪Compilation                                   |
+-----------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| cc_example            | compiler | local.bash   | ['tutorials'] | Example by using␣
↪cc to set C compiler                          |
+-----------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| fc_example            | compiler | local.bash   | ['tutorials'] | Example by using␣
↪fc to set Fortran compiler                    |
+-----------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| cxx_example           | compiler | local.bash   | ['tutorials'] | Example by using␣
↪cxx to set C++ compiler                       |
+-----------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| pre_post_build_run    | compiler | local.bash   | ['tutorials'] | example using␣
↪pre_build, post_build, pre_run, post_run example |
+-----------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
| executable_arguments  | compiler | local.bash   | ['tutorials'] | Passing␣
↪arguments example                                 |
+-----------------------+----------+--------------+---------------+------------------
↪-----------------------------------------------+
```

buildtest will find all buildspecs and validate each 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 since we read from cache. If you make changes to buildspec you may want to rebuild cache using `buildtest buildspec find --clear`.

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";
```

(continues on next page)

(continued from previous page)

```
      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.
Press any key to continue
```

5.4.3 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`

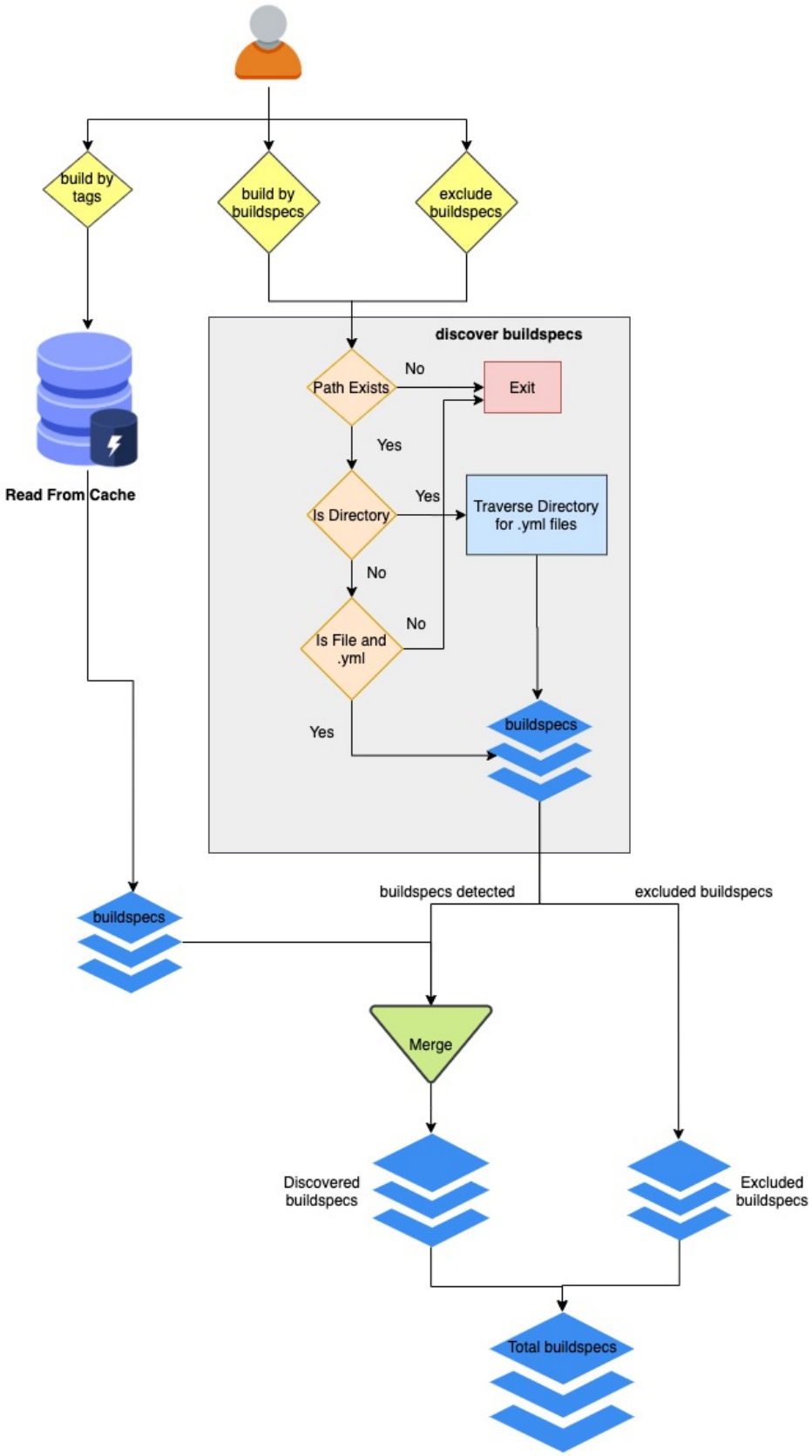
```
$ buildtest build --help
usage: buildtest [options] [COMMANDS] build [-h] [-b BUILDSPEC] [-t TESTDIR] [--
↪settings SETTINGS] [-x EXCLUDE]
                                            [--tags TAGS] [-s {parse,build}]

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.
  -t TESTDIR, --testdir TESTDIR
                        specify a custom test directory. By default, use .buildtest␣
↪in $PWD.
  --settings SETTINGS   Specify an alternate buildtest settings file to use
  -x EXCLUDE, --exclude EXCLUDE
                        Exclude one or more configs from processing. Configs can be␣
↪files or directories.
  --tags TAGS           Specify buildspecs by tags
  -s {parse,build}, --stage {parse,build}
                        control behavior of buildtest build
```

5.4.4 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

5.4.5 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
Paths:
___________
Test Directory: /Users/siddiq90/Documents/buildtest/var/tests

+------------------------------+
| Stage: 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                    | Schema File              | Test Path
↪                                                                 | Buildspec
-------------------------+--------------------------+-----------------------------------
↪-----------------------------------------------------------------+-----------------
↪----------------------------------------
 systemd_default_target | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/systemd/systemd_default_target/generate.sh | /Users/
↪siddiq90/Documents/buildtest/tutorials/systemd.yml

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

 name                    | executor   | status   |   returncode | testpath
-------------------------+------------+----------+--------------+----------------------
↪-------------------------------------------------------------------------------
 systemd_default_target | local.bash | FAIL     |            1 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/systemd/systemd_default_target/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 tests/examples/buildspecs/os.yaml
Paths:

__________
Prefix: /private/tmp
Buildspec Search Path: ['/Users/siddiq90/.buildtest/site']
Test Directory: /private/tmp/tests
tests/examples/buildspecs/os.yaml does not end in file extension .yml
```

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/
Paths:

__________
Test Directory: /Users/siddiq90/Documents/buildtest/var/tests

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

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

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

 schemafile               | validstate   | buildspec
--------------------------+--------------+-----------------------------------------
↪-------------------------------------
 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/tests/
↪examples/buildspecs/shell_examples.yml
 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

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

 Name                        | Schema File              | Test Path
↪                                                                          | Buildspec
----------------------------+--------------------------+-------------------------------
↪-------------------------------------------------------------------------+-----------
↪-------------------------------------------------------------------------
 slurm_down_nodes_reason     | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/slurm/slurm_down_nodes_reason/generate.sh     | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/slurm.yml
 slurm_not_responding_nodes | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/slurm/slurm_not_responding_nodes/generate.sh | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/slurm.yml
 _bin_sh_shell               | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.sh/shell_examples/_bin_sh_shell/generate.sh        | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/shell_examples.yml
```

(continues on next page)

(continued from previous page)

```
 _bin_bash_shell            | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/shell_examples/_bin_bash_shell/generate.sh   | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/shell_examples.yml
 bash_shell                 | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/shell_examples/bash_shell/generate.sh        | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/shell_examples.yml
 sh_shell                   | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.sh/shell_examples/sh_shell/generate.sh            | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/shell_examples.yml
 shell_options              | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.sh/shell_examples/shell_options/generate.sh       | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/shell_examples.yml
 circle_area                | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.python/python-shell/circle_area/generate.py       | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/python-shell.yml
 hello_dinosaur             | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/environment/hello_dinosaur/generate.sh       | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/environment.yml

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

 name                        | executor     | status   |   returncode | testpath
-----------------------------+--------------+----------+--------------+----------------
↪-------------------------------------------------------------------------------------
↪---
 slurm_down_nodes_reason     | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/slurm/slurm_down_nodes_reason/
↪generate.sh
 slurm_not_responding_nodes | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/slurm/slurm_not_responding_nodes/
↪generate.sh
 _bin_sh_shell               | local.sh     | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/_bin_sh_shell/
↪generate.sh
 _bin_bash_shell             | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/_bin_bash_shell/
↪generate.sh
 bash_shell                  | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/bash_shell/
↪generate.sh
 sh_shell                    | local.sh     | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/sh_shell/generate.sh
 shell_options               | local.sh     | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/shell_options/
↪generate.sh
 circle_area                 | local.python | FAIL     |            2 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.python/python-shell/circle_area/
↪generate.py
 hello_dinosaur              | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/environment/hello_dinosaur/
↪generate.sh

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

(continues on next page)

(continued from previous page)

```
Executed 9 tests
Passed Tests: 8/9 Percentage: 88.889%
Failed Tests: 1/9 Percentage: 11.111%
```

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
Paths:
__________
Test Directory: /Users/siddiq90/Documents/buildtest/var/tests

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

/Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/python-shell.yml
/Users/siddiq90/Documents/buildtest/tutorials/systemd.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/
↪tutorials/systemd.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                        | Schema File             | Test Path
↪                                                                  | Buildspec
----------------------------+-------------------------+-------------------------------
↪-------------------------------------------------------------------+-----------
↪-----------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
 circle_area                | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.python/python-shell/circle_area/generate.py       | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/python-shell.yml
 systemd_default_target     | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/systemd/systemd_default_target/generate.sh   | /
↪Users/siddiq90/Documents/buildtest/tutorials/systemd.yml
 hello_dinosaur             | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/environment/hello_dinosaur/generate.sh       | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/environment.yml
 _bin_sh_shell              | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.sh/shell_examples/_bin_sh_shell/generate.sh       | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/shell_examples.yml
 _bin_bash_shell            | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/shell_examples/_bin_bash_shell/generate.sh   | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/shell_examples.yml
 bash_shell                 | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/shell_examples/bash_shell/generate.sh        | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/shell_examples.yml
 sh_shell                   | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.sh/shell_examples/sh_shell/generate.sh            | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/shell_examples.yml
 shell_options              | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.sh/shell_examples/shell_options/generate.sh       | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/shell_examples.yml
 slurm_down_nodes_reason    | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/slurm/slurm_down_nodes_reason/generate.sh    | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/slurm.yml
 slurm_not_responding_nodes | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/slurm/slurm_not_responding_nodes/generate.sh | /
↪Users/siddiq90/Documents/buildtest/tests/examples/buildspecs/slurm.yml

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

 name                         | executor     | status   |   returncode | testpath
------------------------------+--------------+----------+--------------+----------------
↪-------------------------------------------------------------------------------------
↪---
 circle_area                  | local.python | FAIL     |            2 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.python/python-shell/circle_area/
↪generate.py
 systemd_default_target       | local.bash   | FAIL     |            1 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/systemd/systemd_default_target/
↪generate.sh
 hello_dinosaur               | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/environment/hello_dinosaur/
↪generate.sh
 _bin_sh_shell                | local.sh     | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/_bin_sh_shell/
↪generate.sh
 _bin_bash_shell              | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/_bin_bash_shell/
↪generate.sh
 bash_shell                   | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/shell_examples/bash_shell/
↪generate.sh
```

(continues on next page)

(continued from previous page)

```
 sh_shell                    | local.sh     | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/sh_shell/generate.sh
 shell_options               | local.sh     | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.sh/shell_examples/shell_options/
↪generate.sh
 slurm_down_nodes_reason     | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/slurm/slurm_down_nodes_reason/
↪generate.sh
 slurm_not_responding_nodes | local.bash   | PASS     |            0 | /Users/
↪siddiq90/Documents/buildtest/var/tests/local.bash/slurm/slurm_not_responding_nodes/
↪generate.sh

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

Executed 10 tests
Passed Tests: 8/10 Percentage: 80.000%
Failed Tests: 2/10 Percentage: 20.000%
```

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
Paths:

____________
Test Directory: /Users/siddiq90/Documents/buildtest/var/tests
```

(continues on next page)

(continued from previous page)

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

/Users/siddiq90/Documents/buildtest/tutorials/vars.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/gnu_hello.yml
/Users/siddiq90/Documents/buildtest/tutorials/systemd.yml
/Users/siddiq90/Documents/buildtest/tutorials/selinux.yml
/Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml
/Users/siddiq90/Documents/buildtest/tutorials/python-shell.yml
/Users/siddiq90/Documents/buildtest/tutorials/invalid_executor.yml
/Users/siddiq90/Documents/buildtest/tutorials/skip_tests.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/passing_args.yml
/Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
/Users/siddiq90/Documents/buildtest/tutorials/compilers/vecadd.yml

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

[skip] test is skipped.
 schemafile                | validstate   | buildspec
---------------------------+--------------+-----------------------------------------
↪-------------------------------------
 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/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/gnu_hello.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/shell_examples.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/invalid_executor.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/passing_args.yml
 script-v1.0.schema.json   | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/pass_returncode.yml
 compiler-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/compilers/vecadd.yml

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

(continues on next page)

(continued from previous page)

```
 Name                    | Schema File                | Test Path
↪                                                                                |␣
↪Buildspec
------------------------+----------------------------+--------------------------------
↪--------------------------------------------------------------------------------+--------
↪-----------------------------------------------------------------------
 variables               | script-v1.0.schema.json    | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/vars/variables/generate.sh                       | /
↪Users/siddiq90/Documents/buildtest/tutorials/vars.yml
 pre_post_build_run      | compiler-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/pre_post_build_run/pre_post_build_run/generate.sh | /
↪Users/siddiq90/Documents/buildtest/tutorials/compilers/pre_post_build_run.yml
 environment_variables   | script-v1.0.schema.json    | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/environment/environment_variables/generate.sh    | /
↪Users/siddiq90/Documents/buildtest/tutorials/environment.yml
 hello_f                 | compiler-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/gnu_hello/hello_f/generate.sh                    | /
↪Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml
 hello_c                 | compiler-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/gnu_hello/hello_c/generate.sh                    | /
↪Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml
 hello_cplusplus         | compiler-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/gnu_hello/hello_cplusplus/generate.sh            | /
↪Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml
 cc_example              | compiler-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/gnu_hello/cc_example/generate.sh                 | /
↪Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml
 fc_example              | compiler-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/gnu_hello/fc_example/generate.sh                 | /
↪Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml
 cxx_example             | compiler-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/gnu_hello/cxx_example/generate.sh                | /
↪Users/siddiq90/Documents/buildtest/tutorials/compilers/gnu_hello.yml
 systemd_default_target  | script-v1.0.schema.json    | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/systemd/systemd_default_target/generate.sh       | /
↪Users/siddiq90/Documents/buildtest/tutorials/systemd.yml
 selinux_disable         | script-v1.0.schema.json    | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/selinux/selinux_disable/generate.sh              | /
↪Users/siddiq90/Documents/buildtest/tutorials/selinux.yml
 _bin_sh_shell           | script-v1.0.schema.json    | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.sh/shell_examples/_bin_sh_shell/generate.sh           | /
↪Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml
 _bin_bash_shell         | script-v1.0.schema.json    | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/shell_examples/_bin_bash_shell/generate.sh       | /
↪Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml
 bash_shell              | script-v1.0.schema.json    | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/shell_examples/bash_shell/generate.sh            | /
↪Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml
 sh_shell                | script-v1.0.schema.json    | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.sh/shell_examples/sh_shell/generate.sh                | /
↪Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml
 shell_options           | script-v1.0.schema.json    | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.sh/shell_examples/shell_options/generate.sh           | /
↪Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml
 circle_area             | script-v1.0.schema.json    | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.python/python-shell/circle_area/generate.py           | /
↪Users/siddiq90/Documents/buildtest/tutorials/python-shell.yml
```

(continues on next page)

(continued from previous page)

```
 wrongexecutor          | script-v1.0.schema.json   | /Users/siddiq90/Documents/
↪buildtest/var/tests/badexecutor/invalid_executor/wrongexecutor/generate.sh       | /
↪Users/siddiq90/Documents/buildtest/tutorials/invalid_executor.yml
 unskipped              | script-v1.0.schema.json   | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/skip_tests/unskipped/generate.sh                  | /
↪Users/siddiq90/Documents/buildtest/tutorials/skip_tests.yml
 executable_arguments   | compiler-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/passing_args/executable_arguments/generate.sh     | /
↪Users/siddiq90/Documents/buildtest/tutorials/compilers/passing_args.yml
 exit1_fail             | script-v1.0.schema.json   | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.sh/pass_returncode/exit1_fail/generate.sh              | /
↪Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
 exit1_pass             | script-v1.0.schema.json   | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.sh/pass_returncode/exit1_pass/generate.sh              | /
↪Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
 returncode_mismatch    | script-v1.0.schema.json   | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.sh/pass_returncode/returncode_mismatch/generate.sh     | /
↪Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
 vecadd_gnu             | compiler-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/vecadd/vecadd_gnu/generate.sh                     | /
↪Users/siddiq90/Documents/buildtest/tutorials/compilers/vecadd.yml

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

[wrongexecutor]: Failed to Run Test
 name                   | executor     | status   |   returncode | testpath
------------------------+--------------+----------+--------------+--------------------
↪--------------------------------------------------------------------------------
↪----
 variables              | local.bash   | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/vars/variables/generate.sh
 pre_post_build_run     | local.bash   | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/pre_post_build_run/pre_post_build_run/
↪generate.sh
 environment_variables  | local.bash   | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/environment/environment_variables/generate.
↪sh
 hello_f                | local.bash   | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/gnu_hello/hello_f/generate.sh
 hello_c                | local.bash   | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/gnu_hello/hello_c/generate.sh
 hello_cplusplus        | local.bash   | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/gnu_hello/hello_cplusplus/generate.sh
 cc_example             | local.bash   | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/gnu_hello/cc_example/generate.sh
 fc_example             | local.bash   | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/gnu_hello/fc_example/generate.sh
 cxx_example            | local.bash   | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/gnu_hello/cxx_example/generate.sh
 systemd_default_target | local.bash   | FAIL     |            1 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/systemd/systemd_default_target/generate.sh
 selinux_disable        | local.bash   | FAIL     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/selinux/selinux_disable/generate.sh
 _bin_sh_shell          | local.sh     | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.sh/shell_examples/_bin_sh_shell/generate.sh
```

(continues on next page)

(continued from previous page)

```
 _bin_bash_shell         | local.bash   | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/shell_examples/_bin_bash_shell/generate.sh
 bash_shell              | local.bash   | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/shell_examples/bash_shell/generate.sh
 sh_shell                | local.sh     | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.sh/shell_examples/sh_shell/generate.sh
 shell_options           | local.sh     | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.sh/shell_examples/shell_options/generate.sh
 circle_area             | local.python | FAIL     |            2 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.python/python-shell/circle_area/generate.py
 unskipped               | local.bash   | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/skip_tests/unskipped/generate.sh
 executable_arguments    | local.bash   | PASS     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/passing_args/executable_arguments/generate.
↪sh
 exit1_fail              | local.sh     | FAIL     |            1 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_fail/generate.sh
 exit1_pass              | local.sh     | PASS     |            1 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_pass/generate.sh
 returncode_mismatch     | local.sh     | FAIL     |            2 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_mismatch/generate.
↪sh
 vecadd_gnu              | local.bash   | FAIL     |            0 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.bash/vecadd/vecadd_gnu/generate.sh



Error Messages from Stage: Run
_______________________________________________________________________________
[wrongexecutor]: executor badexecutor is not defined in /Users/siddiq90/.buildtest/
↪config.yml



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

Executed 23 tests
Passed Tests: 17/23 Percentage: 73.913%
Failed Tests: 6/23 Percentage: 26.087%
```

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
Paths:
_____________
Test Directory: /Users/siddiq90/Documents/buildtest/var/tests
```

(continues on next page)

(continued from previous page)

```
+------------------------------+
| Stage: 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
Paths:
__________
Test Directory: /Users/siddiq90/Documents/buildtest/var/tests

+------------------------------+
| Stage: 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                    | Schema File              | Test Path
↪                                                                  | Buildspec
-------------------------+--------------------------+-----------------------------------
↪-----------------------------------------------------------------+-----------------
↪-----------------------------------------
 systemd_default_target | script-v1.0.schema.json | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/systemd/systemd_default_target/generate.sh | /Users/
↪siddiq90/Documents/buildtest/tutorials/systemd.yml
```

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
Paths:
___________
Test Directory: /Users/siddiq90/Documents/buildtest/var/tests

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

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

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

 schemafile    | validstate    | buildspec
---------------+---------------+-------------



Error Messages from Stage: Parse
_____________________________________________________________________________
Skipping /Users/siddiq90/Documents/buildtest/tutorials/invalid_buildspec_section.yml␣
↪since it failed to validate

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

 Name    | Schema File    | Test Path    | Buildspec
---------+----------------+--------------+-------------

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

 name    | executor    | status    | returncode    | testpath
---------+-------------+-----------+---------------+------------
No tests were executed
```

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
Paths:
___________
Test Directory: /Users/siddiq90/Documents/buildtest/var/tests

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

(continues on next page)

(continued from previous page)

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

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

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

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

 Name           | Schema File              | Test Path                              ␣
↪                                                              | Buildspec
---------------+--------------------------+-----------------------------------------
↪-------------------------------------------------------------+-------------------------
↪-----------------------------------------
 wrongexecutor | script-v1.0.schema.json | /Users/siddiq90/Documents/buildtest/var/
↪tests/badexecutor/invalid_executor/wrongexecutor/generate.sh | /Users/siddiq90/
↪Documents/buildtest/tutorials/invalid_executor.yml

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

[wrongexecutor]: Failed to Run Test
 name    | executor    | status    | returncode    | testpath
---------+-------------+-----------+---------------+------------



Error Messages from Stage: Run
_____________________________________________________________________________________
[wrongexecutor]: executor badexecutor is not defined in /Users/siddiq90/.buildtest/
↪config.yml


No tests were executed
```

5.4.7 Buildtest Report

The `buildtest report` command will show result of all tests in a tabular form. Shown below is an example

```
$ buildtest report
+-----------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| id                                      | state   |   returncode | starttime    ␣
↪       | endtime             |    runtime | tags      | buildspec                  ␣
↪                                                     |
+=========================================+=========+==============+====================+=======
```

(continues on next page)

(continued from previous page)

```
| systemd_default_target_2020-09-02-21-41     | FAIL    |            1 | 2020/09/02␣
↪21:41:56 | 2020/09/02 21:41:56 | 0.0213593  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/systemd.yml                          |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| systemd_default_target_2020-09-02-21-42     | FAIL    |            1 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.0125463  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/systemd.yml                          |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| systemd_default_target_2020-09-02-21-42     | FAIL    |            1 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00771781 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/systemd.yml                          |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| slurm_down_nodes_reason_2020-09-02-21-42    | PASS    |            0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.0110754  |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/slurm.yml          |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| slurm_down_nodes_reason_2020-09-02-21-42    | PASS    |            0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.00830546 |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/slurm.yml          |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| slurm_not_responding_nodes_2020-09-02-21-42 | PASS    |            0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.00802073 |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/slurm.yml          |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| slurm_not_responding_nodes_2020-09-02-21-42 | PASS    |            0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.00736713 |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/slurm.yml          |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| _bin_sh_shell_2020-09-02-21-42              | PASS    |            0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.0125264  |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| _bin_sh_shell_2020-09-02-21-42              | PASS    |            0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.014642   |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| _bin_bash_shell_2020-09-02-21-42            | PASS    |            0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.00878803 |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
```

(continues on next page)

(continued from previous page)

```
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+----------+-----------------------------
↪-----------------------------------------------------+
| _bin_bash_shell_2020-09-02-21-42          | PASS    |           0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.00996381 |          | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+----------+-----------------------------
↪-----------------------------------------------------+
| bash_shell_2020-09-02-21-42               | PASS    |           0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.00561796 |          | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+----------+-----------------------------
↪-----------------------------------------------------+
| bash_shell_2020-09-02-21-42               | PASS    |           0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.00678369 |          | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+----------+-----------------------------
↪-----------------------------------------------------+
| sh_shell_2020-09-02-21-42                 | PASS    |           0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.0102359  |          | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+----------+-----------------------------
↪-----------------------------------------------------+
| sh_shell_2020-09-02-21-42                 | PASS    |           0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.0110021  |          | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+----------+-----------------------------
↪-----------------------------------------------------+
| shell_options_2020-09-02-21-42            | PASS    |           0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.0104085  |          | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+----------+-----------------------------
↪-----------------------------------------------------+
| shell_options_2020-09-02-21-42            | PASS    |           0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.0115846  |          | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+----------+-----------------------------
↪-----------------------------------------------------+
| circle_area_2020-09-02-21-42              | FAIL    |           2 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.0096071  |          | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/python-shell.yml   |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+----------+-----------------------------
↪-----------------------------------------------------+
| circle_area_2020-09-02-21-42              | FAIL    |           2 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.0117158  |          | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/python-shell.yml   |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+----------+-----------------------------
↪-----------------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| hello_dinosaur_2020-09-02-21-42              | PASS    |            0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.00698703 |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/environment.yml    |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_dinosaur_2020-09-02-21-42              | PASS    |            0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.0071777  |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/environment.yml    |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| variables_2020-09-02-21-42                   | PASS    |            0 | 2020/09/02␣
↪21:42:20 | 2020/09/02 21:42:20 | 0.00946478 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/vars.yml                           |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| pre_post_build_run_2020-09-02-21-42          | PASS    |            0 | 2020/09/02␣
↪21:42:20 | 2020/09/02 21:42:20 | 0.661319   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/pre_post_build_run.yml   |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| environment_variables_2020-09-02-21-42       | PASS    |            0 | 2020/09/02␣
↪21:42:20 | 2020/09/02 21:42:20 | 0.0120327  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/environment.yml                    |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_f_2020-09-02-21-42                     | PASS    |            0 | 2020/09/02␣
↪21:42:20 | 2020/09/02 21:42:20 | 0.00786752 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml            |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_f_2020-09-02-21-42                     | PASS    |            0 | 2020/09/02␣
↪21:42:43 | 2020/09/02 21:42:43 | 0.013958   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml            |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_c_2020-09-02-21-42                     | PASS    |            0 | 2020/09/02␣
↪21:42:20 | 2020/09/02 21:42:21 | 0.229417   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml            |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_c_2020-09-02-21-42                     | PASS    |            0 | 2020/09/02␣
↪21:42:43 | 2020/09/02 21:42:43 | 0.235872   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml            |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_cplusplus_2020-09-02-21-42             | PASS    |            0 | 2020/09/02␣
↪21:42:21 | 2020/09/02 21:42:21 | 0.461781   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml            |
```

(continues on next page)

(continued from previous page)

```
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_cplusplus_2020-09-02-21-42          | PASS    |           0 | 2020/09/02␣
↪21:42:43 | 2020/09/02 21:42:44 | 0.498086   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml           |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| cc_example_2020-09-02-21-42               | PASS    |           0 | 2020/09/02␣
↪21:42:21 | 2020/09/02 21:42:21 | 0.230038   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml           |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| cc_example_2020-09-02-21-42               | PASS    |           0 | 2020/09/02␣
↪21:42:44 | 2020/09/02 21:42:44 | 0.239393   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml           |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| fc_example_2020-09-02-21-42               | PASS    |           0 | 2020/09/02␣
↪21:42:21 | 2020/09/02 21:42:21 | 0.00840816 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml           |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| fc_example_2020-09-02-21-42               | PASS    |           0 | 2020/09/02␣
↪21:42:44 | 2020/09/02 21:42:44 | 0.0103057  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml           |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| cxx_example_2020-09-02-21-42              | PASS    |           0 | 2020/09/02␣
↪21:42:21 | 2020/09/02 21:42:22 | 0.45725    | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml           |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| cxx_example_2020-09-02-21-42              | PASS    |           0 | 2020/09/02␣
↪21:42:44 | 2020/09/02 21:42:44 | 0.505495   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml           |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| selinux_disable_2020-09-02-21-42          | FAIL    |           0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00954966 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/selinux.yml                       |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| _bin_sh_shell_2020-09-02-21-42            | PASS    |           0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.0126632  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/shell_examples.yml                |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| _bin_bash_shell_2020-09-02-21-42           | PASS    |            0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.0085432  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/shell_examples.yml                |
+-------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| bash_shell_2020-09-02-21-42                | PASS    |            0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00603703 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/shell_examples.yml                |
+-------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| sh_shell_2020-09-02-21-42                  | PASS    |            0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00911067 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/shell_examples.yml                |
+-------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| shell_options_2020-09-02-21-42             | PASS    |            0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.0104522  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/shell_examples.yml                |
+-------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| circle_area_2020-09-02-21-42               | FAIL    |            2 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00808543 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/python-shell.yml                  |
+-------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| unskipped_2020-09-02-21-42                 | PASS    |            0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00792155 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/skip_tests.yml                    |
+-------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| unskipped_2020-09-02-21-42                 | PASS    |            0 | 2020/09/02␣
↪21:42:49 | 2020/09/02 21:42:49 | 0.042425   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/skip_tests.yml                    |
+-------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| executable_arguments_2020-09-02-21-42      | PASS    |            0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.232554   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/passing_args.yml        |
+-------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| exit1_fail_2020-09-02-21-42                | FAIL    |            1 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00782586 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/pass_returncode.yml               |
+-------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| exit1_fail_2020-09-02-21-42                | FAIL    |            1 | 2020/09/02␣
↪21:42:48 | 2020/09/02 21:42:48 | 0.0132099  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/pass_returncode.yml               |
```

(continues on next page)

(continued from previous page)

```
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| exit1_pass_2020-09-02-21-42                 | PASS    |           1 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00772625 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/pass_returncode.yml               |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| exit1_pass_2020-09-02-21-42                 | PASS    |           1 | 2020/09/02␣
↪21:42:48 | 2020/09/02 21:42:48 | 0.0110667  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/pass_returncode.yml               |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| returncode_mismatch_2020-09-02-21-42        | FAIL    |           2 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00836054 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/pass_returncode.yml               |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| returncode_mismatch_2020-09-02-21-42        | FAIL    |           2 | 2020/09/02␣
↪21:42:48 | 2020/09/02 21:42:48 | 0.00940911 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/pass_returncode.yml               |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| vecadd_gnu_2020-09-02-21-42                 | FAIL    |           0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.0267169  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/vecadd.yml              |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| vecadd_gnu_2020-09-02-21-42                 | FAIL    |           0 | 2020/09/02␣
↪21:42:46 | 2020/09/02 21:42:46 | 0.0421426  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/vecadd.yml              |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
```

buildtest will store result metadata of each test in a file `var/report.json` which is found in root of buildtest. This file is updated upon every `buildtest build` command. For more information see *Test Reports (buildtest report)*.

5.4.8 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: `DEBUG`, `INFO`, `WARNING`, `ERROR`, `CRITICAL` which controls log level to be displayed in console. buildtest is using logging.setLevel to control log level.

The same content is logged in **buildtest.log** with default log level of `DEBUG`. If you want to get all logs use `-d DEBUG` with your buildtest command:

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

5.4.9 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 behavior of buildtest. There is a json schema file settings.schema.json that defines structure on how to write your configuration file.

For more details on schema attributes see Settings Schema Documentation

5.5.1 Default Configuration

The default configuration for buildtest can be found in the git repo relative to root of buildtest at `buildtest/settings/config.yml`. User may override the default configuration by creating their custom file in `$HOME/.buildtest/config.yml`.

Shown below is the default configuration.

```
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
config:
  editor: vi
```

5.5.2 Executors

Executors are responsible for running jobs, currently buildtest supports the following executors:

- local
- slurm
- lsf

Their is a **ssh** executor is supported in schema but currently not implemented in buildtest.

The local executor is responsible for submitting jobs locally. Currently, buildtest supports `bash`, `sh` and `python` shell. The executors are referenced in your buildspec with the `executor` key as follows:

```
executor: local.bash
```

The `executor` key in buildtest settings is of type `object`, the sub-fields are `local`, `ssh`, and `slurm`.

Local Executors

In this example below we define a local executor named *bash* that is referenced in buildspec `executor: local.bash`:

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

Each local executor requires the `shell` key which takes the pattern `^(/bin/bash|/bin/sh|sh|bash|python).*`

Any buildspec that references the executor `local.bash` will submit job as `bash /path/to/test.sh`.

You can pass options to shell which will get passed into each job submission. For instance if you want bash executor to submit jobs by login mode 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`.

Slurm Executors

The slurm executors are defined in the following section:

```
executors:
  slurm:
    <slurm-executor1>:
    <slurm-executor2>:
```

Slurm executors are responsible for submitting jobs to slurm resource manager. You can define as many slurm executors as you wish, so long as you have a unique name to reference each executor. Generally, you will need one slurm executor per partition or qos that you have at your site. Let's take a look at an example slurm executor called `normal`:

```
executors:
  slurm:
    normal:
      options: ["-C haswell"]
      qos: normal
```

This executor can be referenced in buildspec as `executor: slurm.normal`. This executor defines the following:

- `qos: normal` will add `-q normal` to the launcher command. buildtest will check if qos is found in slurm configuration. If not found, buildtest will reject job submission.
- `options` key is used to pass any options to launcher command. In this example we add `-C haswell`.

5.5.3 buildtest configuration for Cori @ NERSC

Let's take a look at Cori buildtest configuration:

```
executors:

  defaults:
    pollinterval: 10
    launcher: sbatch

  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

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

    shared:
      description: jobs for shared qos
      qos: shared

    bigmem:
      description: bigmem jobs
      cluster: escori
      qos: bigmem

    xfer:
      description: xfer qos jobs
      qos: xfer

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

config:
  editor: vi
  paths:
    prefix: $HOME/cache/
```

In this setting, we define 3 LocalExecutors: `local.bash`, `local.sh` and `local.python` and 5 SlurmExecutors: `slurm.debug`, `slurm.shared`, `slurm.bigmem`, `slurm.xfer`, and `slurm.gpu`. We also introduce section `defaults` section to default configuration for executors.

At the moment, the `launcher` and `pollinterval` are available fields in default which only apply for SlurmExecutor and LSFExecutor. Currently, buildtest supports batch submission via `sbatch` so all SlurmExecutors will inherit `sbatch` as launcher. The `pollinterval` field is used with SlurmExecutor to poll jobs at set interval in seconds

when job active in queue (`PENDING`, `RUNNING`).

At Cori, jobs are submitted via qos instead of partition so each slurm executor has the *qos* key. The `description` key is a brief description of the executor which you can use to document the behavior of the executor. The `cluster` field specifies which slurm cluster to use, at Cori in order to use `bigmem` qos we need to specify `-M escori` where escori is the slurm cluster. buildtest will detect slurm configuration and check if cluster is a valid cluster name. In addition, *sacct* will poll job against the cluster name (`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 this executor for submit to CoriGPU which requires `sbatch -M escori -C gpu`. Any additional #SBATCH options are defined in buildspec using `sbatch` key.

5.5.4 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 define LSF Executor we set top-level key *lsf* in *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.

```
executors:
  defaults:
    launcher: bsub
    pollinterval: 10
  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
    test:
      queue: test
config:
  editor: vi
  paths:
    prefix: /tmp
```

5.5.5 buildspec roots

buildtest can detect buildspec using `buildspec_roots` keyword. For example we clone the repo https://github.com/buildtesters/buildtest-cori at **/Users/siddiq90/Documents/buildtest-cori**

```
config:
  editor: vi
  paths:
    buildspec_roots:
      - /Users/siddiq90/Documents/buildtest-cori
```

If you run `buildtest buildspec find --clear` it will detect all buildspecs in buildspec_roots. buildtest will find all *.yml* extension. By default buildtest will add the `$BUILDTEST_ROOT/tutorials` to search path, where $BUILDTEST_ROOT is root of buildtest repo.

5.5.6 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
Valid State: True
_____________________________________________________________________________


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
config:
  editor: vi
  paths:
    prefix: /tmp
```

(continues on next page)

(continued from previous page)

```
File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/settings.schema.
↪json/valid/ssh-executor.yml
Valid State: True
_____________________________________________________________________________________


executors:
  ssh:
    localhost:
      host: localhost
      user: siddiq90
      identity_file: ~/.ssh/id_rsa
config:
  editor: vi
  paths:
    prefix: /tmp



File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/settings.schema.
↪json/valid/slurm-example.yml
Valid State: True
_____________________________________________________________________________________


executors:
  defaults:
    pollinterval: 20
  slurm:
    normal:
      options: ["-C haswell"]
      qos: normal
      before_script: |
        time
        echo "commands run before job"
      after_script: |
        time
        echo "commands run after job"

config:
  editor: vi
  paths:
    prefix: /tmp



File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/settings.schema.
↪json/valid/combined_executor.yml
Valid State: True
_____________________________________________________________________________________


executors:
  local:
    bash:
      description: submit jobs on local machine
```

(continues on next page)

(continued from previous page)

```
      shell: bash -v

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

  lsf:
    batch:
      launcher: bsub
      options:
        - "-q batch"
        - "-t 00:10"
  ssh:
     login:
       host: cori
       user: root
       identity_file: ~/.ssh/nersc

config:
  editor: vi
  paths:
    prefix: /tmp
    clonepath: /tmp/repo
    logdir: /tmp/logs
    testdir: /tmp/buildtest/tests



File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/settings.schema.
↪json/valid/lsf-example.yml
Valid State: True
_____________________________________________________________________________


executors:
  defaults:
    pollinterval: 10
    launcher: bsub
  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:
      description: "LSF Executor name 'test' that submits jobs to 'test' queue"
      launcher: bsub
      queue: test
      options: ["-W 20"]
```

(continues on next page)

(continued from previous page)

```
config:
  editor: vi
```

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

5.6 Builder Process

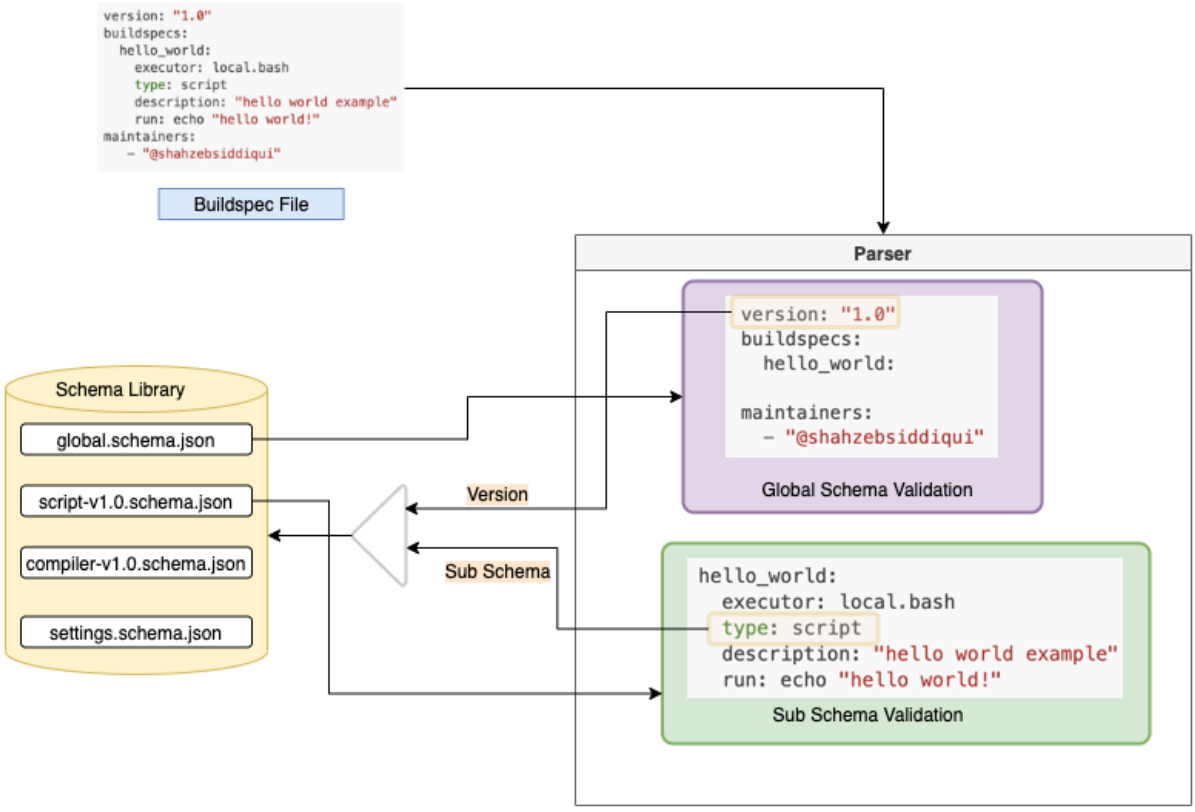
buildtest will process all buildspecs that are discovered see diagram *Discover Buildspecs*. The **BuildspecParser** class is responsible for validating the buildspec. The validation is performed using jsonschema.validate. The parser will validate every buildspec using the global.schema.json which validates 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 *Control builds by Stages* 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.

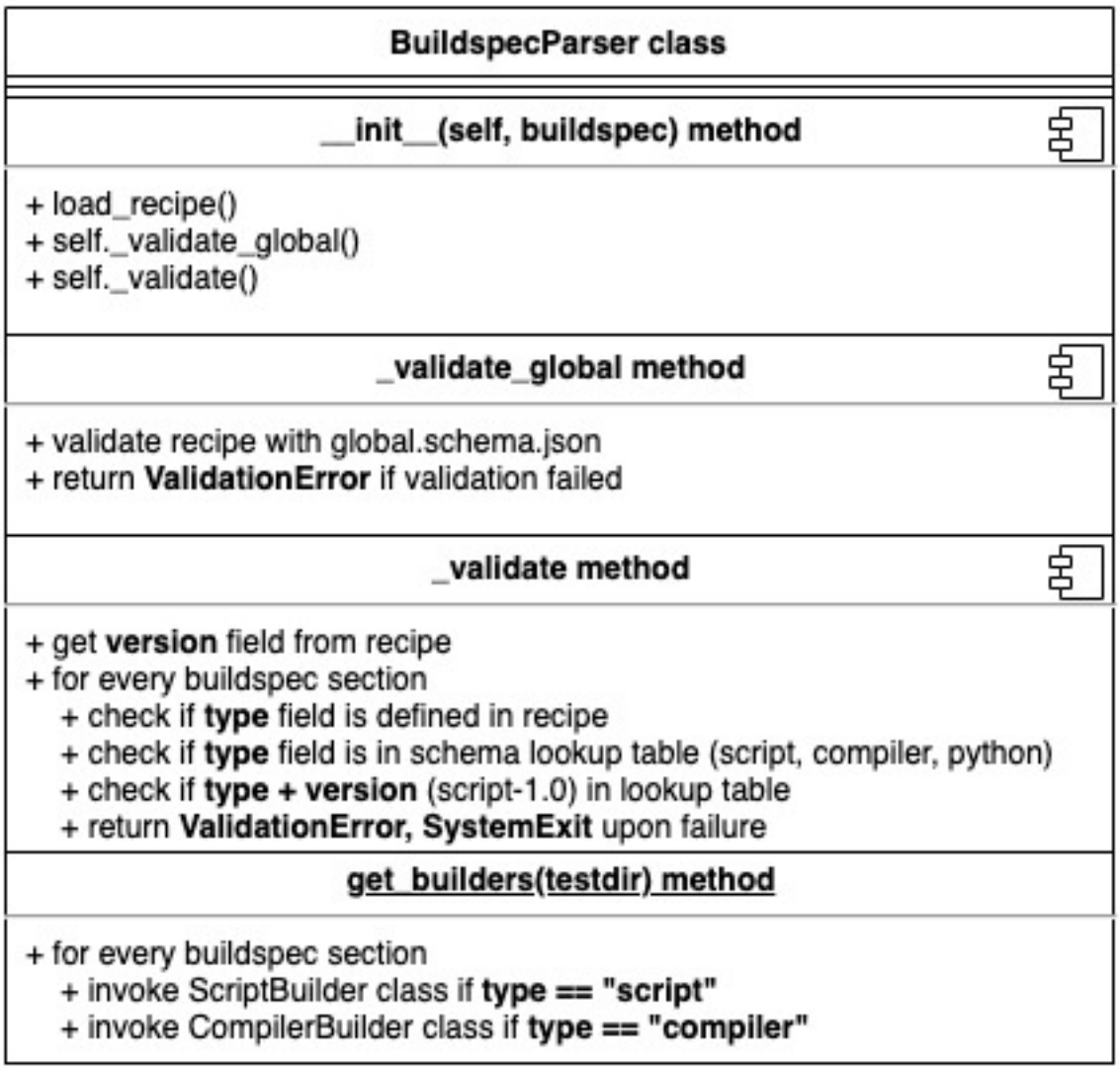
In this diagram, a buildspec file is passed to the Parser and validated with global schema and a sub-schema.



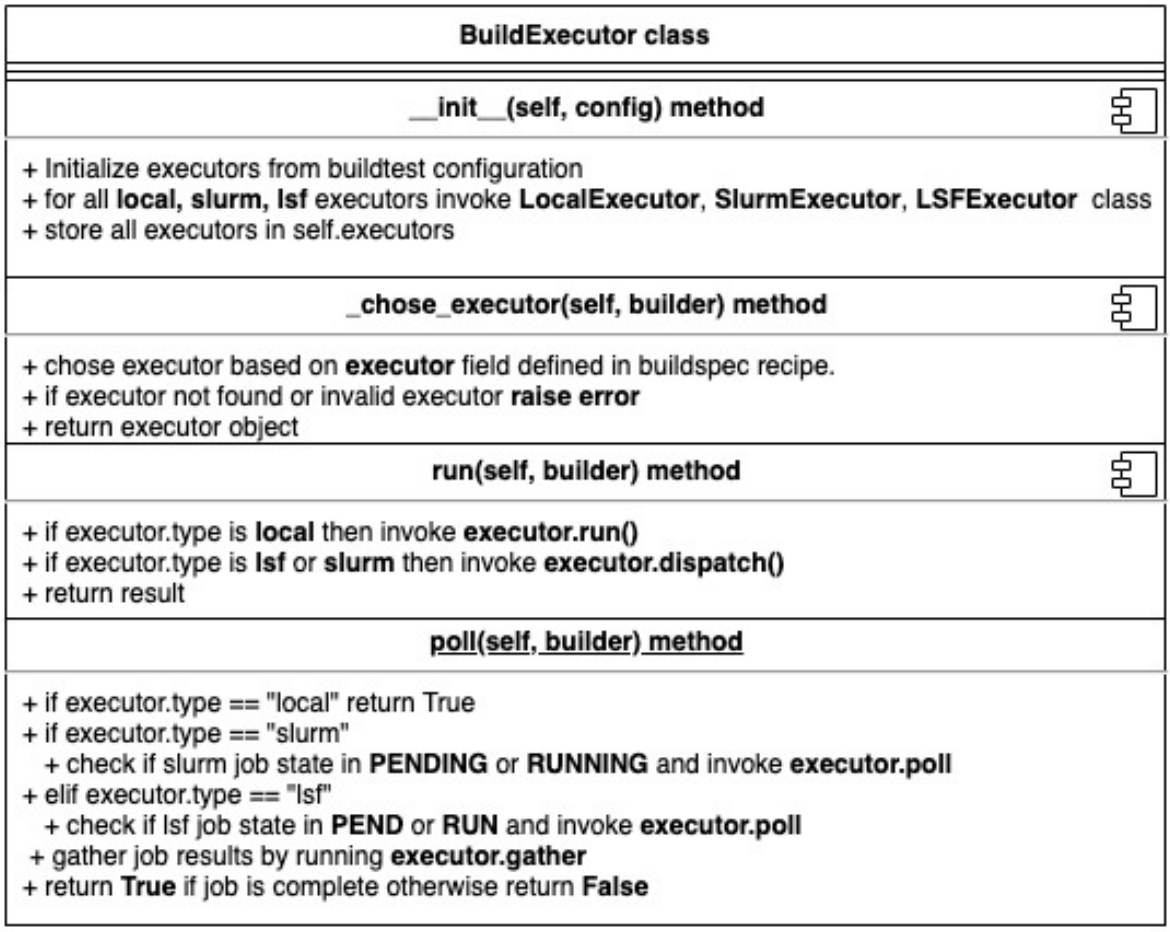
buildtest will invoke BuildspecParser against all discovered buildspecs and catch exceptions **ValidationError** and **SystemExit** and ignore those buildspecs. Next buildtest will build each test, this is implemented using base class *BuilderBase*. The subclass for **BuilderBase** such as **ScriptBuilder** 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 Schemas

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.

For more details see Global Schema Documentation.

Schema Files

- Production Schema
- Development Schema

Global Keys in buildspec

The global keys required for buildspec are `version` and `buildspecs`. The version key is required to validate with sub-schema when used with `type` field. The `buildspecs` is the start of test section. The `maintainers` is an optional field that is an array which can be used to identity maintainer of test. 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
    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 sub-schema is the following section which is validated with the `type` schema:

```
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 type schema, in this example we validate with the schema script-v1.0.schema.json.

Test Names

The **buildspecs** is an object that defines one or more test, the test names take the following pattern `"^[A-Za-z_][A-Za-z0-9_]*$"`. In the previous example the test name is **hello_world**. You must have unique test names in your **buildspecs** section, otherwise you will have an invalid buildspec file.

Note: We refer to the entire YAML content as **buildspec file**, this is not to be confused with the **buildspecs** field.

You may define multiple tests in a single buildspec file, shown below is an example using both script and compiler schema:

```
version: "1.0"
buildspecs:
  hello_f:
    type: compiler
    description: "Hello World Fortran Compilation"
    executor: local.bash
    module:
      - "module purge && module load gcc"
    build:
      source: "src/hello.f90"
      name: gnu
      fflags: -Wall

  environment_variables:
    executor: local.bash
    type: script
    env:
      FIRST_NAME: avocado
      LAST_NAME: dinosaur
    run: |
      hostname
      whoami
      echo $USER
      printf "${FIRST_NAME} ${LAST_NAME}\n"
```

In this example we have two tests **hello_f** and **environment_variables**. The test `hello_f` is using the compiler-v1.0.schema.json for validation because `type: compiler` is set in sub-schema while `environment_variables` test is using script-v1.0.schema.json for validation because `type: script` is set.

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`.

If you have got this far you may proceed with *Writing buildspecs*

Examples

You can see below a list of global schema examples that can be accessible via `buildtest schema -n global.schema.json -e`

```
$ buildtest schema -n global.schema.json -e



File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/global.schema.
↪json/valid/examples.yml
Valid State: True
_____________________________________________________________________________


version: "1.0"
buildspecs:
  # testing all caps
  ABCDEFGHIJKLMNOPQRSTUVWXYZ:
```

(continues on next page)

(continued from previous page)

```
    type: script
    executor: local.bash
    run: "hostname"

  # testing all lowercase letters
  abcdefghijklmnopqrstuvwxyz:
    type: script
    executor: local.bash
    run: "hostname"

  # testing both caps and lowercase and numbers
  ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789:
    type: script
    executor: local.bash
    run: "hostname"

  # testing '_' followed by all caps, lowercase and numbers
  _ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789:
    type: script
    executor: local.bash
    run: "hostname"

  # testing '_' in middle and end of word
  ABCDEFGHIJKLMNOPQRSTUVWXYZ_abcdefghijklmnopqrstuvwxyz_0123456789_:
    type: script
    executor: local.bash
    run: "hostname"



File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/global.schema.
↪json/invalid/maintainers_type_mismatch.yml
Valid State: FAIL
_______________________________________________________________________________


version: "1.0"
# wrong type for maintainers key, expects a string
maintainers: 1
buildspecs:
  hostname:
    type: script
    run: "hostname"
_____________________________________ Validation Error ____________________________
↪_____________
1 is not of type 'array'

Failed validating 'type' in schema['properties']['maintainers']:
    {'description': 'One or more maintainers or aliases',
     'items': {'type': 'string'},
     'minItems': 1,
     'type': 'array'}

On instance['maintainers']:
    1
```

(continues on next page)

(continued from previous page)

```
File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/global.schema.
↪json/invalid/invalid_pattern.yml
Valid State: FAIL
_____________________________________________________________________________


version: "1.0"
buildspecs:
  # invalid pattern for test. Must be matching regex "^[A-Za-z_.][A-Za-z0-9_]*$" when␣
↪declaring a dict
  (badname:
    type: script
    run: "ping login 1"
_____________________________________ Validation Error ____________________________
↪_____________
'(badname' does not match '^[A-Za-z_][A-Za-z0-9_]*$'

Failed validating 'pattern' in schema['properties']['buildspecs']['propertyNames']:
    {'pattern': '^[A-Za-z_][A-Za-z0-9_]*$'}

On instance['buildspecs']:
    '(badname'



File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/global.schema.
↪json/invalid/missing-version.yml
Valid State: FAIL
_____________________________________________________________________________


buildspecs:
  # Shell would be accepted to indicate a single line shell command (or similar)
  login_node_check:
    type: script
    run: "ping login 1"
_____________________________________ Validation Error ____________________________
↪_____________
'version' is a required property

Failed validating 'required' in schema:
    {'$id': 'https://buildtesters.github.io/schemas/schemas/global.schema.json',
     '$schema': 'http://json-schema.org/draft-07/schema#',
     'additionalProperties': False,
     'definitions': {'bsub': {'description': 'This field is used for '
                                             'specifying #BSUB options in '
                                             'test script. buildtest will '
                                             'insert #BSUB in front of '
                                             'each value',
                              'items': {'type': 'string'},
                              'type': 'array'},
                     'env': {'description': 'One or more key value pairs '
                                            'for an environment '
                                            '(key=value)',
                             'items': {'minItems': 1,
                                       'propertyNames': {'pattern': '^[A-Za-z_][A-Za-
↪z0-9_]*$'},
```

(continues on next page)

(continued from previous page)

```
                                    'type': 'object'},
                           'minItems': 1,
                           'type': 'object'},
                  'executor': {'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. ',
                               'type': 'string'},
                  'sbatch': {'description': 'This field is used for '
                                            'specifying #SBATCH options '
                                            'in test script. buildtest '
                                            'will insert #SBATCH in '
                                            'front of each value',
                             'items': {'type': 'string'},
                             'type': 'array'},
                  'skip': {'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.',
                           'type': 'boolean'},
                  'status': {'additionalProperties': False,
                             '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.',
                             'properties': {'regex': {'description': 'Perform '
                                                                     'regular '
                                                                     'expression '
                                                                     'search '
                                                                     'using '
                                                                     '``re.
↪search`` '
                                                                     'python '
                                                                     'module '
                                                                     'on '
                                                                     'stdout/
↪stderr '
                                                                     'stream '
                                                                     'for '
                                                                     'reporting '
                                                                     'if '
                                                                     'test '
                                                                     '``PASS``. ',
                                                          'properties': {'exp': {
↪'description': 'Specify '
```

(continues on next page)

(continued from previous page)

```
↪          'a '                                                                                                    ␣
↪          'regular '                                                                                              ␣
↪          'expression '                                                                                           ␣
↪          'to '                                                                                                   ␣
↪          'run '                                                                                                  ␣
↪          'with '                                                                                                 ␣
↪          'input '                                                                                                ␣
↪          'stream '                                                                                               ␣
↪          'specified '                                                                                            ␣
↪          'by '                                                                                                   ␣
↪          ``stream`` '                                                                                            ␣
↪          'field. '                                                                                               ␣
↪          'buildtest '                                                                                            ␣
↪          'uses '                                                                                                 ␣
↪          're.search '                                                                                            ␣
↪          'when '                                                                                                 ␣
↪          'performing '                                                                                           ␣
↪          'regex',                                                                                                'type
↪': 'string'},                                                                                                   'stream': {
↪'description': 'The '                                                                                                ␣
↪             'stream '                                                                                            ␣
↪             'field '                                                                                             ␣
↪             'can '                                                                                               ␣
↪             'be '                                                                                                ␣
↪             'stdout '                                                                                            ␣
↪             'or '                                                                                                ␣
↪             'stderr. '                                                                                           ␣
↪             'buildtest '                                                                                         ␣
↪             'will '                                                                                              ␣
```

(continues on next page)

(continued from previous page)

```
↪            'read '
↪            'the '
↪            'output '
↪            'or '
↪            'error '
↪            'stream '
↪            'after '
↪            'completion '
↪            'of '
↪            'test '
↪            'and '
↪            'check '
↪            'if '
↪            'regex '
↪            'matches '
↪            'in '
↪            'stream',
↪'enum': ['stdout',
↪      'stderr'],
↪'type': 'string'}},
                                            'required': ['stream',
                                                         'exp'],
                                            'type': 'object'},
                                      'returncode': {'description': 'By '
                                                                    'default,
↪ '
↪'returncode '
                                                                    '0 '
                                                                    'is '
                                                                    'PASS, '
                                                                    'if '
                                                                    'you '
                                                                    'want '
                                                                    'to '
                                                                    'emulate
↪'
```

(continues on next page)

(continued from previous page)

```
                                                                    'a '
                                                                    'non-
↪zero '

↪'returncode '
                                                                    'to '
                                                                    'pass '
                                                                    'then '
                                                                    'specify
↪'
                                                                    'an '

↪'expected '
                                                                    'return '
                                                                    'code. '

↪'buildtest '
                                                                    'will '
                                                                    'match '
                                                                    'actual '

↪'returncode '
                                                                    'with '
                                                                    'one '
                                                                    'defined
↪'
                                                                    'in '
                                                                    'this '
                                                                    'field, '
                                                                    'if '
                                                                    'there '
                                                                    'is '
                                                                    'a '
                                                                    'match '

↪'buildtest '
                                                                    'will '
                                                                    'report '
                                                                    'as '
                                                                    '``PASS``
↪',
                                                      'type': 'integer'},
                                          'slurm_job_state_codes': {'description
↪': 'This '
                                                                             ␣
↪  'field '                                                                  ␣
↪  'can '                                                                    ␣
↪  'be '                                                                     ␣
↪  'used '                                                                   ␣
↪  'for '                                                                    ␣
↪  'checking '                                                               ␣
↪  'Slurm '
```

(continues on next page)

(continued from previous page)

```
↪  'Job '                                                                    ␣
↪  'State, '                                                                 ␣
↪  'if '                                                                     ␣
↪  'there '                                                                  ␣
↪  'is '                                                                     ␣
↪  'a '                                                                      ␣
↪  'match '                                                                  ␣
↪  'buildtest '                                                              ␣
↪  'will '                                                                   ␣
↪  'report '                                                                 ␣
↪  'as '                                                                     ␣
↪  '``PASS`` ',                                                              ␣
                                                                  'enum': [
↪'COMPLETED',
↪'FAILED',
↪'OUT_OF_MEMORY',
↪'TIMEOUT'],
                                                                  'type':
↪'string'}},
                              'type': 'object'},
                   'tags': {'description': 'Classify tests using a tag '
                                           'name, this can be used for '
                                           'categorizing test and '
                                           'building tests using '
                                           '``--tags`` option',
                            'items': {'type': 'string'},
                            'minItems': 1,
                            'type': 'array'}},
    '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',
    'properties': {'buildspecs': {'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_
↪]*$'},
                                  'type': 'object'},
                   'maintainers': {'description': 'One or more '
                                                  'maintainers or aliases',
```

(continues on next page)

(continued from previous page)

```
                                    'items': {'type': 'string'},
                                    'minItems': 1,
                                    'type': 'array'},
                     'version': {'description': 'The semver version of the '
                                                'schema to use (x.x).',
                                 'type': 'string'}},
     'required': ['version', 'buildspecs'],
     'title': 'global schema',
     'type': 'object'}

On instance:
    {'buildspecs': {'login_node_check': {'run': 'ping login 1',
                                         'type': 'script'}}}
```

global.schema.json

```
$ buildtest schema -n global.schema.json -j
{
  "$id": "https://buildtesters.github.io/schemas/schemas/global.schema.json",
  "$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",
      "propertyNames": {
        "pattern": "^[A-Za-z_][A-Za-z0-9_]*$"
      }
    }
  },
  "definitions": {
    "env": {
      "type": "object",
```

(continues on next page)

(continued from previous page)

```
      "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_]*$"
        }
      }
    },
    "tags": {
      "type": "array",
      "description": "Classify tests using a tag name, this can be used for␣
↪categorizing test and building tests using ``--tags`` option",
      "minItems": 1,
      "items": {
        "type": "string"
      }
    },
    "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",
      "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"
      }
    },
    "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_codes": {
          "type": "string",
          "enum": [
```

(continues on next page)

(continued from previous page)

```
            "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": {
          "type": "integer",
          "description": "By default, returncode 0 is PASS, if you want to emulate a␣
↪non-zero returncode to pass then specify an expected return code. buildtest will␣
↪match actual returncode with one defined in this field, if there is a match␣
↪buildtest will report as ``PASS``"
        },
        "regex": {
          "type": "object",
          "description": "Perform regular expression search using ``re.search``␣
↪python module on stdout/stderr stream for reporting if test ``PASS``. ",
          "properties": {
            "stream": {
              "type": "string",
              "enum": [
                "stdout",
                "stderr"
              ],
              "description": "The stream field can be stdout or stderr. buildtest␣
↪will read the output or error stream after completion of test and check if regex␣
↪matches in stream"
            },
            "exp": {
              "type": "string",
              "description": "Specify a regular expression to run with input stream␣
↪specified by ``stream`` field. buildtest uses re.search when performing regex"
            }
          },
          "required": [
            "stream",
            "exp"
          ]
        }
      }
    }
  }
}
```

5.7.2 Writing buildspecs

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.

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 your buildtest settings in *Configuring buildtest*.

In this example we define variables using the *vars* section 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. The Production Schema and Development Schema are kept in sync where any schema development is done in **Development Schema** followed by merge to **Production Schema**.

Return Code Matching

In this next example we will illustrate the concept of returncode match with different exit codes. In this example we have three tests called `exit1_fail`, `exit1_pass` and `returncode_mismatch`. All test are using the `local.sh` executor which is using `sh` to run the test. We expect **exit1_fail** and **returncode_mismatch** to FAIL while **exit1_pass** will PASS since returncode matches

```
version: "1.0"
buildspecs:

  exit1_fail:
    executor: local.sh
    type: script
    description: exit 1 by default is FAIL
    run: exit 1

  exit1_pass:
    executor: local.sh
    type: script
    description: report exit 1 as PASS
    run: exit 1
    status:
      returncode: 1

  returncode_mismatch:
    executor: local.sh
    type: script
    description: exit 2 failed since it failed to match returncode 1
    run: exit 2
    status:
      returncode: 1
```

To demonstrate we will build this test and pay close attention to the Status field in output

```
$ buildtest build -b tutorials/pass_returncode.yml
Paths:
___________
Test Directory: /Users/siddiq90/Documents/buildtest/var/tests

+-------------------------------+
| Stage: 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                 | Schema File              | Test Path
↪                                                                                | Buildspec
---------------------+--------------------------+-----------------------------------
↪--------------------------------------------------------------------------+-----------------
↪-----------------------------------------------
 exit1_fail           | script-v1.0.schema.json | /Users/siddiq90/Documents/buildtest/
↪var/tests/local.sh/pass_returncode/exit1_fail/generate.sh           | /Users/
↪siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
 exit1_pass           | script-v1.0.schema.json | /Users/siddiq90/Documents/buildtest/
↪var/tests/local.sh/pass_returncode/exit1_pass/generate.sh           | /Users/
↪siddiq90/Documents/buildtest/tutorials/pass_returncode.yml
 returncode_mismatch | script-v1.0.schema.json | /Users/siddiq90/Documents/buildtest/
↪var/tests/local.sh/pass_returncode/returncode_mismatch/generate.sh | /Users/
↪siddiq90/Documents/buildtest/tutorials/pass_returncode.yml

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

 name                 | executor    | status    |   returncode | testpath
----------------------+-------------+-----------+--------------+-------------------------
↪-------------------------------------------------------------------------------
 exit1_fail           | local.sh    | FAIL      |            1 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_fail/generate.sh
 exit1_pass           | local.sh    | PASS      |            1 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.sh/pass_returncode/exit1_pass/generate.sh
 returncode_mismatch | local.sh    | FAIL      |            2 | /Users/siddiq90/
↪Documents/buildtest/var/tests/local.sh/pass_returncode/returncode_mismatch/generate.
↪sh

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

(continues on next page)

(continued from previous page)

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

Executed 3 tests
Passed Tests: 1/3 Percentage: 33.333%
Failed Tests: 2/3 Percentage: 66.667%
```

Python example

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: ["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 of 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: true
    run: hostname

  unskipped:
    type: script
```

(continues on next page)

(continued from previous page)

```
    executor: local.bash
    skip: false
    run: hostname
```

The first test *skip* will be skipped by buildtest because `skip:  true` is defined.

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
Paths:
___________
Test Directory: /Users/siddiq90/Documents/buildtest/var/tests

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

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

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

[skip] test is skipped.
 schemafile              | validstate   | buildspec
-------------------------+--------------+-------------------------------------------
↪-----------------
 script-v1.0.schema.json | True         | /Users/siddiq90/Documents/buildtest/
↪tutorials/skip_tests.yml

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

 Name      | Schema File             | Test Path                                      ␣
↪                                          | Buildspec
-----------+-------------------------+-----------------------------------------------
↪-----------------------------------------+---------------------------------------
↪----------------------
 unskipped | script-v1.0.schema.json | /Users/siddiq90/Documents/buildtest/var/tests/
↪local.bash/skip_tests/unskipped/generate.sh | /Users/siddiq90/Documents/buildtest/
↪tutorials/skip_tests.yml

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

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

(continues on next page)

(continued from previous page)

```
 unskipped | local.bash | PASS     |            0 | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/skip_tests/unskipped/generate.sh

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

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

Script Schema and Examples

buildtest command line interface provides access to schemas and example buildspecs for each schemas.

To retrieve the full json schema of script schema you can run the following:

```
buildtest schema -n script-v1.0.schema.json --json
```

The example buildspecs are validated with the schema, so they are self-documentating examples. For example, you can retrieve the script examples using `buildtest schema -n script-v1.0.schema.json -e`. Shown below we show valid and invalid examples. The examples are validated with the schema `script-v1.0.schema.json`.

```
$ buildtest schema -n script-v1.0.schema.json -e



File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/script-v1.0.
↪schema.json/valid/examples.yml
Valid State: True
_____________________________________________________________________________


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
    run: "hostname"

  declare_env:
    executor: local.bash
    type: script
    description: declaring environment variables
    env:
      FOO: BAR
```

(continues on next page)

(continued from previous page)

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


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

(continues on next page)

(continued from previous page)

```
  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:
  executor: local.bash
  type: script
  description: This test pass because using a valid return code
  run: hostname
  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:
  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

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

(continues on next page)

(continued from previous page)

```
  tag_example:
    type: script
    executor: local.bash
    description: This is a tag example
    sbatch:
      - "-t 10:00:00"
      - "-p normal"
      - "-N 1"
      - "-n 8"
    tags: [slurm]
    run: hostname



File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/script-v1.0.
↪schema.json/invalid/examples.yml
Valid State: FAIL
_____


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
```

(continues on next page)

(continued from previous page)

```
  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:
        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"]

  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"]
    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
```

(continues on next page)

(continued from previous page)

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

  invalid_tags_value:
    type: script
    executor: local.bash
    description: invalid tag value must be all string items
    tags: ["compiler", 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

_____________________________________________ Validation Error _____________________________
↪_____________
'invalid_test_name_&!@#$%' does not match '^[A-Za-z_][A-Za-z0-9_]*$'

Failed validating 'pattern' in schema['properties']['buildspecs']['propertyNames']:
    {'pattern': '^[A-Za-z_][A-Za-z0-9_]*$'}

On instance['buildspecs']:
    'invalid_test_name_&!@#$%'
```

5.7.3 Compiler Schema

The compiler schema is used for compilation of programs, currently we support single source file compilation.

Schema Files

- Production Schema
- Development Schema

Compilation Examples

In order to use the compiler schema you must set `type: compiler` in your sub-schema. We assume the reader has basic understanding of *Global Schema* validation.

The **type**, **compiler**, and **executor** are required keys for the 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]
    build:
      source: "src/hello.f90"
      name: gnu
      fflags: -Wall

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

  hello_cplusplus:
    type: compiler
    description: "Hello World C++ Compilation"
    executor: local.bash
    tags: [tutorials]
    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
    tags: [tutorials]
    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]
    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]
```

(continues on next page)

(continued from previous page)

```
    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
Paths:
___________
Test Directory: /Users/siddiq90/Documents/buildtest/var/tests

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

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

+---------------------------+
| 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             | Schema File               | Test Path
↪                                                        | Buildspec
-----------------+---------------------------+-----------------------------------
↪-------------------------------------------------------+--------------------------
↪----------------------------------------
```

(continues on next page)

(continued from previous page)

```
 hello_f         | compiler-v1.0.schema.json | /Users/siddiq90/Documents/buildtest/
↪var/tests/local.bash/gnu_hello/hello_f/generate.sh         | /Users/siddiq90/
↪Documents/buildtest/tutorials/compilers/gnu_hello.yml
 hello_c         | compiler-v1.0.schema.json | /Users/siddiq90/Documents/buildtest/
↪var/tests/local.bash/gnu_hello/hello_c/generate.sh         | /Users/siddiq90/
↪Documents/buildtest/tutorials/compilers/gnu_hello.yml
 hello_cplusplus | compiler-v1.0.schema.json | /Users/siddiq90/Documents/buildtest/
↪var/tests/local.bash/gnu_hello/hello_cplusplus/generate.sh | /Users/siddiq90/
↪Documents/buildtest/tutorials/compilers/gnu_hello.yml
 cc_example      | compiler-v1.0.schema.json | /Users/siddiq90/Documents/buildtest/
↪var/tests/local.bash/gnu_hello/cc_example/generate.sh      | /Users/siddiq90/
↪Documents/buildtest/tutorials/compilers/gnu_hello.yml
 fc_example      | compiler-v1.0.schema.json | /Users/siddiq90/Documents/buildtest/
↪var/tests/local.bash/gnu_hello/fc_example/generate.sh      | /Users/siddiq90/
↪Documents/buildtest/tutorials/compilers/gnu_hello.yml
 cxx_example     | compiler-v1.0.schema.json | /Users/siddiq90/Documents/buildtest/
↪var/tests/local.bash/gnu_hello/cxx_example/generate.sh     | /Users/siddiq90/
↪Documents/buildtest/tutorials/compilers/gnu_hello.yml

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

 name            | executor   | status   |   returncode | testpath
-----------------+------------+----------+--------------+-----------------------------
↪-------------------------------------------------------------------
 hello_f         | local.bash | PASS     |            0 | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/gnu_hello/hello_f/generate.sh
 hello_c         | local.bash | PASS     |            0 | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/gnu_hello/hello_c/generate.sh
 hello_cplusplus | local.bash | PASS     |            0 | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/gnu_hello/hello_cplusplus/generate.sh
 cc_example      | local.bash | PASS     |            0 | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/gnu_hello/cc_example/generate.sh
 fc_example      | local.bash | PASS     |            0 | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/gnu_hello/fc_example/generate.sh
 cxx_example     | local.bash | PASS     |            0 | /Users/siddiq90/Documents/
↪buildtest/var/tests/local.bash/gnu_hello/cxx_example/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
gfortran -Wall -o hello.f90.exe /global/u1/s/siddiq90/tutorials/examples/serial/src/
↪hello.f90
./hello.f90.exe
```

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]
    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:
```

(continues on next page)

(continued from previous page)

```
      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:
    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
-------------+-------------+----------+--------------+---------------------------------
↪------------------------------
 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
__________________________________________

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

(continues on next page)

(continued from previous page)

```
| 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]
    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
Paths:
___________
Test Directory: /Users/siddiq90/Documents/buildtest/var/tests

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

(continues on next page)

(continued from previous page)

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

/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       | Schema File               | Test Path                                 
↪                                          | Buildspec
------------+---------------------------+-------------------------------------------
↪-------------------------------------------+-----------------------------------------
↪----------------------------
 vecadd_gnu | compiler-v1.0.schema.json | /Users/siddiq90/Documents/buildtest/var/
↪tests/local.bash/vecadd/vecadd_gnu/generate.sh | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/vecadd.yml

+----------------------+
| 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/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
------------+------------+----------+--------------+---------------------------------
↪-------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
 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_codes: 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
srun ./vecAdd.c.exe
```

In this next example, we build same test using hpcsdk compiler by NVIDIA that 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 to illustrate this behavior:

```
version: "1.0"
buildspecs:
  executable_arguments:
    type: compiler
    description: example using pre_build, post_build, pre_run, post_run example
    executor: local.bash
    tags: [tutorials]
    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
```

(continues on next page)

(continued from previous page)

```
{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"
```

Compiler Schema Examples

The compiler schema examples can be retrieved via `buildtest schema -n compiler-v1.0.schema.json -e` which shows a list of valid/invalid buildspec examples using `type: compiler`. Each example is validated with schema `compiler-v1.0.schema.json` and error message from invalid examples are also shown in example output.

```
$ buildtest schema -n compiler-v1.0.schema.json -e



File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/compiler-v1.0.
↪schema.json/valid/examples.yml
Valid State: True
--------------------------------------------------------------------------------


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
```

(continues on next page)

(continued from previous page)

```
    cflags: "-O1"

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:
```

(continues on next page)

(continued from previous page)

```
    type: compiler
    description: Example by using cxx to set C++ compiler
    executor: local.bash
    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




File: /Users/siddiq90/Documents/buildtest/buildtest/schemas/examples/compiler-v1.0.
↪schema.json/invalid/examples.yml
Valid State: FAIL
_____________________________________________________________________________


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"
```

(continues on next page)

(continued from previous page)

```
    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'
↪"
    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:
```

(continues on next page)

(continued from previous page)

```
    - "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"


  test_additionalProperties_compiler:
    executor: local.bash
    type: compiler
    description: "test additionalProperties in compiler object. Schema does not allow␣
↪for additional keys"
    module:
    - "module purge && module load gcc/4.0"
    build:
      source: src/hello.c
      foo: bar
      name: gnu
      cflags: "-O1"
      ldflags: "-lm"

  test_additionalProperties_main_schema:
    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"

  type_mismatch_args:
    executor: local.bash
    type: compiler
```

(continues on next page)

(continued from previous page)

```
    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

_____________________________________ Validation Error _____________________________
↪_____________
Did not find 'type' key in test section: missing_type
```

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 (Experimental Feature)

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 a *Slurm Executors* and reference it via `executor`. 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
    executor: slurm.debug
    sbatch:
      - "-t 00:05"
      - "-C haswell"
      - "-N 1"
    run: |
      export SLURM_JOB_NAME="firstjob"
      echo "jobname:" $SLURM_JOB_NAME
```

(continues on next page)

(continued from previous page)

```
      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 --cluster 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 status as `N/A` for jobs submitted to scheduler, this is because we don't have result until we finish polling and gather results. 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  |
+-------------------------------+

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

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

(continues on next page)

(continued from previous page)

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

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

The **SlurmExecutor** class responsible for managing slurm job will retrieve the following format fields using `sacct`

during `gather` stage once job is finished:

- "Account"
- "AllocNodes"
- "AllocTRES"
- "ConsumedEnergyRaw"
- "CPUTimeRaw"
- "End"
- "ExitCode"
- "JobID"
- "JobName"
- "NCPUS"
- "NNodes"
- "QOS"
- "ReqGRES"
- "ReqMem"
- "ReqNodes"
- "ReqTRES"
- "Start"
- "State"
- "Submit"
- "UID"
- "User"
- "WorkDir"

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_codes` 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_codes: "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
---------------+--------------+-----------+--------------+-----------------------------
↪---------------------------
 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"

5.8 Introspection Operation

5.8.1 Config Options (`buildtest config --help`)

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

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

subcommands:
  buildtest configuration

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

The `buildtest config` command allows user to see view or edit your buildtest settings file (`settings.yml`). To see content of your buildtest settings run:

```
buildtest config view
```

Shown below is an example output.

```
$ buildtest config view
executors:
   local:
     bash:
```

(continues on next page)

(continued from previous page)

```
      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
config:
  editor: vi
  paths:
    buildspec_roots:
      - /Users/siddiq90/Documents/buildtest-cori
```

Likewise, you can edit the file by running:

```
buildtest config edit
```

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

```
$ buildtest config validate
/Users/siddiq90/.buildtest/config.yml is valid
```

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

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

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:  0.8.0
 buildtest Path: /Users/siddiq90/.local/share/virtualenvs/buildtest-1gHVG2Pd/bin/
↪buildtest
 Machine Details
 ____________________________________
 Operating System:  Darwin 19.5.0
 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']
 Buildtest Repositories:
 ____________________________________________________________________________
 Repo File: /Users/siddiq90/.buildtest/repo.yaml
 Active Repos: ['buildtesters/tutorials']
 Repo Paths: ['/private/tmp/github.com/buildtesters/tutorials']
 Buildspec Cache File: /Users/siddiq90/Documents/buildtest/var/buildspec.cache
 Number of buildspecs:  13
 Number of Tests: 27
 Tests:  ['systemd_default_target', '_bin_sh_shell', '_bin_bash_shell', 'bash_shell',
↪'sh_shell', 'shell_options', 'environment_variables', 'variables', 'selinux_disable
↪', 'exit1_fail', 'exit1_pass', 'returncode_mismatch', 'wrongexecutor', 'circle_area
↪', 'skip', 'unskipped', 'vecadd_gnu', 'hello_f', 'hello_c', 'hello_cplusplus',
↪'hello_f', 'hello_c', 'hello_cplusplus', 'cc_example', 'fc_example', 'cxx_example',
↪'executable_arguments']
 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/script-v1.0.schema.json
 Examples Directory for schema:  /Users/siddiq90/Documents/buildtest/buildtest/
↪schemas/script/examples
 compiler-v1.0.schema.json : /Users/siddiq90/Documents/buildtest/buildtest/schemas/
↪compiler/compiler-v1.0.schema.json
 Examples Directory for schema:  /Users/siddiq90/Documents/buildtest/buildtest/
↪schemas/compiler/examples
```

5.8.2 Access to 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 buildtest supports.

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

Shown below is the command usage of `buildtest schema`

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

optional arguments:
  -h, --help            show this help message and exit
  -n Schema Name, --name Schema Name
                        show schema by name (e.g., script)
  -e, --example         Show schema examples that are validated by corresponding␣
↪schemafile
  -j, --json            Display json schema file
```

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
```

5.8.3 Buildspec Features

```
$ 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
```

The `buildtest buildspec find` loads all buildspecs specified in *buildspec roots* from your configuration file. To build your cache just run:

```
$ buildtest buildspec find
```

To rebuild your cache, which you may need to do if you add more directories to `buildspec_roots` in your configuration or edit some buildspec run:

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

This will rebuild your cache and validate all buildspecs with updated files. Currently, we don't support automatic rebuild of cache.

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]

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
```

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

```
$ buildtest buildspec find --tags
Searching buildspec in following directories:  /Users/siddiq90/Documents/buildtest/
↪tutorials
+-----------+
| Tags      |
+===========+
| tutorials |
+-----------+
```

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

```
$ buildtest buildspec find --buildspec-files
Searching buildspec in following directories:  /Users/siddiq90/Documents/buildtest/
↪tutorials
+-------------------------------------------------------------------------------+
| buildspecs                                                                    |
+===============================================================================+
| /Users/siddiq90/Documents/buildtest/tutorials/systemd.yml                     |
+-------------------------------------------------------------------------------+
| /Users/siddiq90/Documents/buildtest/tutorials/shell_examples.yml              |
+-------------------------------------------------------------------------------+
| /Users/siddiq90/Documents/buildtest/tutorials/environment.yml                 |
+-------------------------------------------------------------------------------+
| /Users/siddiq90/Documents/buildtest/tutorials/vars.yml                        |
+-------------------------------------------------------------------------------+
| /Users/siddiq90/Documents/buildtest/tutorials/selinux.yml                     |
+-------------------------------------------------------------------------------+
| /Users/siddiq90/Documents/buildtest/tutorials/pass_returncode.yml             |
+-------------------------------------------------------------------------------+
| /Users/siddiq90/Documents/buildtest/tutorials/invalid_executor.yml            |
+-------------------------------------------------------------------------------+
| /Users/siddiq90/Documents/buildtest/tutorials/python-shell.yml                |
+-------------------------------------------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| /Users/siddiq90/Documents/buildtest/tutorials/skip_tests.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        |
+---------------------------------------------------------------------------------+
```

5.8.4 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]

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>,...)
```

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

```
$ buildtest report
+-----------------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| id                                            | state   |   returncode | starttime   ␣
↪       | endtime             |    runtime | tags      | buildspec                   ␣
↪                                                     |
+===============================================+=========+==============+=====================+=======
| systemd_default_target_2020-09-02-21-41       | FAIL    |            1 | 2020/09/02␣
↪21:41:56 | 2020/09/02 21:41:56 | 0.0213593  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/systemd.yml                      |
+-----------------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| systemd_default_target_2020-09-02-21-42       | FAIL    |            1 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.0125463  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/systemd.yml                      |
+-----------------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| systemd_default_target_2020-09-02-21-42       | FAIL    |            1 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00771781 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/systemd.yml                      |
+-----------------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| slurm_down_nodes_reason_2020-09-02-21-42      | PASS    |            0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.0110754  |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/slurm.yml        |
```

(continues on next page)

(continued from previous page)

```
+-----------------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| slurm_down_nodes_reason_2020-09-02-21-42    | PASS    |            0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.00830546 |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/slurm.yml          |
+-----------------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| slurm_not_responding_nodes_2020-09-02-21-42 | PASS    |            0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.00802073 |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/slurm.yml          |
+-----------------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| slurm_not_responding_nodes_2020-09-02-21-42 | PASS    |            0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.00736713 |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/slurm.yml          |
+-----------------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| _bin_sh_shell_2020-09-02-21-42              | PASS    |            0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.0125264  |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+-----------------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| _bin_sh_shell_2020-09-02-21-42              | PASS    |            0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.014642   |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+-----------------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| _bin_bash_shell_2020-09-02-21-42            | PASS    |            0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.00878803 |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+-----------------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| _bin_bash_shell_2020-09-02-21-42            | PASS    |            0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.00996381 |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+-----------------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| bash_shell_2020-09-02-21-42                 | PASS    |            0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.00561796 |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+-----------------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| bash_shell_2020-09-02-21-42                 | PASS    |            0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.00678369 |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+-----------------------------------------------+---------+--------------+--------------
↪-------+---------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| sh_shell_2020-09-02-21-42                    | PASS    |            0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.0102359  |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| sh_shell_2020-09-02-21-42                    | PASS    |            0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.0110021  |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| shell_options_2020-09-02-21-42               | PASS    |            0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.0104085  |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| shell_options_2020-09-02-21-42               | PASS    |            0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.0115846  |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/shell_examples.yml |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| circle_area_2020-09-02-21-42                 | FAIL    |            2 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.0096071  |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/python-shell.yml   |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| circle_area_2020-09-02-21-42                 | FAIL    |            2 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.0117158  |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/python-shell.yml   |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_dinosaur_2020-09-02-21-42              | PASS    |            0 | 2020/09/02␣
↪21:42:01 | 2020/09/02 21:42:01 | 0.00698703 |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/environment.yml    |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_dinosaur_2020-09-02-21-42              | PASS    |            0 | 2020/09/02␣
↪21:42:06 | 2020/09/02 21:42:06 | 0.0071777  |           | /Users/siddiq90/Documents/
↪buildtest/tests/examples/buildspecs/environment.yml    |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| variables_2020-09-02-21-42                   | PASS    |            0 | 2020/09/02␣
↪21:42:20 | 2020/09/02 21:42:20 | 0.00946478 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/vars.yml                           |
+----------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| pre_post_build_run_2020-09-02-21-42          | PASS    |            0 | 2020/09/02␣
↪21:42:20 | 2020/09/02 21:42:20 | 0.661319   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/pre_post_build_run.yml   |
```

(continues on next page)

(continued from previous page)

```
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| environment_variables_2020-09-02-21-42      | PASS    |           0 | 2020/09/02␣
↪21:42:20 | 2020/09/02 21:42:20 | 0.0120327  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/environment.yml                    |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_f_2020-09-02-21-42                    | PASS    |           0 | 2020/09/02␣
↪21:42:20 | 2020/09/02 21:42:20 | 0.00786752 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml             |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_f_2020-09-02-21-42                    | PASS    |           0 | 2020/09/02␣
↪21:42:43 | 2020/09/02 21:42:43 | 0.013958   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml             |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_c_2020-09-02-21-42                    | PASS    |           0 | 2020/09/02␣
↪21:42:20 | 2020/09/02 21:42:21 | 0.229417   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml             |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_c_2020-09-02-21-42                    | PASS    |           0 | 2020/09/02␣
↪21:42:43 | 2020/09/02 21:42:43 | 0.235872   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml             |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_cplusplus_2020-09-02-21-42            | PASS    |           0 | 2020/09/02␣
↪21:42:21 | 2020/09/02 21:42:21 | 0.461781   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml             |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| hello_cplusplus_2020-09-02-21-42            | PASS    |           0 | 2020/09/02␣
↪21:42:43 | 2020/09/02 21:42:44 | 0.498086   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml             |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| cc_example_2020-09-02-21-42                 | PASS    |           0 | 2020/09/02␣
↪21:42:21 | 2020/09/02 21:42:21 | 0.230038   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml             |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| cc_example_2020-09-02-21-42                 | PASS    |           0 | 2020/09/02␣
↪21:42:44 | 2020/09/02 21:42:44 | 0.239393   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml             |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| fc_example_2020-09-02-21-42                 | PASS    |            0 | 2020/09/02␣
↪21:42:21 | 2020/09/02 21:42:21 | 0.00840816 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml            |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| fc_example_2020-09-02-21-42                 | PASS    |            0 | 2020/09/02␣
↪21:42:44 | 2020/09/02 21:42:44 | 0.0103057  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml            |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| cxx_example_2020-09-02-21-42                | PASS    |            0 | 2020/09/02␣
↪21:42:21 | 2020/09/02 21:42:22 | 0.45725    | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml            |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| cxx_example_2020-09-02-21-42                | PASS    |            0 | 2020/09/02␣
↪21:42:44 | 2020/09/02 21:42:44 | 0.505495   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/gnu_hello.yml            |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| selinux_disable_2020-09-02-21-42            | FAIL    |            0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00954966 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/selinux.yml                        |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| _bin_sh_shell_2020-09-02-21-42              | PASS    |            0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.0126632  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/shell_examples.yml                 |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| _bin_bash_shell_2020-09-02-21-42            | PASS    |            0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.0085432  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/shell_examples.yml                 |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| bash_shell_2020-09-02-21-42                 | PASS    |            0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00603703 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/shell_examples.yml                 |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| sh_shell_2020-09-02-21-42                   | PASS    |            0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00911067 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/shell_examples.yml                 |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| shell_options_2020-09-02-21-42              | PASS    |            0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.0104522  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/shell_examples.yml                 |
```

(continues on next page)

(continued from previous page)

```
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| circle_area_2020-09-02-21-42                | FAIL    |           2 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00808543 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/python-shell.yml                 |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| unskipped_2020-09-02-21-42                  | PASS    |           0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00792155 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/skip_tests.yml                   |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| unskipped_2020-09-02-21-42                  | PASS    |           0 | 2020/09/02␣
↪21:42:49 | 2020/09/02 21:42:49 | 0.042425   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/skip_tests.yml                   |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| executable_arguments_2020-09-02-21-42       | PASS    |           0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.232554   | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/passing_args.yml       |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| exit1_fail_2020-09-02-21-42                 | FAIL    |           1 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00782586 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/pass_returncode.yml              |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| exit1_fail_2020-09-02-21-42                 | FAIL    |           1 | 2020/09/02␣
↪21:42:48 | 2020/09/02 21:42:48 | 0.0132099  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/pass_returncode.yml              |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| exit1_pass_2020-09-02-21-42                 | PASS    |           1 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00772625 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/pass_returncode.yml              |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| exit1_pass_2020-09-02-21-42                 | PASS    |           1 | 2020/09/02␣
↪21:42:48 | 2020/09/02 21:42:48 | 0.0110667  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/pass_returncode.yml              |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| returncode_mismatch_2020-09-02-21-42        | FAIL    |           2 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.00836054 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/pass_returncode.yml              |
+-----------------------------------------+---------+-------------+-------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| returncode_mismatch_2020-09-02-21-42        | FAIL    |            2 | 2020/09/02␣
↪21:42:48 | 2020/09/02 21:42:48 | 0.00940911 | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/pass_returncode.yml               |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| vecadd_gnu_2020-09-02-21-42                 | FAIL    |            0 | 2020/09/02␣
↪21:42:22 | 2020/09/02 21:42:22 | 0.0267169  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/vecadd.yml              |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
| vecadd_gnu_2020-09-02-21-42                 | FAIL    |            0 | 2020/09/02␣
↪21:42:46 | 2020/09/02 21:42:46 | 0.0421426  | tutorials | /Users/siddiq90/Documents/
↪buildtest/tutorials/compilers/vecadd.yml              |
+---------------------------------------------+---------+--------------+--------------
↪-------+--------------------+------------+-----------+-----------------------------
↪-----------------------------------------------------+
```

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

```
$ buildtest report --format name,schemafile,executor,state,returncode
+----------------------------+-------------------------+--------------+---------+-----
↪---------+
 | name                       | schemafile              | executor     | state   |  ␣
↪returncode |
␣
↪+============================+=========================+==============+=========+==============+
 | systemd_default_target     | script-v1.0.schema.json | local.bash   | FAIL    |   ␣
↪        1 |
 +----------------------------+-------------------------+--------------+---------+----
↪----------+
```

(continues on next page)

(continued from previous page)

```
| systemd_default_target     | script-v1.0.schema.json | local.bash   | FAIL    |
↪        1 |
+---------------------------+-------------------------+--------------+---------+----
↪----------+
| systemd_default_target     | script-v1.0.schema.json | local.bash   | FAIL    |
↪        1 |
+---------------------------+-------------------------+--------------+---------+----
↪----------+
| hello_dinosaur             | script-v1.0.schema.json | local.bash   | FAIL    |
↪      127 |
+---------------------------+-------------------------+--------------+---------+----
↪----------+
| hello_dinosaur             | script-v1.0.schema.json | local.bash   | PASS    |
↪        0 |
+---------------------------+-------------------------+--------------+---------+----
↪----------+
| circle_area                | script-v1.0.schema.json | local.python | PASS    |
↪        0 |
+---------------------------+-------------------------+--------------+---------+----
↪----------+
| circle_area                | script-v1.0.schema.json | local.python | PASS    |
↪        0 |
+---------------------------+-------------------------+--------------+---------+----
↪----------+
| _bin_sh_shell              | script-v1.0.schema.json | local.sh     | PASS    |
↪        0 |
+---------------------------+-------------------------+--------------+---------+----
↪----------+
| _bin_sh_shell              | script-v1.0.schema.json | local.sh     | PASS    |
↪        0 |
+---------------------------+-------------------------+--------------+---------+----
↪----------+
| _bin_bash_shell            | script-v1.0.schema.json | local.bash   | FAIL    |
↪      127 |
+---------------------------+-------------------------+--------------+---------+----
↪----------+
| _bin_bash_shell            | script-v1.0.schema.json | local.bash   | PASS    |
↪        0 |
+---------------------------+-------------------------+--------------+---------+----
↪----------+
| bash_shell                 | script-v1.0.schema.json | local.bash   | FAIL    |
↪      127 |
+---------------------------+-------------------------+--------------+---------+----
↪----------+
| bash_shell                 | script-v1.0.schema.json | local.bash   | PASS    |
↪        0 |
+---------------------------+-------------------------+--------------+---------+----
↪----------+
| sh_shell                   | script-v1.0.schema.json | local.sh     | PASS    |
↪        0 |
+---------------------------+-------------------------+--------------+---------+----
↪----------+
| sh_shell                   | script-v1.0.schema.json | local.sh     | PASS    |
↪        0 |
+---------------------------+-------------------------+--------------+---------+----
↪----------+
| shell_options              | script-v1.0.schema.json | local.sh     | PASS    |
↪        0 |
```

(continues on next page)

(continued from previous page)

```
+---------------------------+------------------------+-------------+---------+----
↪----------+
| shell_options             | script-v1.0.schema.json | local.sh     | PASS    |  ␣
↪        0 |
+---------------------------+------------------------+-------------+---------+----
↪----------+
| slurm_down_nodes_reason   | script-v1.0.schema.json | local.bash   | FAIL    |  ␣
↪      127 |
+---------------------------+------------------------+-------------+---------+----
↪----------+
| slurm_down_nodes_reason   | script-v1.0.schema.json | local.bash   | PASS    |  ␣
↪        0 |
+---------------------------+------------------------+-------------+---------+----
↪----------+
| slurm_not_responding_nodes | script-v1.0.schema.json | local.bash   | FAIL    |  ␣
↪      127 |
+---------------------------+------------------------+-------------+---------+----
↪----------+
| slurm_not_responding_nodes | script-v1.0.schema.json | local.bash   | PASS    |  ␣
↪        0 |
+---------------------------+------------------------+-------------+---------+----
↪----------+
| selinux_disable           | script-v1.0.schema.json | local.bash   | PASS    |  ␣
↪        0 |
+---------------------------+------------------------+-------------+---------+----
↪----------+
```

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
- Share your tests
- Provide feedback on buildtest options.
    1. What features you *like*/*dislike*
    2. What features you would like to have
    3. What testing capabilities matter most for you

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 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.

5.10.3 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)

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. Please wait for buildtest maintainers to review your PR and provide feedback. If the reviewer requests some changes, then you are requested to make changes and update the branch used for sending PR

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 your PR to be out of date. The maintainers will check for this during PR review or GitHub will report file conflicts during merge.

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.

Anyhow, 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.

It's good practice to check Travis builds since we use Travis to run regression test and Codecov and Coveralls depend on Travis to pass all checks.

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.

- Travis CI - Test and deploy with confidence. Trusted by over 800,000 users, Travis CI is the leading hosted continuous integration system.
- 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.

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 `$BUILDTEST_ROOT/buildtest/docgen/main.py` to automate documentation examples. This script can be run as follows:

```
cd $BUILDTEST_ROOT
python $BUILDTEST_ROOT/buildtest/docgen/main.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
```

For a complete list of options refer to pytest documentation or run `pytest --help`.

You may want to run coverage 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%
```

(continues on next page)

(continued from previous page)

```
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`.

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.

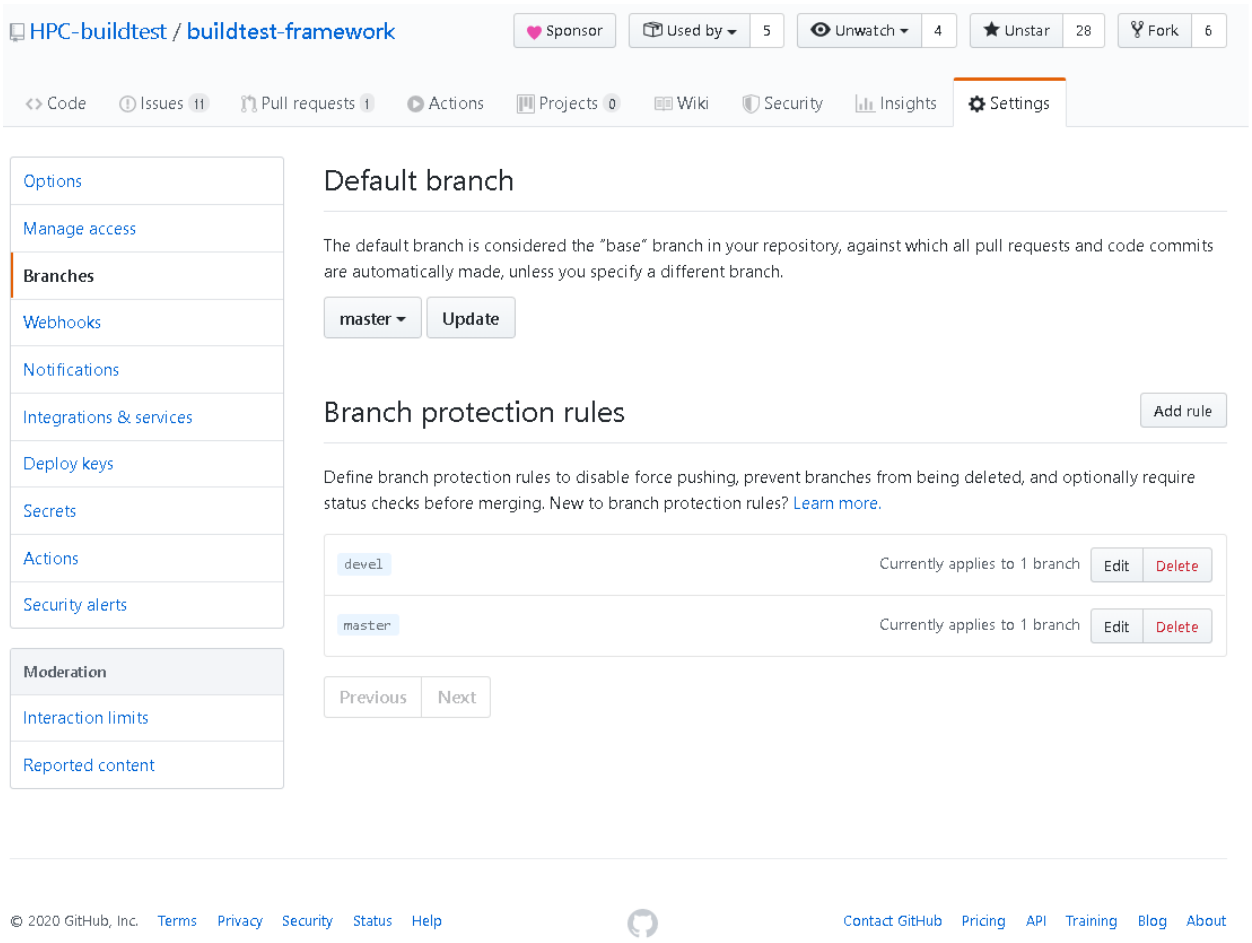
Default Branch

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.

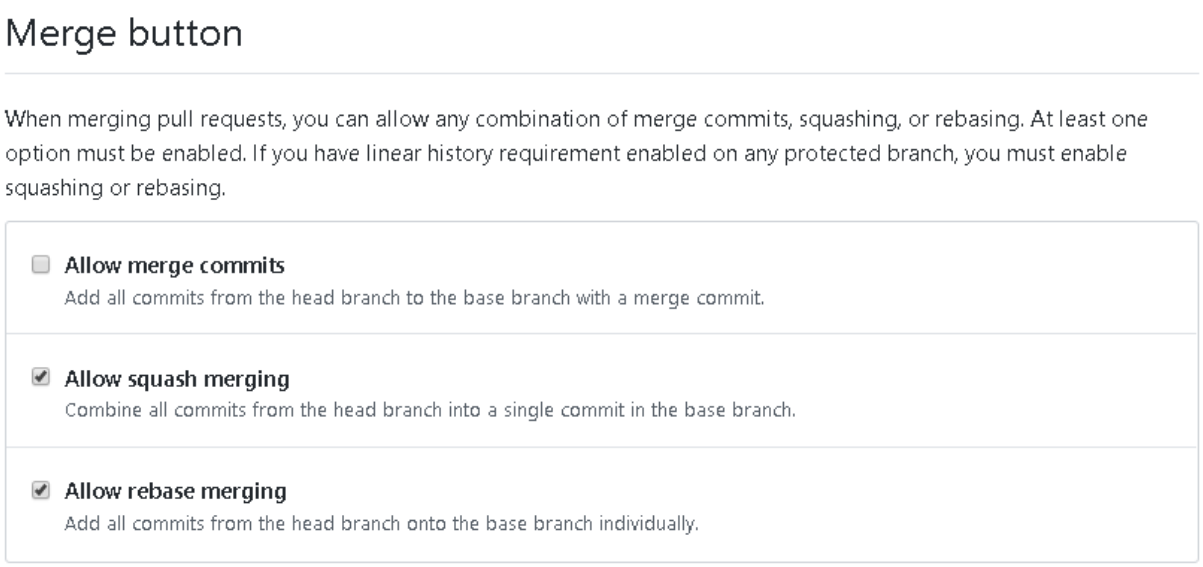
The master and devel branches should be protected branches and master should be enabled as default branch. Shown below is the expected configuration.



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].

[1] Created with sphinx-autoapi

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'

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.

_generate_build_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()`.

generate_script(*self*)
Build the testscript content implemented in each subclass

get_bsub(*self*)

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_sbatch(*self*)

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: python –> py 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

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.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. If a version is provided, honor it. If not, use latest. We also don't allow repeated keys in the same file.

_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*)

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:

Parameters **testdir** (*str, optional*) – Test Destination directory, specified by –testdir

keys(*self*)

Return the list of keys for the loaded Buildspec recipe, not including the metadata keys defined for any global recipe.

buildtest.docgen

Submodules

buildtest.docgen.main

This file is used for generating documentation tests.

Module Contents

Functions

build_helper()	This method will write output of several helper options for all sub-commands in buildtest
generate_tests(prefix, cmd_dict)	
introspection_cmds()	
main()	
run(query)	The `run()` method will execute the command and retrieve the output as part of documentation examples
schemas()	
tutorial()	
writer(fname, out, query)	

buildtest.docgen.main.**build_helper**()
: This method will write output of several helper options for all sub-commands in buildtest

buildtest.docgen.main.**docgen**

buildtest.docgen.main.**generate_tests**(*prefix*, *cmd_dict*)

buildtest.docgen.main.**introspection_cmds**()

buildtest.docgen.main.**main**()

buildtest.docgen.main.**root**

buildtest.docgen.main.**run**(*query*)
: The `run()` method will execute the command and retrieve the output as part of documentation examples

buildtest.docgen.main.**schemas**()

buildtest.docgen.main.**tutorial**()

buildtest.docgen.main.**writer**(*fname*, *out*, *query*)

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
SSHExecutor(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:

Parameters **regex** (*str, required*) – 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.

get_formatted_time(*self*, *key*, *fmt='%Y/%m/%d %X'*)

Given some timestamp key in self.metadata, return a pretty printed version of it. This is intended to log in the console for the user.

Parameters:

key: The key to look up in the metadata fmt: the format string to use

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 :param out: content of output stream :type out: list :param err: content of error stream :type err: list

class buildtest.executors.base.**SSHExecutor**(*name*, *settings*, *config_opts*)
Bases: *buildtest.executors.base.BaseExecutor*

The BaseExecutor is an abstract base class for all executors. All executors must have a listing of steps and dryrun_steps

type = ssh

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 BaseExecutor is an abstract base class for all executors. All

class buildtest.executors.local.**LocalExecutor**(*name*, *settings*, *config_opts*)
Bases: *buildtest.executors.base.BaseExecutor*

The BaseExecutor is an abstract base class for all executors. All executors must have a listing of steps and dryrun_steps

type = local

check(*self*)

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 is written to filesystem. After test

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

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 slurm 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, defined under

class buildtest.executors.setup.**BuildExecutor**(*config_opts*)

A BuildExecutor is a base class some type of executor, defined under the buildtest/settings/default-config.json schema. For example, the types "local" and "slurm" would map to *LocalExecutor* 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:

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*)

run(*self*, *builder*)

Given a buildtest.buildsystem.BuildspecParser (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` defined in executor. 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

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

discover_buildspecs(buildspec)	Given a buildspec file specified by the user with `buildtest build --buildspec`,
discover_buildspecs_by_tags(input_tag)	This method discovers buildspecs by tags, using `--tags` option
func_build_subcmd(args, config_opts)	Entry point for `buildtest build` sub-command. This method will discover

buildtest.menu.build.**discover_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 parse. Examples of intended functionality are documented here. For all of the below, test config root refers to $HOME/.buildtest/site

A relative path to a file in the PWD (outside of test config root, returns single) buildtest build –buildspec relative-folder/hello.sh.yml

A relative path to a file in build test root (returns single) buildtest build –buildspec github.com/buildtesters/tutorials/hello-world/hello.sh.yml

relative directory path (returns multiple) buildtest build –buildspec hello-world

relative directory path in build test root (returns multiple) buildtest build –buildspec github.com/buildtesters/tutorials/hello-world/

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.**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:

Parameters **args** (*dict, required*) – arguments passed from command line

Return type None

buildtest.menu.build.**logger**

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)	
get_buildspecfiles(cache)	
rebuild_buildspec_cache(paths)	

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

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: :param buildspec: buildspec file section to view or edit. :param view: boolean to determine if we want to view buildspec file :param edit: boolean to control if we want to edit buildspec file in editor. :return:

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*)

buildtest.menu.buildspec.**get_buildspecfiles**(*cache*)

buildtest.menu.buildspec.**rebuild_buildspec_cache**(*paths*)

buildtest.menu.config

Module Contents

Functions

func_config_edit(args=None)	Edit buildtest configuration in editor. This implements `buildtest config edit`
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`

buildtest.menu.config.**func_config_edit**(*args=None*)
Edit buildtest configuration in editor. This implements `buildtest config edit`

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.report

Module Contents

Functions

func_report(args=None)	
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.**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

Parameters **valid_builders** – 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

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_edit(args=None)	Edit buildtest configuration in editor. This implements `buildtest config edit`

continues on next page

Table 16 – continued from previous page

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_report(args=None)	
func_schema(args)	This method implements command `buildtest schema` which shows a list

buildtest.menu.**BUILDTEST_VERSION = 0.8.0**

class buildtest.menu.**BuildTestParser**

build_menu(*self*)

This method implements the `buildtest build` command

single buildspec file

```
buildtest build -b <file>
```

single buildspec directory (builds all buildspec in directory)

```
buildtest build -b <dir>
```

build a buildspec and exclude some buildspecs. The exclude (-x) accepts both file and directory

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

multiple buildspecs build and exclude

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

buildspec_menu(*self*)

This method implements `buildtest buildspec` command

Command Usage

find all buildspecs

```
buildtest buildspec find
```

view a buildspec file

```
buildtest buildspec view <name>
```

edit and validate a buildspec file

```
buildtest buildspec edit <name>
```

config_menu(*self*)

This method adds argparse argument for `buildtest config`

Command Usage

view buildtest settings file

```
buildtest config view
```

open buildtest settings in editor and validate upon saving

```
buildtest config edit
```

validate buildtest settings

```
buildtest config validate

# summary of buildtest

buildtest config summary
```

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`

Command Usage

```
# list all schema names

-buildtest schema

# list all schema in json

-buildtest schema -j

# list schema global.schema.json in json

-buildtest schema -n global.schema.json -j

# list schema examples for global.schema.json

-buildtest schema -n global.schema.json -e
```

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

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_edit**(*args=None*)
Edit buildtest configuration in editor. This implements `buildtest config edit`

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_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.**supported_schemas**

buildtest.schemas

Submodules

buildtest.schemas.defaults

Module Contents

buildtest.schemas.defaults.**compiler_schema**

buildtest.schemas.defaults.**compiler_schema_file = compiler-v1.0.schema.json**

buildtest.schemas.defaults.**global_schema**

buildtest.schemas.defaults.**global_schema_file = global.schema.json**

buildtest.schemas.defaults.**here**

buildtest.schemas.defaults.**schema_table**

buildtest.schemas.defaults.**script_schema**

buildtest.schemas.defaults.**script_schema_file = script-v1.0.schema.json**

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 (expected to be under schemas
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 (expected to be under schemas

Parameters:

schema_file: the path to the schema file. name: 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: 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: the path to the schema file.

buildtest.settings

buildtest.utils

Submodules

buildtest.utils.command

Module Contents

Classes

BuildTestCommand(cmd=None)	Class method to invoke shell commands and retrieve output and error.

continues on next page

Table 18 – continued from previous page

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)	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*)

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, required*) – 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)	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.
validate_lsf_executors(lsf_executors)	
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*)
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:

Parameters **settings_path** (*str, optional*) – Path to buildtest settings file

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:

Parameters **settings_path** (*str, optional*) – Path to buildtest settings file

buildtest.config.**logger**

buildtest.config.**validate_lsf_executors**(*lsf_executors*)

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 if `partition`, `qos` or `cluster` field is specified in slurm executor section.

Parameters **slurm_executor** – list of slurm executors defined in settings['executors']['slurm'] dictionary, where settings is loaded buildtest setting

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.**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.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.

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_lmod(*self*)
Check if the system has Lmod installed, determine by setting of LMOD_DIR variable

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" set the linux distrubution name and version

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 = 0.8.0**

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

G

H

I

J

K

L

M

O

P

R

S

T

U

V

W