How to use EasyBuild

EASYBUILD.io

building software with ease

2024-10-10

https://easybuild.io/

History

- The EasyBuild project was started in 2009 by the HPC team of Ghent University (Belgium).
- First public release, v0.5, in April 2012 with v1.0 in November 2012.
- Provides automated and reproducible software builds with differentiation between software, version, compiler etc.
- Integrated with the Lmod advanced modules system.
- Built around principle of open collaboration between HPC sites.

Why do we need EasyBuild

1. Even on systems where a single version of an application is required dependency issues arise with builds.
2. Quantity or chain of dependencies can be overwhelming, conflicting or circular dependencies etc.
3. Need exports of library paths and/or use of an environment modules system and/or build shell scripts.
4. Several build systems (make, GNU Autoconf, CMake, scons, binaries etc).
5. Many people who are building software (e.g., in a HPC space) are spending a lot of time on this!

Feature highlights

1. Fully autonomously building and installing (scientific) software
2. Automatic dependency resolution
3. Automatic generation of module files (Tcl or Lua syntax)
4. Thorough logging of executed build/install procedure
5. Archiving of build specifications (*easyconfig files*)
6. Highly configurable, via config files/environment/command line
7. Dynamically extendable with additional easyblocks, toolchains, etc.
8. Support for custom module naming schemes (incl. hierarchical)
9. Comprehensively tested: lots of unit tests, regression testing, . . .
10. Actively developed, collaboration between various HPC sites
11. Worldwide community

Overview

- **EasyBuild framework**
  - Core of EasyBuild
  - Provides supporting functionality for building and installing software
- **Easyblock**
  - A Python module
  - implements a (generic) software build/install procedure
- **Easyconfig file**
  - Build specification: software name/version, compiler toolchain, etc.
- **Compiler toolchain**
  - Compilers with accompanying libraries (MPI, BLAS/LAPACK, etc.)

LMod Module commands

Command	Abbreviation	Description
module load *[s]/[v]*	ml *[s]/[v]*	Loads software/version
module avail *[s]/[v]*	ml av *[s]/[v]*	List available software
module show *[s]/[v]*	ml show *[s]/[v]*	Show info about software
module list	ml	List currently loaded software
ml spider *[s]*		searches for software

[s]: Software. Optional for *avail* command

[v]: Version. Optional. Latest by default

What modules are available

For global installations

```
module load PDC easybuild-prod
```

- Builds into */pdc/software/23.12/eb/*

For local installations

Good to evaluate prior of global installation and testing purposes

```
module load PDC easybuild-user
```

- Builds into *~/.local/easybuild/*

How to get some help

```
eb --help
```

How are EasyBuild modules configured

```
eb --show-config
```

- Temporary files stored in */tmp*
- EasyBuild searches for available installations in...
    - PDC software stack
    - LUMI software stack
    - CSCS software stack
- EasyBuild searches for **easyblocks** in LUMI easyblocks
- EasyBuild includes LUMI, PDC toolchains

`Current EasyBuild configuration can be looked up in the module and changed if needed`

How to change the configuration

C: command line argument, **D**: default value, **E**: environment variable, **F**: configuration file

```
buildpath           (E) = /tmp/hzazzi
containerpath       (D) = /cfs/klemming/home/h/hzazzi/.local/easybuild/containers
include-easyblocks  (E) = /pdc/software/eb_repo/LUMI-SoftwareStack/easybuild/easyblocks/*/*.py
include-toolchains  (E) = /pdc/software/eb_repo/LUMI-SoftwareStack/easybuild/toolchains/*.py
...
parallel            (E) = 32
```

- To change environment variable

```
EASYBUILD_<NAME OF VARIABLE IN CAPITAL LETTERS>="<VALUE>"
```

- To change command line argument

```
eb --<NAME OF VARIABLE>="<VALUE>" ...
```

External repos and sources

EasyBuild recipes: https://github.com/easybuilders/easybuild-easyconfigs
PDC-software stack: https://github.com/PDC-support/PDC-SoftwareStack
LUMI software stack: https://github.com/Lumi-supercomputer/LUMI-SoftwareStack

Local repos

```
$ ls -l /pdc/software/eb_repo/
drwxrwxr-x+ 10 hzazzi sinstall 4096 Oct 11 12:24 CSCS-production
drwxrwxr-x+  4 hzazzi sinstall 4096 Oct 11 12:26 LUMI-EasyBuild-contrib
drwxrwxr-x+ 11 hzazzi sinstall 4096 Oct 11 12:24 LUMI-SoftwareStack
drwxrwxr-x+  6 hzazzi hzazzi   4096 Oct 21 14:30 PDC-SoftwareStack
drwxrwxr-x+  2 hzazzi hzazzi   4096 Oct 27 09:21 sources
```

How to install software using EasyBuild

Files ending with **eb** are called **easyconfig** files

```
eb <software>-<version>-<toolchain>-<version>.eb
```

dry-run

```
eb Boost-1.75.0-cpeGNU-23.12.eb -dr,--dry-run
```

- Test the installation procedure without installing it
- you can also use *-x,--extended-dry-run* for more information

How to search for software using EasyBuild

Lists available *easyconfig* files

```
eb -S,--search <software>
CFGS1=/pdc/software/eb_repo/PDC-SoftwareStack/easybuild/easyconfigs/g
CFGS2=/pdc/software/eb_repo/LUMI-SoftwareStack/easybuild/easyconfigs/g/GROMACS
CFGS3=/pdc/software/eb_repo/CSCS-production/easybuild/easyconfigs/g/GROMACS
 * $CFGS1/GROMACS-2020.5-cpeCray-23.12.eb
 * $CFGS1/GROMACS-2021.3-cpeCray-23.12.eb
 * $CFGS1/GROMACS-2022-beta1-cpeCray-23.12.eb
```

These can be used to make new easyconfig recipes

```
eb --copy <NAME of easyconfig>
```

How install dependent software

Automatically installs dependency software using easyconfigs that are available in robot paths

```
eb Boost-1.75.0-cpeGNU-23.12.eb -r,--robot
```

To check what dependencies are missing

```
eb Boost-1.75.0-cpeGNU-23.12.eb -M,--missing
```

Easyconfigs

```
easyblock = 'ConfigureMake'

name = 'blast+'
version = '2.15.0'

homepage = 'https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download'
description = """Blast for searching sequences"""

toolchain = {'name': 'cpeGNU', 'version': '23.12'}
toolchainopts = {'usempi': True}
source_urls = ['https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/%(version)s/']
sources = ['ncbi-blast-%(version)s+-src.tar.gz']
dependencies = [
    ('glib', '2.78.0','',SYSTEM),
    ('sqlite', '3.36.0','',SYSTEM),
]
sanity_check_paths = {
    'files': ['bin/blastn', 'bin/blastp', 'bin/blastx'],
    'dirs': []
}
moduleclass = 'bio'
```

Example of an easyconfig file for installing *Blast+*

How to build easyconfig files

What is needed in an easyconfig file

- Name
- Toolchain
- Sources
- Easyblock
- Dependencies
- Sanity_check

Writing easyconfig files
https://docs.easybuild.io/en/latest/Writing_easyconfig_files.html

Parameters and templates in easyconfig files

A full overview of all known easyconfig parameter

```
eb -a,--avail-easyconfig-params
```

A set of variables that can be used in easyconfig files

```
eb --avail-easyconfig-templates
```

Name

- Specifies the name and version of the software
- module will be named accordingly

```
name = 'Blast+'
version = '2.12.0'
homepage = 'https://blast.ncbi.nlm.nih.gov/'
description = """Blast for searching sequences"""
```

Toolchain

If you want to use MPI, OpenMP ...

```
toolchain = {'name': 'cpeGNU', 'version': '23.12'}
```

`Will also have an impact on the dependencies for this easyconfig`

If you want to use supporting tools, libraries...

```
toolchain = SYSTEM
```

Sources

Specify where you can download your source

```
sources = [{
    'source_urls': ['https://example.com'],
    'filename': '%(name)s-%(version)s.tar.gz',
    'extract_cmd': "tar xf %s",  # Optional
}]
```

Easyblock

- A python code to address special needs of the installation
- Adresses that you should first run *configure* > *make* > *make install* or *cmake > make > make install

```
easyblock = 'type'
```

- Many EasyBlock are generic as to describe standard installation patterns
- Easyconfigs without an easyblock entry are special and Easybuild will search for EasyBlocks named **EB_[software]**

To find which Easyblock is specially for you...

```
eb --list-easyblocks
```

See information about parameters for easyblocks

https://docs.easybuild.io/en/latest/version-specific/generic_easyblocks.html

Examples of useful easyblocks

- **ConfigureMake**: implements the standard ./configure, make, make install installation procedure;
- **CMakeMake**: same as ConfigureMake, but with ./configure replaced with cmake for the configuration step;
- **PythonPackage**: implements the installation procedure for a single Python package, by default using "python setup.py install" but other methods like using "pip install" are also supported;
- **Bundle**: a simple generic easyblock to bundle a set of software packages together in a single installation directory;
- **PythonBundle**: a customized version of the Bundle generic easyblock to install a bundle of Python packages in a single installation directory;

Dependencies

- Use **dependencies** or **builddependencies** if the dependencies are only needed during build of the software, or also, when running software.
- Can be installed if found or loaded if a module exists

Main application toolchain

```
dependencies = [
    ('Software', 'version'),
]
```

System toolchain

```
dependencies = [
    ('Software', 'version', '', ('system', '')),
]
```

Special parameters to include in easyconfig

Commands to run before configuration/build

preconfigopts or prebuildopts

```
preconfigopts = "<command> && "
```

Extra parameters for configuration/build

configopts or buildopts

```
buildopts = ' -<PARAM>=True '
```

Add parameter to the module

```
modextravars={'<PARAMETER>': '<PATH>'}
```

Sanity check

- A test to see everything was installed correctly
- **MUST** be present

```
sanity_check_paths = {
    'files': ['bin/reframe',
              'share/completions/file1',
              'share/completions/file2'],
    'dirs': ['bin', 'lib', 'share', 'tutorials']
}

sanity_check_commands = [
    'software --version',
    'software --help',
]
```

Summary

- EasyBuild is a tool for building software with ease
- EasyBuild automatically installs software, its dependencies, and their modules in the correct place
- EasyBuild is very configurable
- Easyconfig are easy to create and you can use other easyconfigs as templates
- More information about how to use EasyBuild can be obtained from https://docs.easybuild.io/