EasyBuild: building software with ease

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

`http://users.ugent.be/~kehoste/EasyBuild-intro-Espoo_20150506.pdf`

User Support Workshop @ NeIC 2015 (Espoo, Finland)
20150506

HPC-UGent in a nutshell

http://www.ugent.be/hpc

- HPC team at central IT dept. of Ghent University (Belgium)
- 9 team members: 1 manager, ∼3 user support, ∼5 sysadmin
- 6(+2) 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

"Please install this on the HPC?"

In the context of high performance computing, *building from source* should be preferred, when possible (i.e., if sources are available).

This allows for controlling used compilers and libraries, optimizing the software for the specific system architecture (e.g., AVX, network), etc.

Installing (lots of) *scientific* software is typically:

- error-prone, trial-and-error
- tedious, hard to get right
- repetitive & boring (well...)
- time-consuming (hours, days, even weeks)
- frustrating ("*Pandora's box*")
- 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- poor/outdated/missing/incorrect documentation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`)

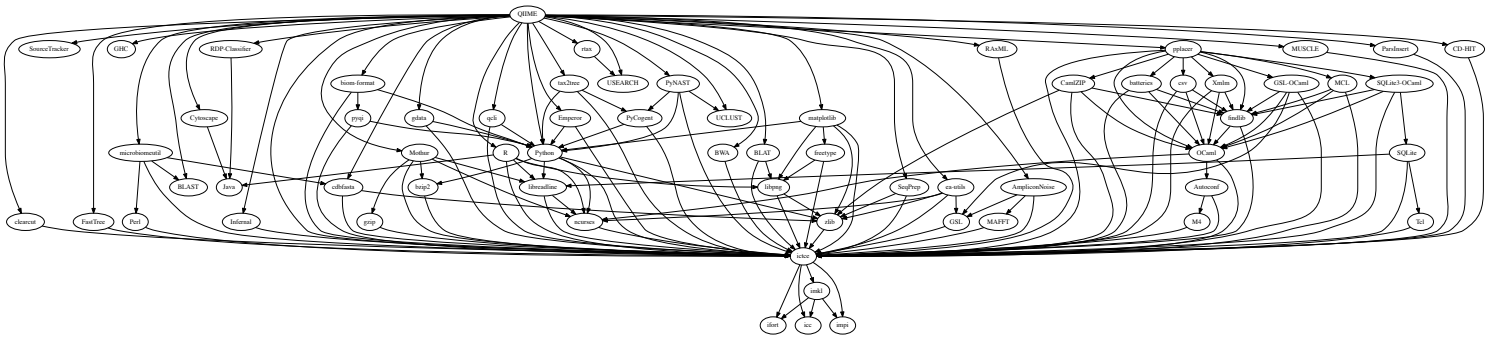
- dozen dependencies: netCDF (C, Fortran), HDF5, tcsh, JasPer, ...
- known issues in last release are (only) documented on website
  - no patch file provided, infrequent bugfix releases
- interactive '`configure`' script :(
- resulting `configure.wrf` needs work:
  - fix hardcoded settings (compilers, 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 (like we wanted)...
- **...now we can (re)build/install it all with a single command!**

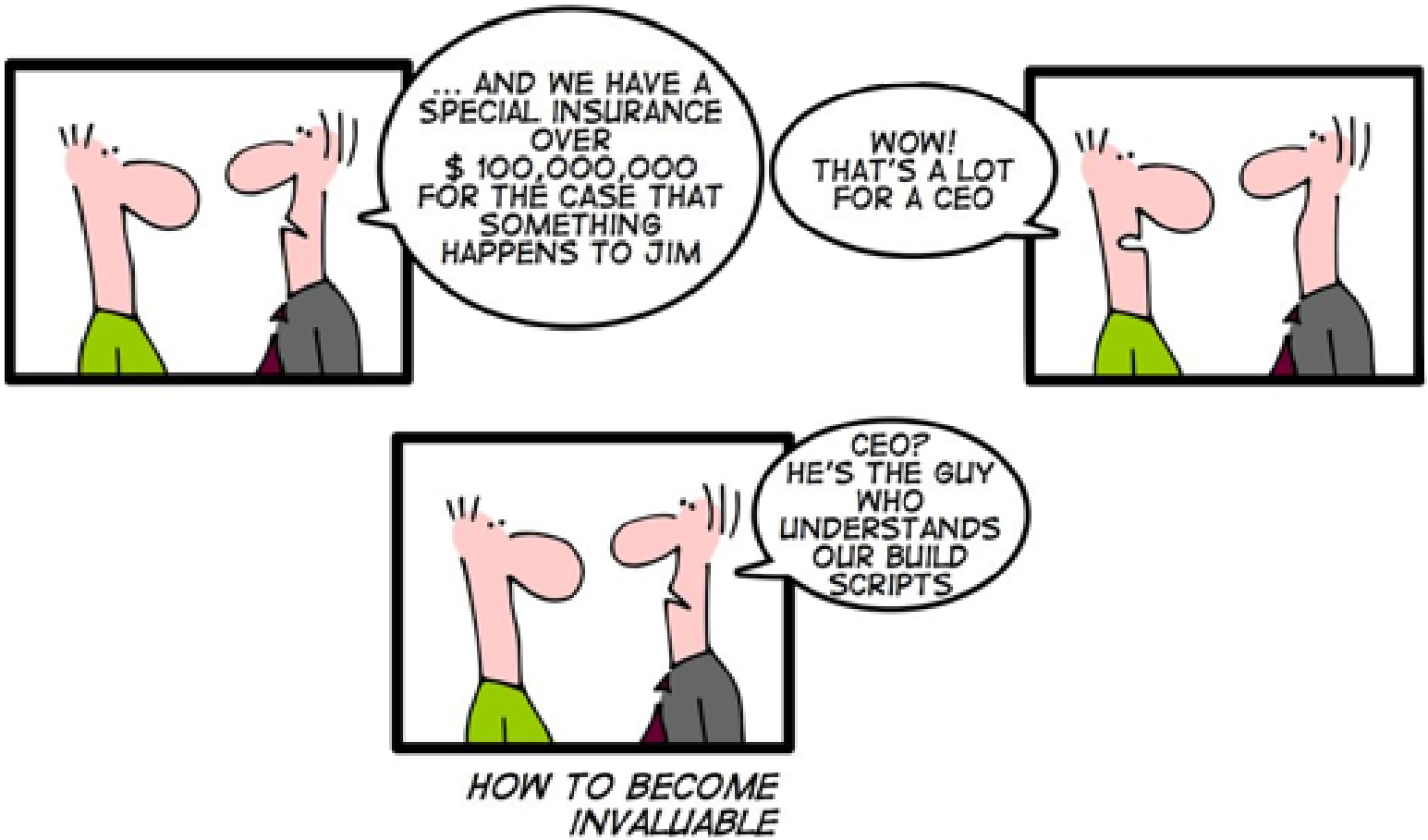
(disclaimer: support for QIIME not included yet in latest version of EasyBuild)

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 time just to get stuff installed
- every site deals with it in its own way (scripting, . . .)
- very little collaboration among HPC sites :(

How to become invaluable



http://geekandpoke.typepad.com/geekandpoke/2010/05/how-to-become-invaluable.html

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/`

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/

- framework for building and installing software
- collection of Python packages and modules
- in-house since 2009, open-source (GPLv2) since Nov 2012
- thriving community: actively contributing, driving development
- new release every 6–8 weeks (latest: EasyBuild v2.1.0, 20150430)

  next release: planned for end of June'15 (v2.2.0)
- supports *over 600* different software packages

  including CP2K, GAMESS-US, GROMACS, NAMD, NWChem, OpenFOAM, PETSc, QuantumESPRESSO, WRF, WPS, ...
- well documented: http://easybuild.readthedocs.org

Similar projects

- **Spack** (LLNL), `http://scalability-llnl.github.io/spack/`
- **iVEC Build System (iBS)**, `http://ivec.org/` *(not public (yet))*
- **Smithy** (NICS, ORNL), `http://anthonydigirolamo.github.io/smithy/`

Major differences with EasyBuild:

- slightly different approach
- smaller community
- fewer supported software packages
- less flexibility

Most (all?) are interested in switching to/merging with EasyBuild.

EasyBuild: requirements

- main target platform (for now) is Linux x86_64 (HPC systems)
  - Red Hat Linux based (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 in 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**

'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

EasyBuild: high-level design overview

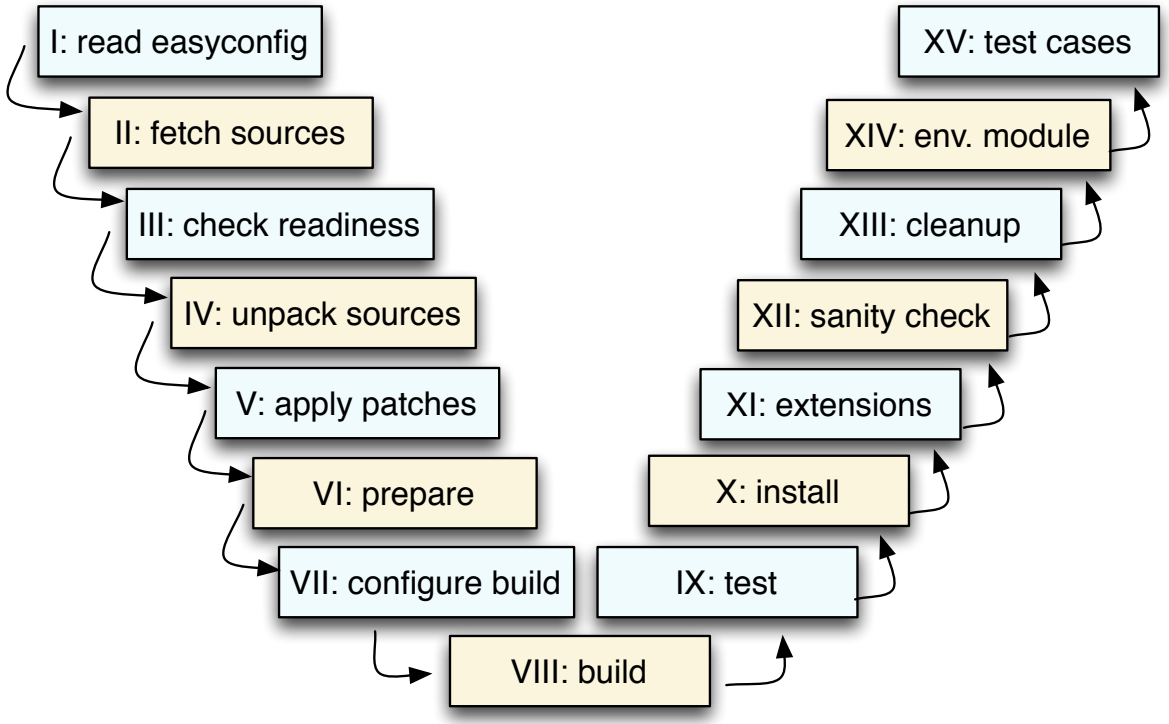
- EasyBuild *framework*
  - core of EasyBuild
  - provides supporting functionality for building and installing software
- *easyblock*
  - a Python module
  - implements a (generic) software build/install procedure
- *easyconfig* file
  - build specification: software name/version, compiler toolchain,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

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*.

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 v2.1.0

- $\sim 20,000$ LoC in framework (14 Python packages, 140 Python modules)

  $+ \sim 5,000$ LoC in `vsc-base` library it requires
  $+ \sim 9,000$ LoC more in unit tests
  $\Longrightarrow \sim 34,000$ LoC in total

- 167 easyblocks in total

  145 software-specific easyblocks
  22 generic easyblocks
  $\sim 15,500$ LoC

- 649 different software packages supported (incl. toolchains)

  bio: 132, tools: 85, lib: 68, devel: 63, vis: 53, data: 41,
  toolchain: 37, math: 46, numlib: 22, mpi: 10, chem: 28,
  lang: 24, system: 18, cae: 10, compiler: 11, perf: 16, phys: 5

- 3,618 easyconfig files: different versions/variants, toolchains, ...

List of supported software packages

a2ps ABAQUS ABINIT ABySS ACML AFNI **ALADIN** Allinea ALLPATHS-LG AMOS AnalyzeFMRI ANSYS ant ANTs APBS ARB argtable aria2 Armadillo arpack-ng ASE ATLAS Autoconf Automake Autotools bam2fastq BamTools Bash BayesTraits bbcp bbFTP bbftpPRO bc BCFtools beagle-lib Beast BEDOPS BEDTools BFAST binutils biodeps BioPerl Biopython BiSearch Bison BitSeq BLACS BLAST BLAST+ BLAT BOINC Bonnie++ Boost Bowtie Bowtie2 bsoft BWA byacc bzip2 cairo CAP3 CBLAS ccache CCfits CD-HIT CDO CEM CFITSIO cflow CGAL cgdb cgmpich cgmpolf cgmvapich2 cgmvolf cgompi cgoolf Chapel CHARMM Chimera Circos Clang ClangGCC CLHEP CLooG Clustal-Omega ClustalW2 CMake Coot Coreutils Corkscrew **CP2K** CPLEX CrayCCE CrayGNU CrayIntel CRF++ ctffind Cube Cuby CUDA Cufflinks cURL cutadapt CVS CVXOPT Cython DB DBD-mysql DBD-SQLite DB_File DIALIGN-TX Diffutils DL_POLY_Classic Docutils **DOLFIN** Doxygen DSRC **EasyBuild** ECore ed Eigen ELinks ELPA ELPH Emacs EMBOSS EPD ErlangOTP ESMF ESPResSo evmix Exonerate expat eXpress FASTA fastahack fastqz FastTree FASTX-Toolkit FCM FDS FDTD_Solutions Ferret FFC ffmpeg FFTW FIAT file findutils fixesproto flex FLTK FLUENT fmri FoldX fontconfig foss fqzcomp FRC_align freeglut FreeSurfer freetype FSA FSL g2clib g2lib **GAMESS-US** GATE GATK gawk GCC gcccuda GD GDAL GDB Geant4 GEM-library GEMSTAT GEOS getdp gettext GHC Ghostscript gimkl gimpi GIMPS git GLib GLIMMER GLPK glproto gmacml GMAP-GSNAP GMP gmpich gmpolf gmsh GMT gmvapich2 gmvolf gnuplot gnutls Go goalf GObject-Introspection gompi gompic goolf goolfc GPAW gperf gperftools gpsmpi gpsolf grace Graphviz GraphViz2 Greenlet Grep grep grib_api GROMACS GSL gsl GSSAPI gtest GTI GTS guile gzip h4toh5 h5py h5utils Hadoop hanythingondemand HarfBuzz Harminv HDF HDF5 HH-suite HMMER Hoard horton HPCBIOS_Bioinfo HPCBIOS_Debuggers HPCBIOS_LifeSciences HPCBIOS_Math HPCBIOS_Profilers HPCG HPL HTSeq HTSlib hwloc Hypre icc iccifort ictce ifort iimpi iiqmpi imake IMB imkl impi Infernal inputproto Inspector Instant intel intel-para intltool iomkl iompi IOR Iperf ipp ipsmpi IPython iqacml ISL IsoInfer ispc itac JAGS Jansson JasPer Java Jellyfish jemalloc Jinja2 JUnit kbproto Kerberos_V5 LAPACK less lftp libcap-ng libcircle libctl libdrm libdwarf libelf libevent libffi libgd libgtextutils libharu libibmad libibumad libibverbs libICE libidn Libint libjpeg-turbo libmatheval libpciaccess libpng libpthread-stubs libreadline libSM libsmm LIBSVM LibTIFF libtool libudev libungif libunistring libunwind libX11 libXau libXaw libxc libxcb libXdmcp libXext libXfixes libXft libXi libXinerama libxml2 libXmu libXp libXpm libXrender libxslt libXt libyaml likwid Lmod Lua LWM2 lxml lynx LZO M4 MAFFT make makedepend Maple MariaDB Mathematica MATLAB matplotlib Maven mawk mc MCL mcpp MDP mdtest Meep MEME Mercurial Mesa Mesquite MetaVelvet MethPipe METIS Minimac Minimac3 MMSEQ Modeller Molden Molekel molmod monty Mothur motif MPC MPFR mpi4py mpiBLAST MPICH MPICH2 MrBayes MTL4 multitail MUMmer MUMPS MUSCLE MUST MUSTANG MVAPICH2 MySQL NAMD nano NASM NCBI-Toolkit ncdf4 **NCL** ncurses ncview NEdit netaddr netCDF netCDF-C++ netCDF-C++4 netCDF-Fortran netcdf4-python netifaces NetLibIDN netloc nettle NEURON nodejs ns numactl numexpr numpy **NWChem** O2scl Oases OCaml Octave Oger OPARI2 OpenBabel OpenBLAS OpenCV **OpenFOAM** **OpenFOAM-Extend** OpenIFS OpenMD OpenMPI OpenPGM OpenSees OpenSSL ORCA orthomcl otcl OTF OTF2 packmol PAML pandas PANDAseq Pango PAPI parallel Paraview ParFlow ParMETIS ParMGridGen Pasha patch paycheck pbs_python PCC PCRE PDT Perl **PETSc** petsc4py phonopy PhyML picard pigz pixman pkg-config PLINK Pmw PnMPI popt PP PRACE PRANK Primer3 printproto problog protobuf pscom PSI psmpi psmpi2 Pygments pyhull pymatgen PyQt PyQuante pysqlite pyTables **Python** python-dateutil python-meep PyYAML PyZMQ Qhull QLogicMPI Qt qtop QuadProg++ **QuantumESPRESSO** Quip R RAxML Ray RCS rCUDA RDP-Classifier RELION renderproto requests rjags RNAz ROOT Rosetta rSeq RSEQtools Ruby runjags S-Lang Sablotron SAMtools ScaLAPACK Scalasca SCALCE ScientificPython scikit-learn scipy SCons SCOOP Score-P SCOTCH SDCC SDPA sed segemehl seqtk setuptools Shapely SHRiMP SIBELia sickle Silo SIP slalib-c SLEPc SMALT SOAPaligner SOAPdenovo SOAPdenovo2 SOAPec SPAdes Spark sparsehash spglib Sphinx SPRNG SQLite SRA-Tookit Stacks stemming Stow STREAM Stride SuiteSparse SURF SWIG sympy systemd Szip TAMkin Tar tbb TCC Tcl tclcl tcsh Tesla-Deployment-Kit texinfo Theano TiCCutils TiMBL TINKER TinySVM Tk tmux TopHat Tornado TotalView TREE-PUZZLE Trilinos Trinity UDUNITS UFC UFL util-linux Valgrind VCFtools Velvet ViennaRNA Vim Viper vsc-base vsc-mympirun vsc-mympirun-scoop vsc-processcontrol VSC-tools VTK VTune WHAM **WIEN2k** wiki2beamer worker **WPS** **WRF** xbitmaps xcb-proto XCrySDen Xerces-C++ xextproto xineramaproto Xmipp XML XML-Dumper XML-LibXML XML-Parser XML-Simple XML-Twig xorg-macros xproto XQilla xtrans XZ yaff YamCha YAML-Syck Yasm YAXT ZeroMQ zlib ZPAQ zsh zsync

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 # specify your install prefix
$ module use /tmp/$USER/modules/all
$ module load EasyBuild
```

Update EasyBuild with . . . EasyBuild!

```
$ module load EasyBuild/1.16.1
$ eb EasyBuild-1.16.0.eb --try-software-version=2.1.0
$ module swap EasyBuild EasyBuild/2.1.0
```

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

- 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` enables dependency resolution:

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

Workflow example

- getting help, overview of options:

```
eb --help
```

- searching for easyconfigs:

```
eb --search HPL   # long output (full paths)
eb -S HPL         # short output
```

- overview of required/available modules:

```
eb HPL-2.1-foss-2015a.eb --dry-run  # long output
eb HPL-2.1-foss-2015a.eb -D         # short output
```

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

```
eb HPL-2.1-foss-2015a.eb --debug -lr
```

Other common command line options

- install to different installation prefix:

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

- build & install different software version:

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

- 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
```

Writing easyconfig files, contributing back

For software packages that follow some type of standard build procedure, just writing an easyconfig file is likely sufficient.

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

If you have working easyconfigs for new software packages, version updates, or using a different toolchain, consider *contributing back* to the central easybuild-easyconfigs repository.

`https://github.com/hpcugent/easybuild/wiki/Contributing-back`

Over 25% of the builds currently supported were defined by *external contributors* (non-HPC-UGent), and that ratio is increasing every day.

Lmod: a modern modules tool

https://tacc.utexas.edu/research-development/tacc-projects/lmod

- drop-in alternative for Tcl-based module tools (a few edge cases)
- improves user experience, without hindering experts
- written in Lua, consumes Lua (and Tcl) module files
- available since Oct'08, *actively developed*, frequent stable releases
- driven by community demands and feedback
- developed by Dr. Robert McLay (TACC, UT Austin)

Example: swapping modules using `ml` shorthand in a module hierarchy

```
$ ml
Currently loaded modules:
 1) GCC/4.8.2   2) MPICH/3.1.1   3) FFTW/3.3.2
$ ml -GCC Clang
The following have been reloaded:
 1) FFTW/3.3.2   2) MPICH/3.1.1
$ ml
Currently loaded modules:
 1) Clang/3.4   2) MPICH/3.1.1   3) FFTW/3.3.2
```

Lmod: feature highlights

- module hierarchy-aware design and functionality
  - searching across entire module tree with 'module `spider`'
  - automatic reloading of dependent modules on '`module swap`'
  - marking missing dependent modules as inactive after '`module swap`'
- caching of module files, for responsive subcommands (e.g., `avail`)
- site-customizable behavior via provided hooks
- `ml` command ('`ml`' is '`module list`', '`ml GCC`' is '`module load GCC`', ...)
- load/unload shortcuts via + and -
- various other useful/advanced features, including:
  - case-insensitive '`avail`' subcommand
  - can send subcommand output to stdout (rather than to stderr)
  - defining module families (e.g., '`compiler`', '`mpi`')
  - assigning properties to modules (e.g., '`sticky`', '`Phi-aware`')
  - stack-based definition of environment variables (using `pushenv`)
  - user-definable collections of modules (`module save`)
  - and a lot more ...

Combined power of EasyBuild and Lmod

Build and install WRF in a module hierarchy, with two different toolchains

```
$ export EASYBUILD_MODULE_NAMING_SCHEME=HierarchicalMNS
$ WRF-3.6.1-intel-2015a-dmpar.eb -dr
$ WRF-3.6.1-intel-2015a-dmpar.eb -dr --try-toolchain-name=foss
```

List existing WRF modules, load GCC/OpenMPI build

```
$ module spider WRF
...
$ ml GCC OpenMPI WRF
```

Swap to Intel build of WRF

```
$ ml -GCC -OpenMPI +icc +ifort +impi
The following have been reloaded:
...
```

Synergy between EasyBuild and Lmod

- EasyBuild can easily build and install hundreds of packages
  - ⇒ lots of modules, may be overwhelming for users
- Lmod's support for hierarchical modules trees can help
  - ⇒ support for using Lmod and hierarchical module naming schemes was added to EasyBuild
- EasyBuild has helped uncover multiple performance issues in Lmod
  - ⇒ Lmod has significantly improved the speed of certain operations
    - '`module --terse avail`' (doesn't parse module files)
    - faster '`module avail`' with Lmod system cache in place (v5.8.6)
- feature requests from the EasyBuild community
  - ⇒ Lmod has added new functionality, for example stack-based definition of environment variables using '`pushenv`'

The synergy between both tools has made them significantly better!

EasyBuild community

- over 110 subscribers to the EasyBuild mailing list
- 20-25 active members on the `#easybuild` IRC channel
- users/contributors at HPC sites and companies around the world

  incl. Flemish Supercomputer Centre sites, Jülich Supercomputer Centre, Univ. of Basel, Stanford Univ., Univ. of Auckland, Bayer AG, Texas A&M, IMP/IMBA (Austria), Univ. of Luxembourg, Cyprus Institute, ...

- *"Getting Scientific Software Installed"* BoF sessions at ISC/SC
- 9 '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**
- proposal for half-day EasyBuild/Lmod tutorial submitted to SC15 (cross your fingers)

Proper documentation covering the basics

`http://easybuild.readthedocs.org`

EasyBuild installation:

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

EasyBuild configuration:

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

Usage of `eb` command line:

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

Writing easyconfig files:

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

Navigating log files:

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

'Sources' are in GitHub repository, `.rst` format; easy to contribute to!

Updating documentation is considered part of release process.

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 software 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`