EasyBuild: getting scientific software installed

a gently introduction for Intel SAT engineers

Kenneth Hoste
HPC-UGent, Ghent University, Belgium
kenneth.hoste@ugent.be

`http://users.ugent.be/~kehoste/EasyBuild-intro-Intel_20151204.pdf`

December 4th 2015

HPC-UGent in a nutshell

`http://www.ugent.be/hpc` – `http://www.vscentrum.be`

- HPC team at central IT dept. of Ghent University (Belgium)
- 9 team members: 1 manager, ∼3 user support, ∼5 sysadmin
- 4 Tier2 clusters + one Tier1 (8.5k cores), >1k servers in total
- ∼1.5k user accounts, across all scientific domains
- tasks: hardware, system administration, user support/training, ...

- member of Flemish Supercomputer Centre (VSC)
  virtual centre, collaboration between Flemish university associations

Tasks for HPC user support teams

- resolving problems that occur when using the HPC system(s)
  - *"I lost my private key/password, and now I can't log in. Help?"*
  - *"My job crashed, and I have no idea why. What happened?"*
  - *"My stuff doesn't work anymore, and I didn't change a thing!"*
- answering questions, from simple to very technical
- **installing (scientific) software tools/libraries/applications**
- helping users improve their workflow (not necessarily by request)
- training: Linux basics, OpenMP, MPI, Python, etc.
- ~~performance analysis and optimisation of large scientific applications~~
- ~~consultancy services w.r.t. developing scientific software~~

Installing scientific software for users

Typical way in which scientific software is installed for users:

- by user request: new software, version upgrades, more variants, ...
- on a (shared NFS) filesystem available on every workernode
- specifically targetted to the HPC cluster it will be used on
  - built from source (if possible)
  - separate installation per cluster
  - highly optimized for system architecture
    - linked with heavily tuned libraries (MPI, BLAS, LAPACK, ...)
    - built with (equivalent of) `-march=native`/`-xHost`
- rebuilt when updates for compilers/libraries become available
- installations remain available during lifetime of system
- accompanying *environment module* is provided for easy access

Environment modules

- canonical way of giving users access to installed (scientific) software
- used on most HPC systems ($>$ 80%[1]), since mid 90's
- module file specifies changes to user environment (in Tcl/Lua subset)
- modules tool applies those changes to the current session (!)
- easy interface for users:
  - available software: '`module avail [software name]`'
  - prepare environment for using software: '`module load <name>`'
  - show loaded modules: '`module list`'
  - rollback changes to environment: '`module unload <name>`'
  - start afresh: '`module purge`'
- Tcl-based environment modules system is most prevalent (for now)
- Lmod: Lua-based modules tool, *vastly* improves user experience

(1) `http://hpcugent.github.io/easybuild/files/SC15_BoF_Getting_Scientific_Software_Installed.pdf`

The magic behind modules

- '`module`' command is a *shell function* (see '`type -f module`')
- actual processing of module file(s) is done in a subprocess
  - via '`modulecmd`', '`modulecmd.tcl`' or '`lmod`' command
- changes to environment are spit out in shell syntax to `stdout`
- output messages are sent to `stderr`
- `stdout` is `eval`'ed by `module` function
  - so the *current* environment is changed (similar as with '`source`'
- `stderr` passes through untouched

Example (magic behind '`module load GCC/4.9.3`' when using Lmod):

```
$ lmod bash load GCC/4.9.3
...
PATH="/opt/easybuild/software/GCC/4.9.3/bin:/usr/bin:...";
export PATH;
...
```

"Please install <software> on the HPC?"

The most common type of support request from users is to install (scientific) software; this covers over 25% of support tickets at HPC-UGent.

Installing (lots of) *scientific* software is:

- error-prone, trial-and-error
- tedious, hard to get right
- repetitive & boring (well...)
- time-consuming (hours, days, even weeks)
- frustrating (e.g., dependency hell)
- sometimes simply not worth the effort...

Common issues with scientific software

Researchers focus on the *science* behind the software they implement, and care little about tools, build procedure, portability, ...

Scientists are not software developers or sysadmins (nor should they be).

"If we would know what we are doing, it wouldn't be called 'research'."

This results in:

- use of non-standard build tools (or broken ones)
- incomplete build procedure, e.g., no configure or install step
- interactive installation scripts
- hardcoded parameters (compilers, libraries, paths, ...)
- poor/outdated/missing/incorrect documentation
- dependency (version) hell

Prime example I: WRF

Weather Research and Forecasting Model (`http://www.wrf-model.org`)
(one of the top 5 applications on Blue Waters)

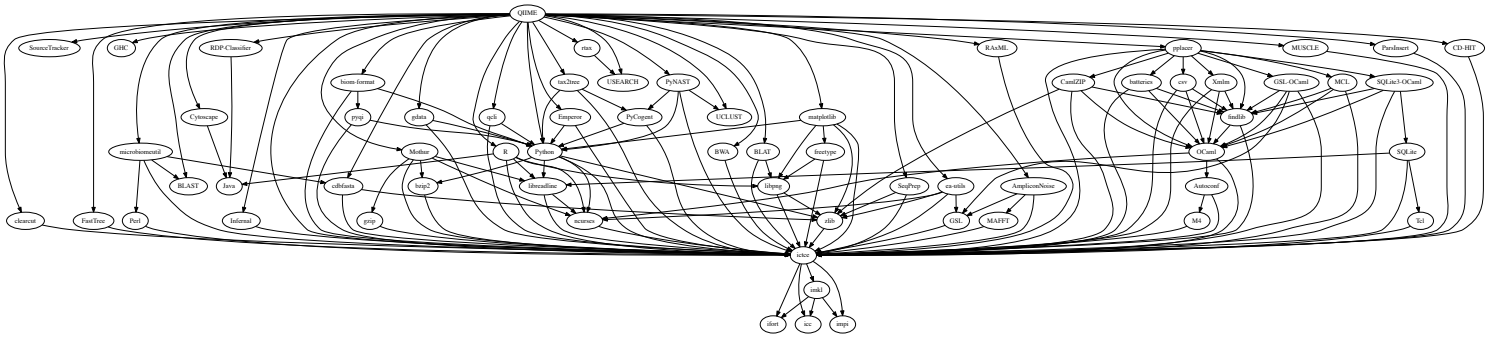
- dozen dependencies: netCDF (C, Fortran), HDF5, tcsh, JasPer, ...
- known issues in last release are (only) documented on website
  no patch file provided, infrequent bugfix releases
- interactive '`configure`' script :(
- resulting `configure.wrf` needs work:
  fix hardcoded settings (compilers, libraries, ...), tweaking of options
- custom '`compile`' script (wraps around '`make`')
  building in parallel is broken without fixing the `Makefile`
- no actual installation step

Wouldn't it be nice to build & install WRF with a single command?

`http://easybuild.readthedocs.org/en/latest/Typical_workflow_example_with_WRF.html`

Prime example II: QIIME

QIIME: Quantitative Insights Into Microbial Ecology (`http://qiime.org/`)



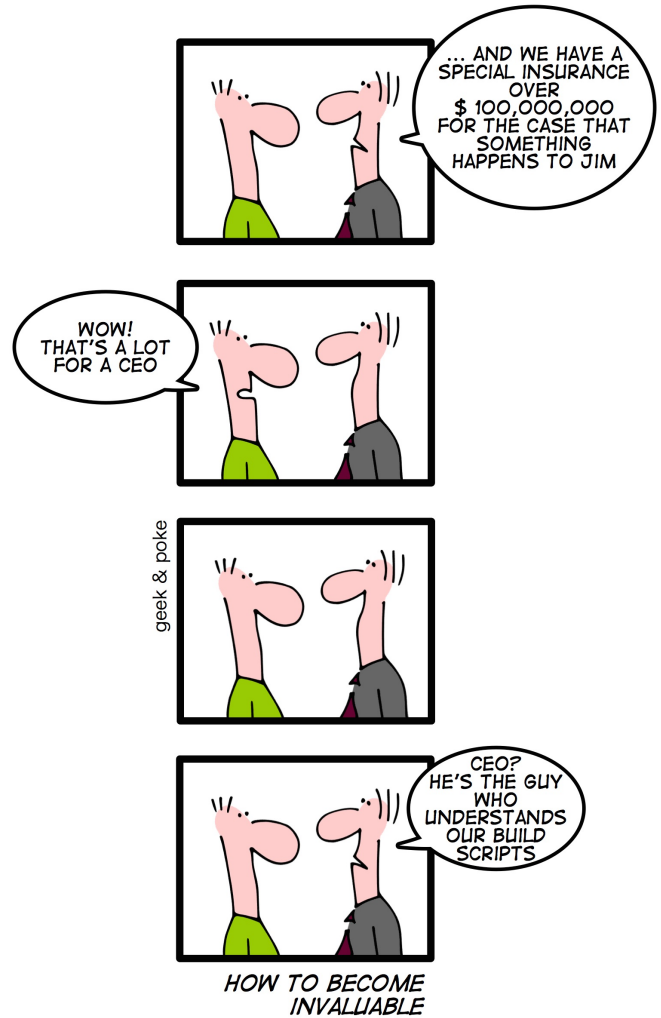
- scientific research domain: bioinformatics ...
- 59 dependencies in total (*without* compiler toolchain), some optional
  - depends on Haskell (GHC), Java, Python, R, Perl, OCaml, ...
  - several deps use a non-standard build procedure (in various degrees)
- *very* picky about dependency versions (e.g., *must* be Python v2.7.*3*)
- took us several weeks to get it installed (using Intel compilers!)...
- **...now we can (re)build/install it all with a single command!**

(disclaimer: support for QIIME not included yet in latest EasyBuild release)

Houston, we have a problem

Installation of scientific software is a *tremendous* problem for HPC sites all around the world.

- huge burden on HPC user support teams
- researchers lose lots of time (waiting)
- sites typically resort to in-house scripting
- very little collaboration among HPC sites :(



What about existing tools?

Existing tools are not well suited to scientific software and HPC systems.

- package managers: **yum** (RPMs), **apt-get** (.deb), ...
- **Homebrew** (Mac OS X), `http://brew.sh/`
- **Linuxbrew**, `http://brew.sh/linuxbrew/`
- **Portage** (Gentoo), `http://wiki.gentoo.org/wiki/Project:Portage`
- **pkgsrc** (NetBSD & (a lot) more), `http://pkgsrc.org/`
- **Nix**, `http://nixos.org/nix`

Common problems:

- usually poor support for multiple versions/builds existing side-by-side
- not flexible enough to deal with idiosyncrasies of scientific software
- hard to maintain (bash, heavy copy-pasting, ...)
- little support for scientific software, other compilers (not GCC), ...

EasyBuild: building software with ease

`http://hpcugent.github.io/easybuild/`

- tool for building/installing (scientific) software on HPC systems
- collection of Python packages and modules
- in-house since 2009, open-source (GPLv2) since 2012
- now: thriving community; actively contributing, driving development
- new release every 6–8 weeks (latest: EasyBuild v2.4.0, Nov 10th 2015)
  next release: planned before end of 2015 (v2.5.0)
- supports over 730 different software packages
  including CP2K, GAMESS-US, GROMACS, NAMD, NWChem, OpenFOAM, PETSc, QuantumESPRESSO, WRF, WPS, ...
- well documented: `http://easybuild.readthedocs.org`

Projects similar to EasyBuild

- **Spack** (LLNL) - `http://scalability-llnl.github.io/spack/`
- **Maali** (Pawsey) - `https://github.com/chrisbpawsey/maali/`
- **Smithy** (NICS, ORNL) - `http://anthonydigirolamo.github.io/smithy/`

Major differences with EasyBuild:

- slightly different approach
- smaller community
- fewer supported software packages
- less flexibility
- not so powerful (except Spack?)

All have expressed interest in cross-community collaboration.

EasyBuild: system requirements

- main target platform (for now) is Linux x86_64 (HPC systems)
  - Red Hat-based systems (Scientific Linux, CentOS, RHEL, ...)
  - also other Linux distros: Debian, Ubuntu, OpenSUSE, SLES, ...
  - also (kind of) works on OS X
  - *no* Windows support (and none planned)
  - experimental support for Cray systems since EasyBuild v2.1.0
  - support for Linux@POWER systems is being looked into
- Python v2.6.x or more recent v2.x (no Python 3 support yet)
- a modules tool:
  - latest release of Tcl/C environment modules (version 3.2.10);
  - or one of the Tcl-only versions of environment modules;
  - or a recent version of *Lmod* (5.6.3 or more recent)
- (a system C/C++ compiler, to get started)

EasyBuild: feature highlights

- fully **autonomously** building and installing (scientific) software
  - automatic dependency resolution
  - automatic generation of module files (Tcl or Lua syntax)
- thorough **logging** of executed build/install procedure
- **archiving** of build specifications ('*easyconfig files*')
- highly **configurable**, via config files/environment/command line
- **dynamically extendable** with additional *easyblocks*, *toolchains*, etc.
- support for **custom module naming schemes** (incl. hierarchical)
- **comprehensively tested**: lots of unit tests, regression testing, ...
- actively developed, **collaboration** between various HPC sites
- worldwide **community**

EasyBuild terminology

- EasyBuild *framework*
  - core of EasyBuild: Python modules & packages
  - provides supporting functionality for building and installing software
- *easyblock*
  - a Python module, 'plugin' for the EasyBuild framework
  - implements a (generic) software build/install procedure
- *easyconfig* file (`*.eb`)
  - build specification: software name/version, compiler toolchain, etc.
- compiler *toolchain*
  - compilers with accompanying libraries (MPI, BLAS/LAPACK, etc.)

Putting it all together

The EasyBuild *framework* leverages *easyblocks* to automatically build and install (scientific) software using a particular *compiler toolchain*, as specified by one or more *easyconfig files*.

'Quick' demo for the impatient

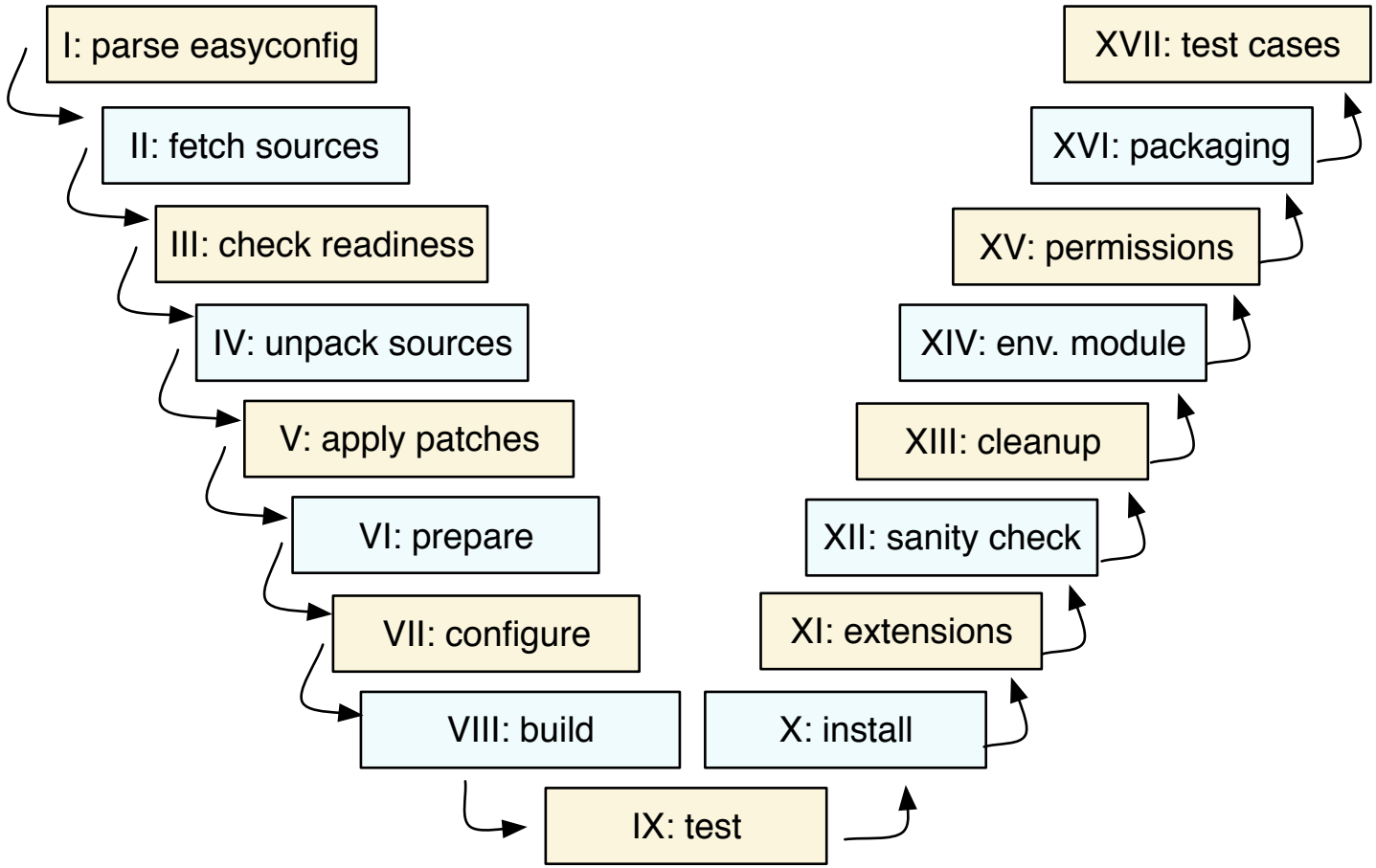
```
eb HPL-2.0-goolf-1.4.10-no-OFED.eb --robot
```

- **downloads** all required sources (best effort)
- **builds/installs** *goolf* toolchain (be patient) + HPL on top of it
  goolf: GCC, OpenMPI, LAPACK, OpenBLAS, FFTW, ScaLAPACK
- **generates module file** for each installed software package
- default: source/build/install dir in `$HOME/.local/easybuild`
  can be easily changed by configuring EasyBuild differently

note: needs a refresh, OpenBLAS version is too old for recent systems

Step-wise install procedure

build and install procedure as implemented by EasyBuild



most of these steps can be customised if required,
via *easyconfig parameters* or a *custom easyblock*

EasyBuild: statistics

EasyBuild v2.4.0 *(Nov'15)*

- $\sim 23,000$ LoC in framework (17 Python packages, 153 Python modules)
  - $+ \sim 5,000$ LoC in `vsc-base` (option parsing/logging)
  - $+ \sim 11,000$ LoC more in unit tests
  - $\Longrightarrow \sim 39,000$ LoC in total
- 182 easyblocks in total ($\sim 16,000$ Loc)
  - 154 software-specific easyblocks
  - 28 generic easyblocks
- 784 different software packages supported (incl. toolchains & bundles)
  - bio: 172, tools: 104, vis: 82, lib: 78, devel 68, math: 49, data: 50, toolchain: 38, chem: 32, lang: 28, mpi: 23, numlib: 22, perf: 21, system: 19, cae: 16, compiler: 12, phys: 6
- 4,596 easyconfig files: different versions/variants, toolchains, ...

Installing EasyBuild

http://easybuild.readthedocs.org/en/latest/Installation.html

Install EasyBuild using the bootstrap script *(highly recommended)*:

```
$ curl -O https://raw.github.com/hpcugent/easybuild-framework/
develop/easybuild/scripts/bootstrap_eb.py
$ python bootstrap_eb.py /tmp/$USER   # replace with your prefix!
$ module use /tmp/$USER/modules/all
$ module load EasyBuild
```

Update EasyBuild with . . . EasyBuild!

```
$ module load EasyBuild/2.3.0
$ eb EasyBuild-2.4.0.eb
$ module swap EasyBuild EasyBuild/2.4.0
```

Configuring EasyBuild

http://easybuild.readthedocs.org/en/latest/Configuration.html

By default, EasyBuild will (ab)use `$HOME/.local/easybuild`.

You should configure EasyBuild to your preferences, via:

- *configuration file(s):* key-value lines, text files (e.g., `prefix=/tmp`)
- *environment variables* (e.g., `$EASYBUILD_PREFIX` set to `/tmp`)
- *command line parameters* (e.g., `--prefix=/tmp`)

Consistency across these options is guaranteed (see `eb --help | tail`).

Priority among different options: cmdline, env vars, config file.
For example:

- `--prefix` overrules `$EASYBUILD_PREFIX`
- `$EASYBUILD_PREFIX` overrules `prefix` in configuration file

Basic usage

http://easybuild.readthedocs.org/en/latest/Using_the_EasyBuild_command_line.html

- specify software name/version and toolchain to 'eb' command
- commonly via name(s) of easyconfig file(s):

```
eb GCC-4.9.2.eb Clang-3.6.0-GCC-4.9.2.eb
```

- or directory name(s) providing set(s) of easyconfig files:

```
eb $HOME/myeasyconfigs
```

- or via command line options:

```
eb --software-name=GCC
```

- `--robot`/`-r`: dependency resolution, `--debug`/`-d`: debug logging

```
eb WRF-3.6.1-intel-2015a-dmpar.eb -dr
```

Workflow example

http://easybuild.readthedocs.org/en/latest/Typical_workflow_example_with_WRF.html

- searching for available easyconfigs (case-insensitive):

```
eb -S hpl
```

- overview of required/available modules (dry run):

```
eb HPL-2.1-foss-2015a.eb -D
```

- show install procedure that will be executed (extended dry run):

```
eb HPL-2.1-foss-2015a.eb -x
```

- enable debug logging, (also) log to stdout, robot build:

```
eb HPL-2.1-foss-2015a.eb -dlr
```

Example 'eb' output

```
$ eb GCC-4.9.3.eb
== temporary log file in case of crash /tmp/eb-GyvPHx/easybuild-U1TkEI.log
== processing EasyBuild easyconfig GCC-4.9.3.eb
== building and installing GCC/4.9.3...
== fetching files...
== creating build dir, resetting environment...
== unpacking...
== patching...
== preparing...
== configuring...
== building...
== testing...
== installing...
== taking care of extensions...
== postprocessing...
== sanity checking...
== cleaning up...
== creating module...
== permissions...
== packaging...
== COMPLETED: Installation ended successfully
== Results of the build can be found in the log file /opt/easybuild/software/GCC/4...
== Build succeeded for 1 out of 1
== Temporary log file(s) /tmp/eb-GyvPHx/easybuild-U1TkEI.log* have been removed.
== Temporary directory /tmp/eb-GyvPHx has been removed.
```

Other common command line options

- install to different installation prefix:

```
eb HPL-2.1-foss-2015a.eb --installpath=/tmp/$USER
```

- (try to) build & install different software version:

```
eb HPL-2.1-foss-2015a.eb --try-software-version=2.0
```

- (try to) build & install with a different toolchain (version):

```
eb HPL-2.0-foss-2014b.eb --try-toolchain-version=2015a -r
eb HPL-2.1-foss-2015a.eb --try-toolchain=intel,2015a -r
```

- grab (specific) easyconfigs from a GitHub pull request:

```
eb --from-pr 1239 OpenMPI-1.8.4-GCC-4.9.2.eb --robot
```

EasyBuild community

- ∼150 subscribers to the EasyBuild mailing list
- ∼20 active members on the `#easybuild` IRC channel
- bi-weekly conf calls: recent developments, discussing problems, ...
- users/contributors at HPC sites and companies around the world

  incl. Flemish Supercomputer Centre sites, Jülich Supercomputer Centre, Univ. of Basel, OHRI (Canada), Univ. of Auckland, Bayer AG, Texas A&M, IMP/IMBA (Austria), Univ. of Luxembourg, Cyprus Institute, University of Colorado Boulder, CSCS (Switzerland), ...
- *"Getting Scientific Software Installed"* BoF sessions at ISC/SC
- 10 'hackathon' workshops (2-3 days) since Aug'12
  - Ghent (Belgium), Luxembourg, Nicosia (Cyprus), Jülich (Germany), Vienna (Austria), Basel (Switzerland), Espoo (Finland)
  - 10th hackathon at TACC (Austin, Texas), Nov'15, before SC15
- **1st EasyBuild User Meeting: Jan 27-29 2016, Ghent**

HUST'14 paper

Modern Scientific Software Management Using EasyBuild and Lmod

Markus Geimer (JSC)

Kenneth Hoste (HPC-UGent)

Robert McLay (TACC)

`http://hpcugent.github.io/easybuild/files/hust14_paper.pdf`

- paper at HPC User Support Tools workshop (HUST'14 @ SC14)
- explains basics of module tools, EasyBuild and Lmod
- highlights issues with current approaches in software installation
- advocates use of a hierarchical module naming scheme
- presents EasyBuild and Lmod as adequate tools for (automated) software/module management on HPC systems

Do you want to know more?

- EasyBuild website: `http://hpcugent.github.io/easybuild`
- EasyBuild documentation: `http://easybuild.readthedocs.org`
- stable EasyBuild releases: `http://pypi.python.org/pypi/easybuild`

  EasyBuild framework: `http://pypi.python.org/pypi/easybuild-framework`
  easyblocks: `http://pypi.python.org/pypi/easybuild-easyblocks`
  easyconfigs `http://pypi.python.org/pypi/easybuild-easyconfigs`
- source repositories on GitHub

  EasyBuild meta package + docs: `https://github.com/hpcugent/easybuild`
  EasyBuild framework: `https://github.com/hpcugent/easybuild-framework`
  easyblocks: `https://github.com/hpcugent/easybuild-easyblocks`
  easyconfigs: `https://github.com/hpcugent/easybuild-easyconfigs`
- EasyBuild mailing list: easybuild@lists.ugent.be

  `https://lists.ugent.be/wws/subscribe/easybuild`
- Twitter: @easy_build
- IRC: `#easybuild` on `chat.freenode.net`