EasyBuild Documentation

Release 20210907.0

Ghent University

Tue, 07 Sep 2021 07:19:38

Contents

EasyBuild documentation

Welcome to the documentation of EasyBuild, a software build and installation framework that allows you to manage (scientific) software on High Performance Computing (HPC) systems in an efficient way.

This documentation is intended for EasyBuild version 4.4.2, and was last rebuilt on Tue, 07 Sep 2021 07:18:35.

Note: Please consult the overview of major changes in EasyBuild v4.0 **before upgrading!**

CHAPTER 1

What is EasyBuild?

EasyBuild is a software build and installation framework that allows you to manage (scientific) software on High Performance Computing (HPC) systems in an efficient way. It is motivated by the need for a tool that combines the following features:

- a **flexible framework** for building/installing (scientific) software
- fully **automates** software builds
- divert from the standard `configure` / `make` / `make install` with custom procedures
- allows for easily **reproducing** previous builds
- keep the software build recipes/specifications **simple and human-readable**
- supports **co-existence of versions/builds** via dedicated installation prefix and module files
- enables **sharing** with the HPC community (win-win situation)
- automagic **dependency resolution**
- **retain logs** for traceability of the build processes

Some key features of EasyBuild:

- build & install (scientific) software **fully autonomously**
  - also interactive installers, code patching, generating module file, ...
- easily *configurable*: config file/environment/command line
  - including aspects like module naming scheme
- thorough logging and archiving (see *Understanding EasyBuild logs*)
  - entire build process is logged thoroughly, logs are stored in install directory;
  - easyconfig file used for build is archived (install directory + file/svn/git repo)
- automatic **dependency resolution** (see *Enabling dependency resolution, –robot / -r and –robot-paths*)
  - build entire software stack with a single command, using `--robot`

- building software in **parallel**
- robust and thoroughly tested code base, fully unit-tested before each release
- thriving, growing **community**

Take a look at our HUST'14 workshop paper *Modern Scientific Software Management Using EasyBuild and Lmod* (download PDF here) and use that as a reference in case you present academic work mentioning EasyBuild.

CHAPTER 2

Concepts and terminology

EasyBuild consists of a collection of Python modules and packages that interact with each other, dynamically picking up additional Python modules as needed for building and installing a (stack of) software package(s) specified via simple specification files.

Or, in EasyBuild terminology: **the EasyBuild framework leverages easyblocks to automatically build and install software using particular compiler toolchains, as specified by one or multiple easyconfig files**.

Contents

2.1 EasyBuild framework

The EasyBuild **framework** embodies the core of the tool, providing functionality commonly needed when installing scientific software on HPC systems. For example, it deals with downloading, unpacking and patching of sources, loading module files for dependencies, setting up the build environment, autonomously running (interactive) shell commands, creating module files that match the specification files, etc.

Included in the framework is an *abstract* implementation of a software build and install procedure, which is split up into different *steps*:

- unpacking sources
- configuration
- build
- installation
- module generation
- etc.

Most of these steps, i.e., the ones which are generally more-or-less analogous across different software packages, have appropriate (default) implementations. The only exceptions are the configuration, build and installation steps that are purposely left unimplemented (since there is no common procedure for them).

Each of the steps can be tweaked and steered via different parameters known to the framework, for which values are either obtained from the provided specification files or set to reasonable default values. See *Easyconfig files*.

In EasyBuild version 4.4.2 the framework source code consists of about 19000 lines of code, organized across about 125 Python modules in roughly a dozen Python package directories, next to almost 7000 lines of code for tests. This provides some notion of the size of the EasyBuild framework and the amount of supporting functionality it has to offer.

2.2 Easyblocks

The implementation of a particular software build and install procedure is done in a Python module, which is aptly referred to as an **easyblock**.

Each easyblock ties in with the framework API by defining (or extending/replacing) one or more of the step functions that are part of the abstract procedure used by the EasyBuild framework. Easyblocks typically heavily rely on the supporting functionality provided by the framework, for example for (autonomously) executing (interactive) shell commands and obtaining the command's output and exit code.

A distinction is made between **software-specific** and **generic** easyblocks. Software-specific easyblocks implement a build and install procedure which is entirely custom to one particular software package (e.g., WRF), while generic easyblocks implement a procedure using standard tools (e.g., CMake). Since easyblocks are implemented in an object-oriented scheme, the step methods implemented by a particular easyblock can be reused in others via inheritance, enabling code reuse across build procedure implementations.

For each software package being built, the EasyBuild framework will determine which easyblock should be used, based on the name of the software package or the value of the `easyblock` specification parameter (see *Easyblock specification*). Since EasyBuild v2.0, an easyblock *must* be specified in case no matching easyblock is found based on the software name (cfr. *Automagic fallback to ConfigureMake*).

EasyBuild version 2.4.0 includes 154 software-specific easyblocks and 28 generic easyblocks (see also *List of easyblocks*), providing support for automatically installing a wide range of software packages. Examples range from fairly easy-to-build programs like gzip, other basic tools like compilers, various MPI stacks and commonly used libraries, primarily for x86_64 architecture systems, to large scientific software packages that are notorious for their involved and tedious install procedures, such as: *CP2K*, *NWChem*, *OpenFOAM*, *QuantumESPRESSO*, *WRF*.

2.3 Toolchains

EasyBuild employs so-called **compiler toolchains** or, simply *toolchains* for short, which are a major concept in handling the build and installation processes.

A typical toolchain consists of one or more compilers, usually put together with some libraries for specific functionality, e.g., for using an MPI stack for distributed computing, or which provide optimized routines for commonly used math operations, e.g., the well-known BLAS/LAPACK APIs for linear algebra routines.

For each software package being built, the toolchain to be used must be specified in some way.

The EasyBuild framework prepares the *build environment* for the different toolchain components, by loading their respective modules and defining environment variables to specify compiler commands (e.g., via `$F90`), compiler and linker options (e.g., via `$CFLAGS` and `$LDFLAGS`), the list of library names to supply to the linker (via `$LIBS`), etc. This enables making easyblocks largely *toolchain-agnostic* since they can simply rely on these environment variables; that is, unless they need to be aware of, for example, the particular compiler being used to determine the build configuration options.

Recent releases of EasyBuild include out-of-the-box toolchain support for:

- various compilers, including GCC, Intel, Clang, CUDA
- common MPI libraries, such as Intel MPI, MPICH, MVAPICH2, OpenMPI
- various numerical libraries, including ATLAS, Intel MKL, OpenBLAS, ScalaPACK, FFTW

Please see the *Common toolchains* page for details about the two most common toolchains, one for "free and open source software" (`foss`) based on GCC and one based on the Intel compilers (`intel`).

2.3.1 `system` toolchain

The `system` toolchain is a special case. It is an *empty* toolchain, i.e. a toolchain without any components, and corresponds to using the readily available compilers and libraries (e.g., the ones provided by the operating system, or by modules which were loaded before issuing the `eb` command).

When the `system` toolchain is used, a corresponding `system` module file is not required/loaded and no build environment is being defined.

2.3.2 `dummy` toolchain *(DEPRECATED)*

The `dummy` toolchain has been deprecated in EasyBuild v4.0, and replaced by the system_toolchain.

2.3.3 Common toolchains

For more information on the concept of *common toolchains*, see *Common toolchains*.

2.4 Easyconfig files

The specification files that are supplied to EasyBuild are referred to as **easyconfig files** (or simply *easyconfigs*), which are basically plain text files containing (mostly) key-value assignments for build parameters supported by the framework, also referred to as **easyconfig parameters** (see *Writing easyconfig files: the basics* for more information).

Note that easyconfig files only provide the bits of information required to determine the corresponding module name; the module name itself is computed by EasyBuild framework by querying the module naming scheme being used. The

complete list of supported easyconfig parameters can be easily obtained via the EasyBuild command line using `eb -a` (see also *All available easyconfig parameters, –avail-easyconfig-params / -a*).

As such, each easyconfig file provides a complete specification of which particular software package should be installed, and which settings should be used for building it. After completing an installation, EasyBuild copies the used easyconfig file to the install directory, as a template, and also supports maintaining an easyconfig archive which is updated on every successful installation. Therefore, reproducing installations becomes trivial.

2.5 Extensions

Some software packages support installing additional add-ons alongside the 'main' software, either in the same installation prefix, or in a separate location.

In EasyBuild, we use the neutral term '**extensions**' to refer these add-ons.

Well-known examples include:

- Perl modules (http://www.cpan.org/modules/)
- Python packages (https://pypi.python.org/pypi)
- R libraries (http://cran.r-project.org/web/packages/)
- Ruby gems (http://guides.rubygems.org/what-is-a-gem/)

CHAPTER 3

Typical workflow example: building and installing WRF

This section shows an example case of building Weather Research and Forecasting (WRF) scientific software application, which is a notoriously complex software application to build and install. With EasyBuild however, WRF can be installed quite easily and here is how.

First, you search which easyconfigs are available for WRF, using `--search` (see *Searching for easyconfigs, –search / -S*) and you select one based on the software version, toolchain, etc.

Using the selected easyconfig file, you can get an overview of the planned installations using `--dry-run` (see *Getting an overview of planned installations –dry-run / -D*).

Finally, building and installing WRF is done by specifying the matching easyconfig file in the eb command line, and using `--robot` (see *Enabling dependency resolution, –robot / -r and –robot-paths*) to enable dependency resolution. That way WRF and all of its dependencies are installed with *a single command*!

3.1 Searching for available easyconfigs files

Searching for build specification for a particular software package can be done using the `--search`/`-S` command line options (see *Searching for easyconfigs, –search / -S*); for example, to get a list of available easyconfig files for WRF:

```
$ eb -S WRF
CFGS1=/home/example/.local/easybuild/software/EasyBuild/4.1.1/easybuild/easyconfigs
 * $CFGS1/w/WPS/WPS-4.0.1_find-wrfdir.patch
 * $CFGS1/w/WPS/WPS-4.0.2_find-wrfdir.patch
 [ . . . ]
 * $CFGS1/w/WRF/WRF-4.0.1-intel-2018b-dmpar.eb
 * $CFGS1/w/WRF/WRF-4.0.2-foss-2018b-dmpar.eb
 [ . . . ]

Note: 16 matching archived easyconfig(s) found, use --consider-archived-easyconfigs␣
↪to see them
```

Various easyconfig files are found: for different versions of WRF (e.g., v4.0.1 and v4.0.2), for different (versions of) compiler toolchains (e.g., foss 2018b, intel 2018b), etc.

For the remainder of this example, we will use the available `WRF-4.0.2-foss-2018b-dmpar.eb` easyconfig file to specify to EasyBuild to build and install WRF v4.0.2 using version 2018b of the `foss` toolchain, which is one of the *Common toolchains*. The `foss` toolchain stands for `GCC, OpenMPI, OpenBLAS/LAPACK, ScaLAPACK, and FFTW`. See *List of known toolchains* for a list of all available toolchains.

3.2 Getting an overview of planned installations

To get an overview of the software that EasyBuild is going to build and install we can use the `--dry-run`/`-D` (see *Getting an overview of planned installations –dry-run / -D*) command line option. This will show a list of easyconfig files that will be used, together with the module files that will be installed, as well as their current availability (`[x]` marks available modules).

Note that EasyBuild will take care of all of the dependencies of WRF as well, and can even install the compiler toolchain as well if the corresponding modules are not available yet:

```
$ eb WRF-4.0.2-foss-2018b-dmpar.eb -Dr
== temporary log file in case of crash /tmp/eb-eEnBF5/easybuild-pvrvNP.log
Dry run: printing build status of easyconfigs and dependencies
CFGS=/home/example/.local/easybuild/software/EasyBuild/4.1.1/easybuild/easyconfigs
 * [x] $CFGS/m/M4/M4-1.4.18.eb (module: M4/1.4.18)
 * [x] $CFGS/z/zlib/zlib-1.2.11.eb (module: zlib/1.2.11)
 * [x] $CFGS/h/help2man/help2man-1.47.4.eb (module: help2man/1.47.4)
 * [x] $CFGS/m/M4/M4-1.4.17.eb (module: M4/1.4.17)
 * [x] $CFGS/b/Bison/Bison-3.0.4.eb (module: Bison/3.0.4)
 * [x] $CFGS/f/flex/flex-2.6.4.eb (module: flex/2.6.4)
 * [x] $CFGS/b/binutils/binutils-2.30.eb (module: binutils/2.30)
 * [x] $CFGS/g/GCCcore/GCCcore-7.3.0.eb (module: GCCcore/7.3.0)
 * [x] $CFGS/h/help2man/help2man-1.47.4-GCCcore-7.3.0.eb (module: help2man/1.47.4-
↪GCCcore-7.3.0)
 * [x] $CFGS/m/M4/M4-1.4.18-GCCcore-7.3.0.eb (module: M4/1.4.18-GCCcore-7.3.0)
 * [x] $CFGS/z/zlib/zlib-1.2.11-GCCcore-7.3.0.eb (module: zlib/1.2.11-GCCcore-7.3.0)
 * [x] $CFGS/b/Bison/Bison-3.0.4-GCCcore-7.3.0.eb (module: Bison/3.0.4-GCCcore-7.3.0)
 * [x] $CFGS/b/Bison/Bison-3.0.5-GCCcore-7.3.0.eb (module: Bison/3.0.5-GCCcore-7.3.0)
 * [x] $CFGS/f/flex/flex-2.6.4-GCCcore-7.3.0.eb (module: flex/2.6.4-GCCcore-7.3.0)
 * [x] $CFGS/b/binutils/binutils-2.30-GCCcore-7.3.0.eb (module: binutils/2.30-GCCcore-
↪7.3.0)
 * [ ] $CFGS/n/ncurses/ncurses-6.1-GCCcore-7.3.0.eb (module: ncurses/6.1-GCCcore-7.3.
↪0)
 * [ ] $CFGS/c/CMake/CMake-3.11.4-GCCcore-7.3.0.eb (module: CMake/3.11.4-GCCcore-7.3.
↪0)
 * [x] $CFGS/a/Autoconf/Autoconf-2.69-GCCcore-7.3.0.eb (module: Autoconf/2.69-GCCcore-
↪7.3.0)
 * [x] $CFGS/a/Automake/Automake-1.16.1-GCCcore-7.3.0.eb (module: Automake/1.16.1-
↪GCCcore-7.3.0)
 * [x] $CFGS/g/GCC/GCC-7.3.0-2.30.eb (module: GCC/7.3.0-2.30)
 * [ ] $CFGS/p/pkg-config/pkg-config-0.29.2-GCCcore-7.3.0.eb (module: pkg-config/0.29.
↪2-GCCcore-7.3.0)
 * [ ] $CFGS/c/cURL/cURL-7.60.0-GCCcore-7.3.0.eb (module: cURL/7.60.0-GCCcore-7.3.0)
 * [x] $CFGS/l/libtool/libtool-2.4.6-GCCcore-7.3.0.eb (module: libtool/2.4.6-GCCcore-
↪7.3.0)
 * [ ] $CFGS/s/Szip/Szip-2.1.1-GCCcore-7.3.0.eb (module: Szip/2.1.1-GCCcore-7.3.0)
 * [x] $CFGS/o/OpenBLAS/OpenBLAS-0.3.1-GCC-7.3.0-2.30.eb (module: OpenBLAS/0.3.1-GCC-
↪7.3.0-2.30)
```

(continues on next page)

(continued from previous page)

```
 * [ ] $CFGS/t/tcsh/tcsh-6.20.00-GCCcore-7.3.0.eb (module: tcsh/6.20.00-GCCcore-7.3.0)
 * [ ] $CFGS/j/JasPer/JasPer-2.0.14-GCCcore-7.3.0.eb (module: JasPer/2.0.14-GCCcore-7.
↪3.0)
 * [x] $CFGS/a/Autotools/Autotools-20180311-GCCcore-7.3.0.eb (module: Autotools/
↪20180311-GCCcore-7.3.0)
 * [ ] $CFGS/d/Doxygen/Doxygen-1.8.14-GCCcore-7.3.0.eb (module: Doxygen/1.8.14-
↪GCCcore-7.3.0)
 * [x] $CFGS/n/numactl/numactl-2.0.11-GCCcore-7.3.0.eb (module: numactl/2.0.11-
↪GCCcore-7.3.0)
 * [x] $CFGS/x/xorg-macros/xorg-macros-1.19.2-GCCcore-7.3.0.eb (module: xorg-macros/1.
↪19.2-GCCcore-7.3.0)
 * [x] $CFGS/l/libpciaccess/libpciaccess-0.14-GCCcore-7.3.0.eb (module: libpciaccess/
↪0.14-GCCcore-7.3.0)
 * [x] $CFGS/n/ncurses/ncurses-6.0.eb (module: ncurses/6.0)
 * [x] $CFGS/g/gettext/gettext-0.19.8.1.eb (module: gettext/0.19.8.1)
 * [x] $CFGS/x/XZ/XZ-5.2.4-GCCcore-7.3.0.eb (module: XZ/5.2.4-GCCcore-7.3.0)
 * [x] $CFGS/l/libxml2/libxml2-2.9.8-GCCcore-7.3.0.eb (module: libxml2/2.9.8-GCCcore-
↪7.3.0)
 * [x] $CFGS/h/hwloc/hwloc-1.11.10-GCCcore-7.3.0.eb (module: hwloc/1.11.10-GCCcore-7.
↪3.0)
 * [x] $CFGS/o/OpenMPI/OpenMPI-3.1.1-GCC-7.3.0-2.30.eb (module: OpenMPI/3.1.1-GCC-7.3.
↪0-2.30)
 * [x] $CFGS/g/gompi/gompi-2018b.eb (module: gompi/2018b)
 * [x] $CFGS/f/FFTW/FFTW-3.3.8-gompi-2018b.eb (module: FFTW/3.3.8-gompi-2018b)
 * [x] $CFGS/s/ScaLAPACK/ScaLAPACK-2.0.2-gompi-2018b-OpenBLAS-0.3.1.eb (module:␣
↪ScaLAPACK/2.0.2-gompi-2018b-OpenBLAS-0.3.1)
 * [x] $CFGS/f/foss/foss-2018b.eb (module: foss/2018b)
 * [ ] $CFGS/h/HDF5/HDF5-1.10.2-foss-2018b.eb (module: HDF5/1.10.2-foss-2018b)
 * [ ] $CFGS/n/netCDF/netCDF-4.6.1-foss-2018b.eb (module: netCDF/4.6.1-foss-2018b)
 * [ ] $CFGS/n/netCDF-Fortran/netCDF-Fortran-4.4.4-foss-2018b.eb (module: netCDF-
↪Fortran/4.4.4-foss-2018b)
 * [ ] $CFGS/w/WRF/WRF-4.0.2-foss-2018b-dmpar.eb (module: WRF/4.0.2-foss-2018b-dmpar)
== Temporary log file(s) /tmp/eb-eEnBF5/easybuild-pvrvNP.log* have been removed.
== Temporary directory /tmp/eb-eEnBF5 has been removed.
```

3.3 Installing a software stack

To make EasyBuild build and install WRF, including all of its dependencies, a **single command** is sufficient.

By using the `--robot`/`-r` (see *Enabling dependency resolution, –robot / -r and –robot-paths*) command line option, we enable dependency resolution such that the entire software stack is handled:

```
$ eb WRF-4.0.2-foss-2018b-dmpar.eb --robot
== temporary log file in case of crash /tmp/eb-LfQa8b/easybuild-TBXLTy.log
== resolving dependencies ...
== processing EasyBuild easyconfig /home/example/.local/easybuild/software/EasyBuild/
↪4.1.1/easybuild/easyconfigs/n/ncurses/ncurses-6.1-GCCcore-7.3.0.eb
== building and installing ncurses/6.1-GCCcore-7.3.0...
[...]
== building and installing tcsh/6.20.00-GCCcore-7.3.0...
[...]
== building and installing CMake/3.11.4-GCCcore-7.3.0...
[...]
== building and installing JasPer/2.0.14-GCCcore-7.3.0...
```

(continues on next page)

(continued from previous page)

```
[...]
== building and installing pkg-config/0.29.2-GCCcore-7.3.0...
[...]
== building and installing Doxygen/1.8.14-GCCcore-7.3.0...
[...]
== building and installing cURL/7.60.0-GCCcore-7.3.0...
[...]
== building and installing Szip/2.1.1-GCCcore-7.3.0...
[...]
== building and installing HDF5/1.10.2-foss-2018b...
[...]
== building and installing netCDF/4.6.1-foss-2018b...
[...]
== building and installing netCDF-Fortran/4.4.4-foss-2018b...
[...]
== building and installing WRF/4.0.2-foss-2018b-dmpar...
[...]
== Build succeeded for 12 out of 12
== Temporary log file(s) /tmp/eb-LfQa8b/easybuild-TBXLTy.log* have been removed.
== Temporary directory /tmp/eb-LfQa8b has been removed.
```

Once the installation has succeeded, modules will be available for WRF and all of its dependencies:

```
$ module load WRF
$ module list
$ module list

Currently Loaded Modules:
  1) EasyBuild/4.1.1                  13) gompi/2018b
  2) GCCcore/7.3.0                    14) FFTW/3.3.8-gompi-2018b
  3) zlib/1.2.11-GCCcore-7.3.0        15) ScaLAPACK/2.0.2-gompi-2018b-OpenBLAS-0.3.1
  4) binutils/2.30-GCCcore-7.3.0      16) foss/2018b
  5) GCC/7.3.0-2.30                   17) JasPer/2.0.14-GCCcore-7.3.0
  6) numactl/2.0.11-GCCcore-7.3.0     18) Szip/2.1.1-GCCcore-7.3.0
  7) XZ/5.2.4-GCCcore-7.3.0           19) HDF5/1.10.2-foss-2018b
  8) libxml2/2.9.8-GCCcore-7.3.0      20) cURL/7.60.0-GCCcore-7.3.0
  9) libpciaccess/0.14-GCCcore-7.3.0  21) netCDF/4.6.1-foss-2018b
 10) hwloc/1.11.10-GCCcore-7.3.0      22) netCDF-Fortran/4.4.4-foss-2018b
 11) OpenMPI/3.1.1-GCC-7.3.0-2.30     23) WRF/4.0.2-foss-2018b-dmpar
 12) OpenBLAS/0.3.1-GCC-7.3.0-2.30
```

For more information, see the other topics discussed in the documentation (see *Introductory topics*).

CHAPTER 4

Getting started

4.1 Installing EasyBuild

EasyBuild is Python software, so there are a couple of ways to install it.

We recommend installing EasyBuild using `pip`. This method is described at *Using pip to Install EasyBuild*.

It is also possible to install EasyBuild as a module. To do this, use the 3-step procedure outlined at *Installing EasyBuild with EasyBuild*.

Do take into account the required and optional dependencies (see *Requirements* and *Dependencies*).

Notes on other ways of installing EasyBuild are available under section *Alternative installation methods*.

Contents

4.1.1 Requirements

The only strict requirements are:

- a **GNU/Linux distribution** as operating system
- **Python**:
  - Python 2.7, or Python 3.x (>= 3.5). Since Python 2 is end-of-life (https://www.python.org/doc/sunset-python-2/) we recommend using Python 3 if it is available
  - **note**: only EasyBuild v4.0 (or newer) is compatible with Python 3, earlier EasyBuild releases require Python 2
  - no Python packages other than the ones included in the Python standard library are strictly required
    * **note**: only EasyBuild versions prior to v4.0 require `vsc-base` (& `vsc-install`), see also *Required Python packages for older EasyBuild versions*
  - for some specific features, additional Python packages are needed though, see *Optional Python packages*
- a **modules tool**: Tcl(/C) environment modules or Lmod
  - the actual module command/script (`modulecmd`, `modulecmd.tcl` or `lmod`) *must* be available via `$PATH`
  - see *Required modules tool* for more details

For more information on (optional) dependencies, see *Dependencies*.

4.1.2 Using pip to Install EasyBuild

Since EasyBuild is released as a Python package on PyPI (https://pypi.org/project/easybuild) you can install it using `pip`, the most commonly used tool for installing Python packages.

Install EasyBuild with:

```
pip install easybuild
```

You may need to tweak this command a bit, depending on your setup, see *Additional pip install options*.

Note: There are various other ways of installing Python packages, which we won't cover here. If you are familiar with other tools like `virtualenv` or `pipenv`, feel free to use those instead to install EasyBuild.

Sanity check

Compare the version of `eb`, the main EasyBuild command, with the version of the EasyBuild module that was installed. For example:

```
$ module load EasyBuild
$ module list

Currently Loaded Modules:
  1) EasyBuild/4.4.0

$ eb --version
This is EasyBuild 4.4.0 (framework: 4.4.0, easyblocks: 4.4.0) on host example.local
```

Tip: The Tcl-based or Lmod implementations of environment modules do their default sorting differently. The former will normally sort in the lexicographic order, while Lmod follows an approach that is closer to Python's construct `LooseVersion` way of ordering. Such aspects may make a big difference, if you have installed both versions 1.9.0 and 1.15.2, with respect to what is the version being loaded by default.

You can also run `eb --show-system-info` to see system information relevant to EasyBuild, or run''eb –show-config'' to see the default EasyBuild configuration (see also *Configuring EasyBuild*).

Updating an existing EasyBuild installation

To upgrade to a newer EasyBuild version than the one currently installed:

- `pip install --upgrade easybuild` will upgrade EasyBuild to the latest release.

Additional pip install options

For the `pip` install, you may wish to slightly change this command depending on the context and your personal preferences:

- to install EasyBuild *system-wide*, you can use `sudo` (if you have admin privileges):

  ```
  sudo pip install easybuild
  ```

- To install EasyBuild *in your personal home directory*, you can use the `--user` option:

  ```
  pip install --user easybuild
  ```

  This will result in an EasyBuild installation in `$HOME/.local/`.

- To install EasyBuild in a *specific directory* you can use the `--prefix` option:

  ```
  pip install --prefix _PREFIX_ easybuild
  ```

In this command, you should replace '_PREFIX_' with the location where you want to have EasyBuild installed (for example, `$HOME/tools` or `/tmp/$USER`).

Keep in mind that you may need to update your environment too when using `--user` or `--prefix`, see *Updating your environment*.

`pip` vs `pip3`

On systems where both Python 2 and Python 3 are installed you may also have different `pip` commands available. Or maybe `pip` is not available at all, and only "versioned" `pip` commands like `pip3` are available.

If you (only) have `pip3` available, you can replace `pip` with `pip3` in any of the `pip install` commands above:

```
pip3 install easybuild
```

If you want to ensure that you are using the `pip` installation that corresponds to the Python 3 installation that you intend to use, you can use `python3 -m pip` rather than `pip3`.

```
python3.6 -m pip install easybuild
```

Note that you may also need to instruct the `eb` command to use the correct Python version at runtime, via `$EB_PYTHON` (see *Setting $EB_PYTHON*).

Updating your environment

If you used the `--user` or `--prefix` option in the `pip install` command, or if you installed EasyBuild with a `pip` version that does not correspond to your default Python installation, you will need to update your environment to make EasyBuild ready for use. This is not required if you did a system-wide installation in a standard location with the default Python version.

Note: Keep in mind that you will have to make these environment changes again if you start a new shell session. To avoid this, you can update one of the shell startup scripts in your home directory (`.bashrc` for example).

Updating `$PATH`

Update the `$PATH` environment variable to make sure the `eb` command is available:

```
export PATH=_PREFIX_/bin:$PATH
```

Replace `_PREFIX_` in this command with the directory path where EasyBuild was installed into (use `$HOME/.local` if you used `pip install --user`).

This is not required if you installing EasyBuild in a standard system location.

You can check with the `which eb` command to determine whether or not you need to update the `$PATH` environment variable.

Updating `$PYTHONPATH`

If you installed EasyBuild to a non-standard location using `pip install --prefix`, you also need to update the Python search path environment variable `$PYTHONPATH` to instruct Python where it can find the EasyBuild Python packages.

This is not required if you used the `--user` option, since Python will automatically consider `$HOME/.local` when searching for installed Python packages, or if you installed EasyBuild in a standard system-wide location.

Update `$PYTHONPATH` by running a command like:

```
export PYTHONPATH=_PREFIX_/lib/pythonX.Y/site-packages:$PYTHONPATH
```

Here, you need to replace the `X` and `Y` with the major and minor version of your Python installation, which you can determine by running `python -V`. For example, if you are using Python 3.6, make sure you are using `/python3.6/` in the command to update `$PYTHONPATH`.

And of course, you again need to **replace '''_PREFIX_'''** with the installation prefix where EasyBuild was installed into.

For example:

```
# update $PYTHONPATH if EasyBuild was installed in $HOME/tools with Python 3.6
export PYTHONPATH=$HOME/tools/lib/python3.6/site-packages:$PYTHONPATH
```

Setting `$EB_PYTHON`

If you want to control which Python version is used to run EasyBuild, you can specify the name or the full path to the `python` command that should be used by the `eb` command via the `$EB_PYTHON` environment variable.

This may be required when you installing EasyBuild with a version of `pip` that does not correspond with the default Python version.

For example, to ensure that `eb` uses `python3.6`:

```
export EB_PYTHON=python3.6
```

Setting `$EB_VERBOSE`

To determine which `python` commands are being considered by the `eb` command, you can define the `$EB_VERBOSE` environment variable. For example:

```
$ EB_VERBOSE=1 eb --version
>> Considering 'python3.6'...
>> 'python3' version: 3.6.8, which matches Python 3 version requirement (>= 3.5)
>> Selected Python command: python3 (/usr/bin/python3.6)
>> python3.6 -m easybuild.main --version
This is EasyBuild 4.3.3 (framework: 4.3.3, easyblocks: 4.3.3) on host example
```

4.1.3 Installing EasyBuild with EasyBuild

If you prefer having EasyBuild available through an environment module file, you can consider installing EasyBuild with EasyBuild. This can be done in 3 steps:

- Step 1: Installing EasyBuild with `pip` into a temporary location (only needed if EasyBuild is not installed yet)
- Step 2: Using EasyBuild to install EasyBuild as a module
- Step 3: Loading the EasyBuild module

Step 1: Installing EasyBuild into a temporary location

If you don't have EasyBuild installed yet, you need to install it in a temporary location first. The recommended way of doing this is using *Using pip to Install EasyBuild*.

For example, to install EasyBuild into a subdirectory `/tmp/$USER` using the default Python 3 version:

```
# pick installation prefix, and install EasyBuild into it
export EB_TMPDIR=/tmp/$USER/eb_tmp
python3 -m pip install --ignore-installed --prefix $EB_TMPDIR easybuild

# update environment to use this temporary EasyBuild installation
export PATH=$EB_TMPDIR/bin:$PATH
export PYTHONPATH=$(/bin/ls -rtd -1 $EB_TMPDIR/lib*/python*/site-packages | tail -1):
↪$PYTHONPATH
export EB_PYTHON=python3
```

Step 2: Using EasyBuild to install EasyBuild

Once you have a working (recent) temporary EasyBuild installation, you can use it to install EasyBuild as a module. Usually this is done in the location where you would like to install other software too.

You can use the `eb --install-latest-eb-release` command for this, combined with the `--prefix` option to control which directories are used by EasyBuild for the installation.

For example, to install the latest version of EasyBuild as a module into `$HOME/easybuild`:

```
eb --install-latest-eb-release --prefix $HOME/easybuild
```

Note: You may see a harmless deprecation warning popping up when performing this installation, just ignore it.

Step 3: Loading the EasyBuild module

Once *Step 2: Using EasyBuild to install EasyBuild* is completed, you should be able to load the module that was generated alongside the EasyBuild installation. You will need to do this every time you start a new shell session.

First, make the module available by running the following command (which will update the module search path environment variable `$MODULEPATH`):

```
module use _PREFIX_/modules/all
```

Replace `_PREFIX_` with the path to the directory that you used when running *Step 2: Using EasyBuild to install EasyBuild* (for example, `$HOME/easybuild`).

Then, load the EasyBuild module to update your environment and make EasyBuild available for use:

```
module load EasyBuild
```

Note: Note that in this case, we don't need to make any changes to our environment for EasyBuild to work correctly. The environment module file that was generated by EasyBuild specifies all changes that need to be made.

4.1.4 Dependencies

EasyBuild has a couple of dependencies, some are optional.

Required dependencies

- a **GNU/Linux** distribution as operating system
  - some common shell tools are expected to be available, see *Required shell tools*
- Python:
  - Python 2.7, or Python 3.x (>= 3.5);
  - since Python 2 is end-of-life (https://www.python.org/doc/sunset-python-2/) we strongly recommend using Python 3 if it is available;
  - no third-party Python packages are strictly required (the Python standard library is sufficient);
  - for some *specific* EasyBuild features additional Python packages are required however, see *Optional Python packages*;
- a **modules tool**: Tcl(/C) environment modules or Lmod
  - the actual modules tool *must* be available via `$PATH`, see *Required modules tool*
- a C/C++ compiler (e.g., `gcc` and `g++`)
  - only required to build and install GCC with, or as a dependency for the Intel compilers, for example

Required shell tools

A couple of shell tools may be required, depending on the particular use case (in relative order of importance):

- shell builtin commands:
  - `type`, for inspecting the `module` function (if defined)
  - `ulimit`, for querying user limits
- tools for unpacking (source) archives:
  - commonly required: `tar`, `gunzip`, `bunzip2`
  - occasionally required: `unzip`, `unxz`
- `patch`, for applying patch files to unpacked sources (relatively common)
- `rpm` or `dpkg`, for querying OS dependencies (only needed occasionally)
- `locate`, only as a (poor mans) fallback to `rpm`/`dpkg` (rarely needed)
- `sysctl`, for querying system characteristics (only required on non-Linux systems)

Required modules tool

EasyBuild not only generates module files to be used along with the software it installs, it also depends on the generated modules, mainly for resolving dependencies. Hence, a modules tool must be available to consume module files with.

Supported module tools:

- Tcl/C environment-modules (version >= 3.2.10)

- Tcl-only variant of environment modules
- Lmod (version >= 6.5.1), *highly recommended*

Note: The path to the actual modules tool binary/script used *must* be included in `$PATH`, to make it readily available to EasyBuild.

- for Tcl/C environment modules: `modulecmd`
- for Tcl-only environment modules: `modulecmd.tcl`
- for Lmod: `lmod`

The path where the modules tool binary/script is located can be determined via the definition of the `module` function; for example, using `type module` or `type -f module`.

Note: For Lmod specifically, EasyBuild will try to fall back to finding the `lmod` binary via the `$LMOD_CMD` environment variable, in case `lmod` is not available in `$PATH`.

In EasyBuild versions *prior* to 2.1.1, the path specified by `$LMOD_CMD` was (erroneously) preferred over the (first) `lmod` binary available via `$PATH`.

Additional notes:

- Tcl(/C) environment-modules requires Tcl to be installed (with header files and development libraries)
- Lmod requires Lua and a couple of non-standard Lua libraries (`lua-posix`, `lua-filesystem`) to be available
  - Tcl (`tclsh`) must also be available for Lmod to support module files in `Tcl` syntax
- a guide to installing Tcl/C environment modules without having root permissions is available at *Installing environment modules without root permissions*.
- a guide to installing Lmod without having root permissions is available at *Installing Lmod without root permissions*.

Required Python packages

Since EasyBuild v4.0, *no* Python packages outside of the Python standard library are required.

Required Python packages for older EasyBuild versions

For EasyBuild versions prior to version 4.0, a couple of additional Python packages are required:

- `setuptools`: used to define the `easybuild` namespace across different directories
  - available at https://pypi.python.org/pypi/setuptools
  - must be version 0.6 or more recent
  - strictly required since EasyBuild v2.7.0
- `vsc-install`: provides setuptools functions and support for unit test suites for Python tools
  - also required to install `vsc-base` (see below)
  - available at https://pypi.python.org/pypi/vsc-install

  - the required version depends primarily on the `vsc-base` version
- `vsc-base`: a Python library providing the `fancylogger` and `generaloption` Python modules
  - available at https://pypi.python.org/pypi/vsc-base and https://github.com/hpcugent/vsc-base
  - the required version of `vsc-base` depends on the EasyBuild version

---

Note: `vsc-base` is installed automatically along with EasyBuild 3.x, if an installation procedure is used that consumes the `setup.py` script that comes with the EasyBuild framework (e.g., EasyBuild or the EasyBuild bootstrap script, `pip`, `easy_install`, ...)

Other Python packages are optional dependencies, see *Optional Python packages*.

Optional dependencies

Some dependencies are optional and are only required to support certain features.

Optional Python packages

- GC3Pie, only needed when using *GC3Pie* as a backend for `--job`, see also *Submitting jobs using –job*;
- GitPython, only needed if EasyBuild is hosted in a git repository or if you're using a git repository for easyconfig files (.eb);
- graphviz for Python, only needed for building nice-looking dependency graphs using `--dep-graph *.pdf` / `*.png`;
- keyring, only needed for securely storing a GitHub token (see *Integration with GitHub*);
- pycodestyle, only required for `--check-style` and `--check-contrib`;
- pysvn, only needed if you're using an SVN repository for easyconfig files;
- python-graph-dot, only needed for building nice-looking dependency graphs using `--dep-graph *.dot`

4.1.5 Sources

EasyBuild is split up into three different packages, which are available from the Python Package Index (PyPi):

- easybuild-framework - the EasyBuild framework, which includes the easybuild.framework and easybuild.tools Python packages that provide general support for building and installing software
- easybuild-easyblocks - a collection of easyblocks that implement support for building and installing (collections of) software packages
- easybuild-easyconfigs - a collection of example easyconfig files that specify which software to build, and using which build options; these easyconfigs will be well tested with the latest compatible versions of the easybuild-framework and easybuild-easyblocks packages

Next to these packages, a meta-package named easybuild is also available on PyPi, in order to easily install the full EasyBuild distribution.

The source code for these packages is also available on GitHub:

- easybuild-framework git repository
- easybuild-easyblocks git repository

- easybuild-easyconfigs git repository
- the main EasyBuild repository mainly hosts *this* EasyBuild documentation

4.1.6 In case of installation issues...

Should the installation of EasyBuild fail for you, **'please open an issue'_** to report the problems you're running into.

How to collect info in case sanity checks fail or there is another issue

In order to get a better understanding in which kind of environment you are using the bootstrap script, please copy-paste the commands below and provide the output in your problem report. **Do not worry if some of these commands fail or spit out error messages.**

```
python -V
type module
type -f module
module --version
module av EasyBuild
which -a eb
eb --version
```

4.2 Configuring EasyBuild

This page discusses the recommended style of configuring EasyBuild, which is supported since EasyBuild v1.3.0.

A demo on configuring EasyBuild is available here.

Contents

4.2.1 Supported configuration types

Configuring EasyBuild can be done by:

- using `eb` with **command line arguments**

- setting **environment variables** (`$EASYBUILD_...`)
- providing one or more **configuration files**

Of course, combining any of these types of configuration works too (and is even fairly common).

The order of preference for the different configuration types is as listed above, that is:

- environment variables override the corresponding entries in the configuration file
- command line arguments in turn override the corresponding environment variables *and* matching entries in the configuration file

Consistentency across supported configuration types

Note that the various available configuration options are handled **consistently** across the supported configuration types.

For example: to configure EasyBuild to use Lmod as modules tool, the following alternatives are available:

- configuration file entry (key-value assignment):

```
[config]
modules-tool = Lmod
```

- environment variable (upper case, `EASYBUILD_` prefix, -'s becomes _'s):

```
$ export EASYBUILD_MODULES_TOOL=Lmod
```

- command line argument (long options preceded by `--` and (optionally) using =):

```
$ eb --modules-tool=Lmod
```

or

```
$ eb --modules-tool Lmod
```

For more details w.r.t. each of the supported configuration types, see below.

Configuration file(s)

List of used configuration files

The list of configuration files that will be used by EasyBuild is determined in the following order of preference:

- the path(s) specified via the **command line argument** `--configfiles`
- the path(s) specified via the `$EASYBUILD_CONFIGFILES` **environment variable**
- the **default paths** for EasyBuild configuration files

Default configuration files

By default, EasyBuild will use existing configuration files at the following paths:

- `$dir/easybuild.d/*.cfg`, for each directory `$dir` listed in `$XDG_CONFIG_DIRS` (where `$XDG_CONFIG_DIRS` defaults to `/etc`)
- `$XDG_CONFIG_HOME/easybuild/config.cfg` (where `$XDG_CONFIG_HOME` defaults to `$HOME/.config`)

Hence, if `$XDG_CONFIG_HOME` and `$XDG_CONFIG_DIRS` are not defined, EasyBuild will only consider default configuration files at `/etc/easybuild.d/*.cfg` and `$HOME/.config/easybuild/config.cfg`.

The configuration file located in `$XDG_CONFIG_HOME` will be listed *after* the ones obtained via `$XDG_CONFIG_DIRS`, such that user-defined configuration settings have preference over system defaults.

A detailed overview of the list of default configuration files is available via `eb --show-default-configfiles` (available since EasyBuild v2.1.0). For example:

```
$ XDG_CONFIG_DIRS=/tmp/etc:/tmp/moreetc eb --show-default-configfiles
Default list of configuration files:

[with $XDG_CONFIG_HOME: (not set), $XDG_CONFIG_DIRS: /tmp/etc:/tmp/moreetc]

* user-level: ${XDG_CONFIG_HOME:-$HOME/.config}/easybuild/config.cfg
  -> /home/example/.config/easybuild/config.cfg => found
* system-level: ${XDG_CONFIG_DIRS:-/etc}/easybuild.d/*.cfg
  -> {/tmp/etc, /tmp/moreetc}/easybuild.d/*.cfg => /tmp/etc/easybuild.d/config.cfg, /
↪tmp/moreetc/easybuild.d/bar.cfg, /tmp/moreetc/easybuild.d/foo.cfg

Default list of existing configuration files (4): /tmp/etc/easybuild.d/config.cfg, /
↪tmp/moreetc/easybuild.d/bar.cfg, /tmp/moreetc/easybuild.d/foo.cfg, /home/example/.
↪config/easybuild/config.cfg
```

Multiple configuration files

If multiple configuration files are listed via a mechanism listed above, the settings in the last configuration file have preference over the others.

Each available configuration file will be used, and the configuration settings specified in these files will be retained according to the order of preference as indicated above: settings in configuration files specified via `--configfiles` override those in configuration files specified via `$EASYBUILD_CONFIGFILES`, which in turns override settings in default configuration files.

Ignored configuration files

On top of this, the `--ignoreconfigfiles` configuration option allows to specify configuration files that should be *ignored* by EasyBuild (regardless of whether they are specified via any of the options above).

Configuration file format

The EasyBuild configuration file follows the default Python configuration format as parsed by the `configparser` module (see http://docs.python.org/2/library/configparser.html).

Configuration files are organized in sections, the section name for a particular configuration setting is indicated in the output of `eb --help`. Some examples sections are: `MAIN`, `basic`, `config`, `informative`, `override`, `regtest`, `software`, `unittest`, etc.

Sections are indicated by specifying the section name in square brackets on a dedicated line, e.g., `[basic]`.

Configuration settings are specified in a `key = value` or `key: value` format, **without using quotes for string-like values**. For boolean configuration settings, values that evaluated to `True` (e.g., `true`, `1`, ...) are all equivalent to enabling the setting.

Comment lines start with a hash character # (just like in Python code).

An example configuration file that should make everything clear is shown below.

```
[basic]
# always enable logging to stdout
logtostdout = true
[config]
# use Lmod as modules tool
modules-tool: Lmod
# use different default installation path
prefix=/home/you/work/easybuild/
```

Templates and constants supported in configuration files

Two types of template values `%(...)s` are supported in configuration files:

- for configuration options defined in the configuration file (and only those)
  - *syntax:* `%(opt)s`, i.e., using the (lowercase) name of the configuration option
- for the default value of selected configuration options (see `eb --avail-cfgfile-constants`)
  - *syntax:* `%(DEFAULT_OPT)s`, i.e., using the uppercase name of the configuration option and prefixed with `DEFAULT_`

Note: These template values are only supported in configuration files, *not* in environment variable values or command line option values.

Note: Using an unknown template value, i.e. either one for a configuration option that was not defined in the configuration file, or a non-existing one for a particular default value, will result in an error like: `ConfigParser.InterpolationMissingOptionError: Bad value substitution`.

Example

To include both the (custom) location for the easyconfigs archive repository and the default list of robot search paths in the active robot search path, the following configuration file entry can be used, featuring the template for the `repositorypath` configuration option and the provided `DEFAULT_ROBOT_PATHS` constant:

```
[basic]
repositorypath = /home/example/easybuild/easyconfigs_archive
robot-paths = %(repositorypath)s:%(DEFAULT_ROBOT_PATHS)s
```

See also *Controlling the robot search path*.

Generating a template configuration file

Since EasyBuild v1.10, a command line option `--configihelp` is available that prints out the help text as an annotated configuration file. This can be used as an empty template configuration file:

```
$ mkdir -p $HOME/.config/easybuild
$ eb --configihelp > $HOME/.config/easybuild/config.cfg
```

```
$ head $HOME/.easybuild/config.cfg
[MAIN]
# Enable debug log mode (def False)
#debug=
# Enable info log mode (def False)
#info=
# Enable info quiet/warning mode (def False)
#quiet=

[basic]
# Print build overview incl. dependencies (full paths) (def False)
```

Environment variables

All configuration settings listed as long options in `eb --help` can also be specified via `EASYBUILD_`-prefixed environment variables.

Configuration settings specified this way always override the corresponding setting specified in a configuration file.

For example, to enable debug logging using an environment variable:

```
$ export EASYBUILD_DEBUG=1
```

More examples of using environment variables to configure EasyBuild are shown in the sections below.

Tip: Any configuration option of EasyBuild which can be tuned by command line or via the configuration file, can also be tuned via a corresponding environment variable.

Note: If any `$EASYBUILD`-prefixed environment variables are defined that do not correspond to a known configuration option, EasyBuild will report an error message and exit.

Command line arguments

The configuration type with the highest precedence are the `eb` command line arguments, which override settings specified through environment variables or in configuration files.

For some configuration options, both short and long command line arguments are available (see `eb --help`); the long options indicate how the configuration setting should be specified in a configuration file or via an environment variable (`$EASYBUILD_<LONGOPTION>`).

For boolean configuration settings, both the `--<option>` and `--disable-<option>` variants are always available.

Examples (more below):

- enable debug logging (long option) and logging to stdout (short option)

```
$ eb --debug -l ...
```

- use `/dev/shm` as build path, install to temporary install path, disable debug logging

```
$ eb --buildpath=/dev/shm --installpath=/tmp/$USER --disable-debug ...
```

4.2.2 Overview of current configuration (`--show-config`, `--show-full-config`)

To get an overview of the current EasyBuild configuration across all configuration types, you can use `eb --show-config`.

The output will specify:

- any configuration setting for which the current value is different from the default value
- a couple of selected important configuration settings (even if they are still set to the default value), i.e.:
  - build path (see *Build path (–buildpath)*)
  - install path (see *Software and modules install path (–installpath, –installpath-software, –installpath-modules)*)
  - path to easyconfigs repository (see *Easyconfigs repository (–repository, –repositorypath)*)
  - the robot search path (see *Searching for easyconfigs: the robot search path*)
  - source path (see *Source path (–sourcepath)*)
- through which configuration type each setting was defined
  - i.e., default value, configuration file, environment variable or command line argument

Example output:

```
$ cat $HOME/.config/easybuild/config.cfg
[config]
buildpath = /tmp/eb-build

$ export EASYBUILD_MODULES_TOOL=Lmod
$ export EASYBUILD_OPTARCH=''

$ eb --show-config --installpath=$HOME/apps --job-cores=4
#
# Current EasyBuild configuration
# (C: command line argument, D: default value, E: environment variable, F:␣
↪configuration file)
#
buildpath      (F) = /tmp/eb-build
installpath    (C) = /Users/example/apps
job-cores      (C) = 4
modules-tool   (E) = Lmod
optarch        (E) = ''
repositorypath (D) = /Users/example/.local/easybuild/ebfiles_repo
robot-paths    (D) = /Users/example/easybuild-easyconfigs/easybuild/easyconfigs
sourcepath     (D) = /Users/example/.local/easybuild/sources
```

For a full overview of the current configuration, including *all* configuration settings, see `eb --show-full-config`.

4.2.3 Available configuration settings

To obtain a full and up-to-date list of available configuration settings, see `eb --help`. We refrain from listing all available configuration settings here, to avoid outdated documentation.

A couple of selected configuration settings are discussed below, in particular the mandatory settings.

Mandatory configuration settings

A handful of configuration settings are **mandatory**, and should be provided using one of the supported configuration types.

The following configuration settings are currently mandatory (more details in the sections below):

- *Source path (–sourcepath)*
- *Build path (–buildpath)*
- *Software and modules install path (–installpath, –installpath-software, –installpath-modules)*
- *Easyconfigs repository (–repository, –repositorypath)*
- *Logfile format (–logfile-format)*

If any of these configuration settings is not provided in one way or another, EasyBuild will complain and exit.

In practice, all of these have reasonable defaults (see `eb --help` for the default settings).

Note: The mandatory path-related options can be tweaked collectively via `--prefix`, see *Overall prefix path (–prefix)* for more information.

Source path (`--sourcepath`)

default: `$HOME/.local/easybuild/sources/` (determined via *Overall prefix path (–prefix)*)

The `sourcepath` configuration setting specifies the parent path of the directory in which EasyBuild looks for software source and install files.

Looking for the files specified via the `sources` parameter in the .eb easyconfig file is done in the following order of preference:

- `<sourcepath>/<name>`: a subdirectory determined by the name of the software package
- `<sourcepath>/<letter>/<name>`: in the style of the `easyblocks`/`easyconfigs` directories: in a subdirectory determined by the first letter (in lower case) of the software package and by its full `name`
- `<sourcepath>`: directly in the source path

Note that these locations are also used when EasyBuild looks for patch files in addition to the various `easybuild/easyconfigs` directories that are listed in the $PYTHONPATH.

You can specify multiple paths, separated with `:`, in which EasyBuild will look for sources, but only the first one will be used for downloading, so one needs to make sure at least the first path is writable by the user invoking `eb`.

Build path (`--buildpath`)

default: `$HOME/.local/easybuild/build/` (determined via *Overall prefix path (–prefix)*)

The `buildpath` configuration setting specifies the parent path of the (temporary) directories in which EasyBuild builds its software packages.

Each software package is (by default) built in a subdirectory of the specified `buildpath` under `<name>/<version>/<toolchain><versionsuffix>`.

Note that the build directories are emptied and removed by EasyBuild when the installation is completed (by default).

Tip: Using `/dev/shm` as build path can significantly speed up builds, if it is available and provides a sufficient amount of space. Setting up the variable `EASYBUILD_BUILDPATH` in your shell startup files makes this default. However be aware that, fi., two parallel GCC builds may fill up `/dev/shm` !

Software and modules install path (`--installpath`, `--installpath-software`, `--installpath-modules`)

defaults:

- *software install path:* `$HOME/.local/easybuild/software` (determined via *Overall prefix path (–prefix)* and `--subdir-software`)
- *modules install path:* `$HOME/.local/easybuild/modules/all` (determined via *Overall prefix path (–prefix)*, `--subdir-modules` and `--suffix-modules-path`)

There are several ways in which the software and modules install path used by EasyBuild can be configured:

- using the direct configuration options `--installpath-software` and `--installpath-modules` (see below)
- via the parent install path configuration option `--installpath` (see below)
- via the overall prefix path configuration option `--prefix` (see *Overall prefix path (–prefix)*)

Direct options: `--installpath-software` and `--installpath-modules`

default: *(no default specified)*

The `--installpath-software` and `--installpath-modules` configuration options (available since EasyBuild v2.1.0) allow to directly specify the software and modules install paths, respectively.

These configuration options have precedence over all of the other configuration options that relate to specifying the install path for software and/or modules (see below).

Parent install path: `--installpath`

default: *(no default specified)*

The `--installpath` configuration option specifies the *parent* path of the directories in which EasyBuild should install software packages and the corresponding module files.

The install path for software and modules specifically is determined by combining `--installpath` with `--subdir-software`, and combining `--installpath` with `--subdir-modules` and `--suffix-modules-path`, respectively.

For more information on these companion configuration options, see *Software and modules install path subdirectories (–subdir-software, –subdir-modules, –suffix-modules-path)*.

Full install path for software and module file

The full software and module install paths for a particular software package are determined by the active module naming scheme along with the general software and modules install paths specified by the EasyBuild configuration.

Both the software itself and the corresponding module file will be installed in a subdirectory of the corresponding install path named according to the active module naming scheme (default format: `<name>/<version>-<toolchain><versionsuffix>`). Additionally, symlinks to the actual module file are installed in a subdirectory of the modules install path named according to the value of the `moduleclass` easyconfig parameter.

For more information on the module naming scheme used by EasyBuild, see *Active module naming scheme (–module-naming-scheme)*.

Updating `$MODULEPATH`

To make the modules generated by EasyBuild available, the `$MODULEPATH` environment variable must be updated to include the modules install path.

The recommended way to do this is to use the `module use` command. For example:

```
$ eb --installpath=$HOME/easybuild
$ module use $HOME/easybuild/modules/all
```

It is probably a good idea to add this to your (favourite) shell `.rc` file, e.g., `~/.bashrc`, and/or the `~/.profile` login scripts, so you do not need to adjust `$MODULEPATH` every time you start a new session.

Note: Updating `$MODULEPATH` is not required for EasyBuild itself, since `eb` updates `$MODULEPATH` itself at runtime according to the modules install path it is configured with.

Easyconfigs repository (`--repository`, `--repositorypath`)

default: `FileRepository` at `$HOME/.local/easybuild/ebfiles_repo` (determined via *Overall prefix path (–prefix)*)

EasyBuild has support for archiving (tested) `.eb` easyconfig files. After successfully installing a software package using EasyBuild, the corresponding `.eb` file is uploaded to a repository defined by the `repository` and `repositorypath` configuration settings.

Currently, EasyBuild supports the following repository types (see also `eb --avail-repositories`):

- `FileRepository('path', 'subdir')`: a plain flat file repository; `path` is the path where files will be stored, `subdir` is an *optional* subdirectory of that path where the files should be stored
- `GitRepository('path', 'subdir/in/repo'`: a *non-empty* **bare** git repository (created with `git init --bare` or `git clone --bare`); `path` is the path to the git repository (can also be a URL); `subdir/in/repo` is optional, and specifies a subdirectory of the repository where files should be stored in
- `SvnRepository('path', 'subdir/in/repo')`: an SVN repository; `path` contains the subversion repository location (directory or URL), the optional second value specifies a subdirectory in the repository

You need to set the `repository` setting inside a configuration file like this:

```
[config]
repository = FileRepository
repositorypath = <path>
```

Or, optionally an extra argument representing a subdirectory can be specified, e.g.:

```
$ export EASYBUILD_REPOSITORY=GitRepository
$ export EASYBUILD_REPOSITORYPATH=<path>,<subdir>
```

You do not have to worry about importing these classes, EasyBuild will make them available to the configuration file.

Using `git` requires the `GitPython` Python modules, using `svn` requires the `pysvn` Python module (see *Dependencies*).

If access to the easyconfigs repository fails for some reason (e.g., no network or a missing required Python module), EasyBuild will issue a warning. The software package will still be installed, but the (successful) easyconfig will not be automatically added to the archive (i.e., it is not considered a fatal error).

Logfile format (`--logfile-format`)

default: `easybuild, easybuild-%(name)s-%(version)s-%(date)s.%(time)s.log`

The *logfile format* configuration setting contains a tuple specifying a log directory name and a template log file name. In both of these values, using the following string templates is supported:

- `%(name)s`: the name of the software package to install
- `%(version)s`: the version of the software package to install
- `%(date)s`: the date on which the installation was performed (in `YYYYMMDD` format, e.g. `20120324`)
- `%(time)s`: the time at which the installation was started (in `HHMMSS` format, e.g. `214359`)

Note: Because templating is supported in configuration files themselves (see *Templates and constants supported in configuration files*), the '`%`' character in these template values must be escaped when used in a configuration file (and only then), e.g., '`%%(name)s`'. Without escaping, an error like `InterpolationMissingOptionError: Bad value substitution` will be thrown by `ConfigParser`.

For example, configuring EasyBuild to generate a log file mentioning only the software name in a directory named `easybuild` can be done via the `--logfile-format` command line option:

```
eb --logfile-format="easybuild,easybuild-%(name)s.log" ...
```

or the `$EASYBUILD_LOGFILE_FORMAT` environment variable:

```
export EASYBUILD_LOGFILE_FORMAT="easybuild,easybuild-%(name)s.log"
```

or by including the following in an EasyBuild configuration file (note the use of '`%%`' to escape the `name` template value here):

```
logfile-format = easybuild,easybuild-%%(name)s.log
```

Optional configuration settings

The subsections below discuss a couple of commonly used optional configuration settings.

Overall prefix path (`--prefix`)

default: `$HOME/.local/easybuild`

The overall prefix path used by EasyBuild can be specified using the `--prefix` configuration option.

This affects the default value of several configuration options:

- source path (see *Source path (–sourcepath)*)
- build path (see *Build path (–buildpath)*)
- software and modules install path (see *Software and modules install path (–installpath, –installpath-software, –installpath-modules)*)
- easyconfigs repository path (see *Easyconfigs repository (–repository, –repositorypath)*)
- package path (see *Configuration options*)
- container path (see *Location for generated container recipes & images (–containerpath)*)

Software and modules install path subdirectories (`--subdir-software`, `--subdir-modules`, `--suffix-modules-path`)

defaults:

- *software install path subdirectory* (`--subdir-software`): `software`
- *modules install path subdirectory* (`--subdir-modules`): `modules`
- *modules install path suffix* (`--suffix-modules-path`): `all`

The subdirectories for the software and modules install paths (relative to `--installpath`, see *Software and modules install path (–installpath, –installpath-software, –installpath-modules)*) can be specified using the corresponding dedicated configuration options (available since EasyBuild v1.14.0).

For example:

```
$ export EASYBUILD_SUBDIR_SOFTWARE=installs
$ eb --installpath=$HOME/easybuild --subdir-modules=module_files ...
```

Modules tool (`--modules-tool`)

default: `Lmod`

Specifying the modules tool that should be used by EasyBuild can be done using the `modules-tool` configuration setting. A list of supported modules tools can be obtained using `eb --avail-modules-tools`.

Currently, the following modules tools are supported:

- `Lmod` *(default)*: Lmod, an modern alternative to environment modules, written in Lua (`lmod`)
- `EnvironmentModules`: modern Tcl-only version of environment modules (4.x) (`modulecmd.tcl`)
- `EnvironmentModulesC`: Tcl/C version of environment modules, usually version 3.2.10 (`modulecmd`)
- `EnvironmentModulesTcl`: (ancient) Tcl-only version of environment modules (`modulecmd.tcl`)

You can determine which modules tool you are using by checking the output of `type -f module` (in a `bash` shell), or `alias module` (in a `tcsh` shell).

The actual module command (i.e., `modulecmd`, `modulecmd.tcl`, `lmod`, ...) must be available via `$PATH` (which is not standard), except when using Lmod (in that case the `lmod` binary can also be located via `$LMOD_CMD`).

For example, to indicate that EasyBuild should be using `Lmod` as modules tool:

```
$ eb --modules-tool=Lmod ...
```

Active module naming scheme (`--module-naming-scheme`)

default: `EasyBuildModuleNamingScheme`

The module naming scheme that should be used by EasyBuild can be specified using the `module-naming-scheme` configuration setting.

```
$ eb --module-naming-scheme=HierarchicalMNS ...
```

For more details, see the dedicated page: https://github.com/easybuilders/easybuild/wiki/Using-a-custom-module-naming-scheme .

Module files syntax (`--module-syntax`)

default: `Lua`

supported since: EasyBuild v2.1

The syntax to use for generated module files can be specified using the `--module-syntax` configuration setting.

Possible values are:

- `Lua`: generate module files in Lua syntax
  - this requires the use of Lmod as a modules tool to consume the module files (see *Modules tool (–modules-tool)*)
  - module file names will have the `.lua` extension
- `Tcl`: generate module files in Tcl syntax
  - Tcl module files can be consumed by all supported modules tools
  - module files will contain a header string `#%Module` indicating that they are composed in Tcl syntax

---

Note: Lmod is able to deal with having module files in place in both Tcl and Lua syntax. When a module file in Lua syntax (i.e., with a `.lua` file name extension) is available, a Tcl module file with the same name will be ignored. The Tcl-based environment modules tool will simply ignore module files in Lua syntax, since they do not contain the header string that is included in Tcl module files.

Note: Using module files in Lua syntax has the advantage that Lmod does not need to translate from Lua to Tcl internally when processing the module files, which benefits responsiveness of Lmod when used interactively by users. In terms of Lmod-specific aspects of module files, the syntax of the module file does *not* matter; Lmod-specific statements supported by EasyBuild can be included in Tcl module files as well, by guarding them by a condition that only evaluates positively when Lmod is consuming the module file, i.e. '`if { [ string match "*tcl2lua.tcl" $env(_) ] } { ... }`'. Only conditional load statements like '`load(atleast("gcc","4.8"))`' can only be used in Lua module files.

4.3 Common toolchains

This page documents the concept of *common toolchains* in the EasyBuild community; for a more general definition of what (compiler) toolchains are, see *Toolchains*.

Contents

4.3.1 Definition and motivation

Picking a *compiler toolchain* to use is one of the first things you (need to) do when starting to use EasyBuild. This can be a daunting task, since a whole bunch of toolchains and different toolchain versions are readily available in EasyBuild. It may be difficult to determine which toolchain would be most rewarding to use, in terms of stability, performance of the resulting binaries and readily available easyconfig files.

In an attempt to focus the effort of the EasyBuild community, the concept of so-called *common toolchains* was introduced.

The idea is to compose and maintain a limited set of specific compiler toolchains, and try and convince many HPC sites to employ these toolchains. This helps in assuring stability of these toolchains w.r.t. which software can be built (correctly) with them, since they get significantly more testing. In addition, the expectation/hope is that more easyconfigs are contributed back to the central easyconfigs repository (https://github.com/easybuilders/easybuild-easyconfigs), resulting in a wide range of readily available easyconfig files using the common toolchains.

The intention is to revise/update the definitions of the common toolchains regularly (see *Update cycle for common toolchains*), which again can be a joint effort that benefits many HPC sites.

Currently, two different common toolchains are being maintained: `foss` and `intel`; see below for more details, and also *Overview of common toolchains*.

`foss` toolchain

The `foss` common compiler toolchain consists entirely of open source software (hence the name, derived from the common term 'FOSS', which is short for "Free and Open Source Software").

This toolchain consists of:

- binutils (https://www.gnu.org/software/binutils/)
- the GNU Compiler Collection (GCC, https://gcc.gnu.org/), i.e. `gcc` (C), `g++` (C++) and `gfortran` (Fortran)
- the Open MPI library (https://www.open-mpi.org/)
- the OpenBLAS (http://www.openblas.net/) + LAPACK (http://netlib.org/lapack) libraries (for `foss` < 2021a)
- the FlexiBLAS library (https://www.mpi-magdeburg.mpg.de/projects/flexiblas), with OpenBLAS + LAPACK as backend (for `foss` >= 2021a)
- the ScaLAPACK (http://netlib.org/scalapack) library is also included
- the FFTW library (http://fftw.org/)

Note: The toolchain name was deliberately chosen to be generic, to allow for swapping any of the toolchain components with a better (open source) alternative in the future, should the need or opportunity arise.

`intel` toolchain

The `intel` common compiler toolchain consists of the Intel compilers and libraries, i.e.:

- the Intel C/C++/Fortran compilers (https://software.intel.com/en-us/intel-compilers), i.e. `icc`, `icpc` and `ifort`,
  - binutils and GCC, which serve as a base for the Intel compilers, are also included
- the Intel MPI library (https://software.intel.com/en-us/intel-mpi-library)
- the Intel Math Kernel Library (MKL, https://software.intel.com/en-us/intel-mkl) for BLAS/LAPACK/FFT functionality

Note: This compiler toolchain includes licensed software; valid licenses must be available to install and use it.

4.3.2 Versioning scheme for common toolchains

The common toolchains follow a specific versioning scheme, which takes the *6-month update cycle* into account.

Each revision of the common toolchains is versioned as the *year* in which it was defined, plus an additional 'a' or 'b' to indicate whether the toolchain was defined at the start of the year ('a') or halfway through the year ('b'); in short, the common toolchains are versioned as `<year>{a,b}`.

For example, `foss/2016b` is a revision of the `foss` that was composed mid-2016.

A full historic overview of the `foss` and `intel` common toolchains is available in *Overview of common toolchains*.

Note: Next to the versions that follow the `<year>{a,b}` versioning scheme, additional versions of the `foss` and `intel` versions are available as well.

These versions are **not** considered to be part of the series of common toolchains (even though they consists of the same toolchain components). These versions may be site-specific, or compositions that were put in place to evaluate a potential future common toolchain.

Typically, they are versioned as `<year>.<month>`, indicating when the most recent component included was released, or when that particular toolchain composition was defined.

4.3.3 Update cycle for common toolchains

The intention is to revise and update the common toolchains every 6 months: once in late December/early January (version `<year>a`), and once in late June/early July (version `<year>b`).

This is meant be to be a community effort, in the sense that a proposal for an updated composition is shared and discussed before it is set in stone.

Recent versions of each of the toolchain components are considered, taking into account stability, performance improvements, added features, known bugs/issues and experiences with those versions.

Moreover, the proposed toolchain compositions are tested extensively, typically by rebuilding all available easyconfigs that are using the most recent revision of the common toolchains at that time.

4.3.4 Overview of common toolchains

Component versions in `foss` toolchain

foss	*date*	*binutils*	*GCC*	*Open MPI*	*FlexiBLAS*	*OpenBLAS*	*LAPACK*	*ScaLAPACK*	*FFTW*
2014b	Jul '14	*(none)*	4.8.3	1.8.1	*(none)*	0.2.9	3.5.0	2.0.2	3.3.4
2015a	Jan '15	*(none)*	4.9.2	1.8.4	*(none)*	0.2.13	3.5.0	2.0.2	3.3.4
2015b	Jul '15	2.25	4.9.3	1.8.8	*(none)*	0.2.14	3.5.0	2.0.2	3.3.4
2016a	Jan '16	2.25	4.9.3	1.10.2	*(none)*	0.2.15	3.6.0	2.0.2	3.3.4
2016b	Jul '16	2.26	5.4.0	1.10.3	*(none)*	0.2.18	3.6.1	2.0.2	3.3.4
2017a	Jan '17	2.27	6.3.0	2.0.2	*(none)*	0.2.19	3.7.0	2.0.2	3.3.6(-pl2)
2017b	Jul '17	2.28	6.4.0	2.1.1	*(none)*	0.2.20*	(incl. with OpenBLAS)	2.0.2	3.3.6(-pl2)
2018a	Jan '18	2.28	6.4.0	2.1.2	*(none)*	0.2.20*	(incl. with OpenBLAS)	2.0.2	3.3.7
2018b	Jul '18	2.30	7.3.0	3.1.1	*(none)*	0.3.1	(incl. with OpenBLAS)	2.0.2	3.3.8
2019a	Jan '19	2.31.1	8.2.0	3.1.3	*(none)*	0.3.5	(incl. with OpenBLAS)	2.0.2	3.3.8
2019b	Sept '19	2.32	8.3.0	3.1.4	*(none)*	0.3.7	(incl. with OpenBLAS)	2.0.2	3.3.8
2020a	May '20	2.34	9.3.0	4.0.3	*(none)*	0.3.9	(incl. with OpenBLAS)	2.1.0	3.3.8
2020b	Nov '20	2.35	10.2.0	4.0.5	*(none)*	0.3.12	(incl. with OpenBLAS)	2.1.0	3.3.8
2021a	May '21	2.36.1	10.3.0	4.1.1	3.0.4	0.3.15	(incl. with OpenBLAS)	2.1.0	3.3.9

*(components marked with * were patched)*

Component versions in `intel` toolchain

intel	*date*	*binutils*	*GCC*	*Intel compilers*	*Intel MPI*	*Intel MKL*
2014b (*)	Jul '14	'*(none)*	4.8.3	2013.5.192	4.1.3.049	11.1.2.144
2015a (*)	Jan '15	'*(none)*	4.9.2	2015.1.133	5.0.2.044	11.2.1.133
2015b (*)	Jul '15	2.25	4.9.3	2015.3.187	5.0.3.048	11.2.3.187
2016a	Jan '16	2.26	4.9.3	2016.1.150	5.1.2.150	11.3.1.150
2016b	Jul '16	2.26	5.4.0	2016.3.210	5.1.3.181	11.3.3.210
2017a	Jan '17	2.27	6.3.0	2017.1.132	2017.1.132	2017.1.132
2017b	Jul '17	2.28	6.4.0	2017.4.196	2017.3.196	2017.3.196
2018a	Jan '18	2.28	6.4.0	2018.1.163	2018.1.163	2018.1.163
2018b	Jul '18	2.30	7.3.0	2018.3.222	2018.3.222	2018.3.222
2019a	Jan '19	2.31.1	8.2.0	2019.1.144	2018.4.274	2019.1.144
2019b	Sept '19	2.32	8.3.0	2019.5.281	2018.5.288	2019.5.281
2020a	May'20	2.34	9.3.0	2020.1.217	2019.7.217	2020.1.217
2020b	Nov'20	2.35	10.2.0	2020.4.304	2019.9.304	2020.4.304
2021a	May'21	2.36.1	10.3.0	2021.2.0	2021.2.0	2021.2.0

(*): This toolchain is deprecated, see *Deprecated toolchains* for more information.

4.3.5 Customizing common toolchains

Sometimes the need arises to customize one or more components of a common toolchain w.r.t. site-specific aspects. One common example is using additional configuration options for Open MPI.

To customize a toolchain component, you should copy the corresponding easyconfig file, modify according to your needs, and make sure it is included in a location in the robot search path that precedes the location of the easyconfig files that are included with EasyBuild (see also *Searching for easyconfigs: the robot search path*), before building and installation the toolchain.

4.3.6 More information about toolchains

Please see the *List of known toolchains* for how to obtain a listing of the currently known toolchains.

For a detailed listing of the compiler options available with each toolchain, please see avail_toolchain_opts.

CHAPTER 5

Basic usage topics

5.1 Using the EasyBuild command line

Basic usage of EasyBuild is described in the following sections, covering the most important range of topics if you are new to EasyBuild.

`eb` is EasyBuild's main command line tool, to interact with the EasyBuild framework and hereby the most common command line options are being documented.

5.1.1 Specifying builds

To instruct EasyBuild which software packages it should build/install and which build parameters it should use, one or more *easyconfig files* (see *Easyconfig files*) must be specified.

This can be done in a number of ways:

- *By providing a single easyconfig file*
- *Via command line options*
- *By providing a set of easyconfig files*
- using `eb --from-pr` (see *Using easyconfigs from pull requests (–from-pr)*)

Whenever EasyBuild searches for *explicitly specified* easyconfig files, it considers a couple of locations, i.e. (in order of preference):

(i) the local working directory

(ii) the robot search path (see *Searching for easyconfigs: the robot search path*)

(iii) the path to easyconfig files that are part of the active EasyBuild installation (which is included in the default robot search path)

These locations are only considered for easyconfig files that are specified only by filename or using a relative path, *not* for easyconfig files that are specified via an absolute path. The dependencies are resolved using the robot search path (see *Searching for easyconfigs: the robot search path*).

Note: For easyconfig files specified on the `eb` command line, the *full* robot search path is only considered since EasyBuild v2.0.0. Earlier versions only considered the local working directory and the easyconfig files that are part of the active EasyBuild installation for *explicitly specified* easyconfig files.

By providing a single easyconfig file

The most basic usage is to simply provide the name of an easyconfig file to the `eb` command. EasyBuild will (try and) locate the easyconfig file, and perform the installation as specified by that easyconfig file.

For example, to build and install `HPL` using the `goolf` toolchain:

```
$ eb HPL-2.0-goolf-1.4.10.eb --robot
[...]
== building and installing GCC/4.7.2...
[...]
== building and installing goolf/1.4.10...
[...]
== building and installing HPL/2.0-goolf-1.4.10...
== fetching files...
== creating build dir, resetting environment...
== unpacking...
== patching...
== preparing...
== configuring...
== building...
== testing...
== installing...
== taking care of extensions...
== packaging...
== postprocessing...
== sanity checking...
== cleaning up...
== creating module...
== COMPLETED: Installation ended successfully
== Results of the build can be found in the log file /home/example/.local/easybuild/
↪software/HPL/2.0-goolf-1.4.10/easybuild/easybuild-HPL-2.0-20141031.223237.log
== Build succeeded for 9 out of 9
== temporary log file /tmp/easybuild-UOEWix/easybuild-NiswcV.log has been removed.
== temporary directory /tmp/easybuild-UOEWix has been removed.
```

Then, we can actually load the freshly installed HPL module:

```
$ module load HPL/2.0-goolf-1.4.10
$ which xhpl
/home/example/.local/easybuild/software/HPL/2.0-goolf-1.4.10/bin/xhpl
```

All easyconfig file names' suffixes are `.eb` and follow format:

```
``<name>-<version>-<toolchain>-<versionsuffix>``
```

This is a crucial design aspect, since the dependency resolution mechanism (see *Enabling dependency resolution, –robot / -r and –robot-paths*) relies upon this convention.

Tip: You may wish to modify the installation prefix (e.g., using `--prefix` or by defining `$EASYBUILD_PREFIX`),

in order to redefine the build/install/source path prefix to be used; default value is: `$HOME/.local/easybuild`.

Via command line options

An alternative approach is to only use command line options to specify which software to build. Refer to the `Software search and build options` section in the `eb --help` output for an overview of the available command line options for this purpose (cfr. basic_usage_help).

Here is how to build and install last version of HPCG (that EasyBuild is aware of) using the `goolf/1.4.10` toolchain:

```
$ eb --software-name=HPCG --toolchain=goolf,1.4.10
[...]
== building and installing HPCG/2.1-goolf-1.4.10...
[...]
== COMPLETED: Installation ended successfully
[...]
```

At this point, a module `HPCG/2.1-goolf-1.4.10` should have been installed.

By providing a set of easyconfig files

Multiple easyconfig files can be provided as well, either directly or by specifying a directory that contains easyconfig files.

For example, to build and install both HPCG and GCC with a single command, simply list the easyconfigs for both on the `eb` command line (note that HPCG is not being reinstalled, since a matching module is already available):

```
$ eb HPCG-2.1-goolf-1.4.10.eb GCC-4.8.3.eb
[...]
== HPCG/2.1-goolf-1.4.10 is already installed (module found), skipping
[...]
== building and installing GCC/4.8.3...
[...]
== Build succeeded for 1 out of 1
[...]
```

When one or more directories are provided, EasyBuild will (recursively) traverse them to find easyconfig files. For example:

```
$ find set_of_easyconfigs/ -type f
set_of_easyconfigs/GCC-4.8.3.eb
set_of_easyconfigs/foo.txt
set_of_easyconfigs/tools/HPCG-2.1-goolf-1.4.10.eb
```

```
$ eb set_of_easyconfigs/
== temporary log file in case of crash /tmp/easybuild-1yxCvv/easybuild-NeNmZr.log
== HPCG/2.1-goolf-1.4.10 is already installed (module found), skipping
== GCC/4.8.3 is already installed (module found), skipping
== No easyconfigs left to be built.
== Build succeeded for 0 out of 0
== temporary log file /tmp/easybuild-1yxCvv/easybuild-NeNmZr.log has been removed.
== temporary directory /tmp/easybuild-1yxCvv has been removed.
```

Note: EasyBuild will only pick up the files which end with `.eb` ; anything else will be ignored.

Tip: Calling EasyBuild is designed as an *idempotent* operation; if a module is available that matches with an provided easyconfig file, the installation will simply be skipped.

5.1.2 Commonly used command line options

Command line help, `--help` / `-H`

Detailed information about the usage of the eb command is available via the `--help`, `-H`, `-h` help options.

Refer to page basic_usage_help for more detailed information.

Note: `--help` / `-H` spit out the long help info (i.e. including long option names), `-h` only includes short option names.

Tip: This is the best way to query for certain information, esp. recent features, since this is in sync with the actual EasyBuild version being used.

Report version, `--version`

You can query which EasyBuild version you are using with `--version`:

```
$ eb --version
This is EasyBuild 1.15.2 (framework: 1.15.2, easyblocks: 1.15.2) on host example.
↪local.
```

Tip: Asking EasyBuild to print own its version is a quick way to ensure that `$PYTHONPATH` is set up correctly, so that the entire EasyBuild installation (framework and easyblocks) is available.

List of known toolchains, `--list-toolchains`

For an overview of known toolchains, use `eb --list-toolchains`.

Toolchains have brief mnemonic names, for example:

- `goolf` stands for `GCC, OpenMPI, OpenBLAS/LAPACK, FFTW and ScaLAPACK`
- `iimpi` stands for `icc/ifort, impi`
- `cgmvolf` stands for `Clang/GCC, MVAPICH2, OpenBLAS/LAPACK, FFTW`

The complete table of available toolchains is available at *List of known toolchains*.

List of available easyblocks, `--list-easyblocks`

You can obtain a list of available *Easyblocks* via `--list-easyblocks`.

The `--list-easyblocks` command line option prints the easyblocks in a hierarchical way, showing the inheritance patterns, with the "base" easyblock class `EasyBlock` on top.

Software-specific easyblocks have a name that starts with `EB_`; the ones that do not are generic easyblocks. (cfr. *Easyblocks* for the distinction between both types of easyblocks).

For example, a list of easyblocks can be obtained with:

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

To see more details about the available easyblocks, i.e., in which Python module the class is defined, and where it is located, use `--list-easyblocks=detailed`.

Refer to page *List of easyblocks* for more information.

All available easyconfig parameters, `--avail-easyconfig-params` / `-a`

EasyBuild provides a significant amount of easyconfig parameters. An overview of all available easyconfig parameters can be obtained via `eb --avail-easyconfig-params`, or `eb -a` for short.

Refer to page easyconfig_params for more information, the possible parameters are a very rich set.

Combine -a with `--easyblock/-e` to include parameters that are specific to a particular easyblock. For example:

```
$ eb -a -e EB_WRF
```

If you want to see the full output of running this command, look at *Available easyconfig parameters for EB_WRF*.

Enable debug logging, `--debug` / `-d`

Use `eb --debug/-d` to enable debug logging, to include all details of how EasyBuild performed a build in the log file:

```
$ eb HPCG-2.1-goolf-1.4.10.eb -d
```

Tip: You may enable this by default via adding `debug = True` in your EasyBuild configuration file

Note: Debug log files are significantly larger than non-debug logs, so be aware.

Rebuild installation, `--rebuild`

Use `eb --rebuild` to rebuild a given easyconfig/module.

Warning: Use with care, since the reinstallation of existing modules will be done without requesting confirmation first!

Tip: Combine `--rebuild` with `--dry-run` to get a good view on which installations will be rebuilt. (cfr. *Getting an overview of planned installations –dry-run / -D*)

Note: To create a backup of existing modules that are regenerated when `--rebuild` is used, use `--backup-modules` (see also *Backing up of existing modules (–backup-modules)*).

Forced reinstallation, `--force` / `-f`

Use `eb --force/-f` to force the reinstallation of a given easyconfig/module. The behavior of `--force` is the same as `--rebuild` and `--ignore-osdeps`.

Warning: Use with care, since the reinstallation of existing modules will be done without requesting confirmation first!

Tip: Combine `--force` with `--dry-run` to get a good view on which installations will be forced. (cfr. *Getting an overview of planned installations –dry-run / -D*)

Note: To create a backup of existing modules that are regenerated when `--force` is used, use `--backup-modules` (see also *Backing up of existing modules (–backup-modules)*).

Searching for easyconfigs, `--search` / `-S`

Searching for available easyconfig files can be done using the `--search` (long output) and `-S` (short output) command line options. All easyconfig files available in the robot search path are considered (see *Searching for easyconfigs: the robot search path*), and searching is done *case-insensitive*.

For example, to see which easyconfig files are available for the software package named *Mesquite*:

```
$ eb --search mesquite
== temporary log file in case of crash /tmp/eb-_TYdTf/easybuild-iRJ2vb.log
== Searching (case-insensitive) for 'mesquite' in /home/example/easybuild-easyconfigs/
↪easybuild/easyconfigs
 * /Users/kehoste/work/easybuild-easyconfigs/easybuild/easyconfigs/m/Mesquite/
↪Mesquite-2.3.0-goolf-1.4.10.eb
 * /Users/kehoste/work/easybuild-easyconfigs/easybuild/easyconfigs/m/Mesquite/
↪Mesquite-2.3.0-ictce-4.1.13.eb
 * /Users/kehoste/work/easybuild-easyconfigs/easybuild/easyconfigs/m/Mesquite/
↪Mesquite-2.3.0-ictce-5.3.0.eb
== temporary log file(s) /tmp/eb-_TYdTf/easybuild-iRJ2vb.log* have been removed.
== temporary directory /tmp/eb-_TYdTf has been removed.
```

The same query with `-S` is more readable, when there is a joint path that can be collapsed to a variable like `$CFGS1`:

```
$ eb -S mesquite
== temporary log file in case of crash /tmp/eb-5diJjn/easybuild-nUXlkj.log
== Searching (case-insensitive) for 'mesquite' in /home/example/easybuild-easyconfigs/
↪easybuild/easyconfigs
```

(continues on next page)

(continued from previous page)

```
CFGS1=/home/example/easybuild-easyconfigs/easybuild/easyconfigs/m/Mesquite
 * $CFGS1/Mesquite-2.3.0-goolf-1.4.10.eb
 * $CFGS1/Mesquite-2.3.0-ictce-4.1.13.eb
 * $CFGS1/Mesquite-2.3.0-ictce-5.3.0.eb
== temporary log file(s) /tmp/eb-5diJjn/easybuild-nUXlkj.log* have been removed.
== temporary directory /tmp/eb-5diJjn has been removed.
```

For more specific searching, a regular expression pattern can be supplied (since EasyBuild v2.1.1).

For example, to search which easyconfig files are available for GCC v4.6.x, without listing easyconfig files that use GCC v4.6.x as a toolchain:

```
$ eb -S ^GCC-4.6
== temporary log file in case of crash /tmp/eb-PpwTwm/easybuild-b8yrOG.log
== Searching (case-insensitive) for '^GCC-4.6' in /home/example/easybuild-easyconfigs/
↪easybuild/easyconfigs
CFGS1=/home/example/easybuild-easyconfigs/easybuild/easyconfigs/g/GCC
 * $CFGS1/GCC-4.6.3-CLooG-PPL.eb
 * $CFGS1/GCC-4.6.3.eb
 * $CFGS1/GCC-4.6.4.eb
== temporary log file(s) /tmp/eb-PpwTwm/easybuild-b8yrOG.log* have been removed.
== temporary directory /tmp/eb-PpwTwm has been removed.
```

Or, to find all easyconfig files for Python versions 2.7.8 and 2.7.9 that use the `intel` toolchain:

```
$ eb -S '^Python-2.7.[89].*intel'
== temporary log file in case of crash /tmp/eb-Dv5LEJ/easybuild-xpGGSF.log
== Searching (case-insensitive) for '^Python-2.7.[89].*intel' in /home/example/
↪easybuild-easyconfigs/easybuild/easyconfigs
CFGS1=/home/example/easybuild-easyconfigs/easybuild/easyconfigs/p/Python
 * $CFGS1/Python-2.7.8-intel-2014.06.eb
 * $CFGS1/Python-2.7.8-intel-2014b.eb
 * $CFGS1/Python-2.7.8-intel-2015a.eb
 * $CFGS1/Python-2.7.9-intel-2015a-bare.eb
 * $CFGS1/Python-2.7.9-intel-2015a.eb
== temporary log file(s) /tmp/eb-Dv5LEJ/easybuild-xpGGSF.log* have been removed.
== temporary directory /tmp/eb-Dv5LEJ has been removed.
```

Note: Prior to EasyBuild v2.1.1, the full path to easyconfig files was considered when matching the search pattern. Starting with EasyBuild v2.1.1, only the filename of the easyconfig file itself is taken into account.

Enabling dependency resolution, `--robot` / `-r` and `--robot-paths`

EasyBuild supports installing an entire software stack, including the required toolchain if needed, with a single `eb` invocation.

To enable dependency resolution, use the `--robot` command line option (or `-r` for short):

```
$ eb mpiBLAST-1.6.0-goolf-1.4.10.eb --robot
[...]
== building and installing GCC/4.7.2...
[...]
== building and installing hwloc/1.6.2-GCC-4.7.2...
[...]
```

(continues on next page)

(continued from previous page)

```
== building and installing OpenMPI/1.6.4-GCC-4.7.2...
[...]
== building and installing gompi/1.4.10...
[...]
== building and installing OpenBLAS/0.2.6-gompi-1.4.10-LAPACK-3.4.2...
[...]
== building and installing FFTW/3.3.3-gompi-1.4.10...
[...]
== building and installing ScaLAPACK/2.0.2-gompi-1.4.10-OpenBLAS-0.2.6-LAPACK-3.4.2...
[...]
== building and installing goolf/1.4.10...
[...]
== building and installing mpiBLAST/1.6.0-goolf-1.4.10...
[...]
== Build succeeded for 9 out of 9
```

The dependency resolution mechanism will construct a full dependency graph for the software package(s) being installed, after which a list of dependencies is composed for which no module is available yet. Each of the retained dependencies will then be built and installed, in the required order as indicated by the dependency graph.

Tip: Using `--robot` is particularly useful for software packages that have an extensive list of dependencies, or when reinstalling software using a different compiler toolchain (you can use the `--try-toolchain` command line option in combination with `--robot`).

Note: Unless dependency resolution is enabled, EasyBuid requires that modules are available for every dependency. If `--robot` is not used and one or more modules are missing, `eb` will exit with an error stating that a module for a particular dependency could not be found. For example:

```
add_dependencies: no module 'GCC/4.7.2' found for dependency {...}
```

Searching for easyconfigs: the robot search path

For each dependency that does not have a matching module installed yet, EasyBuild requires a corresponding easyconfig file. If no such easyconfig file was specified on the `eb` command line, the dependency resolution mechanism will try to locate one in the *robot search path*.

Searching for easyconfigs is done based on filename (see also *What is an easyconfig (file)?*), with filenames being derived from the dependency specification (i.e. software name/version, toolchain and version suffix). For each entry in the robot search path, a couple of different filepaths are considered, mostly determined by the software name.

For example, when looking for an easyconfig for `OpenMPI` version `1.6.4` and version suffix `-test` with toolchain `GCC/4.7.2`, the following filepaths are considered (relative to each entry in the robot search path):

- `OpenMPI/1.6.4-GCC-4.7.2-test.eb`
- `OpenMPI/OpenMPI-1.6.4-GCC-4.7.2-test.eb`
- `o/OpenMPI/OpenMPI-1.6.4-GCC-4.7.2-test.eb`
- `OpenMPI-1.6.4-GCC-4.7.2-test.eb`

Note: Sometimes easyconfig files are needed even when the modules for the dependencies are already available, i.e.,

whenever the information provided by the dependency specification (software name/version, toolchain and version suffix) is not sufficient. This is the case when using `--dry-run` to construct the complete dependency graph, or when the active module naming scheme requires some additional information (e.g., the `moduleclass`).

Note: If EasyBuild is unable to locate required easyconfigs, an appropriate error message will be shown. For example:

```
Irresolvable dependencies encountered: GCC/4.7.2
```

or:

```
Failed to find easyconfig file 'GCC-4.7.2.eb' when determining module name for: {...}
```

Default robot search path

By default, EasyBuild will only include the collection of easyconfig files that is part of the EasyBuild installation in the robot search path. More specifically, only directories listed in the *Python search path* (partially specified by the `$PYTHONPATH` environment variable) that contain a subdirectory named `easybuild/easyconfigs` are considered part of the robot search path (in the order they are encountered).

Controlling the robot search path

To control the robot search path, you can specify a (colon-separated list of) path(s) to `--robot`/`-r` and/or `--robot-paths` (or, equivalently, `$EASYBUILD_ROBOT` and/or `$EASYBUILD_ROBOT_PATHS`):

```
eb --robot=$PWD:$HOME ...
```

These two configuration options each serve a particular purpose, and together define the robot search path:

- `--robot`, `-r`:
  - intended to be used (only) as a command line option to `eb` (although it can also be defined through another configuration type)
  - enables the dependency resolution mechanism (disabled by default)
  - optionally a list of paths can be specified, which is included *first* in the robot search path
  - by default, the corresponding list of paths is *empty*
- `--robot-paths`:
  - intended to be defined in an EasyBuild configuration file, or via `$EASYBUILD_ROBOT_PATHS`
  - does *not* enable the dependency resolution mechanism
  - the specified list of paths is included *last* in the robot search path
  - by default, only the path to the easyconfig files that are part of the EasyBuild installation is listed
  - **note**: setting this configuration option implies redefining the default robot search path, unless a prepending/appending list of paths is specified, see *Prepending and/or appending to the default robot search path*

For both options, the list of paths should be specified as a colon-separated (`:`) list.

By combining `--robot` and `--robot-paths` using the different configuration types (see also *Supported configuration types*), you have full control over the robot search path: which paths are included, the order of those paths, whether or not the easyconfig files that are part of the EasyBuild installation should be considered, etc.

A constant named `DEFAULT_ROBOT_PATHS` is available that can be used (only) in EasyBuild configuration files to refer to the default robot search path, i.e. the path to the easyconfigs that are part of the EasyBuild installation. For more information on using constants in EasyBuild configuration files, see *Templates and constants supported in configuration files*.

Tip: Only use `--robot` to enable the dependency resolution mechanism; define `robot-paths` in your EasyBuild configuration file or via `$EASYBUILD_ROBOT_PATHS` to specify which sets of easyconfig files EasyBuild should consider, and in which order. By means of exception, a path can be specified to `--robot` to give a specific set of easyconfig files precedence over others, for example when testing modified easyconfig files.

Note: The paths specified on the configuration type with the highest order of preference *replace* any paths specified otherwise, i.e. values are not retained across configuration types. That is: `--robot` *overrides* the value in `$EASYBUILD_ROBOT`, `$EASYBUILD_ROBOT_PATHS` *overrides* the `robot-paths` specification in an EasyBuild configuration file, etc. Of course, the value specified for `--robot` does not directly affect the value specified for `--robot-paths`, on any configuration level, and vice versa. For more information on the relation between the different configuration types, see *Supported configuration types*.

Prepending and/or appending to the default robot search path

Prepending or appending to the default robot search path is supported via the `--robot-paths` configuration option.

To *prepend* one or more paths, a list of paths followed by a '`:`' should be specified.

Analogously, to *append* one or more paths, a list of paths preceded by a '`:`' should be specified.

For example:

- `export EASYBUILD_ROBOT_PATHS=/tmp/$USER:` specifies to prepend `/tmp/$USER` to the robot search path
- `--robot-paths :$HOME/eb:$HOME/test` specifies to append `$HOME/eb` and `$HOME/test` to the robot search path (in that order)
- `--robot-paths=/tmp/$USER::$HOME/test` specifies to prepend `/tmp/$USER` *and* append `$HOME/test` to the robot search path

Example use case

For example, say we want to configure EasyBuild to behave as follows w.r.t. the robot search path:

- (always) prefer easyconfig files in the archive/repository over the ones that are included in the EasyBuild installation (i)
- consider easyconfig files located in the current directory or home directory first (in that order), before any others (ii)

Matching setup:

- satisfy (i) using `robot-paths` in one of the active EasyBuild configuration files (see also *List of used configuration files*):

```
[basic]
repositorypath = /home/example/easybuild/easyconfigs_archive
robot-paths = %(repositorypath)s:%(DEFAULT_ROBOT_PATHS)s
```

- satisfy (ii) via `--robot` on the `eb` command line:

```
eb mpiBLAST-1.6.0-goolf-1.4.10.eb --robot $PWD:$HOME
```

Getting an overview of planned installations `--dry-run` / `-D`

You can do a "dry-run" overview by supplying `-D`/`--dry-run` (typically combined with `--robot`, in the form of `-Dr`):

```
$ eb mpiBLAST-1.6.0-goolf-1.4.10.eb -Dr
== temporary log file in case of crash /tmp/easybuild-vyNQhw/easybuild-pO8EJv.log
Dry run: printing build status of easyconfigs and dependencies
CFGS=/home/example/.local/easybuild/software/EasyBuild/1.15.2/lib/python2.7/site-
↪packages/easybuild_easyconfigs-1.15.2.0-py2.7.egg/easybuild/easyconfigs
 * [x] $CFGS/g/GCC/GCC-4.7.2.eb (module: GCC/4.7.2)
 * [x] $CFGS/h/hwloc/hwloc-1.6.2-GCC-4.7.2.eb (module: hwloc/1.6.2-GCC-4.7.2)
 * [x] $CFGS/o/OpenMPI/OpenMPI-1.6.4-GCC-4.7.2.eb (module: OpenMPI/1.6.4-GCC-4.7.2)
 * [x] $CFGS/g/gompi/gompi-1.4.10.eb (module: gompi/1.4.10)
 * [ ] $CFGS/o/OpenBLAS/OpenBLAS-0.2.6-gompi-1.4.10-LAPACK-3.4.2.eb (module: OpenBLAS/
↪0.2.6-gompi-1.4.10-LAPACK-3.4.2)
 * [ ] $CFGS/f/FFTW/FFTW-3.3.3-gompi-1.4.10.eb (module: FFTW/3.3.3-gompi-1.4.10)
 * [ ] $CFGS/s/ScaLAPACK/ScaLAPACK-2.0.2-gompi-1.4.10-OpenBLAS-0.2.6-LAPACK-3.4.2.eb␣
↪(module: ScaLAPACK/2.0.2-gompi-1.4.10-OpenBLAS-0.2.6-LAPACK-3.4.2)
 * [ ] $CFGS/g/goolf/goolf-1.4.10.eb (module: goolf/1.4.10)
 * [ ] $CFGS/m/mpiBLAST/mpiBLAST-1.6.0-goolf-1.4.10.eb (module: mpiBLAST/1.6.0-goolf-
↪1.4.10)
== temporary log file /tmp/easybuild-vyNQhw/easybuild-pO8EJv.log has been removed.
== temporary directory /tmp/easybuild-vyNQhw has been removed.
```

Note how the different status symbols denote distinct handling states by EasyBuild:

- `[ ]` The build is not available, EasyBuild will deliver it
- `[x]` The build is available, EasyBuild will skip building this module
- `[F]` The build is available, however EasyBuild has been asked to force a rebuild via `--force` and will do so
- `[R]` The build is available, and the application will be rebuilt as request via `--rebuild`

Note: Since EasyBuild v2.4.0, a detailed overview of the build and install procedure that EasyBuild will be execute can be obtained using `--extended-dry-run` or `-x`, see *Extended dry run*.

Getting an overview of missing installations `--missing-modules` / `-M`

Since EasyBuild v3.9.1, you can obtain a list of missing installations (i.e. easyconfigs for which no corresponding environment module file is available yet) using `eb --missing-modules` (or `-eb -M` for short):

```
$ eb TensorFlow-1.13.1-foss-2019a-Python-3.7.2.eb --missing-modules
== temporary log file in case of crash /tmp/eb-bjCz9b/easybuild-CSAvhK.log
```

(continues on next page)

(continued from previous page)

```
2 out of 54 required modules missing:

* Bazel/0.20.0-GCCcore-8.2.0 (Bazel-0.20.0-GCCcore-8.2.0.eb)
* TensorFlow/1.13.1-foss-2019a-Python-3.7.2 (TensorFlow-1.13.1-foss-2019a-Python-3.7.
↪2.eb)

== Temporary log file(s) /tmp/eb-bjCz9b/easybuild-CSAvhK.log* have been removed.
== Temporary directory /tmp/eb-bjCz9b has been removed.
```

Note that dependency resolution is enabled automatically (i.e., `--robot` is implied when using `--missing-modules` or `-M`).

Tweaking existing easyconfig files, using `--try-*`

Making minor changes to existing easyconfig files can be done straight from the `eb` command line. This way, having to manually copying and editing easyconfig files can be avoided.

Tweaking existing easyconfig files can be done using the `--try-*` command line options. For each of the software build options that can be used as an alternative to specifying easyconfig file names, a matching `--try-X` command line options is available:

- `--try-toolchain` to try using the toolchain with the given name and version
  - format: `--try-toolchain=<name>,<version>`
  - `--try-toolchain-name` to try using the latest toolchain version of a toolchain
  - `--try-toolchain-version` to try using a different toolchain version
- `--try-software-version` to try building a different software version
- `--try-amend` to try tweaking a different easyconfig parameter
  - format: `--try-amend=<param>=<value>`
  - only supports string and list-of-strings value types

For example, to build and install WRF and its dependencies with a different toolchain version:

```
$ eb WRF-3.5.1-goolf-1.4.10-dmpar.eb --try-toolchain-version=1.5.14 -Dr
== temporary log file in case of crash /tmp/easybuild-Y9WApt/easybuild-VmPiOH.log
Dry run: printing build status of easyconfigs and dependencies
 * [x] /home/example/work/easybuild-easyconfigs/easybuild/easyconfigs/g/GCC/GCC-4.8.2.
↪eb (module: GCC/4.8.2)
 * [x] /home/example/work/easybuild-easyconfigs/easybuild/easyconfigs/h/hwloc/hwloc-1.
↪8.1-GCC-4.8.2.eb (module: hwloc/1.8.1-GCC-4.8.2)
 * [x] /home/example/work/easybuild-easyconfigs/easybuild/easyconfigs/o/OpenMPI/
↪OpenMPI-1.6.5-GCC-4.8.2.eb (module: OpenMPI/1.6.5-GCC-4.8.2)
 * [x] /home/example/work/easybuild-easyconfigs/easybuild/easyconfigs/g/gompi/gompi-1.
↪5.14.eb (module: gompi/1.5.14)
 * [x] /home/example/work/easybuild-easyconfigs/easybuild/easyconfigs/o/OpenBLAS/
↪OpenBLAS-0.2.8-gompi-1.5.14-LAPACK-3.5.0.eb (module: OpenBLAS/0.2.8-gompi-1.5.14-
↪LAPACK-3.5.0)
 * [x] /home/example/work/easybuild-easyconfigs/easybuild/easyconfigs/f/FFTW/FFTW-3.3.
↪4-gompi-1.5.14.eb (module: FFTW/3.3.4-gompi-1.5.14)
 * [x] /home/example/work/easybuild-easyconfigs/easybuild/easyconfigs/s/ScaLAPACK/
↪ScaLAPACK-2.0.2-gompi-1.5.14-OpenBLAS-0.2.8-LAPACK-3.5.0.eb (module: ScaLAPACK/2.0.
↪2-gompi-1.5.14-OpenBLAS-0.2.8-LAPACK-3.5.0)
 * [x] /home/example/work/easybuild-easyconfigs/easybuild/easyconfigs/g/goolf/goolf-1.
↪5.14.eb (module: goolf/1.5.14)
```

(continues on next page)

(continued from previous page)

```
 * [ ] /tmp/easybuild-Y9WApt/tweaked_easyconfigs/zlib-1.2.7-goolf-1.5.14.eb (module:␣
↪zlib/1.2.7-goolf-1.5.14)
 * [ ] /tmp/easybuild-Y9WApt/tweaked_easyconfigs/Szip-2.1-goolf-1.5.14.eb (module:␣
↪Szip/2.1-goolf-1.5.14)
 * [ ] /tmp/easybuild-Y9WApt/tweaked_easyconfigs/ncurses-5.9-goolf-1.5.14.eb (module:␣
↪ncurses/5.9-goolf-1.5.14)
 * [ ] /tmp/easybuild-Y9WApt/tweaked_easyconfigs/flex-2.5.37-goolf-1.5.14.eb (module:␣
↪flex/2.5.37-goolf-1.5.14)
 * [ ] /tmp/easybuild-Y9WApt/tweaked_easyconfigs/M4-1.4.16-goolf-1.5.14.eb (module:␣
↪M4/1.4.16-goolf-1.5.14)
 * [ ] /tmp/easybuild-Y9WApt/tweaked_easyconfigs/JasPer-1.900.1-goolf-1.5.14.eb␣
↪(module: JasPer/1.900.1-goolf-1.5.14)
 * [ ] /tmp/easybuild-Y9WApt/tweaked_easyconfigs/HDF5-1.8.10-patch1-goolf-1.5.14.eb␣
↪(module: HDF5/1.8.10-patch1-goolf-1.5.14)
 * [ ] /tmp/easybuild-Y9WApt/tweaked_easyconfigs/tcsh-6.18.01-goolf-1.5.14.eb␣
↪(module: tcsh/6.18.01-goolf-1.5.14)
 * [ ] /tmp/easybuild-Y9WApt/tweaked_easyconfigs/Bison-2.7-goolf-1.5.14.eb (module:␣
↪Bison/2.7-goolf-1.5.14)
 * [ ] /tmp/easybuild-Y9WApt/tweaked_easyconfigs/Doxygen-1.8.3.1-goolf-1.5.14.eb␣
↪(module: Doxygen/1.8.3.1-goolf-1.5.14)
 * [ ] /tmp/easybuild-Y9WApt/tweaked_easyconfigs/netCDF-4.2.1.1-goolf-1.5.14.eb␣
↪(module: netCDF/4.2.1.1-goolf-1.5.14)
 * [ ] /tmp/easybuild-Y9WApt/tweaked_easyconfigs/netCDF-Fortran-4.2-goolf-1.5.14.eb␣
↪(module: netCDF-Fortran/4.2-goolf-1.5.14)
 * [ ] /tmp/easybuild-Y9WApt/tweaked_easyconfigs/WRF-3.5.1-goolf-1.5.14-dmpar.eb␣
↪(module: WRF/3.5.1-goolf-1.5.14-dmpar)
== temporary log file /tmp/easybuild-Y9WApt/easybuild-VmPiOH.log has been removed.
== temporary directory /tmp/easybuild-Y9WApt has been removed.
```

Note: The `--try-*` command line options behave as expected when combined with `--robot`. For example: a modified toolchain specified via `--try-toolchain` only trickles down until the toolchain level (not deeper). This makes for a particularly powerful combo for rebuilding entire software stacks using a different toolchain.

Note: Modifying the software version does **not** trickle down the entire software stack, even when combined with `--robot`, since a software version is tied to a particular software package.

5.2 Writing easyconfig files: the basics

This page explains all the basic information about how to write easyconfig files.

For software builds that follow established build patterns, an easyconfig is all that you need to create in order to build and install the software and the corresponding module file.

Luckily, the majority of software delivery mechanisms are being designed around either autotools or CMake or, perhaps, some simple file extraction/copy pattern. In that case, a *generic easyblock* can be leveraged; see generic_easyblocks.

Yet, in case the software build calls for more elaborate steps (scientific software never fails to surprise us in this regard), a software-specific easyblock may be required; see *Implementing easyblocks*.

Contents

5.2.1 What is an easyconfig (file)?

An easyconfig file serves as a *build specification* for EasyBuild.

It consists of a plain text file (in Python syntax) with mostly *key-value* assignment to define **easyconfig parameters**.

Easyconfigs typically follow a (fixed) strict naming scheme, i.e. `<name>-<version>[-<toolchain>][<versionsuffix>].eb`.

The `-<toolchain>` label (which includes the toolchain name and version) is omitted when a system_toolchain is used. The `<versionsuffix>` label is omitted when the version suffix is empty.

Note: the filename of an easyconfig is only important w.r.t. dependency resolution (`--robot`), see *Enabling dependency resolution, –robot / -r and –robot-paths*.

Example:

```
# easyconfig file for GCC v4.8.3
name = 'GCC'
version = '4.8.3'
...
```

Tip: Comments can be included in easyconfig files using the hash (`#`) character (just like in Python code).

5.2.2 Available easyconfig parameters

About 50 different (generic) easyconfig parameters are supported currently. An overview of all available easyconfig parameters is available via the `-a` command line option.

By default, the parameters specific to generic (default) easyblock `ConfigureMake` are included; using `--easyblock`/`-e` parameters that are specific to a particular easyblock can be consulted.

See easyconfig_params for more details.

Example:

```
$ eb -a -e Binary
Available easyconfig parameters (* indicates specific for the Binary EasyBlock)
MANDATORY
---------
[..]
name:           Name of software (default: None)
[...]
EASYBLOCK-SPECIFIC
------------------
install_cmd(*):     Install command to be used. (default: None)
[...]
```

5.2.3 Mandatory easyconfig parameters

A handful of easyconfig parameters are *mandatory*:

- **name, version**: specify what software (version) to build
- **homepage, description**: metadata (used for module help)
- **toolchain**: specifies name and version of compiler toolchain to use
  - format: dictionary with name/version keys, e.g., `{'name': 'foo', 'version': '1.2.3'}`
  - a list of supported toolchains can be found here

Remarks:

- some others are planned to be required in the future
  - *docurls, software license, software license urls*

Example:

```
name = 'HPL'
version = '2.0'

homepage = 'http://www.netlib.org/benchmark/hpl/'
description = "High Performance Computing Linpack Benchmark"

toolchain = {'name': 'goolf', 'version': '1.4.10'}
[...]
```

5.2.4 Common easyconfig parameters

This section includes an overview of some commonly used (optional) easyconfig parameters.

Source files, patches and checksums

- **sources**: list of source files (filenames only)
- **source urls**: list of URLs where sources can be downloaded
- **patches**: list of patch files to be applied (`.patch` extension)
- **checksums**: list of checksums for source and patch files

Remarks:

- sources are downloaded (best effort), unless already available
- proxy settings are taken into account, since the urllib2 Python package is used for downloading (since EasyBuild v2.0)
- patches need to be EasyBuild-compatible
  - unified diff format (`diff -ruN`)
  - patched locations relative to unpacked sources
- see *Checksums* for more information on `checksums`
- `sources` is usually specified as a list of strings representing filenames for source files, but other formats are supported too, see *Alternative formats for sources*

Example:

```
name = 'HPL'
version = '2.2'

[...]

source_urls = ['http://www.netlib.org/benchmark/hpl']
sources = ['hpl-%(version)s.tar.gz']

# fix Make dependencies, so parallel build also works
patches = ['HPL_parallel-make.patch']

checksums = ['ac7534163a09e21a5fa763e4e16dfc119bc84043f6e6a807aba666518f8df440']

[...]
```

Note: Rather than hardcoding the version (and name) in the list of sources, a string template *%(version)s* can be used, see also *Dynamic values for easyconfig parameters*.

Checksums

Checksums for source files and patches can be provided via the `checksums` easyconfig parameter.

EasyBuild does not enforce checksums to be available for all source files and patches. Provided checksums will be 'consumed' first for the specified sources (in order), and subsequently also for patches.

Nevertheless, providing checksums for *all* source files and patches is highly recommended.

If checksums are provided, the checksum of the corresponding source files and patches is verified to match.

The `checksums` easyconfig parameter is usually defined as a list of strings.

Until EasyBuild v3.3.0, only MD5 checksums could be provided through a list of strings. Since EasyBuild v3.3.0, the checksum type is determined by looking at the length of the string:

- 32-character strings are considered to be MD5 checksums (`md5`)
- 64-character strings are considered to be SHA256 checksums (`sha256`)
- (other lengths will result in an error message)

The intention is to move towards making `sha256` the recommended and default checksum type.

Other checksum types are also supported: `adler32`, `crc32`, `sha1`, `sha512`, `size` (filesize in bytes). To provide checksum values of a specific type, elements of the `checksums` list can also be 2-element tuples of the form `('<checksum type>', '<checksum value>')`. For example:

```
checksums = [('sha512',
↪'f962008105639f58e9a4455c8057933ab0a5e2f43db8340ae1e1afe6dc2d24105bfca3b2e1f79cb242495ca4eb363c982(
↪')]
```

Adding or replacing checksums using `--inject-checksums`

Using the `--inject-checksums` command line option, you can let EasyBuild add or update checksums in one or more easyconfig files (which is significantly more convenient than doing it manually).

With `--inject-checksums`, checksums are injected for all sources and patches (if any), as well as for all sources & patches of every extension listed in `exts_list` (if any, see module_extensions).

If the sources (& patches) are not available yet, EasyBuild will try to download them first; i.e., the `fetch` step is run prior to computing & injecting the checksums.

A backup is created of every easyconfig file that is touched by `--inject-checksums`, to avoid accidental loss of information. Backups are given an additional extension of the form `.bak_<year><month><day><hour><min><sec>`.

Note: To clean up backup easyconfig files, you can use this one-liner:

```
find . -name '*.eb.bak_*' | xargs rm -v
```

The `-v` option makes `rm` print the path of files that are being removed.

Do use this with care; just run `find . -name '*.eb.bak_*'` first in case of doubt!

Multilple easyconfigs can be specified when using `--inject-checksums`, they will be processed in sequence. In addition, you can also combine `--inject-checksums` with `--robot`, see *Synergy between –inject-checksums and –robot*.

Adding checksums when none are specified yet

If the easyconfig file does not specify any checksums yet, they are simply injected after the `sources` (or `patches`, if present) specification when `--inject-checksums` is used.

For example:

```
$ eb bzip2-1.0.6.eb --inject-checksums
== temporary log file in case of crash /tmp/eb-Vm6w3e/easybuild-cAVQl6.log
== injecting sha256 checksums in /example/bzip2-1.0.6.eb
```

(continues on next page)

(continued from previous page)

```
== fetching sources & patches for bzip2-1.0.6.eb...
== backup of easyconfig file saved to /example/bzip2-1.0.6.eb.bak_20170824200906...
== injecting sha256 checksums for sources & patches in bzip2-1.0.6.eb...
== * bzip2-1.0.6.tar.gz:␣
↪a2848f34fcd5d6cf47def00461fcb528a0484d8edef8208d6d2e2909dc61d9cd
== Temporary log file(s) /tmp/eb-Vm6w3e/easybuild-cAVQl6.log* have been removed.
== Temporary directory /tmp/eb-Vm6w3e has been removed.
```

The backup easyconfig file can be used to double-check the difference between the original easyconfig file and the one produced by `--inject-checksums`:

```
$ diff -u /example/bzip2-1.0.6.eb.bak_20170824200906 /example/bzip2-1.0.6.eb
diff --git a//example/bzip2-1.0.6.eb.bak_20170824200906 b/example/bzip2-1.0.6.eb
index 46b2debed..2eb73f15a 100644
--- a/example/bzip2-1.0.6.eb.bak_20170824200906
+++ b/example/bzip2-1.0.6.eb
@@ -9,8 +9,9 @@ compressors), whilst being around twice as fast at compression and␣
↪six times fas
 toolchain = SYSTEM
 toolchainopts = {'pic': True}

-sources = [SOURCE_TAR_GZ]
 source_urls = ['http://www.bzip.org/%(version)s/']
+sources = [SOURCE_TAR_GZ]
+checksums = ['a2848f34fcd5d6cf47def00461fcb528a0484d8edef8208d6d2e2909dc61d9cd']

 buildopts = "CC=gcc CFLAGS='-Wall -Winline -O3 -fPIC -g $(BIGFILES)'"
```

Note: Along with injecting checksums, EasyBuild will also reorder the `source_urls`, `sources` and `patches` specifications, in that order and if they are present, and include the `checksums` specification afterwards. This is done to facilitate working towards a uniform style in easyconfig files, which also applies to the order of specified easyconfig parameters.

Replacing existing checksums

When one or more checksums are already specified, EasyBuild requires the use of `--force` together with `--inject-checksums` to replace those checksums. A clear warning will be printed to notify that existing checksums will be replaced.

For example:

```
$ eb bzip2-1.0.6.eb --inject-checksums
== temporary log file in case of crash /tmp/eb-WhSwVH/easybuild-HCODnl.log
== injecting sha256 checksums in /example/bzip2-1.0.6.eb
== fetching sources & patches for bzip2-1.0.6.eb...
ERROR: Found existing checksums, use --force to overwrite them
```

```
$ eb bzip2-1.0.6.eb --inject-checksums --force
== temporary log file in case of crash /tmp/eb-dS2QLa/easybuild-JGxOzC.log
== injecting sha256 checksums in /example/bzip2-1.0.6.eb
== fetching sources & patches for bzip2-1.0.6.eb...
```

(continues on next page)

(continued from previous page)

```
WARNING: Found existing checksums in bzip2-1.0.6.eb, overwriting them (due to use of -
↪-force)...

== backup of easyconfig file saved to /example/bzip2-1.0.6.eb.bak_20170824203850...
== injecting sha256 checksums for sources & patches in bzip2-1.0.6.eb...
...
```

Note: Any existing checksums are *blindly* replaced when `--inject-checksums --force` is used: the existing checksums are *not verified* to be correct as during normal use of EasyBuild (since that would kind of defeat the purpose of `--inject-checksums`).

In addition, it also doesn't matter whether or not checksums are available for all sources & patches: with `--inject-checksums`, checksums will be added for *all* sources and patches, including for extensions listed in `exts_list` (if any).

Synergy between `--inject-checksums` and `--robot`

When `--inject-checksums` is combined with `--robot`, checksums are injected for *each* easyconfig file in the dependency graph for which no module is available yet.

For example, to inject checksums in *every* easyconfig file required to build HPL 2.2 with the `foss/2017a` toolchain:

```
$ MODULEPATH= eb HPL-2.2-foss-2017a.eb --installpath /tmp/$USER/sandbox --inject-
↪checksums --robot
== temporary log file in case of crash /tmp/eb-8HpJc3/easybuild-H35khM.log
== resolving dependencies ...
...
== injecting sha256 checksums in /example/GCCcore-6.3.0.eb
...
== injecting sha256 checksums in /example/OpenMPI-2.0.2-GCC-6.3.0-2.27.eb
...
== injecting sha256 checksums in /example/FFTW-3.3.6-gompi-2017a.eb
...
== injecting sha256 checksums in /example/HPL-2.2-foss-2017a.eb
...
```

Note: We are clearing `$MODULEPATH` and specifying a custom (empty) location to `--installpath` to avoid that EasyBuild skips any easyconfig because a corresponding module is already available.

Type of checksum to inject

By default, `--inject-checksums` will compute & inject `SHA256` checksums, but a different checksum type can be specified as an argument (e.g., `--inject-checksums md5`).

Note: Because of the optional argument that can be passed to `--inject-checksums`, you should not specify an easyconfig file name directly after the `--inject-checksums`, since it will be assumed to specify a checksum type, which will result in an error message like:

```
$ eb --inject-checksums bzip2-1.0.6.eb
Usage: eb [options] easyconfig [...]

eb: error: option --inject-checksums: invalid choice: 'bzip2-1.0.6.eb' (choose from
↪'adler32', 'crc32', 'md5', 'sha1', 'sha256', 'sha512', 'size')
```

Alternative formats for `sources`

In some cases, it can be required to provide additional information next to the name of a source file, e.g., a custom extraction command (because the one derived from the file extension is not correct), or an alternname filename that should be used to download the source file.

This can be specified using a Python dictionary value in the `sources` easyconfig parameter.

Since EasyBuild v3.3.0, three keys are supported:

- `filename` (*mandatory*): filename of source file
- `download_filename`: filename that should be used when downloading this source file; the downloaded file will be saved using the `filename` value
- `extract_cmd`: custom extraction command for this source file
- `source_urls`: source URLs to consider for downloading this source file
- `git_config`: see *Downloading from a Git repository*

For example:

```
sources = [{
    'source_urls': ['https://example.com'],
    'filename': 'example-%(version)s.gz',
    'download_filename': 'example.gz',  # provided source tarball is not versioned...
    'extract_cmd': "tar xfvz %s",  # source file is actually a gzipped tarball␣
↪(filename should be .tar.gz)
}]
```

Note: Custom extraction commands can also be specified as a 2-element tuple, but this format has been deprecated in favor of the Python dictionary format described above; see also *Specifying source files as 2-element tuples to provide a custom extraction command*.

Downloading from a Git repository

Since EasyBuild v3.7.0, support for downloading directly from a Git repository is available.

When `git_config` is provided for a particular source file (see *Alternative formats for sources*), EasyBuild will create a source tarball after downloading the specified Git repository.

The value for `git_config` is a Python dictionary, where the following keys are *mandatory*:

- `url`: the URL where the Git repository is located
- `repo_name`: the name of the Git repository

The value for `filename` in the source specification *must* end in `.tar.gz` (because a gzipped tarball will be created from the cloned repository).

In addition, either of the following keys *must* also be defined:

- `tag`: the specific tag to download (could be a branch name or an actual tag)
- `commit`: the specific commit ID to download

If a recursive checkout should be made of the repository, the `recursive` key can be set to `True`.

To also retain the `.git` directory (which holds the Git metadata for the repository), you can set the `keep_git_dir` to `True` (supported since EasyBuild v4.2.0).

Examples:

- creating a source tarball named `example-main.tar.gz` of the `main` branch of a (fictional) `test` repository from `https://agitserver.org/example`:

```
sources = [{
    'filename': 'example-main.tar.gz',
    'git_config': {
        'url': 'https://agitserver.org/example',
        'repo_name': 'test',
        'tag': 'main',
    },
}]
```

- creating a source tarball named `example-20180920.tar.gz` of the recursive checkout of commit `abcdef12` of the `test` repository from `https://agitserver.org/example`:

```
sources = [{
    'filename': 'example-20180920.tar.gz',
    'git_config': {
        'url': 'https://agitserver.org/example',
        'repo_name': 'test',
        'commit': 'abcdef12',
        'recursive': True,
        'keep_git_dir': True,
    },
}]
```

Note: Because the source tarball is created locally (by running `tar cfvz` on the directory containing the cloned repository), the (SHA256) checksum is not guaranteed to be the same across different systems.

Whenever you have the option to download a source tarball (or equivalent) directly (for example from GitHub, which also allows downloading a tarball of a specific commit), we strongly recommend you to do so, `git_config` is intended for other Git repos.

Dependencies

- **dependencies**: build/runtime dependencies
- **builddependencies**: build-only dependencies (not in module)
- **hiddendependencies**: dependencies via hidden modules (see also *Installing dependencies as hidden modules using –hide-deps*)
- **osdependencies**: system dependencies (package names)

Remarks:

- modules must exist for all (non-system) dependencies
- (non-system) dependencies can be resolved via `--robot`
- format: `(<name>, <version>[, <versionsuffix>[, <toolchain>]])`

Example:

```
name = 'GTI'
...
toolchain = {'name': 'goolf', 'version': '1.5.14'}
dependencies = [('PnMPI', '1.2.0')]
builddependencies = [('CMake', '2.8.12', '', ('GCC', '4.8.2'))]
```

For each of the specified (build) dependencies, the corresponding module will be loaded in the build environment defined by EasyBuild. For the *runtime* dependencies, `module load` statements will be included in the generated module file.

Note: By default, EasyBuild will try to resolve dependencies using the same toolchain as specified for the software being installed. As of v3.0, if no easyconfig exists to resolve a dependency using the default toolchain EasyBuild will search for the dependency using a compatible subtoolchain.

A different toolchain can be specified on a per-dependency level (cfr. the `CMake` build dependency in the example above).

Alternatively, you can instruct EasyBuild to use the most minimal (sub)toolchain when resolving dependencies, see *Using minimal toolchains for dependencies*.

Loading of modules for dependencies with a `system` toolchain

When a system_toolchain is used, the modules for each of the (build) dependencies are *always* loaded, regardless of the toolchain version (as opposed the behaviour with the `dummy` toolchain in EasyBuild versions prior to v4.0, see system_toolchain_motivation_deprecating_dummy).

Specifying dependencies using `system` toolchain

To make EasyBuild resolve a dependency using the `system` toolchain, either specify '`system`' as toolchain name in the tuple representing the dependency specification, or simply use `True` as 4th value in the tuple.

For example, to specify PnMPI version 1.2.0 built with the `system` toolchain as a (runtime) dependency:

```
dependencies = [('PnMPI', '1.2.0', '', ('system', ''))]
```

which is equivalent to:

```
dependencies = [('PnMPI', '1.2.0', '', True)]
```

Using external modules as dependencies

Since EasyBuild v2.1, specifying modules that are not provided via EasyBuild as dependencies is also supported. See *Using external modules* for more information.

Extensions

Besides dependencies, which are found outside the software being built but are part of the site's EasyBuild installation, it is also possible to incorporate extensions to the software within the build. This is done via the `exts_list` array.

Each entry in `exts_list` is a three-component tuple, with the name and version number, and a dictionary of configuration options for the entry:

```
exts_list = [
    ('name', 'version', { 'option':'value', 'option':'value' })
]
```

The latter may contain essentially any of the full easyconfig parameters, including `buildopts`, `installopts`, etc. Among those options, the following exceptions and special cases should be noted:

- **nosource**: If set `True`, no download will be done
- **source_tmpl**: Template string for the file to be downloaded * default is `'%(name)s-%(version)s.tar.gz'` * `%(name)s` and `%(version)s` come from the `exts_list` entry (above)
- **sources**: Dictionary specifying details of where to download the extension * equivalent to a single entry from the easyconfig `sources` list * preferred to use of `source_tmpl`
- **start_dir**: If not set, will be derived; the easyconfig value will not be used

```
exts_list = [
    ('llvmlite', '0.26.0', {
        'source_urls': ['https://pypi.python.org/packages/source/l/llvmlite/'],
        'patches': ['llvmlite-0.26.0_fix-ffi-Makefile.patch'],
        'checksums': [
            '13e84fe6ebb0667233074b429fd44955f309dead3161ec89d9169145dbad2ebf',     #␣
↪llvmlite-0.26.0.tar.gz
            '40e6fe6de48709b45daebf8082f65ac26f73a4afdf58fc1e8099b97c575fecae',     #␣
↪llvmlite-0.26.0_fix-ffi-Makefile.patch
        ],
    }),
    ('singledispatch', '3.4.0.3', {
        'source_urls': ['https://pypi.python.org/packages/source/s/singledispatch/'],
        'checksums': [
↪'5b06af87df13818d14f08a028e42f566640aef80805c3b50c5056b086e3c2b9c'],
    }),
    (name, version, {
        'source_urls': ['https://pypi.python.org/packages/source/n/numba/'],
        'checksums': [
↪'c62121b2d384d8b4d244ef26c1cf8bb5cb819278a80b893bf41918ad6d391258'],
    }),
]
```

That third instance uses the `name` and `version` variables defined in the easyconfig file. Since EasyBuild v4.2.2, a single-entry `sources` dictionary (see _common_easyconfig_param_sources_alt) may be included in an `exts_list` entry. For example, to download Git sources from a private repository and convert them to a tar-ball for installation:

```
exts_defaultclass = 'PythonPackage'
exts_list = [
    ('pyCAP', '0.1', {
        'sources': {
            'filename': '%(name)s-%(version)s.tar.gz',
            'git_config': {
                'url': 'ssh://nero.stanford.edu/data/git/Analysis',
```

(continues on next page)

(continued from previous page)

```
                'repo_name': 'pyCAP',
                'tag': '%(version)s',
            }
        }
    }),
]
```

Here, the template strings `%(name)s` and `%(version)s` will be substituted from the `exts_list` entry elements ("pyCAP" and "0.1", respectively), not from the easyconfig values.

Configure/build/install command options

- **configopts**: options for configure command
- **preconfigopts**: options used as prefix for configure command

In analogy to *configure*, also *build* and *install* commands are tuneable:

- **buildopts, prebuildopts**: options for build command
- **installopts, preinstallopts**: options for install command

Example:

```
easyblock = 'ConfigureMake'
...
# configure with: ./autogen.sh && ./configure CC="$CC" CFLAGS="$CFLAGS"
preconfigopts = "./autogen.sh && "
buildopts = 'CC="$CC" CFLAGS="$CFLAGS"'
# install with: make install PREFIX=<installation prefix>
installopts = 'PREFIX=%(installdir)s'
```

Note: For more details w.r.t. use of string templates like `%(installdir)s`, see *Dynamic values for easyconfig parameters*.

List of configure/build/install options

In some cases, the *configure-build-install* cycle must be executed multiple times during a single installation, using different options for one or more steps.

EasyBuild supports specifying a *list* of strings, each of which specifying a particular set of options to use.

For example, to perform the installation procedure with three different sets of configuration options:

```
configopts = [
    "--common-opt --one --one-more",
    "--common-opt --two",
    "--common-opt --three",
]
```

This way, EasyBuild will perform the *configure-build-install* cycle **three** times:

- configure using `--common-opt --one --one-more`, build and install
- configure using `--common-opt --two`, build and install on top of the existing installation

- configure using `--common-opt --three`, build and install once more on top of what is installed already

During this process, the environment is reset and the build directory is cleaned up after each cycle, while the installation directory is left untouched (in order to not destroy the result of earlier cycles).

If several `(pre){config|build|install}opts` parameters are defined as being a list of strings, the number of items in the lists must be the same. Any of these parameters defined as a single string value are just reused for each of the cycles performed. For example:

```
easyblock = 'ConfigureMake'
configopts = ['--one', '--two', '--three']
buildopts = 'lib'
preinstallopts = ['TYPE=one', 'TYPE=two', 'TYPE=three']
```

would result in:

- `./configure --prefix=... --one; make lib; TYPE=one make install`
- `./configure --prefix=... --two; make lib; TYPE=two make install`
- `./configure --prefix=... --three; make lib; TYPE=three make install`

An example use case of this is building FFTW with different precisions, see the FFTW easyconfig files.

Sanity check

Custom paths and commands to be used in the sanity check step can be specified using the respective parameters. These are used to make sure that an installation didn't (partly) fail unnoticed.

- **sanity_check_paths**: files/directories that must get installed
- **sanity_check_commands**: (simple) commands that must work when the installed module is loaded

Remarks:

- format: Python dictionary with (*only*) `files`/`dirs` keys
- values must be lists of (tuples of) strings, one of both **must** be non-empty
  - paths are *relative* to installation directory
  - for a path specified as a tuple, only one of the specified paths must be available
- default values:
  - paths: non-empty `bin` and `lib` or `lib64` directories
  - commands: none

Example:

```
sanity_check_paths = {
    'files': ["bin/xhpl"],
    'dirs': [],
}
```

Easyblock specification

To make EasyBuild use a specific (usually generic) easyblock the **easyblock** parameter can be used.

By default, EasyBuild will assume that the easyblock to use can be derived from the software name. For example: for `GCC`, EasyBuild will look for an easyblock class named `EB_GCC` in the Python module `easybuild.easyblocks.gcc`.

A list of available easyblocks is available via `--list-easyblocks` (see also *List of available easyblocks, –list-easyblocks*); generic easyblocks are the ones for which the name does *not* start with `EB_`.

Example:

```
easyblock = 'CMakeMake'
name = 'GTI'
version = '1.2.0'
...
```

Tip: It is highly recommended to use existing (generic) easyblocks, where applicable. This avoids the need for creating (and maintaining) new easyblocks. Typically, generic easyblocks support several custom easyconfig parameters which allow to steer their behavior (see also *All available easyconfig parameters, –avail-easyconfig-params / -a*).

Example:

```
easyblock = 'Binary'
[...]
install_cmd = "./install.bin"
[...]
```

Module class

The category to which the software belongs to can be specified using the **moduleclass** easyconfig parameter. By default, the `base` module class is used (which should be replaced with a more appropriate category).

EasyBuild enforces that only known module classes can be specified (to avoid misclassification due to typos).

The default list of module classes is available via `--show-default-moduleclasses`; additional module classes can be defined via the `--moduleclasses` configure option.

Example:

```
name = 'GCC'
[...]
moduleclass = 'compiler'
```

Note: By default, EasyBuild will create a symlink to the generated module file in a module class-specific path. This behavior is configurable through the module naming scheme being used.

Tip: The module class may play a significant role in other aspects. For example, the alternative (hierarchical) module naming scheme `HierarchicalMNS` heavily relies on the `moduleclass` parameter for discriminating compilers and MPI libraries.

5.2.5 Tweaking existing easyconfig files

The ability to modify easyconfig files on the fly with EasyBuild, provides a very powerful and flexible feature to describe builds, without having to manually create all the input files.

Tweaking existing easyconfigs can be done using the `--try-*` command lines options. See *Tweaking existing easyconfig files, using –try-** for more details.

Example:

- GCC version update:

```
eb GCC-4.9.0.eb --try-software-version=4.9.1
```

- install WRF + its dozen dependencies with a different toolchain (!):

```
eb WRF-3.5.1-ictce-5.3.0-dmpar.eb --try-toolchain=intel,2014b -r
```

5.2.6 Dynamic values for easyconfig parameters

String templates are completed using the value of particular easyconfig parameters, typically `name` and/or `version`. These help to avoid hardcoding values in multiple locations.

A list of available string templates can be obtained using `--avail-easyconfig-templates`.

Additionally, constants that can be used in easyconfig files are available via `--avail-easyconfig-constants`.

Example:

```
name = 'GCC'
version = '4.8.3'
...
source_urls = [
    # http://ftpmirror.gnu.org/gcc/gcc-4.8.3
    'http://ftpmirror.gnu.org/%(namelower)s/%(namelower)s-%(version)s',
]
sources = [SOURCELOWER_TAR_GZ]  # gcc-4.8.3.tar.gz
...
```

Note: Proper use of string templates is important, in particular to avoid hardcoding the software version in multiple locations of an easyconfig file; this is critical to make `--try-software-version` behave as expected (see also *Tweaking existing easyconfig files, using –try-**).

5.2.7 Version-specific documentation relevant to easyconfigs

- Available config file constants
- Available easyconfig parameters
- Constants available for easyconfig files
- License constants available for easyconfig files
- List of available easyblocks
- List of available toolchain options

- List of known toolchains
- List of supported software
- Overview of generic easyblocks
- Templates available for easyconfig files

5.2.8 Contributing easyconfigs

Contribute your working easyconfig files!

Share your expertise with the community, avoid duplicate work, especially if:

- the software package is not supported yet
- an existing easyconfig needs (non-trivial) changes for a different version/toolchain
- it is a frequently used software package (compilers, MPI, etc.)

See *Contributing* for more information.

5.3 Understanding EasyBuild logs

EasyBuild thoroughly keeps track of the executed build and install procedures. This page details some of the specifics, to help you making sense of them.

5.3.1 Basic information

During an invocation of the `eb` command, a temporary log file is provided. This log can be consulted in case any problems occur during the process. Right before completing successfully, EasyBuild will clean up this temporary log file.

A separate log file is created for each build and install procedure that is performed. After each successful installation, this application log file is copied to the install directory for future reference.

By default, the application log file is copied to a subdirectory of the installation prefix named `easybuild`, and has a filename like `easybuild-HPL-2.0-20141103.104412.log` for example, which corresponds to the filename template `easybuild-%(name)s-%(version)s-%(date)s.%(time)s.log`. This aspect can be tweaked via the `--logfile-format` configuration option.

Example:

```
$ eb HPL-2.0-goolf-1.4.10.eb
== temporary log file in case of crash /tmp/easybuild-rHHgBu/easybuild-XD0Ae_.log
[...]
== building and installing HPL/2.0-goolf-1.4.10...
[...]
== COMPLETED: Installation ended successfully
== Results of the build can be found in the log file /home/example/.local/easybuild/
↪software/HPL/2.0-goolf-1.4.10/easybuild/easybuild-HPL-2.0-20141103.104412.log
== Build succeeded for 1 out of 1
== temporary log file /tmp/easybuild-rHHgBu/easybuild-XD0Ae_.log has been removed.
== temporary directory /tmp/easybuild-rHHgBu has been removed.
```

Note: Enabling debug mode using the `--debug` or `-d` command line option ensures that all details of the executed build and installation procedure are included in the log file, but will also result is significantly bigger and more verbose logs.

Tip: Always include a reference to a log file (even if partial) when reporting a potential bug in EasyBuild. A particularly useful way of doing so is by creating a Gist (https://gist.github.com/), and sharing the corresponding URL. This is much better than sending a lengthy log file via email, since it can be easily shared across different communication channels (mailing list, IRC, IM, etc.).

5.3.2 Navigating log files

Extracting the information you're interested in from an EasyBuild log file may be a daunting task, especially for debug logs. The information and guidelines in this section should make navigating logs less scary.

Log message format

Each log message as emitted by EasyBuild follows a well-defined format. Example:

```
== 2014-11-03 13:34:31,906 main.EB_HPL INFO This is EasyBuild 1.15.2 (framework: 1.15.
↪2, easyblocks: 1.15.2) on host example.
```

Each log line consists of the following parts:

- a prefix label ==, which is useful to discriminate between EasyBuild log messages and the output of executed shell commands;
- date and time information (e.g., `2014-11-03 13:34:31,906`);
- the Python module/class/function that is responsible for the log message (e.g., `main.EB_HPL`);
- the log level (e.g., `INFO`);
- and a string with the actual log message at the end

Useful handles in log files

Next to looking for a particular search pattern (e.g., `[Ee]rror`), there are a couple of handles that can be used to jump around in log files.

Step markers

For each step performed in the build and installation process, corresponding log messages is emitted. For example:

```
== 2014-11-03 13:34:48,816 main.EB_HPL INFO configuring...
== 2014-11-03 13:34:48,817 main.EB_HPL INFO Starting configure step
[...]
== 2014-11-03 13:34:48,823 main.EB_HPL INFO Running method configure_step part of␣
↪step configure
```

This allows you to navigate a log file step by step, for example using the `_step` search pattern.

Executed shell commands

For each executed shell command, log messages are included with the full command line, the location where the command was executed and the command's output and exit code. For example:

```
== 2014-11-03 13:34:48,823 main.run DEBUG run_cmd: running cmd /bin/bash make_generic␣
↪(in /tmp/user/easybuild_build/HPL/2.0/goolf-1.4.10/hpl-2.0/setup)
== 2014-11-03 13:34:48,823 main.run DEBUG run_cmd: Command output will be logged to /
↪tmp/easybuild-W85p4r/easybuild-run_cmd-XoJwMY.log
== 2014-11-03 13:34:48,849 main.run INFO cmd "/bin/bash make_generic" exited with␣
↪exitcode 0 and output:
```

If you are primarily interested in the different commands as they were executed by EasyBuild, you can use `INFO cmd` (or `run_cmd`, in debug logs) as a search pattern.

Note: Next to the configure/build/install commands, EasyBuild also runs a couple of other commands to obtain system information, or to query the modules tool. Typically, a single invocation of `eb` involves executing a dozen or so different shell commands, minimally.

CHAPTER 6

Advanced usage topics

6.1 Archived easyconfigs

Since EasyBuild v3.0.0, easyconfig files using deprecated (i.e., old and inactive) toolchains are *archived*.

Contents

6.1.1 Toolchain deprecation

Once in a blue moon, we review the list of toolchains (& versions) that are included in EasyBuild.

Easyconfig files that use toolchains that become *deprecated* are then moved to the *easyconfigs archive*, i.e. the `__archive__` subdirectory in the `easybuild-easyconfigs` repository (see https://github.com/easybuilders/easybuild-easyconfigs/tree/main/easybuild/easyconfigs/__archive__).

What are deprecated toolchains?

Toolchains become deprecated if:

- no easyconfig files using that toolchain have been contributed recently (e.g., in the last year)
- that toolchain is considered to be inactive, after consulting the EasyBuild community (via mailing list, bi-weekly conf calls)

Deprecating a toolchain implies that all easyconfigs using that toolchain are moved to the easyconfigs archive, and that they are no longer included in the EasyBuild regression test. In addition, these easyconfigs are 'hidden' from plain sight, in the sense that you need to use `--consider-archived-easyconfigs` to make EasyBuild consider them when it is looking for easyconfigs (e.g., with `--search` or `--robot`).

This does *not* mean that the support for using these toolchains is removed from the EasyBuild framework, although not testing them anymore may imply that using them may no longer work at some point in time.

For toolchains for which no active versions are available (outside of the easyconfigs archive), it is possible that they will be reactivated, if a new toolchain version is contributed.

Why are toolchains being deprecated?

- using old toolchains (incl. old compilers and/or libraries) is likely to become more and more difficult on modern operating systems
- these toolchains put a significant burden on the regression testing for EasyBuild releases
- easyconfigs using old toolchains are likely to be for old software versions, which may no longer be relevant anyway

6.1.2 Using `--consider-archived-easyconfigs`

To make EasyBuild consider archived easyconfig files, you need to enable the `--consider-archived-easyconfigs` configuration option:

```
$ eb -S '^goolfc'
CFGS=/home/example/work/easybuild-easyconfigs/easybuild/easyconfigs
 * $CFGS1/g/goolfc/goolfc-2016.08.eb
 * $CFGS1/g/goolfc/goolfc-2016.10.eb

Note: 6 matching archived easyconfig(s) found, use --consider-archived-easyconfigs to␣
↪see them
```

```
$ eb -S '^goolfc' --consider-archived-easyconfigs
CFGS=/home/example/easybuild-easyconfigs/easybuild/easyconfigs
 * $CFGS/g/goolfc/goolfc-2016.08.eb
 * $CFGS/g/goolfc/goolfc-2016.10.eb

Matching archived easyconfigs:

 * $CFGS/__archive__/g/goolfc/goolfc-1.3.12.eb
 * $CFGS/__archive__/g/goolfc/goolfc-1.4.10.eb
 * $CFGS/__archive__/g/goolfc/goolfc-2.6.10.eb
```

6.2 Backing up of existing modules (`--backup-modules`)

While regenerating existing module files, you may want to preserve the existing module files to compare and assess that the changes in the newly generated module file match expectations.

Backing up of existing modules can be enabled with `--backup-modules`.

Backups are stored in the same directory as where the module file was located, and the files are given an additional extension of the form `.bak_<year><month><day><hour><min><sec>`.

- With module files in Tcl syntax, the backup module file is hidden by adding a leading dot to the filename; this is done to avoid that it is displayed as a normal module in `module avail`.
- With module files in Lua syntax, the backup module file is not made hidden (unless Lmod 6.x is used), since the additional `.bak_*` extension prevents from picking it up as a valid module file; only files ending in `.lua` are considered to be module files by Lmod 7+ .

The location of the backed up module file will be printed, as well as a "unified diff" comparison (very similar to what `diff -u` produces) between the backed up module file and the newly generated module file (or a message mentioning that no differences were found).

6.2.1 Disabling automatic backup of modules

When `--skip` or `--module-only` is used, backing up of existing modules is enabled automatically.

This can be disabled with `--disable-backup-modules`.

6.2.2 Example

Suppose existing modules in both Tcl & Lua syntax are present (`bzip2/1.0.6`).

Both these module files are missing an update statement for $PATH because the `/bin` subdirectory was missing in the installation, for the purpose of this example:

```
$ ls -la $EASYBUILD_PREFIX/modules/all/bzip2
total 16
drwxr-xr-x  2 example  example   136 Aug 24 10:20 .
drwxr-xr-x  3 example  example   102 Aug 24 10:18 ..
-rw-r--r--  1 example  example  1256 Aug 24 10:19 1.0.6
-rw-r--r--  1 example  example  1303 Aug 24 10:18 1.0.6.lua
```

Using `--force` and `--backup-modules`, we can reinstall the `bzip2/1.0.6` modules and get a clear view on what has changed.

To reinstall the `bzip2/1.0.6` module in Lua syntax while retaining a backup of the existing module:

```
$ eb bzip2-1.0.6.eb --module-syntax=Lua --force --backup-modules
...
== creating build dir, resetting environment...
== backup of existing module file stored at /example/modules/all/bzip2/1.0.6.lua.bak_
↪20170824102603
...
== creating module...
== comparing module file with backup /example/modules/all/bzip2/1.0.6.lua.bak_
↪20170824102603; diff is:
--- /example/modules/all/bzip2/1.0.6.lua.bak_20170824102603
+++ /example/modules/all/bzip2/1.0.6.lua
@@ -25,9 +25,10 @@
 prepend_path("LD_LIBRARY_PATH", pathJoin(root, "lib"))
 prepend_path("LIBRARY_PATH", pathJoin(root, "lib"))
 prepend_path("MANPATH", pathJoin(root, "man"))
+prepend_path("PATH", pathJoin(root, "bin"))
 setenv("EBROOTBZIP2", root)
 setenv("EBVERSIONBZIP2", "1.0.6")
 setenv("EBDEVELBZIP2", pathJoin(root, "easybuild/bzip2-1.0.6-easybuild-devel"))

...
```

Equivalently, we can reinstall the module in Tcl syntax using:

```
$ eb bzip2-1.0.6.eb --module-syntax=Tcl --force --backup-modules
```

Afterwards, both the newly generated modules and the backups are in place:

```
$ ls -la $EASYBUILD_PREFIX/modules/all/bzip2
total 32
drwxr-xr-x  2 example  example   204 Aug 24 10:26 .
drwxr-xr-x  3 example  example   102 Jul 11 10:18 ..
-rw-r--r--  1 example  example  1227 Aug 24 10:24 .1.0.6.bak_20170824102412
-rw-r--r--  1 example  example  1256 Jul 11 01:24 1.0.6
-rw-r--r--  1 example  example  1303 Jul 11 01:26 1.0.6.lua
-rw-r--r--  1 example  example  1259 Aug 24 10:26 1.0.6.lua.bak_20170824102603
```

Cleaning up the backup module files can be done with the following command (for example):

```
$ find $EASYBUILD_PREFIX/modules/all/bzip2 -name '*.bak*' | xargs rm -v
/example/modules/all/bzip2/.1.0.6.bak_20170824102412
/example/modules/all/bzip2/1.0.6.lua.bak_20170824102603
```

6.3 Generating container recipes & images

EasyBuild has support for generating Singularity and Docker *container recipes* which will use EasyBuild to build and install a specified software stack. In addition, EasyBuild can (optionally) leverage the build tool provided by the container software of choice to create *container images*.

Note: The features documented here have been available since EasyBuild v3.6.0 but are still *experimental*, which implies they are subject to change in upcoming versions of EasyBuild.

You will need to enable the `--experimental` configuration option in order to use them.

See *Experimental features* for more information.

Initially (since EasyBuild v3.6.0), only Singularity (https://www.sylabs.io/singularity) was supported. Since EasyBuild v3.6.2, generating (recipes for) Docker (https://www.docker.com/) containers is also supported.

In EasyBuild v3.9.2 the support for generating Singularity container recipes/images was enhanced significantly.

Contents

6.3.1 Requirements

- Docker, or Singularity version 2.4 (or more recent, incl. version 3.x)
- `sudo` permissions *(only required to actually build container images, see Building container images (–container-build-image))*

6.3.2 Usage

Generating container recipes (`--containerize` / `-C`)

To generate container recipes, use `eb --containerize`, or `eb -C` for short.

The resulting container recipe will, in turn, leverage EasyBuild to build and install the software that corresponds to the easyconfig files that are specified as arguments to the `eb` command (and all required dependencies, if needed).

Note: EasyBuild will refuse to overwrite existing container recipes.

To re-generate an already existing recipe file, use the `--force` command line option.

Container template recipe (`--container-template-recipe`)

Via the `--container-template-recipe` configuration option, you can specify a specific container template recipe that EasyBuild should use to generate container recipes.

This gives you control over a variety of aspects, including:

- the operating system (version) used in the container image
- the location where EasyBuild installs software within the container
- how EasyBuild is configured when installing software in the container
- etc.

When generating container recipes, EasyBuild will replace the following template values:

- `%(bootstrap)s`: bootstrap agent to use
  - see also https://www.sylabs.io/guides/latest/user-guide/definition_files.html#header
- `%(bootstrap_config)s`: configuration for the bootstrap agent

  - this is expected to include lines that specify `From:`, `MirrorURL:`, etc.
  - for more information, see *Container configuration (–container-config)*
- `%(easyconfigs)s`: (list of) easyconfig file name(s)/path(s) to pass to `eb` command
- `%(eb_args)s`: additional arguments for 'eb' command
- `%(include)s`: list of additional OS packages to include
  - see also *include keyword: OS packages to include*
- `%(install_eb)s`: list of commands to install EasyBuild
- `%(install_os_deps)s`: list of commands to install required OS packages (for example `yum install -y openssl`)
  - incl. `osdependencies` specified in easyconfig files
- `%(mirrorurl)`: URI to use to download OS
  - see also *mirrorurl keyword: mirror URL to use to download OS*
- `%(modname)s`: module name(s) to load in environment
- `%(osversion)`: OS version to use
  - see also *osversion keyword: OS version to use*
- `%(post_commands)s`: additional commands for the `post` section of the (Singularity) container recipe

Container configuration (`--container-config`)

Using `--container-config`, values for specific template values can be specified.

Values can be specified as a comma-separated list of `<key>=<value>` pairs; for example: `--container-config bootstrap=localimage,from:example.sif`.

Currently supported keywords include:

- `bootstrap`: bootstrap agent to use
  - two types of values are supported:
    - *Image-based bootstrap agents (docker, library, localimage, shub)*
    - *Linux distro bootstrap agents (arch, busybox, debootstrap, yum, zypper)*
- `eb_args`: additional arguments for 'eb' command
- `from`: argument to pass to bootstrap agent
  - *required/only valid with* `docker`, `library`, `localimage` *and* `shub` *bootstrap agents*
  - for more details, see *Image-based bootstrap agents (docker, library, localimage, shub)*
- `include`: list of additional OS packages to include
  - see also *include keyword: OS packages to include*
- `install_eb`: commands to install EasyBuild
- `mirrorurl`: URI to use to download OS
  - see also *mirrorurl keyword: mirror URL to use to download OS*
- `osversion`: OS version to use
  - see also *osversion keyword: OS version to use*

- `post_commands`: additional commands for `post` section of (Singularity) container recipe

For more details on the last three, see *Linux distro bootstrap agents (arch, busybox, debootstrap, yum, zypper)*.

Note: Specifying any unknown keywords will results in an error.

Image-based bootstrap agents (`docker`, `library`, `localimage`, `shub`)

These bootstrap agents involve using an existing container image as a base.

Supported values include:

- `docker`: base container image hosted on Docker Hub (https://hub.docker.com/)
- `library`: base container image hosted on Sylabs Container Library (https://cloud.sylabs.io/)
- `localimage`: local base container image file
- `shub`: base container image hosted on Singularity Hub (https://singularity-hub.org/)

The `from` keyword must also be specified when using one of these bootstrap agents.

The `localimage` bootstrap agents corresponds to using a local container image file as a base, where it's path is specified using the `from` keyword. For example: "`bootstrap=localimage,from=/home/example/base.sif`".

Each of the other image-based bootstrap agents imply that the container image to use as a base is downloaded from the corresponding registry, ad specified through the `from` keyword, with a specific format:

- for `docker` bootstrap agent: `<registry>/<namespace>/<container>:<tag>@<digest>`
- for `library` bootstrap agent: `<entity>/<collection>/<container>:<tag>`
- for `shub` bootstrap agent: `<registry>/<username>/<container-name>:<tag>@digest`

For more details, see https://www.sylabs.io/guides/latest/user-guide/appendix.html#build-modules.

Requirements for base container image

There are a couple of specific requirements for the base container image:

- all dependencies of EasyBuild must be installed, including:
  - Python 2.6 or 2.7
  - Lmod
  - standard tools & utilities like `make`, `patch`, `tar`, etc.
  - OS packages for system libraries like OpenSSL

See also *Requirements*.

Each generated container recipe will include commands to create the `easybuild` user if it doesn't exist yet, as well as commands to create the `/app` and `/scratch` directories and give the `easybuild` user write permissions to those locations.

Note: The generated container recipe currently hardcodes some of this. We intend to make this more configurable in a future version of EasyBuild.

Linux distro bootstrap agents (`arch`, `busybox`, `debootstrap`, `yum`, `zypper`)

Dedicated bootstrap agents are supported for different flavors of Linux distributions, including:

- `arch`: Arch Linux
- `busybox`: BusyBox Linux
- `debootstrap`: apt-based systems like Ubuntu/Debian
- `yum`: yum-based systems like CentOS
- `zypper`: zypper-based systems like openSUSE

When one of these bootstrap agents is used, additional keywords can be specified:

- *include keyword: OS packages to include*
- *mirrorurl keyword: mirror URL to use to download OS*
- *osversion keyword: OS version to use*

`include` keyword: OS packages to include

Via the `include` keywords, a list of packages can be specified that should be include on top of the base OS installation.

For some bootstrap agents, a default value is used if no value is specified:

- for the `yum` bootstrap agent: `yum`
- for the `zypper` bootstrap agent: `zypper`

See also https://www.sylabs.io/guides/latest/user-guide/appendix.html#yum-bootstrap-agent and https://www.sylabs.io/guides/latest/user-guide/appendix.html#zypper-bootstrap-agent.

`mirrorurl` keyword: mirror URL to use to download OS

For most of the Linux distro bootstrap agents (all except `arch`), Singularity requires that a mirror URL is specified that will be used when downloading the corresponding OS.

You can specify a value using the `mirrorurl` keyword. For example: "`bootstrap=yum, mirrorurl=https://example.com`".

EasyBuild will use a default value for `mirrorurl` if no other value is specified:

- `busybox`: `https://www.busybox.net/downloads/binaries/%{OSVERSION}/busybox-x86_64`
- `debootstrap`: `http://us.archive.ubuntu.com/ubuntu/`
- `yum`: `http://mirror.centos.org/centos-%{OSVERSION}/%{OSVERSION}/os/x86_64/`
- `zypper`:: `http://download.opensuse.org/distribution/leap/%{OSVERSION}/repo/oss/`

`osversion` keyword: OS version to use

Using the `osversion` keyword you can specify which OS version should be installed.

Note that is this only required/used if value for the `mirrorurl` value contains `%{OSVERSION}s`.

For example: "`bootstrap=yum,osversion=7`".

Building container images (`--container-build-image`)

To instruct EasyBuild to also build a container image from the generated container recipe, use `--container-build-image` (in combination with `-C` or `--containerize`).

EasyBuild will leverage functionality provided by the container software of choice (see containers_cfg_image_type) to build the container image.

For example, in the case of Singularity, EasyBuild will run `sudo /path/to/singularity build` on the generated container recipe.

Note: In order to leverage the image building functionality of the container software, admin privileges are typically required. Therefore, EasyBuild will run the command to build the container image with `sudo`. You may need to enter your password to let the command execute.

EasyBuild will only run the actual container image build command with `sudo`. It will not use elevated privileges for anything else.

In case of doubt, you can use `--extended-dry-run` or `-x` do perform a dry run, so you can evaluate which commands will be executed (see also *Extended dry run*).

If you're not comfortable with this, you can just let EasyBuild generate the container recipe, and then use that to build the actual container images yourself, either locally or through Singularity Hub (https://singularity-hub.org).

The container image will be placed in the location specified by the `--containerpath` configuration option (see *Location for generated container recipes & images (–containerpath)*), next to the generated container recipe that was used to build the image.

Note: When building container images, make sure to use a file system location with sufficient available storage space. Singularity may pull metadata during the build, and each image can range from several hundred MBs to GBs, depending on software stack you are including in the container image.

Note: EasyBuild will refuse to overwrite existing container images.

To re-generate an already existing image file, use the `--force` command line option.

Example usage

In this example, we will use a pre-built base container image located at `example.sif` (see also *Image-based bootstrap agents (docker, library, localimage, shub)*).

To let EasyBuild generate a container recipe for GCC 6.4.0 + binutils 2.28:

```
eb GCC-6.4.0-2.28.eb --containerize --container-config bootstrap=localimage,
↪from=example.sif --experimental
```

With other configuration options left to default (see output of `eb --show-config`), this will result in a Singularity container recipe using `example.sif` as base image, which will be stored in `$HOME/.local/easybuild/containers`:

```
$ eb GCC-6.4.0-2.28.eb --containerize --container-config bootstrap=localimage,
↪from=example.sif --experimental
== temporary log file in case of crash /tmp/eb-dLZTNF/easybuild-LPLeG0.log
== Singularity definition file created at /home/example/.local/easybuild/containers/
↪Singularity.GCC-6.4.0-2.28
== Temporary log file(s) /tmp/eb-dLZTNF/easybuild-LPLeG0.log* have been removed.
== Temporary directory /tmp/eb-dLZTNF has been removed.
```

Example of a generated container recipe

Below is an example of container recipe for that was generated by EasyBuild, using the following command:

```
eb Python-3.6.4-foss-2018a.eb -C --container-config bootstrap=yum,osversion=7 --
↪experimental
```

```
Bootstrap: yum
OSVersion: 7
MirrorURL: http://mirror.centos.org/centos-%{OSVERSION}/%{OSVERSION}/os/x86_64/
Include: yum

%post
yum install --quiet --assumeyes epel-release
yum install --quiet --assumeyes python setuptools Lmod
yum install --quiet --assumeyes python-pip
yum install --quiet --assumeyes bzip2 gzip tar zip unzip xz
yum install --quiet --assumeyes curl wget
yum install --quiet --assumeyes patch make
yum install --quiet --assumeyes file git which
yum install --quiet --assumeyes gcc-c++
yum install --quiet --assumeyes perl-Data-Dumper
yum install --quiet --assumeyes perl-Thread-Queue
yum --skip-broken --quiet --assumeyes install libibverbs-dev libibverbs-devel rdma-
↪core-devel
yum --skip-broken --quiet --assumeyes install openssl-devel libssl-dev libopenssl-
↪devel

# install EasyBuild using pip
pip install -U setuptools
pip install 'vsc-install<0.11.4' 'vsc-base<2.9.0'
pip install easybuild

# create 'easybuild' user (if missing)
id easybuild || useradd easybuild

# create /app software installation prefix + /scratch sandbox directory
if [ ! -d /app ]; then mkdir -p /app; chown easybuild:easybuild -R /app; fi
if [ ! -d /scratch ]; then mkdir -p /scratch; chown easybuild:easybuild -R /scratch;␣
↪fi

# install Lmod RC file
cat > /etc/lmodrc.lua << EOF
scDescriptT = {
```

(continues on next page)

(continued from previous page)

```
  {
    ["dir"]       = "/app/lmodcache",
    ["timestamp"] = "/app/lmodcache/timestamp",
  },
}
EOF

# change to 'easybuild' user
su - easybuild

# verbose commands, exit on first error
set -ve

# configure EasyBuild

# use /scratch as general prefix, used for sources, build directories, etc.
export EASYBUILD_PREFIX=/scratch

# also use /scratch for temporary directories
export EASYBUILD_TMPDIR=/scratch/tmp

# download sources to /scratch/sources, but also consider files located in /tmp/
↪easybuild/sources;
# that way, source files that can not be downloaded can be seeded in
export EASYBUILD_SOURCEPATH=/scratch/sources:/tmp/easybuild/sources

# install software & modules into /app
export EASYBUILD_INSTALLPATH=/app

# use EasyBuild to install specified software
eb Python-3.6.4-foss-2018a.eb --robot

# update Lmod cache
mkdir -p /app/lmodcache
$LMOD_DIR/update_lmod_system_cache_files -d /app/lmodcache -t /app/lmodcache/
↪timestamp /app/modules/all

# exit from 'easybuild' user
exit

# cleanup, everything in /scratch is assumed to be temporary
rm -rf /scratch/*

%runscript
eval "$@"

%environment
# make sure that 'module' and 'ml' commands are defined
source /etc/profile
# increase threshold time for Lmod to write cache in $HOME (which we don't want to do)
export LMOD_SHORT_TIME=86400
# purge any modules that may be loaded outside container
module --force purge
# avoid picking up modules from outside of container
module unuse $MODULEPATH
# pick up modules installed in /app
module use /app/modules/all
```

(continues on next page)

(continued from previous page)

```
# load module(s) corresponding to installed software
module load Python/3.6.4-foss-2018a

%labels
```

The generated container recipe includes a bunch of `yum install` commands to install additional required/useful OS packages, `pip install` commands to install EasyBuild (if it's not installed yet), commands to create the `easybuild` user and provide write access to the `/app` and `/scratch` directories, and to configure Lmod and update the Lmod cache after software was installed with EasyBuild.

In addition, the generated module files will follow the default module naming scheme (`EasyBuildMNS`). The modules that correspond to the easyconfig files that were specified on the command line will be loaded automatically, see the statements in the `%environment` section of the generated container recipe.

Example of building container image

You can instruct EasyBuild to also build the container image by also using `--container-build-image`.

Note that you will need to enter your `sudo` password (unless you recently executed a `sudo` command in the same shell session):

```
$ eb GCC-6.4.0-2.28.eb --containerize --container-config bootstrap=localimage,from=/
↪tmp/example.sif --container-build-image --experimental
== temporary log file in case of crash /tmp/eb-aYXYC8/easybuild-8uXhvu.log
== Singularity tool found at /usr/bin/singularity
== Singularity version '2.4.6' is 2.4 or higher ... OK
== Singularity definition file created at /home/example/.local/easybuild/containers/
↪Singularity.GCC-6.4.0-2.28
== Running 'sudo /usr/bin/singularity build  /home/example/.local/easybuild/
↪containers/GCC-6.4.0-2.28.sif /home/example/.local/easybuild/containers/Singularity.
↪GCC-6.4.0-2.28', you may need to enter your 'sudo' password...
== (streaming) output for command 'sudo /usr/bin/singularity build  /home/example/.
↪local/easybuild/containers/GCC-6.4.0-2.28.sif /home/example/.local/easybuild/
↪containers/Singularity.GCC-6.4.0-2.28':
Using container recipe deffile: /home/example/.local/easybuild/containers/Singularity.
↪GCC-6.4.0-2.28
Sanitizing environment
Adding base Singularity environment to container
...
== temporary log file in case of crash /scratch/tmp/eb-WnmCI_/easybuild-GcKyY9.log
== resolving dependencies ...
...
== building and installing GCCcore/6.4.0...
...
== building and installing binutils/2.28-GCCcore-6.4.0...
...
== building and installing GCC/6.4.0-2.28...
...
== COMPLETED: Installation ended successfully
== Results of the build can be found in the log file(s) /app/software/GCC/6.4.0-2.28/
↪easybuild/easybuild-GCC-6.4.0-20180424.084946.log
== Build succeeded for 15 out of 15
...
Building Singularity image...
Singularity container built: /home/example/.local/easybuild/containers/GCC-6.4.0-2.28.
↪sif
```

(continues on next page)

(continued from previous page)

```
Cleaning up...
== Singularity image created at /home/example/.local/easybuild/containers/GCC-6.4.0-2.
↪28.sif
== Temporary log file(s) /tmp/eb-aYXYC8/easybuild-8uXhvu.log* have been removed.
== Temporary directory /tmp/eb-aYXYC8 has been removed.
```

The inspect the container image, you can use `singularity shell` to start a shell session *in* the container:

```
$ singularity shell --shell "/bin/bash --norc" $HOME/.local/easybuild/containers/GCC-
↪6.4.0-2.28.sif

Singularity GCC-6.4.0-2.28.sif:~> module list

Currently Loaded Modules:
  1) GCCcore/6.4.0   2) binutils/2.28-GCCcore-6.4.0   3) GCC/6.4.0-2.28

Singularity GCC-6.4.0-2.28.sif:~> which gcc
/app/software/GCCcore/6.4.0/bin/gcc

Singularity GCC-6.4.0-2.28.sif:~> gcc --version
gcc (GCC) 6.4.0
...
```

Note: We are passing `--shell "/bin/bash --norc` to `singularity shell` to avoid that the `.bashrc` login script that may be present in your home directory is sourced, since that may include statements that are not relevant in the container environment.

Or, you can use `singularity exec` to execute a command in the container.

Compare the output of running `which gcc` and `gcc --version` locally:

```
$ which gcc
/usr/bin/gcc
$ gcc --version
gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-16)
...
```

and the output when running the same commands in the container:

```
$ singularity exec GCC-6.4.0-2.28.sif which gcc
/app/software/GCCcore/6.4.0/bin/gcc

$ singularity exec GCC-6.4.0-2.28.sif gcc --version
gcc (GCC) 6.4.0
...
```

6.3.3 Configuration

Note: You can specify each of these configuration options either as options to the `eb` command, via the equivalent `$EASYBUILD_CONTAINER*` environment variable, or via an EasyBuild configuration file; see *Supported configuration types*.

Location for generated container recipes & images (`--containerpath`)

To control the location where EasyBuild will put generated container recipes & images, use the `--containerpath` configuration setting. Next to providing this as an option to the `eb` command, you can also define the `$EASYBUILD_CONTAINERPATH` environment variable or specify `containerpath` in an EasyBuild configuration file.

The default value for this location is `$HOME/.local/easybuild/containers`, unless the `--prefix` configuration setting was provided, in which case it becomes `<prefix>/containers` (see *Overall prefix path (–prefix)*).

Use `eb --show-full-config | grep containerpath` to determine the currently active setting.

Container image format (`--container-image-format`)

Note: This is only relevant when creating Singularity container images; the value for `--container-image-format` is ignored when creating Docker container images.

The format for container images that EasyBuild is produces via the functionality provided by the container software can be controlled via the `--container-image-format` configuration setting.

For Singularity containers (see *Type of container recipe/image to generate (–container-type)*), three image formats are supported:

- `squashfs` *(default when using Singularity 2.x)*: compressed images using `squashfs` read-only file system
- `sif` *(default when using Singularity 3.x)*: compressed read-only Singularity Image File (SIF)
- `ext3`: writable image file using `ext3` file system *(only supported with Singularity 2.x)*
- `sandbox`: container image in a regular directory

See also https://www.sylabs.io/guides/latest/user-guide/build_a_container.html .

Name for container recipe & image (`--container-image-name`)

By default, EasyBuild will use the name of the first easyconfig file (without the `.eb` suffix) as a name for both the container recipe and image.

You can specify an alternname name using the `--container-image-name` configuration setting.

The filename of generated container recipe will be `Singularity.<name>`.

The filename of the container image will be `<name><extension>`, where the value for `<extension>` depends on the image format (see *Container image format (–container-image-format)*):

- '`.simg`' for `squashfs` Singularity container images *(only with Singularity 2.x)*
- '`.img`' for `ext3` Singularity container images *(only with Singularity 2.x)*
- '`.sif`' for Singularity container images in Singularity Image Format (SIF) *(only with Singularity 3.x)*
- *empty* for `sandbox` Singularity container images (in which case the container image is actually a directory rather than a file)
- *empty* for Docker container images

Temporary directory for creating container images (`--container-tmpdir`)

The container software that EasyBuild leverages to build container images may be using a temporary directory in a location that doesn't have sufficient free space.

You can instruct EasyBuild to pass an alternate location via the `--container-tmpdir` configuration setting.

For Singularity, the default is to use `/tmp`, see https://www.sylabs.io/guides/latest/user-guide/build_env.html#temporary-folders. If `--container-tmpdir` is specified, the `$SINGULARITY_TMPDIR` environment variable will be defined accordingly to let Singularity use that location instead.

Type of container recipe/image to generate (`--container-type`)

With the `--container-type` configuration option, you can specify what type of container recipe/image EasyBuild should generated. Possible values are:

- `docker`: Docker (https://docs.docker.com/) container recipe & images (supported since EasyBuild v3.6.2)
- `singularity` *(default)*: Singularity (https://www.sylabs.io/singularity) container recipes & images

6.3.4 'Stacking' container images

To avoid long build times and excessive large container images, you can construct your target container image step-by-step, by first building a base container image for the compiler toolchain you want to use, and then using it to build a container images for a particular (set of) software package(s).

For example, to build a container image for Python 3.6.4 built with the `foss/2018a` toolchain:

```
$ cd /tmp

# use current directory as location for generated container recipes & images
$ export EASYBUILD_CONTAINERPATH=$PWD

# build base container image for OpenMPI + GCC parts of foss/2018a toolchain, on top␣
↪of CentOS 7.4 base image
$ eb -C --container-build-image OpenMPI-2.1.2-GCC-6.4.0-2.28.eb --container-config␣
↪bootstrap=yum,osversion=7 --experimental
...
== Singularity image created at /tmp/OpenMPI-2.1.2-GCC-6.4.0-2.28.sif
...

$ ls -lh OpenMPI-2.1.2-GCC-6.4.0-2.28.sif
-rwxr-xr-x 1 root root 590M Apr 24 11:43 OpenMPI-2.1.2-GCC-6.4.0-2.28.sif

# build another container image for the for the full foss/2018a toolchain, using the␣
↪OpenMPI + GCC container as a base
$ eb -C --container-build-image foss-2018a.eb --container-config bootstrap=localimage,
↪from=OpenMPI-2.1.2-GCC-6.4.0-2.28.sif --experimental
...
== Singularity image created at /tmp/foss-2018a.sif
...

$ ls -lh foss-2018a.sif
-rwxr-xr-x 1 root root 614M Apr 24 13:11 foss-2018a.sif

# build container image for Python 3.6.4 with foss/2018a toolchain by leveraging base␣
↪container image foss-2018a.sif
```

(continues on next page)

(continued from previous page)

```
$ eb -C --container-build-image Python-3.6.4-foss-2018a.eb --container-config␣
↪bootstrap=localimage,from=foss-2018a.sif --experimental
...
== Singularity image created at /tmp/Python-3.6.4-foss-2018a.sif
...

$ ls -lh Python-3.6.4-foss-2018a.sif
-rwxr-xr-x 1 root root 759M Apr 24 14:01 Python-3.6.4-foss-2018a.sif

$ singularity exec Python-3.6.4-foss-2018a.sif which python
/app/software/Python/3.6.4-foss-2018a/bin/python

$ singularity exec Python-3.6.4-foss-2018a.sif python -V
vsc40023 belongs to gsingularity
Python 3.6.4
```

6.3.5 Seeding in source files for container build process

In some cases, you may need to "seed in" manually downloaded source files into the container build environment, because the sources can not be downloaded automatically.

As shown in *Example of a generated container recipe*, the container recipe generated by EasyBuild includes `/tmp/easybuild/sources/` as a fallback directory in the list of locations considered by EasyBuild when looking for sources/patches (see also *Source path (–sourcepath)*).

That way, you can copy source files that should be available when building the container image into `/tmp/easybuild/sources/`.

6.4 Contributing

This documentation explains how you can contribute to EasyBuild, and discusses the review process for contributions.

Contents

6.4.1 How to contribute

It is a common misconception that contributing to an open source project requires being intimately familiar with its codebase.

There are various ways to contribute, even if you don't have any experience with the programming language(s) being used by the project you are interested in.

You can contribute to EasyBuild by:

- *Giving feedback*
- *Reporting bugs*
- *Submitting feature/change requests*
- *Helping others*
- *Contributing easyconfig files*
- *Contributing code changes*
- *Writing documentation*
- *Joining the conversation*

Giving feedback

An easy way to contribute to EasyBuild, even without having a lot of hands-on experience with it, is to **provide feedback** on your experiences.

Feedback from people new to EasyBuild is valuable, since it provides a perspective that is easily overlooked by more experienced users.

You can contact the EasyBuild community via the EasyBuild mailing list (`easybuild@lists.ugent.be`), the `#easybuild` IRC channel (see also *Getting help*).

Reporting bugs

If you run into unexpected problems when using EasyBuild, please **open a bug report** in the issue tracker of the relevant GitHub repository:

- https://github.com/easybuilders/easybuild-framework/issues: for general problems with `eb`, the EasyBuild framework, etc.
- https://github.com/easybuilders/easybuild-easyblocks/issues: for problems specific to a particular (generic or software-specific) easyblock, etc.
- https://github.com/easybuilders/easybuild-easyconfigs/issues: for problems specific to a particular software package, e.g., with building and installing a particular version of that software, or when using a particular toolchain, etc.
- https://github.com/easybuilders/easybuild/issues: for problems with the EasyBuild documentation, etc.

Please try and provide all relevant information, which may include:

- the EasyBuild version you are using
- the specifics of the system you are using, incl. OS + version, Python version, modules tool & version, etc.
- the active EasyBuild configuration; usually the output of `eb --show-config` is sufficient
- the full `eb` command line that was used
- warning or error messages, or other relevant informative messages
- filename or contents of easyconfig file(s) being used
- EasyBuild log file (for example via https://gist.github.com/), preferably collected with `eb --debug`

Submitting feature/change requests

If you have **suggestions for enhancements or ideas for new features** that could be useful, please open an issue in the relevant GitHub repository (see *Reporting bugs*).

Describe what you have in mind, and maybe also provide a concrete example to illustrate how your suggestion would improve the current functionality.

Helping others

Try and **help others** based on your experience, or help them figure out an answer to their question or a solution to their problem using the EasyBuild documentation or by reaching out to someone else in the EasyBuild community that may know the answer.

The EasyBuild community is known to be very welcoming and helpful, and you too can be a part of that.

Contributing easyconfig files

Please **contribute easyconfig files** that are not included yet in (the `develop` branch of) the `easybuild-easyconfigs` GitHub repository at https://github.com/easybuilders/easybuild-easyconfigs.

This includes easyconfigs for software that is not supported yet by EasyBuild, as well as updates to existing easyconfigs (e.g., version updates, using a different toolchain, etc.), even if you consider the updates to be trivial. Keep in mind that being able to use something that is known to work out-of-the-box can save quite a bit of time compared to having to tweak existing easyconfigs and validate the result installations.

We highly recommend using the `--new-pr` and `--update-pr` command line options for contributing easyconfig files; see *Submitting new and updating pull requests (–new-pr, –update-pr)*.

Contributing code changes

Of course you can also **contribute code changes**, including bug fixes, enhancements or additional features in the EasyBuild framework, the easyblocks repository, the test suites or in related scripts.

Do keep in mind that this requires some experience with Python, although you should be able to go a long way by using the existing code base as support.

See *Pull requests* for more information on the practical aspects of contributing code.

Writing documentation

Another way to contribute to EasyBuild without having to implement Python code is by **writing documentation**, i.e. enhancing or updating existing documentation or documenting features that are not covered yet.

To contribute to the documentation hosted at http://easybuild.readthedocs.io/, you should open a pull request to the `develop` branch of the main EasyBuild repository at https://github.com/easybuilders/easybuild; see the `docs` subdirectory.

A particularly useful tool is http://rst.ninjs.org/, which can be used to preview how the documentation written in `reStructuredText` format will be rendered on `readthedocs.org` (select the `Nature` theme for optimal results).

Joining the conversation

Last but not least, you can actively **join the conversation** that arise on the EasyBuild mailing list, the `#easybuild` IRC channel and during the bi-weekly EasyBuild conference calls (see https://github.com/easybuilders/easybuild/wiki/Conference-calls).

Engage with the EasyBuild community, and help steer EasyBuild development by participating in the conversations regarding a variety of topics related to building and installing (scientific) software.

6.4.2 Pull requests

To contribute to the EasyBuild framework, easyblocks, easyconfig files or the EasyBuild documentation, you will need to open a **pull request** to the corresponding GitHub repository:

- EasyBuild framework: https://github.com/easybuilders/easybuild-framework
- easyblocks: https://github.com/easybuilders/easybuild-easyblocks
- easyconfigs: https://github.com/easybuilders/easybuild-easyconfigs
- documentation: https://github.com/easybuilders/easybuild (see `docs` subdirectory)

Doing this the traditional way requires some knowledge about using `git` (i.e. creating commits, using branches, pushing to and pulling from remote Git repositories, etc.), and being familiar with GitHub.

However, this can be largely circumvented by using the GitHub integration provided by EasyBuild, see *Integration with GitHub* **.**

Setting up

Note: These instructions assume that you already have a working GitHub account.

If you don't have a GitHub account yet, visit https://github.com/ to create one.

We highly recommend registering your SSH public key in your GitHub account, via https://github.com/settings/keys. This allows pushing to your fork of the GitHub repositories without using a password.

Before you can open pull requests, you will need to **fork** the GitHub repository you are targeting, and create a local **working copy** of that repository. This only needs to be done *once* for every of the EasyBuild repositories.

Note: For the purpose of this guide, we will be using the `easybuild-framework` repository.

However, the instructions are equivalent for the other EasyBuild repositories.

Forking the repository

First, create your own *fork* of the repository in your GitHub account using the '`Fork`' button on the top right at https://github.com/easybuilders/easybuild-framework.

This will create a 'copy' of the `easybuild-framework` repository owned by the `easybuilders` GitHub organisation to your own personal GitHub account.

You will have to use this fork as a staging area for your work, to prepare your contributions before creating the actual pull requests.

Creating a working copy

Note: Do not forget to replace '`EXAMPLE`' with your GitHub account name in the instructions below.

In addition to forking the repository on GitHub, you also need to create a local *working copy* of the repository. This is basically a local checkout of the repository where you can track the changes you are making.

First, *clone* your fork of the repository:

```
git clone git@github.com:EXAMPLE/easybuild-framework.git
```

If that worked as expected, you should have a new directory named `easybuild-framework`. Move into the `easybuild-framework` directory:

```
cd easybuild-framework
```

Finally, we recommended to also check out the `develop` branch, which will be used as the base for your own branches:

```
git checkout -b develop origin/develop
```

With this in place, you are all set to open pull requests for your contributions.

Keeping the `develop` branch in sync

It is important to keep the `develop` branch in your working copy in sync with the upstream repository in the GitHub `easybuilders` organization.

For this, you need to add the upstream repository as a '*remote*' repository:

```
git remote add upstream git@github.com:easybuilders/easybuild-framework.git
```

Note: '`upstream`' is just a name that you give to the remote the central `easybuilders` repository on GitHub; you can modify this to your liking if desired (but do take that into account for the further instructions if you do so).

For your fork of the repository, a default remote named `origin` should have been created via `git clone` (see the output of `git remote -v`).

To sync your `develop` branch, use `git pull upstream develop` after making sure you have the `develop` branch checked out:

```
git checkout develop
git pull upstream develop
```

Opening a new pull request

Note: This section describes the manual procedure to open a new pull request.

Please consider using `eb --new-pr` instead, see *Submitting pull requests (–new-pr)*.

Note: We assume you are already located in your local working copy of the repository you want to contribute to (e.g., `easybuild-framework`).

To open a pull request for your contribution, you must follow these steps:

i. *Creating a new branch*

ii. *Committing your changes*

iii. *Pushing your branch*

iv. *Pull request title & description*

v. *Opening the pull request*

Creating a new branch

First, create a new branch for your work. You can do this either before or after making the changes that you would like to contribute, but we recommend to create a new branch before making any changes.

Make sure you have the `develop` branch checked out before creating your branch:

```
git checkout develop
```

To create a new branch, you should use `git branch <branch_name>` followed by `git checkout <branch_name>`, or equivalently `git checkout -b <branch_name>`.

For example, to create a new branch named `mybranch`:

```
git checkout -b mybranch
```

You can choose the branch name freely, but make it sufficiently descriptive – your future self will thank you.

Committing your changes

To 'save' your changes, you should create one or more *commits* in the branch you created. We recommended making separate commits for different 'units of work'.

First, make sure you have checked out the branch were you want to commit the changes to. For example, to prepare for committing changes to the `mybranch` branch:

```
git checkout mybranch
```

To check which branch is currently checked out, use `git branch`.

To get a high-level overview of the changes before committing them, you can use `git status`.

To see the actual changes that were made, use `git diff`.

To commit the changes you want to contribute, use `git add <files>` to *stage* the changes, followed by `git commit -m "<message>"` to create the actual commit.

For example, to commit the changes that were made to `easybuild/tools/filetools.py`:

```
git add easybuild/tools/filetools.py
git status  # check which files are staged for commit
git commit -m "example commit message for changes to filetools.py"
```

Note: Please use a concise commit message that describes the changes you made.

Note: For files that are already being tracked, you can use `git commit -am <message>` to commit all changes at once.

To verify that your work was committed, use `git log` to see all commits on the current branch. Use `git log --stat` and/or `git log --diff` to see more details about which changes are included in each of the commits.

Pushing your branch

Once you have committed your changes to a branch, you should *push* your branch to your fork of the GitHub repository using `git push`.

For example, to push the `mybranch` branch to your fork of the GitHub repository (via the `origin` remote, see *Creating a working copy*):

```
git push origin mybranch
```

Note: this will make your work public.

Pull request title & description

Please use a descriptive (short) title for your pull requests, and clarify (if needed) in the pull request description.

If any other pull requests are required, refer to them in the description using `#<PR number>` (only for pull requests to the same repository), or by copy-pasting the URL to the pull request.

For pull requests to the easyconfig repository, we recommend using this format for the pull request title when contributing new easyconfig files:

```
{<moduleclass>}[<toolchain>] <software name> <software version> <extra info>
```

For example:

- `{tools}[dummy] EasyBuild v3.2.1`
- `{math}[intel/2017a] numpy 1.13.0 w/ Python 2.7.13`
- `{math,numlib}[foss/2017a] METIS v5.1.0, ParMETIS v4.0.3, SuiteSparse v4.5.5, ...`

If you are opening a work-in-progress pull request, for example to solicit feedback, tag it using `(WIP).` in the pull request title.

Opening the pull request

To open a pull request using the branch you pushed, you should use the GitHub interface, and follow the steps outlined below.

Note: Replace '`EXAMPLE`' with your GitHub account name, and '`easybuild-framework`' with the name of the target EasyBuild repository.

---

i. visit https://github.com/EXAMPLE/easybuild-framework;

ii. switch to the branch that includes the changes you want to contribute using the '`Branch: main`' button on the left, for example by selecting `Branch: mybranch` from the dropdown list

iii. click the '`New pull request`' button;

iv. change the target branch to `develop` using the '`base: main`' button;

v. review your changes using the 'diff' view presented by GitHub;

vi. provide an appropriate title and description for your contribution;

vii. open the pull request by clicking the green '`Create pull request`' button

Next, your pull request will be reviewed & tested, see *Review process for contributions*.

Updating existing pull requests

Note: This section describes the manual procedure to create a new pull request.

Please consider using `eb --update-pr` instead, see *Updating existing pull requests (–update-pr)*.

It is quite common to update a pull request after creating it, for example if the test suite run in GitHub Actions reports problems with the changes being made, or as a response to someone reviewing your contribution.

To update an existing pull request, it suffices to add commits to the branch that was used for opening the pull request, and pushing the updated branch to GitHub.

For example, to update the pull request that was created using the `mybranch` branch:

```
git checkout mybranch
# make changes...
git add <paths to changed files>
git commit -m "example commit message for additional changes"
git push origin mybranch
```

Updating a pull request will trigger GitHub Actions to re-test your contribution, and a notification will be sent out to whoever is 'watching' your pull request.

Merging of pull requests

Once your pull request has been given the green light by GitHub Actions and one or more people reviewing have approved the changes, it can be merged into the `develop` branch.

This can only be done by a member of the EasyBuild maintainers team. Only pull requests that meet the requirements are eligible for merging, see *Requirements for pull requests* **.**

Merging a pull request usually implies that the changes will be part of the next EasyBuild release.

6.4.3 Review process for contributions

Each contribution is thoroughly reviewed and tested before it gets merged in. Some aspects of this are automated, others require human intervention.

Requirements for pull requests

Only pull requests (PRs) that fulfill these requirements are eligible to be merged:

(i) *the PR must target the* `develop` *branch of the repository*; see *develop branch*

(ii) *the test suite must (still) pass*, i.e. GitHub Actions *must* give a green light; see contributing_review_process_travis

- *tests should be added or enhanced when appropriate*; see *Adding tests*, especially for PRs to the `easybuild-framework` repository

(iii) *backward compatibility should be retained*; see *Backward compatibility*

(iv) *code style must be kept consistent*; see *Code style review*

- *easyconfigs should be kept consistent across versions & toolchains*; see *Comparing with existing easyconfigs (–review-pr)*

(v) *(successful) test reports must be submitted for easyconfig PRs*; see *Test reports for easyconfig contributions (upload-test-report)*

(vi) *the PR is approved by one or more maintainers of the repository*; see *EasyBuild maintainers*

(vii) *the PR should be merged by one of the maintainers, other than the author of the PR*; see *Pull requests are merged by a maintainer other than the author*

`develop` branch

Pull requests are only merged in the '`develop`' **branch** of the EasyBuild repositories, which contains the changes that will be included in the next EasyBuild release.

The '`main`' branch provides the latest stable release of EasyBuild at all times. Only the EasyBuild release manager should issue a pull request to the EasyBuild '`main`' branch, when preparing a new EasyBuild release.

Occasionally, an additional version branch (e.g. '`3.3.x`') may be introduced temporarily, in case an intermittent bugfix release is being worked on.

Test suite (GitHub Actions)

Each pull request is tested automatically by GitHub Actions and the test result is reported in the pull request.

Only pull requests that have been tested and approved by GitHub Actions are eligible for being merged!

Note that GitHub Actions will only run the *test suite* for that particular repository. That is, for easyconfig contributions it does *not* include actually building and installing software.

For more information on the test suites, see *Unit tests*.

Adding tests

An implicit requirement for contributions, in particular contributions to the EasyBuild framework, is that they **must be accompanied by additional tests or test cases**.

For new features or enhancements, a dedicated test (case) must be added which verifies that the feature implementation works as expected.

For bug fixes, a test (case) must be added that triggers the code path where the bug manifested, and which verifies that the bug was indeed fixed.

Tests not only confirm that the implementation is correct, it also helps to ensure that any future changes will not break the functionality you contributed.

Backward compatibility

Contributions should retain backward compatibility, i.e., they should *not* make any changes that alter the (default) functionality of the existing code base. Of course, enhancements to existing code that retain backward compatibility can be made.

One exception to this rule is *bug fixes*, where the whole point is usually to fix functionality that was implemented incorrectly.

This also applies to existing easyconfig files; for example, the versions of dependencies should *not* be altered. Adding dependencies that were missing or otherwise enhancing existing easyconfigs (e.g., adding extra extensions, enabling additional features, etc.) are usually considered acceptable.

Changes that break backward compatibility have to be motivated well with technical arguments, and must be approved by the EasyBuild maintainers.

Code style review

Next to functional evaluation of contributions, care is also taken to maintain a consistent code style across the EasyBuild code base (see also *Code style*); **contributions must take the (mostly PEP8) code style into account.**

This aspect is sometimes considered to be needless overhead, yet it is an important aspect of the review process. A consistent code style is invaluable in a large code base that is constantly being updated by a worldwide community.

This also applies to easyconfig files, where we try to maintain a strict style that mostly matches the established PEP8 coding style for Python (since easyconfigs are written in Python syntax). However, also the grouping and ordering of easyconfig parameters is a part of the 'code' style we maintain.

An automated (partial) check to see whether easyconfig files are ready to be contributed can be performed via `eb --check-contrib`. This will check:

- style aspects for the specified easyconfig files
- whether SHA256 checksums are included for all source files & patches

This check is also a part of the test suite run by GitHub Actions for easyconfig PRs.

Comparing with existing easyconfigs (`--review-pr`)

We try to maintain **consistency across easyconfig files** for a particular software package, across multiple software versions, toolchains and variants (with a different `versionsuffix`).

Therefore, easyconfig contributions are also reviewed using `eb --review-pr <PR #>`, which compares the touched easyconfig files to those in the current `develop` branch that are most closely related.

The easyconfig files to compare with are selected based on similarity, by combining two criteria, in order.

First, the software version is taken into account, using one of the following criteria:

- exact match on software version match
- match on major/minor software version
- match on major software version
- no match on software version

This is combined with one of the criteria below (in order):

- matching versionsuffix and toolchain name/version
- matching versionsuffix and toolchain name (any toolchain version)
- matching versionsuffix (any toolchain name/version)
- matching toolchain name/version (any versionsuffix)
- matching toolchain name (any versionsuffix, toolchain version)
- no extra requirements (any versionsuffix, toolchain name/version)

The first combination of one of the software version criteria with one of the other criteria that yields one or more matching easyconfig files is used. If none of the combinations match, no easyconfig files for that particular software package are available yet, and no comparison is made.

The output of `--review-pr` provides a 'multidiff' comparison, which highlights the differences between the easyconfig file in the pull request and the most similar selected ones from the current `develop` branch.

For example:

Interpreting this output is a quick and easy way to assess how different the contributed easyconfig files are from the existing easyconfigs, although it does require a bit of practice because of the density of the provided information.

Test reports for easyconfig contributions (`upload-test-report`)

For easyconfig contributions, one or more accompanying **test reports must be submitted** to confirm that the added and/or changed easyconfig files (still) work as expected.

We recommend that you submit a test report for your own easyconfig pull requests. Other people can also submit test reports to confirm that your contribution works as expected on their system(s).

With EasyBuild being properly configured (see *Configuration*), this should be as simple as running `eb --from-pr <PR#> --upload-test-report --force --robot`.

See *Uploading test reports (–upload-test-report)* for more information.

Pull requests are merged by a maintainer other than the author

A pull request should never be merged by its author.

This policy is maintained in order to ensure a *"two-pairs-of-eyes"* review process of all contributions.

Why a pull request may be closed by a maintainer

Although it is generally avoided, there are a few reasons why maintainers might close a pull request:

- uses an archived toolchain
- no activity for > 6 months
- obsolete because more recent PRs for newer versions of the software have been merged already

This is done routinely as a way of focusing everyone's efforts on relevant contributions, and should not be seen as a rejection. In fact, contributors are encouraged to reopen the pull request if they feel it is still relevant.

6.5 Controlling compiler optimization flags

This page provides an overview on the different ways in which compiler optimization flags used by EasyBuild can be controlled.

Contents

6.5.1 Controlling target architecture specific optimizations via `--optarch`

Default behaviour

By default, EasyBuild optimizes builds for the CPU architecture of the build host, by instructing the compiler to generate instructions for the highest instruction set supported by the process architecture of the build host processor.

This is done by including specific compiler flags in `$CFLAGS`, `$CXXFLAGS`, `$FFLAGS`, `$F90FLAGS`, etc.

For example:

- for toolchains using the GCC compilers, `--march=native` will be used (see https://gcc.gnu.org/onlinedocs/gcc-4.9.0/gcc/i386-and-x86-64-Options.html)
- for toolchains using the Intel compilers, `-xHost` will be used (https://software.intel.com/en-us/node/522846)

Caveats

Heterogeneous clusters

Optimizing for the processor architecture of the build host is usually what you want in an HPC cluster, but it has some implications if your cluster is heterogeneous (i.e., has different processor generations), or if you want to execute your applications in a machine with a processor architecture that is different from the one of the build host.

For example, if you compile your application optimized for an Intel Haswell processor (i.e. using AVX2 instructions), it will not run on a system with an older Intel Nehalem processor.

One possible workaround for heterogeneous HPC clusters is to build as many copies of the software stack as you have processor generations in your cluster, and to configure your system so each compute node uses the right software stack matching its processor architecture type. Details for one way of doing this, using automounter/autofs are available at https://easybuilders.github.io/easybuild/files/sciCORE-software-management_20150611.pdf .

Another solution is to configure EasyBuild to not optimize for the processor architecture of the build host via `--optarch`, see below.

Build environment vs hardcoding in build scripts

Be aware that that using `--optarch` as described below does not provide hard guarantees that the build will be executed using the intended compiler flags.

EasyBuild will define the appropriate environment variables (`$CFLAGS` and co) to use the compiler flags as specified, but some MakeFiles or build systems could have hardcoded values that have not been dealt with yet (for example, via a patch file or by specifying options to the `make` command).

For example, the OpenBLAS build system will autodetect the processor architecture in the build host, and will optimize for that processor architecture by default.

If you want a generic OpenBLAS build you will need to tweak the OpenBLAS easyconfig file to define the desired `TARGET` to use. For this you will need to modify the `buildopts` easyconfig parameter, for example:

```
buildopts = 'TARGET=PRESCOTT BINARY=64 ' + threading + ' CC="$CC" FC="$F77"'
```

See these links for more details w.r.t. OpenBLAS:

- https://github.com/xianyi/OpenBLAS/blob/develop/TargetList.txt
- https://github.com/xianyi/OpenBLAS/issues/685

Specifying target architecture specific optimization flags to use via `--optarch=<flags>`

Using the `--optarch` EasyBuild configuration option, specific compiler flags can be provided that EasyBuild should use, rather than the ones used by default (depending on the compiler in the toolchain being used).

Like any other configuration setting, this can also be specified via `$EASYBUILD_OPTARCH`, or by defining `optarch` in an EasyBuild configuration file (cfr. *Consistentency across supported configuration types*).

For example, by specifying `--optarch=march=core2`, EasyBuild will use `-march=core2` rather than the default flag `--march=native` (when using GCC compilers).

Likewise, to avoid using the default `-xHost` flag with the Intel compilers and using `-xSSSE3` instead, you can define `$EASYBUILD_OPTARCH` to be equal to `xSSSE3`.

Note: The first dash (`-`) is added automagically to the value specified to `--optarch`, because of technicalities with the current implementation.

The `--optarch` configuration option gives you flexibility to define the specific target architecture optimization flags you want, but requires that you take care of specifying different flags for different compilers and choose the right flags depending on your specific processor architecture.

Optimizing for a generic processor architecture via `--optarch=GENERIC`

To make EasyBuild optimize for a *generic* processor architecture, `--optarch` can be set to '`GENERIC`'.

When this is the case, EasyBuild will use the right compiler flags to optimize for a generic processor architecture, i.e. avoid using machine instructions that are only supported by very recent processors.

The `GENERIC` keyword for `--optarch` is recognized since EasyBuild v2.5.0, and is supported for GCC and Intel compilers on x86-64 systems (Intel or AMD). For other compilers that can be used in toolchains and other system architectures, the necessary compiler flags will be defined in later EasyBuild versions.

Currently, using `--optarch=GENERIC` will result in the following target architecture optimization flags being used, (on a Linux x86-64 system):

- for toolchains using GCC compilers: `-march=x86-64 -mtune=generic`
- for toolchains using Intel compilers: `-xSSE2`

On other systems or for other compilers, you can check which compiler flags will be used via *Extended dry run*.

Setting architecture flags for different compilers via `--optarch=<compiler:flags>;<compiler:flags>`

Starting with version 3.1.0, EasyBuild supports specifying architecture flags on a per-compiler basis. This enables to "set and forget" the `--optarch` option for your compilers of interest, as opposed to change it depending on the compiler used on the packages to be installed.

The syntax is `<compiler:flags>;<compiler:flags>`, where `:` separates the compiler name from the compiler flags, and `;` separates different compilers. This is an example for the Intel and GCC compilers: `--optarch='Intel:xHost;GCC:march=x86-64 -mtune=generic'`. As in the simple cases, EasyBuild adds one `-` to the flags specified, so the flags passed to the Intel and GCC compilers in this case are `-xHost` and `-march=x86-64 -mtune=generic`. Please note the quotes to escape the space in the GCC flags.

Additionally, `GENERIC` is also supported on a compiler basis, allowing to specify a generic compilation for the desired compilers. This is an example of this usage: `--optarch=Intel:xHost;GCC:GENERIC`. Of course, this is supported just for compiler toolchains that recognize `GENERIC`.

The options for each compiler are set independently. That means that if a GCC-based toolchain is used, but the only compiler specified is `Intel` (for example with `--optarch=Intel:xCORE-AVX2`), then EasyBuild will behave as if `--optarch` was not specified for this toolchain.

The compiler name corresponds to the value of the `COMPILER_FAMILY` constant of the toolchain. Two common examples are `GCC` and `Intel`.

Due to the special treatment of `--optarch` in Cray environments, this feature is not supported on this platform.

6.6 EasyBuild on Cray

As of EasyBuild v2.7.0, the support for using EasyBuild on Cray systems is considered stable. Tt enables building/installing software using the `PrgEnv` modules provided by Cray. This page provides an overview of the current status.

For more information about this, contact:

- Guilherme Peretti-Pezzi (CSCS, Switzerland, peretti@cscs.ch)
- Kenneth Hoste (HPC-UGent, Belgium; kenneth.hoste@ugent.be)
- Petar Forai (IMBA/IMP, Austria; petar.forai@imp.ac.at).

Thanks to:

- Olli-Pekka Letho (CSC.fi)
- Tim Robinson and Guilherme Peretti-Pezzi (CSCS.ch)
- Eric Bavier (Cray)
- Brett Bode (NCSA)

for providing us access to Cray systems, for their support and for testing and contributing to this work.

6.6.1 Test systems

- Piz Daint & Piz Dora @ CSCS.ch (http://www.cscs.ch/computers/piz_daint_piz_dora/index.html)
- Santis & Brisi (TDS) @ CSCS.ch
- Sisu @ CSC.fi (https://research.csc.fi/sisu-supercomputer)
- Swan (TDS) @ Cray
- Blue Waters @ NCSA

6.6.2 EasyBuild toolchains

- `CrayCCE`: `PrgEnv-cray` with pinned versions of `cce`, `cray-libsci` and `cray-mpich`
- `CrayGNU`: `PrgEnv-gnu` with pinned versions of `gcc`, `cray-libsci` and `cray-mpich`
- `CrayIntel`: `PrgEnv-intel` with pinned versions of `intel`, `cray-libsci` and `cray-mpich`
- `CrayPGI`: `PrgEnv-pgi` with pinned versions of `pgi` and `cray-mpich`

versions:

- `Cray{CCE,GNU,Intel}/2015.06` (requires Cray PE June/2015)
- `Cray{CCE,GNU,Intel}/2015.11` (requires Cray PE November/2015)

- `CrayGNU/2016.03` (requires Cray PE March/2016)
- `Cray{GNU,PGI}/2016.04` (requires Cray PE April/2016)
- `Cray{GNU,Intel}/2016.06` (requires Cray PE June/2016)

6.6.3 What works already?

(see below for more information)

- **HPL** (LINPACK) benchmark version 2.1

Major scientific software applications

- **CP2K** 2.6.0
- **GROMACS** 4.6.7
- **Python** 2.7.9 + numpy 1.9.2 + scipy 0.15.1
- **WRF** 3.6.1 (pending on Sisu)

An up-to-date list of software applications built on Cray systems at CSCS can be found https://github.com/eth-cscs/production/, see https://github.com/eth-cscs/production/tree/master/jenkins-builds .

6.6.4 Required EasyBuild configuration

Modules tool

- Sisu: self-installed Lmod 5.8
- Piz Daint, Dora, Swan, Santis, Brisi: system-provided environment modules 3.2.10

Example for environment modules 3.2.10:

```
source /opt/modules/3.2.10.3/init/bash
export PATH=/opt/modules/3.2.10.3/bin/:$PATH
export EASYBUILD_MODULES_TOOL=EnvironmentModulesC
export EASYBUILD_MODULE_SYNTAX=Tcl
```

Architecture

- the `craype-<target>` module to load must be specified using `--optarch`
  - e.g., `--optarch=sandybridge` results in `craype-sandybridge` being loaded in the build environment used by EasyBuild

You can also export this option as a shell variable. Example for `sandybridge`:

```
$ export EASYBUILD_OPTARCH=sandybridge
```

Metadata for Cray-provided modules

- Easybuild provides a sample metadata file in order to use modules provided by Cray:

```
easybuild-framework/etc/cray_external_modules_metadata.cfg
```

This file is loaded by default and contains enough information to build the easyconfig files shipped with EasyBuild.

If you need to use a customized file, it can be specified using `--external-modules-metadata`. For more details see *Metadata for external modules*.

6.6.5 Major supported/tested applications

(in alphabetical order)

CP2K

- http://www.cp2k.org
- version(s): 2.6.0

```
$ eb CP2K-2.6.0-CrayGNU-2015.06.eb -dr
```

GROMACS

- http://www.gromacs.org
- version(s): 4.6.7

```
eb GROMACS-4.6.7-CrayGNU-2015.06-mpi.eb -dr
```

HPL

- http://www.netlib.org/benchmark/hpl
- version(s): 2.1

```
eb HPL-2.1-CrayCCE-2015.06.eb -dr
eb HPL-2.1-CrayGNU-2015.06.eb -dr
eb HPL-2.1-CrayIntel-2015.06.eb -dr
```

Python + numpy/scipy

- http://python.org, http://www.numpy.org, http://www.scipy.org
- version(s): Python 2.7.9, numpy 1.9.2, scipy 0.15.1

```
eb Python-2.7.9-CrayGNU-2015.06.eb -dr
# includes a few python packages (such as mpi4py, numpy and scipy)
```

WRF

- http://www.wrf-model.org
- version(s): 3.6.1

```
eb WRF-3.6.1-CrayGNU-2015.06-dmpar.eb --dr
```

6.7 Detection of loaded modules

Since EasyBuild v3.3.0, the `eb` command performs a check to see if any (EasyBuild-generated) modules have been loaded in the current environment.

This check can be controlled via the `--detect-loaded-modules` and `--allow-loaded-modules` configuration options.

In addition any unknown `$EBROOT*` environment variables are detected and acted upon, according to how the `--check-ebroot-env-vars` configuration option was set.

Contents

6.7.1 Motivation

Running EasyBuild in an environment where one or more (EasyBuild-generated) modules are loaded may interfere with the software installations performed by EasyBuild, i.e.:

- they may cause installations failures, for example due to incompatibilities with the modules being loaded during the installation procedure being performed;

- they may cause installations to work in that particular environment, for example by providing a necessary dependency

Since manually loading modules may affect the reproducibility of software installations, it should be discouraged.

In EasyBuild versions before v3.3.0, having a loaded module for the same software packages as the one being installed resulted in an EasyBuild error message.

Since EasyBuild v3.3.0 a more extensive detection mechanism is available and the action that should be taken for loaded modules can be controlled via `--detect-loaded-modules`. Having a module loaded for any software package that is being installed still results in a hard error.

6.7.2 Detection mechanism

Detecting loaded EasyBuild-generated modules is done by checking the environment for defined `$EBROOT*` environment variables. If any are found, the corresponding loaded module is determined for better reporting.

In case defined `$EBROOT*` environment variables are found that do not match a loaded modules, action is taken as well; see *Checking of $EBROOT* environment variables*.

6.7.3 Action to take if loaded modules are detected

The action that should be taken when one or more loaded (EasyBuild-generated) modules are detected can be specified via the `--detect-loaded-modules` configuration option.

Supported values include:

- *error: report error and stop EasyBuild upon detection of loaded modules*
- *ignore: ignore any loaded modules*
- *purge: run 'module purge' to clean environment of loaded modules*
- *unload: unload loaded modules*
- *warn: print warning and continue upon detection of loaded modules* (current default)

The specified action is only taken for non-allowed loaded modules, see *Allowing particular loaded modules*.

`error`: report error and stop EasyBuild upon detection of loaded modules

When EasyBuild is configured with `--detect-loaded-modules=error`, it will print a clear error and stop when any (non-allowed) loaded modules are detected.

For example:

```
$ eb example.eb --detect-loaded-modules=error
== temporary log file in case of crash /tmp/eb-UlKMRN/easybuild-MgfEHi.log
ERROR: Found one or more non-allowed loaded (EasyBuild-generated) modules in current␣
↪environment:
* Spack/0.10.0

To make EasyBuild allow particular loaded modules, use the --allow-loaded-modules␣
↪configuration option.
Use --detect-loaded-modules={error,ignore,purge,unload,warn} to specify action to␣
↪take when loaded modules are detected.

See http://easybuild.readthedocs.io/en/latest/Detecting_loaded_modules.html for more␣
↪information.
```

(continues on next page)

(continued from previous page)

`ignore`: ignore any loaded modules

With `--detect-loaded-modules=ignore` in place, any loaded modules are simply ignored and EasyBuild continues silently.

This is equivalent to the behaviour of EasyBuild versions prior to version 3.3.0.

Note: This is **not** recommended!

`purge`: run '`module purge`' to clean environment of loaded modules

Using `--detect-loaded-modules=purge`, EasyBuild will run `module purge` if any (non-allowed) loaded modules are detected, in an attempt to restore the environment to a clean state before starting software installations.

A short warning message is printed in case `module purge` was used to clean up the environment:

```
$ export EASYBUILD_DETECT_LOADED_MODULES=purge
$ eb example.eb
== temporary log file in case of crash /tmp/eb-QLTV9v/easybuild-6mOmIN.log

WARNING: Found non-allowed loaded (EasyBuild-generated) modules (Spack/0.10.0),␣
↪running 'module purge'

...
```

Note: Whether or not `module purge` is a suitable action is site-specific, since this will unload *all* loaded modules (except for 'sticky' modules when Lmod is used), including modules that were not installed with EasyBuild and which may be always required.

Configuring EasyBuild to use `module purge` when (non-allowed) loaded modules are found should *not* be done on Cray systems, since it will result in a broken environment.

`unload`: unload loaded modules

When `--detect-loaded-modules=unload` is used, EasyBuild will only unload the (non-allowed & EasyBuild-generated) modules. The modules are unloaded in reverse order, i.e. the last loaded module is unloaded first.

This is an alternative to using `module purge`, in case some other (allowed) modules are loaded and should remain loaded.

A short warning message is printed when loaded modules are unloaded:

```
$ eb example.eb --detect-loaded-modules=unload
== temporary log file in case of crash /tmp/eb-JyyaEF/easybuild-WyGqZs.log

WARNING: Unloading non-allowed loaded (EasyBuild-generated) modules: Spack/0.10.0

...
```

`warn`: print warning and continue upon detection of loaded modules

When EasyBuild is configured with `--detect-loaded-modules=warn`, EasyBuild will print a warning mentioning that one or more loaded (EasyBuild-generated) were detected, before continuing as normal.

The warning is intended to make the user aware that the environment in which EasyBuild is being run is not clean.

For example:

```
$ export EASYBUILD_DETECT_LOADED_MODULES=warn
$ eb example.eb
== temporary log file in case of crash /tmp/eb-9HD20m/easybuild-WAYzK2.log

WARNING: Found one or more non-allowed loaded (EasyBuild-generated) modules in␣
↪current environment:
* Spack/0.10.0

To make EasyBuild allow particular loaded modules, use the --allow-loaded-modules␣
↪configuration option.
Use --detect-loaded-modules={error,ignore,purge,unload,warn} to specify action to␣
↪take when loaded modules are detected.

See http://easybuild.readthedocs.io/en/latest/Detecting_loaded_modules.html for more␣
↪information.

...
```

Note: This is the default behaviour in EasyBuild v3.3.0 and newer.

6.7.4 Allowing particular loaded modules

By default, only EasyBuild itself is considered as an allowed module.

EasyBuild can be configured to allow particular modules to be loaded via `--allow-loaded-modules`, by specifying a comma-seperated list of software names.

For example:

```
$ export EASYBUILD_DETECT_LOADED_MODULES=error
$ export EASYBUILD_ALLOW_LOADED_MODULES=EasyBuild,GC3Pie

$ module load EasyBuild
$ module load GC3Pie
$ eb example.eb
...
```

When `--allow-loaded-modules` is used, the EasyBuild module is no more allowed by default and must be explicitly listed if you want to allow an EasyBuild installation provided through a module.

6.7.5 Checking of `$EBROOT*` environment variables

The detection mechanism for loaded modules relies on defined `$EBROOT*` environment variables, and determining by which loaded module they were set.

When one or more `$EBROOT*` environment variables are found for which the corresponding loaded modules can not be found, this can lead to problems.

Hence, EasyBuild will detect this and act on it, according to the value specified to `--check-ebroot-env-vars`:

- *error: report error and stop EasyBuild on unknown $EBROOT* environment variables*
- *ignore: ignore unknown $EBROOT* environment variables*
- *unset: unset unknown $EBROOT* environment variables and print warning*
- *warn: print warning for unknown $EBROOT* environment variables and continue* (current default)

`error`: report error and stop EasyBuild on unknown `$EBROOT*` environment variables

When configured with `--check-ebroot-env-vars=error`, EasyBuild will stop with a clear error message when it discovers one or more `$EBROOT*` environment variables that do not correspond to a loaded module:

```
$ export EBROOTUNKNOWN=just_an_example

$ eb example.eb --check-ebroot-env-vars=error

== temporary log file in case of crash /tmp/eb-cNqOzG/easybuild-xmV8vc.log
ERROR: Found defined $EBROOT* environment variables without matching loaded module:
↪$EBROOTUNKNOWN
(control action via --check-ebroot-env-vars={error,ignore,unset,warn})
```

`ignore`: ignore unknown `$EBROOT*` environment variables

To simply ignore any defined `$EBROOT*` environment variables that do not correspond with a loaded module, configure EasyBuild with `--check-ebroot-env-vars=ignore`.

`unset`: unset unknown `$EBROOT*` environment variables and print warning

If you are confident that the defined `$EBROOT*` environment variables for which no matching loaded module was found are harmless, you can specify that EasyBuild should clean up the environment by unsetting them, before continuing. A clear warning message will be printed whenever this occurs:

```
$ export EBROOTUNKNOWN=just_an_example

$ eb bzip2-1.0.6.eb --check-ebroot-env-vars=unset

== temporary log file in case of crash /tmp/eb-IvXik8/easybuild-cjHjhs.log

WARNING: Found defined $EBROOT* environment variables without matching loaded module:
↪$EBROOTUNKNOWN; unsetting them

...
```

Note that these unknown `$EBROOT*` will only be unset in the environment of the running EasyBuild session, not in the current session in which `eb` is being run.

`warn`: print warning for unknown `$EBROOT*` environment variables and continue

If EasyBuild was configured with `--check-ebroot-env-vars=warn`, a warning will be printed when one or more defined `$EBROOT*` environment variables are encountered for which no corresponding loaded module was found:

```
$ export EBROOTUNKNOWN=just_an_example

$ export EASYBUILD_CHECK_EBROOT_ENV_VARS=warn
$ eb example.eb

== temporary log file in case of crash /tmp/eb-1h_LQ9/easybuild-BHrPk4.log
WARNING: Found defined $EBROOT* environment variables without matching loaded module:
↪$EBROOTUNKNOWN
(control action via --check-ebroot-env-vars={error,ignore,unset,warn})

...
```

Note: This is the default behaviour in EasyBuild v3.3.0 and newer.

6.8 Local variables in easyconfig files

This page discusses the use of *local variables in easyconfig files*.

For more general information on writing easyconfig files, please see *Writing easyconfig files: the basics*.

Contents

6.8.1 Motivation & context

When composing easyconfig files, it can sometimes make sense to use one or more *local variables* to define easyconfig parameters in a cleaner way, for example to avoid copy-pasting values that are used multiple times, or to avoid too long lines (longer than the maximum width of 120 characters as specified in code style). A local variable within the context of easyconfig files is any variable that does not correspond to a known easyconfig parameter (either general of easyblock-specific).

One thing that is easily overlooked is the importance of how these local variables are named: you should try and avoid that the names of local variables will match with the name of an easyconfig parameter that may get introduced in future EasyBuild versions. If that would happen, the semantics of the easyconfig file will change which may either result in a broken installation or perhaps a different installation than was intended (which could be worse than a broken one).

Avoiding that names of local variables are *close* to the name of (future) easyconfig parameters is also important, because of the "typo detection" feature that EasyBuild has: if the name of a local variable is too close to a known easyconfig parameter, EasyBuild will assume you have made a mistake, and will report a typo error, for example:

```
ERROR: Failed to process easyconfig example.eb:
You may have some typos in your easyconfig file: configopt -> configopts
```

6.8.2 Changes in EasyBuild v4.0 w.r.t. local variables in easyconfig files

Some changes were made in EasyBuild v4.0 to discourage the use of poorly named local variables:

- a *Recommended naming scheme for local variables in easyconfig files* was defined;
- a mechanism was implemented to detect the use of local variables in easyconfig files, which will print a *Warning for local variables that do not follow the recommended naming scheme*
- EasyBuild can be configured to report an error for local variables that do not follow the recommended naming scheme; see easyconfig_files_local_variables_strict_naming
- using `eb --fix-deprecated-easyconfigs`, the names of local variables can be changed automatically to match the recommended naming scheme; see *Renaming local variables to match the recommended naming scheme using eb –fix-deprecated-easyconfigs*

6.8.3 Recommended naming scheme for local variables in easyconfig files

To ensure that local variables used in easyconfig files do not clash with easyconfig parameters that get added in future EasyBuild versions, we have defined a recommended naming scheme for local variables.

Names of local variables should either:

- **start with** `local_`, to make it explicit that it is a *local* variable (examples: `local_example`, `local_var`)
- **consist of a single letter** (examples: `f`, `l`, `x`); this is typically only done within the context of a list comprehension
- **start with a single underscore** (_), which matches the common convention in Python code that these variables are only for "internal use" (examples: `_example`, `_local_var`)

Note: A check was added in EasyBuild v4.0 to ensure that the names of known easyconfig parameters do *not* conform to any of these rules, to ensure that local variables can always be discriminated from known easyconfig parameters.

6.8.4 Warning for local variables that do not follow the recommended naming scheme

By default, EasyBuild will produce a clear warning when one or more local variables are used that do not conform to the recommended naming scheme easyconfig_files_local_variables_naming_scheme.

For example, when using an easyconfig file that includes a local variable named `var`:

```
$ eb example.eb
...
WARNING: Use of 1 unknown easyconfig parameters detected in example.eb: var
If these are just local variables please rename them to start with 'local_', or try␣
↪using --fix-deprecated-easyconfigs to do this automatically.
```

To get rid of this warning, you can either:

- rename the local variable, either manually to something like `local_var` or using `eb --fix-deprecated-easyconfigs example.eb` (see also *Renaming local variables to match the recommended naming scheme using eb –fix-deprecated-easyconfigs*)
- configure EasyBuild to only log the warning (not print it), via `--local-var-naming-check=log` (see also *Specifying what should be done when non-confirming local variables are found via –local-var-naming-check*; note that silencing the printed warning is **not recommended**, see the motivation in motivation above easyconfig_files_local_variables_motivation

6.8.5 Specifying what should be done when non-confirming local variables are found via `--local-var-naming-check`

Using the `--local-var-naming-check` configuration option, you can specify what should be done when one or more local variables are found that do not follow the recommended naming scheme easyconfig_files_local_variables_naming_scheme:

- `--local-var-naming-check=error`: **fail with an error** if any easyconfig file that was parsed includes local variables that do not follow the recommended naming scheme;
- `--local-var-naming-check=log`: only *log* a warning (but do not print it) if any easyconfig file that was parsed includes local variables that do not follow the recommended naming scheme;
- `--local-var-naming-check=warn` *[default]*: *print* a warning if any easyconfig file that was parsed includes local variables that do not follow the recommended naming scheme;

The default is set such that non-confirming local variables are only reported through a printed warning, but do not result in cancelling the installation (since they're usually not actually problematic).

6.8.6 Renaming local variables to match the recommended naming scheme using `eb --fix-deprecated-easyconfigs`

To fix one or more easyconfig files that includes local variables that do not follow the recommended naming scheme easyconfig_files_local_variables_naming_scheme, `eb --fix-deprecated-easyconfigs` can be used.

For example:

```
eb --fix-deprecated-easyconfigs bzip2.eb zlib.eb
== temporary log file in case of crash /tmp/eb-Z7r_KJ/easybuild-dHtPY4.log

* [1/2] fixing /tmp/example/bzip2.eb... FIXED!
  (changes made in place, original copied to /tmp/example/bzip2.eb.orig_
↪20190815180106_53972)

* [2/2] fixing /tmp/example/zlib.eb... FIXED!
  (changes made in place, original copied to /tmp/example/zlib.eb.orig_20190815180106_
↪53972)

All done! Fixed 2 easyconfigs (out of 2 found).
```

(continues on next page)

(continued from previous page)

```
== Temporary log file(s) /tmp/eb-Z7r_KJ/easybuild-dHtPY4.log* have been removed.
== Temporary directory /tmp/eb-Z7r_KJ has been removed.
```

There are a couple of caveats here though...

While `--fix-deprecated-easyconfigs` can be very useful, it's certainly not perfect since all it does is simple search and replace of the names of non-conforming local variables (as whole words) to prefix them with `local_`.

This means that it may make some unintended changes, so make sure to **always double check which changes were made!**

In addition, it sometimes make more sense to simply *eliminate* the local variable rather than renaming it, for example when it wasn't really needed at all: maybe it was only actually used once, or maybe using a template like `%(pyver)s` or `(pyshortver)s` (see also avail_easyconfig_templates) renders it obsolete.

6.9 Using an index to speed up searching for easyconfigs

EasyBuild often needs to search for *Easyconfig files* (or accompanying files like patches), either based on command line arguments (like the name of an easyconfig file) or options to the `eb` command (like `--search`, see *Searching for easyconfigs, –search / -S*), or to resolve dependencies for a particular easyconfig file that was parsed already.

Searching for easyconfig files or patches may take a while, since it can potentially involve weeding through thousands of files, which may be located on a shared filesystem (where metadata-intensive operations like file searching can be rather slow).

If EasyBuild turns out to be sluggish when searching for easyconfig files in your setup, using an *index* file could make a big difference.

Note: Support for index files was implemented in EasyBuild version 4.2.0 .

Contents

6.9.1 Creating an index (`--create-index`)

`eb --create-index` can be used to create an index file for a particular directory that holds a (large) set of easyconfig files.

The index file (a hidden file named `.eb-path-index`) will be created in the specified directory. It will contain a list of (relative) paths for all files in that directory, along with some metadata: the time at which the index file was created and a timestamp that indicates how long the index is considered to be valid (see *Controlling how long the index is valid (–index-max-age)*).

Note: Make sure to create an index for the correct path.

The search for easyconfig files performed by EasyBuild will *not* recurse into subdirectories of the locations it considers (see *Searching for easyconfigs: the robot search path*), other than those with a name matching the software for which it is trying to find an easyconfig file (like `t/TensorFlow/` when searching for an easyconfig file for `TensorFlow`).

Hence, if the directory housing your easyconfig files has an `easybuild/easyconfigs` subdirectory (for example, a working copy of the `easybuild-easyconfigs` repository), create the index *in* that subdirectory, rather than in the higher-level directory.

Example of creating an index

```
$ eb --create-index $HOME/easybuild-easyconfigs/easybuild/easyconfigs
== temporary log file in case of crash /tmp/eb-tUu6_w/easybuild-SKBnVO.log
Creating index for /home/example/easybuild-easyconfigs/easybuild/easyconfigs...
== found valid index for /home/example/easybuild-easyconfigs/easybuild/easyconfigs,␣
↪so using it...
Index created at /home/example/easybuild-easyconfigs/easybuild/easyconfigs/.eb-path-
↪index (738 files)

$ head -n 5 $HOME/easybuild-easyconfigs/easybuild/easyconfigs/.eb-path-index
# created at: 2020-04-13 14:19:57.008981
# valid until: 2020-04-20 14:19:57.008981
a/Arrow/Arrow-0.16.0-intel-2019b-Python-3.7.4.eb
b/Boost/Boost-1.71.0-gompi-2019b.eb
b/bokeh/bokeh-1.4.0-foss-2019b-Python-3.7.4.eb
```

Note: The "`found valid index ...`" message being printed occurs because EasyBuild tries to load the index file right after creating it, as a sanity check.

6.9.2 Updating an existing index (`--create-index --force`)

To update an existing index, you can use `--create-index --force`.

Without using `--force`, EasyBuild will refuse to overwrite the existing index file:

```
$ eb --create-index $HOME/easybuild-easyconfigs/easybuild/easyconfigs
== temporary log file in case of crash /tmp/eb-tUu6_w/easybuild-SKBnVO.log
Creating index for /home/example/easybuild-easyconfigs/easybuild/easyconfigs...
ERROR: File exists, not overwriting it without --force: /home/example/easybuild-
↪easyconfigs/easybuild/easyconfigs/.eb-path-index
```

6.9.3 Using index files

EasyBuild will automatically pick up and use any index file that it runs into while searching for easyconfig files or patches. If an index file is found, it will be preferred over walking through the directory tree to check for the target file, which is likely to significantly speed up the search operation.

When a (valid) index file is found for a particular path, a message will be printed mentioning "`found valid index for...`":

```
 $ eb --search TensorFlow-2.1.0-foss-2019b
 == found valid index for /home/example/easybuild-easyconfigs/easybuild/easyconfigs,␣
↪so using it...
* /home/example/easybuild-easyconfigs/easybuild/easyconfigs/t/TensorFlow/TensorFlow-2.
↪1.0-foss-2019b-Python-3.7.4.eb
```

6.9.4 Ignoring indices (`--ignore-index`)

One potential issue with having an index in place is that it may get outdated: new files may have been added to the directory since the index was created or last updated.

If updating the indexes is not an option (see *Updating an existing index (–create-index –force)*), you can instruct EasyBuild to ignore any existing indices using the `--ignore-index` configuration option.

The only downside of this option is that searching for easyconfig files may be significantly slower. Any existing index files are left untouched (they will *not* be updated or removed).

6.9.5 Controlling how long the index is valid (`--index-max-age`)

When creating an index file, you can specify how long the index should be considered valid.

Using the `--index-max-age` configuration option, you can specify how long after the creation time the index remains valid (in seconds).

By default, EasyBuild will consider index files to remain valid for 1 week (7 * 24 * 60 * 60 = 604,800 seconds).

To create an index that *always* remains valid (never expires), use zero (`0`) as value for `--index-max-age`:

```
$ eb --index-max-age=0 --create-index $HOME/easybuild-easyconfigs/easybuild/
↪easyconfigs

$ head -n 2 $HOME/easybuild-easyconfigs/easybuild/easyconfigs/.eb-path-index
# created at: 2020-04-13 15:10:07.173191
# valid until: 9999-12-31 23:59:59.999999
```

Note: Trust us here, December 31st 9999 is the end of times. Better get prepared.

6.9.6 Index included with EasyBuild releases

Each EasyBuild release (since EasyBuild v4.2.0) comes with an index file for the easyconfig (and patch) files that are included with that release.

Hence, you only need to use `--create-index` to create/update the index file for any additional directories with easyconfig files you may have on the side (and only if searching those easyconfig files is rather slow).

6.9.7 Should I create an index?

Whether or not you should create an index file for your directories housing additional easyconfig files depends on a number of factors, including:

- how often files are added and/or removed in those directories, since files listed in the index are assumed to be there and any files not included in the index will be overlooked by EasyBuild when it's searching for files;
- the filesystem on which those directories are located, since an index file will only make a significant difference on filesystems where metadata-intensive operations are relatively slow;
- how many files there are in those directories, since performance benefits will only be apparent if the number if files is sufficiently large;

Note: Keep in mind that creating an index implies also updating it frequently, to ensure that EasyBuild will take all available files in account.

6.10 Easystack files

This documentation covers aspects of specifying a software stack to install with Easybuild with *easystack files*.

Note: this is an *experimental feature*. **Some of the mentioned functionality may be subject to change or be prone to errors.**

Contents

6.10.1 The basics

Easystack files describe an entire software stack, and can be used to specify to EasyBuild what to install.

6.10.2 Usage

To build software with *Easystack*, type:

```
eb --easystack example.yaml --experimental
```

where `example.yaml` is the file with specifications that you just created (more about this in the next section).

6.10.3 Structure of an easystack file

Easystack files are written in YAML syntax.

General options, which should be applied to each software (for example `robot`), are defined at the top of the file.

Afterwards, particular software specifications follow.

It is mandatory to specify basic software-related keywords: *software name*, *toolchains* and *versions*.

- *software name*: Name of the software.
- *toolchains*: Names and versions of compiler *Toolchains*.
- *versions*: Versions of software. If multiple entries are provided, EasyBuild will install all of them.

Can be in form of a list or consecutive line entries (see example).

General structure of YAML-formatted easystack:

```
software:
  <software_name>:
    toolchains:
      <toolchain name and version>:
          <software_version>:
            versionsuffix: '<-example>'
```

Example of YAML-formatted easystack:

```
software:
  Bioconductor:
    toolchains:
      foss-2020a:
        versions:
          3.11:
  EasyBuild:
    toolchains:
      SYSTEM:
        versions: [4.3.1]
  GROMACS:
    toolchains:
      foss-2020a:
        versions:
          2020.1:
          2020.3:
      fosscuda-2020a:
        versions: [2020.1]
  OpenFOAM:
    toolchains:
      foss-2020a:
        versions: [8, v2006]
  R:
    toolchains:
      foss-2020a:
        versions: [4.0.0]
```

To install the software specified in this *easystack file* named '`myeasystack.yaml`', run:

```
eb --easystack myeasystack.yaml
```

This is equivalent to running:

```
eb Bioconductor-3.11-foss-2020a.eb EasyBuild-4.3.1-SYSTEM.eb GROMACS-2020.1-foss-
↪2020a.eb GROMACS-2020.3-foss-2020a.eb GROMACS-2020.1-fosscuda-2020a.eb OpenFOAM-8-
↪foss-2020a.eb OpenFOAM-v2006-foss-2020a.eb R-4.0.0-foss-2020a.eb
```

6.10.4 To be developed

Optionally, more advanced keywords can be specified: *easybuild_version*, *robot*, *from_pr*, *versionsuffix*, *include-labels*, *exclude-labels*.

- *easybuild_version:* if present, EasyBuild will check if the easystack file was intended for the current version of EasyBuild.
- *robot:* enables dependency resolution; see *Using the EasyBuild command line* for more details.
- *from_pr:* easyconfig files that are added or modified by a particular pull request to the easybuild-easyconfigs GitHub repository can be used (regardless of whether the pull request is merged or not). (see *Integration with GitHub* for more details).
- *versionsuffix:* additional suffix for software version (placed after toolchain name)
- *include-labels:* only include this software when EasyBuild is configured with one of the specified labels
- *exclude-labels:* **do not** include this software when EasyBuild is configured with one of the specified labels

6.11 Experimental features

First introduced in EasyBuild v2.1.0 (see *EasyBuild v2.1.0 Release Notes*), experimental features can only be used by enabling the `--experimental` configuration option.

An experimental feature indicates to users that these features may change significantly in a future release and should be used only for testing, not (yet) for production usage.

Currently enabled experimental features include:

- support for easyconfig files in YAML syntax (see easyconfig_yeb_format)
- support for generating container recipes & images (see *Generating container recipes & images*)
- support for using easystack files (see *Easystack files*)

6.12 Extended dry run

Using `--extended-dry-run` or `-x` (supported since EasyBuild v2.4.0, see release notes for *EasyBuild v2.4.0 (November 10th 2015)*), a detailed overview of the build and install procedure that EasyBuild is going to execute can be obtained almost instantly.

All time-consuming operations, including executing commands to configure/build/install the software, are only *reported* rather than being actually performed.

Example output is available at extended_dry_run_examples.

Contents

6.12.1 Important notes

There are a couple of things you should be aware of when using `--extended-dry-run` and interpreting the output it produces.

Build/install procedure reported by dry run may be (slightly) different

The actual build and install procedure may (slightly) differ from the one reported by `--extended-dry-run`, due to conditional checks in the easyblock being used.

For example, expressions that are conditional on the presence of certain files or directories in the build directory will always be false, since the build directory is never actually populated.

Errors are ignored (by default) during dry run

Any errors that occur are ignored, and are reported with a clear warning message. This is done because it is possible that these errors occur because of the dry run mechanism.

For example, the install step could assume that certain files created by a previous step will be present, but they will not be there since the commands that are supposed to produce them were not actually performed in dry run mode.

Errors are ignored *on a per-step basis*. When an error is ignored in a particular step, that step is aborted, which may result in partial dry run output for that particular step. Subsequent steps will still be run (in dry run mode), however.

Since it's possible that these errors occur due to a bug in the easyblock being used, it's important to pay attention to these ignored errors.

Ignored errors are reported as follows, for example:

```
== testing... [DRY RUN]

[test_step method]
!!!
!!! WARNING: ignoring error "[Errno 2] No such file or directory: 'test'"
!!!
```

At the end of dry run output, anonother warning message is shown if any ignored errors occurred:

```
== COMPLETED: Installation ended successfully

!!!
!!! WARNING: One or more errors were ignored, see warnings above
!!!
```

Disabling ignoring errors during dry run

Ignoring errors that occur during a dry run is enabled by default; it can be disabled using the configuration option that is available for it, i.e. by:

- the `--disable-extended-dry-run-ignore-errors` command line option
- by defining the `$EASYBUILD_DISABLE_EXTENDED_DRY_RUN_IGNORE_ERRORS` environment variable
- or by defining `disable-extended-dry-run-ignore-errors` in an EasyBuild configuration file

(see also *Configuring EasyBuild*)

6.12.2 Overview of dry run mechanism

During an extended dry run, several operations are not performed, or are only simulated.

The sections below give a detailed overview of the dry run mechanism.

Temporary directories as build/install directories

To make very sure that EasyBuild does not touch any files or directories during the dry run, the build and (software/module) install directories are replaced by subdirectories of the temporary directory used by that particular EasyBuild session.

In the background, the values for `self.builddir`, `self.installdir` and `self.installdir_mod` are changed in the `EasyBlock` instance(s) being used; this also affects the use of the `%(builddir)s` and `$(installdir)s` values in easyconfig files.

Although the build and install directories are effectively temporary directories during a dry run (under a prefix like `/tmp/eb-aD_yNu/__ROOT__`), this is not visible in the dry run output: the 'fake' build and install directories are replaced by the corresponding original value in the dry run output. For example:

```
[extract_step method]
  running command "tar xzf /home/example/easybuild/sources/b/bzip2/bzip2-1.0.6.tar.gz"
  (in /tmp/example/eb_build/bzip2/1.0.6/GCC-4.9.2)
```

Note on build directory in dry run mode

The build (sub)directory used during an actual (non-dry run) EasyBuild session may be different than the one mentioned in the dry run output.

This is because during a dry run, EasyBuild will *guess* the name of the subdirectory that is created by unpacking the first source file in the build directory as being `<name>-<version>`. Although this is a common pattern, it is not always 100% correct.

For example, you may see this in the dry run output for WRF (for which a build-in-installdir procedure is used):

```
[build_step method]
  running command "tcsh ./compile -j 4 wrf"
  (in /home/example/eb/software/WRF/3.6.1-intel-2015a-dmpar/WRF-3.6.1)
```

The actual build (and install) subdirectory is slightly different while not in dry run mode however, i.e.: `/home/example/eb/software/WRF/3.6.1-intel-2015a-dmpar/WRFV3`.

No downloading of missing source files/patches

Required files (source files/patches) are not downloaded during a dry run if they are not available yet.

The dry run output will specify whether files are found (and if so, at which path) or not; the exact output for files that were not found depends on whether or not source URLs are available.

For example: if the required source file for `bzip2` is not available yet, it is indicated where EasyBuild will try to download it to:

```
[fetch_step method]
Available download URLs for sources/patches:
  * http://www.bzip.org/1.0.6/$source

List of sources:
  * bzip2-1.0.6.tar.gz downloaded to /home/example/easybuild/sources/b/bzip2/bzip2-1.
↪0.6.tar.gz

List of patches:
(none)
```

If the source file is already available in the source path that EasyBuild was configured with, it is indicated as such:

```
List of sources:
  * bzip2-1.0.6.tar.gz found at /home/example/easybuild/sources/b/bzip2/bzip2-1.0.6.
↪tar.gz
```

In case no source URLs are available and required files are missing, they are simply marked as such:

```
Available download URLs for sources/patches:
(none)

List of sources:
  * bzip2-1.0.6.tar.bz2 (MISSING)
```

However, since the dry run mechanism never actually uses the source files/patches, this does not affect the remainder of the output of `--extended-dry-run`/`-x`.

Checksum verification is skipped

Computing checksums of sources files/patches, and verifying them against specified checksums (if available) is *skipped* during a dry run, because it is considered potentially too time-consuming. In addition, source files/patches may not be available anyway.

If checksums are available they are only reported, for example (for GCC v4.9.3):

```
[checksum_step method]
* expected checksum for gcc-4.9.3.tar.bz2: 6f831b4d251872736e8e9cc09746f327
* expected checksum for gmp-6.0.0a.tar.bz2: b7ff2d88cae7f8085bd5006096eed470
* expected checksum for mpfr-3.1.2.tar.gz: 181aa7bb0e452c409f2788a4a7f38476
* expected checksum for mpc-1.0.2.tar.gz: 68fadff3358fb3e7976c7a398a0af4c3
* expected checksum for mpfr-3.1.2-allpatches-20141204.patch:␣
↪58aec98d15982f9744a043d2f1c5af82
```

Source files are not unpacked

Source files are *not* unpacked, since this may require too much time (in case of large source files). Additionally, source files may not be available anyway.

This has a number of implications:

- files or directories that may be expected to be there are not, which may lead to (ignored) errors if the used easyblock does not take this into account (see also *Errors are ignored (by default) during dry run*)
- the build directory in which commands are executed may be incorrect in the dry run output (see also *Note on build directory in dry run mode*)

The extraction command is mentioned in the dry run output however, for example:

```
[extract_step method]
  running command "tar xjf bzip2-1.0.6.tar.bz2"
  (in /tmp/example/eb_build/bzip2/1.0.6/GCC-4.9.2)
```

Patch files are not applied, no runtime patching

Since source files are not unpacked, patch files can not applied either.

The dry run output does provide an overview of patch files, together with where they are found and how they are applied:

```
[patch_step method]
* applying patch file WRF_parallel_build_fix.patch
  running command "patch -b -p<derived> -i /home/example/easybuild/sources/w/WRF/WRF_
↪parallel_build_fix.patch"
  (in /home/example/easybuild/easybuild/software/WRF/3.6.1-intel-2015a-dmpar)
* applying patch file WRF-3.6.1_known_problems.patch
  running command "patch -b -p<derived> -i /home/example/easybuild/sources/w/WRF/WRF-
↪3.6.1_known_problems.patch"
  (in /home/example/easybuild/easybuild/software/WRF/3.6.1-intel-2015a-dmpar)
```

Likewise, runtime patching performed by the easyblock itself can not work either. If the `apply_regex_substitutions` function (available from `easybuild.tools.filetools`) is used, a clear overview is included in the dry run output (see also *Runtime patching of files: apply_regex_substitutions*).

For example, in the `configure` step of the WRF easyblock when using the Intel compilers, this yields:

```
[configure_step method]
...
applying regex substitutions to file configure.wrf
  * regex pattern '^(DM_FC\s*=\s*).*$', replacement string '\1 mpif90'
  * regex pattern '^(DM_CC\s*=\s*).*$', replacement string '\1 mpicc -DMPI2_SUPPORT'
```

If the `apply_regex_substitutions` function provided for runtime patching is not used (and `fileinput` is used directly, for example), runtime patching performed by the easyblock will most likely result in an error, leading to the step in which it is being performed being aborted (see *Errors are ignored (by default) during dry run*).

Module `load` statements are executed or simulated

`module load` statements are either effectively executed or simulated, dependending on whether the corresponding module files are available or not.

Available modules are loaded

`module load` statements are fairly light-weight, so they are effectively executed if the module being loaded is available.

The dry run output includes an overview of the modules being loaded. In addition an overview of all loaded modules, including the ones that were loaded indirectly, is shown.

For example:

```
[prepare_step method]
Defining build environment, based on toolchain (options) and specified dependencies...

Loading toolchain module...

module load GCC/4.9.2

Loading modules for dependencies...

module load M4/1.4.17-GCC-4.9.2

Full list of loaded modules:
  1) GCC/4.8.2
  2) M4/1.4.17-GCC-4.9.2
```

Loading of non-available modules is simulated

If the module file required to execute a particular `module load` statement is not available, the dry run mechanism will *simulate* the loading of the module.

The `module load` statements that were simulated rather than actually performed are clearly indicated using `[SIMULATED]` in the dry run output, for example:

```
[prepare_step method]
Defining build environment, based on toolchain (options) and specified dependencies...

Loading toolchain module...

```

(continues on next page)

(continued from previous page)

```
module load intel/2015a

Loading modules for dependencies...

module load JasPer/1.900.1-intel-2015a
module load netCDF/4.3.2-intel-2015a [SIMULATED]
module load netCDF-Fortran/4.4.0-intel-2015a [SIMULATED]
module load tcsh/6.18.01-intel-2015a
```

Only modules that were effectively loaded will appear in the full list of modules being printed; modules for which the load was simulated will not be included.

Simulated loading of non-available *dependency* modules

For dependencies, simulating a `module load` statement basically (only) entails defining the `$EBROOT*` and `$EBVERSION*` environment variables (the full variable names are determined by the software name), which are picked up by resp. the `get_software_root` and `get_software_version` functions often used in easyblocks.

The `$EBVERSION*` environment variable is defined with the actual software version of the dependency.

For the `$EBROOT*` environment variable, the name of the environment variable itself prefixed with a '`$`' is used as a dummy value, rather than using an fake installation software prefix. For example, when simulating the load statement for a `GCC` module, the environment variable `$EBROOTGCC` is defined as the string value `'$EBROOTGCC'` (literally).

This results in sensible output when this value is picked up via `get_software_root` by the easyblock.

For example, for netCDF used as a dependency for WRF the following is included in the module file contents included in the dry run output:

```
setenv  NETCDF          "$EBROOTNETCDF"
setenv  NETCDFF         "$EBROOTNETCDFMINFORTRAN"
```

Simulated loading of non-available *toolchain* module

When the module that corresponds to the toolchain being used is not available, the dry run mechanism will also simulate the `module load` statements for the individual toolchain components, to ensure that version checks on the toolchain components can work as expected.

For example, if the toolchain module `intel/2015a` is not available, the loading of the `icc`, `ifort`, `impi` and `imkl` modules that would be loaded by the `intel` module is also simulated:

```
[prepare_step method]
Defining build environment, based on toolchain (options) and specified dependencies...

Loading toolchain module...

module load icc/2015.1.133-GCC-4.9.2 [SIMULATED]
module load ifort/2015.1.133-GCC-4.9.2 [SIMULATED]
module load impi/5.0.2.044-iccifort-2015.1.133-GCC-4.9.2 [SIMULATED]
module load imkl/11.2.1.133-iimpi-7.2.3-GCC-4.9.2 [SIMULATED]
module load intel/2015a [SIMULATED]
```

Build environment is reported

The build environment that is set up based on the toolchain (and toolchain options) being used, and the dependencies being loaded is reported as a part of the dry run output.

For example, when `GCC` is used as a toolchain something like this will be included in the `prepare_step` part of the dry run output:

```
Defining build environment...

  export CC="gcc"
  export CFLAGS="-O2"
  export CXX="g++"
  export CXXFLAGS="-O2"
  export F77="gfortran"
  export F90="gfortran"
  export F90FLAGS="-O2"
  export FFLAGS="-O2"
  export FLIBS="-lgfortran"
  export LDFLAGS="-L/home/example/eb/software/GCC/4.8.2/lib"
  export LIBS="-lm -lpthread"
  export OPTFLAGS="-O2"
  export PRECFLAGS=""
```

This is particularly useful as an overview of which environment variables that are defined by the toolchain mechanism, and to assess the effect of changing toolchain options.

The output is deliberately formatted such that is can be easily copy-pasted, which can be useful to mimic the environment in which EasyBuild will perform the build and install procedure.

Shell commands are not executed

Any shell commands that are executed via the `run_cmd` and `run_cmd_qa` functions that are provided by the EasyBuild framework via the `easybuild.tools.run` are *not* executed, only reported (see also *Executing commands: run_cmd and run_cmd_qa*).

This typically includes the commands that are defined in the easyblock to be run as a part of the configure/build/install steps.

For example:

```
configuring... [DRY RUN]

[configure_step method]
  running command " ./configure --prefix=/home/example/eb/software/make/3.82-GCC-4.8.
↪2 "
  (in /home/example/eb/build/make/3.82/GCC-4.8.2/make-3.82)

building... [DRY RUN]

[build_step method]
  running command " make -j 4 "
  (in /home/example/eb/build/make/3.82/GCC-4.8.2/make-3.82)

...

installing... [DRY RUN]
```

(continues on next page)

(continued from previous page)

```
[stage_install_step method]

[make_installdir method]

[install_step method]
  running command " make install "
  (in /home/example/eb/build/make/3.82/GCC-4.8.2/make-3.82)
```

There are a couple of minor exceptions though. Some (light-weight) commands are always run by the EasyBuild framework, even in dry run mode, and an easyblock can specify that particular commands *must* always be run (see also *Executing commands: run_cmd and run_cmd_qa*).

Sanity check paths/commands are not checked

Since nothing is actually being installed during a dry run, the sanity check paths/commands can not be checked.

Instead, the dry run mechanism will produce a clear overview of which paths are expected to be found in the installation directory, and which commands are expected to work (if any).

For example:

```
sanity checking... [DRY RUN]

[sanity_check_step method]
Sanity check paths - file ['files']
  * WRFV3/main/ideal.exe
  * WRFV3/main/libwrflib.a
  * WRFV3/main/ndown.exe
  * WRFV3/main/nup.exe
  * WRFV3/main/real.exe
  * WRFV3/main/tc.exe
  * WRFV3/main/wrf.exe
Sanity check paths - (non-empty) directory ['dirs']
  * WRFV3/main
  * WRFV3/run
Sanity check commands
  (none)
```

Module file is incomplete and only printed

During a dry run, the contents of the module file that would be installed is still generated, but only printed; it is not actually written to file.

More importantly however, the module file being reported is bound to be **incomplete**, since the module generator only includes certain statements conditionally, for example only if the files/directories to which they relate actually exist. This typically affects `prepend-path` statements, e.g. for `$PATH`, `$LD_LIBRARY_PATH`, etc.

For example, the reported module file for make v3.82 built with `GCC/4.8.2` may look something like:

```
creating module... [DRY RUN]

[make_module_step method]
Generating module file /home/example/eb/modules/all/make/3.82-GCC-4.8.2, with␣
↪contents:
```

(continues on next page)

(continued from previous page)

```
    #%Module
    proc ModulesHelp { } {
        puts stderr { make-3.82: GNU version of make utility - Homepage: http://www.
↪gnu.org/software/make/make.html
        }
    }

    module-whatis {Description: make-3.82: GNU version of make utility - Homepage:␣
↪http://www.gnu.org/software/make/make.html}

    set root /home/example/eb/software/make/3.82-GCC-4.8.2

    conflict make

    if { ![ is-loaded GCC/4.8.2 ] } {
        module load GCC/4.8.2
    }

    setenv  EBROOTMAKE              "$root"
    setenv  EBVERSIONMAKE           "3.82"
    setenv  EBDEVELMAKE             "$root/easybuild/make-3.82-GCC-4.8.2-easybuild-
↪devel"

    # Built with EasyBuild version 2.4.0
```

Note that there is no `prepend-path PATH` statement for the `bin` subdirectory, for example.

6.12.3 Guidelines for easyblocks

To ensure useful output under `--extended-dry-run`, easyblocks should be implemented keeping in mind that some operations are possible not performed, to avoid generating errors in dry run mode.

Although errors are just ignored by the dry run mechanism on a per-step basis, they may hide subsequent operations and useful information for the remainder of the step (see also *Errors are ignored (by default) during dry run*).

Detecting dry run mode and enhancing the dry run output

To detect whether an easyblock is being used in dry run mode, it suffices to check the `self.dry_run` class variable.

Additional messages can be included in the dry run output using the `self.dry_run_msg` method.

For example:

```
class Example(EasyBlock):

    def configure_step(self):

        if self.dry_run:
            self.dry_run_msg("Dry run mode detected, not reading template␣
↪configuration files")
            ...
```

Check whether files/directories exist before accessing them

Rather than assuming that particular files or directories will be there, easyblocks should take into that they may not be, for example because EasyBuild is being run in dry run mode.

For example, instead of simply assuming that a directory named '`test`' will be there, the existence should be checked first. If not, an appropriate error should be produced, but only when the easyblock is *not* being used in dry run mode.

Bad example:

```
# *BAD* example: maybe the 'test' directory is not there (e.g., because we're in dry␣
↪run mode)!
try:
    testcases = os.listdir('test')
except OSError as err:
    raise EasyBuildError("Unexpected error when determining list of test cases: %s",␣
↪err)
```

Good example:

```
# make sure the 'test' directory is there before trying to access it
if os.path.exists('test'):
    try:
        testcases = os.listdir('test')
    except OSError as err:
        raise EasyBuildError("Unexpected error when determining list of test cases: %s
↪", err)

# only raise an error if we're not in dry run mode
elif not self.dry_run:
    raise EasyBuildError("Test directory not found, failed to determine list of test␣
↪cases")
```

Easyblocks that do not take this into account are likely to result in ignored errors during a dry run (see also *Errors are ignored (by default) during dry run*). For example, for the bad example shown above:

```
!!!
!!! WARNING: ignoring error "Unexpected error when determining list of test cases:␣
↪[Errno 2] No such file or directory: 'test'"
!!!
```

Use functions provided by the EasyBuild framework

The EasyBuild framework provides a bunch of functions that are "*dry run-aware*", and which can significantly help in keeping easyblocks free from conditional statements checking `self.dry_run`:

- *Defining environment variables: setvar*
- *Writing or appending to files: write_file*
- *Runtime patching of files: apply_regex_substitutions*
- *Executing commands: run_cmd and run_cmd_qa*

Defining environment variables: `setvar`

For defining environment variables, the `setvar` function available in the `easybuild.tools.environment` module should be used.

For example, from the WRF easyblock:

```
jasper = get_software_root('JasPer')
if jasper:
    env.setvar('JASPERINC', os.path.join(jasper, 'include'))
```

When triggered in dry run mode, this will result in a clear dry run message like:

```
export JASPERINC="$EBROOTJASPER/include"
```

The actual output depends on whether or not the required module for `JasPer` is available (see *Simulated loading of non-available dependency modules*).

Silently defining environment variables

The `setvar` function also supports defining environment variables *silently*, i.e. without producing a corresponding dry run message, via the named argument `verbose`.

This is used in a couple of places in the EasyBuild framework, to avoid some environment variables being defined cluttering the dry run output without added value. It can be used for similar reasons in easyblocks.

For example, the `PythonPackage` uses it in the *install* step, to modify `$PYTHONPATH` as required by the `python setup.py install` procedure (which is considered not relevant to include in the dry run output, since it's a technicality):

```
env.setvar('PYTHONPATH', new_pythonpath, verbose=False)
```

Writing or appending to files: `write_file`

For writing and appending to files, the EasyBuild framework provides the `write_file` function (available from the `easybuild.tools.filetools` module).

Using it is straightforward, for example:

```
write_file('example.txt', "Contents for the example file")
```

To append to an existing file, `write_file` support a named argument `append`.

When used in dry run mode, `write_file` does not actually (attempt to) write to the file; instead, it just produces an appropriate dry run message and returns.

For example:

```
file written: /tmp/eb-ksVC07/tmp.conf
```

Runtime patching of files: `apply_regex_substitutions`

To make runtime patching of files in easyblocks easier, and to do it with taking the possibility of being in dry run module into account, the EasyBuild framework provides the `apply_regex_substitutions` function (available from the `easybuild.tools.filetools` module, since EasyBuild v2.4.0).

This function takes two arguments: a path to the file that should be patched, and a list of tuples specifying the regular expression pattern to match on, and the string value that should be used as replacement text.

For example (simple fictional example):

```
# replace value for C++ compiler
apply_regex_substitutions('config.mk', [('^(CPLUSPLUS\s*=).*', '\1 %s' % os.environ[
↪'CXX'])])
```

When used in dry run mode, it will produce a message like:

```
applying regex substitutions to file config.mk
  * regex pattern '^(CPLUSPLUS\s*=\s).*', replacement string '\1 g++'
```

Executing commands: `run_cmd` and `run_cmd_qa`

To execute shell commands, the `run_cmd` and `run_cmd_qa` functions are provided by the EasyBuild framework in the `easybuild.tools.run` module, with the latter providing support for running interactive commands.

In their simplest form, they simply take the command to execute as a string. For example:

```
run_cmd("tcsh ./compile -j %s wrf" % self.cfg['parallel'])
```

In dry run mode, these functions just produce a dry run message, rather than actually executing the specified command. For example:

```
running command "tcsh ./compile -j 4 wrf"
(in /home/example/eb/software/WRF/3.6.1-intel-2015a-dmpar/WRF-3.6.1)
```

Take into account that the directory included in the message may not be 100% accurate, see *Note on build directory in dry run mode*.

Silently executing commands

The `verbose` named argument supported by the `run_cmd` function allows to execute a particular command silently, i.e. without producing a dry run message.

For example:

```
# only run for debugging purposes
run_cmd("ulimit -v", verbose=False)
```

Forced execution of particular commands

Sometimes, it can be required that specific (light-weight) commands are *always* executed, because they have side-effects that are assumed to have taken place later in the easyblock.

For this, the `run_cmd` function support another named argument, i.e. `force_in_dry_run`. When set to `True`, the specified command will always be executed, even when in dry run mode.

This is mainly intended for use in the EasyBuild framework itself, where commands that verify certain things must be executed, but it can also be useful for easyblocks (if used correctly).

For example:

```
out, exit_code = run_cmd("type module", simple=False, force_in_dry_run=True)
```

6.12.4 Example output

Output examples for `eb --extended-dry-run`/`eb -x`:

- extended_dry_run_examples_make382_GCC482
- extended_dry_run_examples_WRF361_intel2015a

6.13 Hooks

Since v3.5.0, EasyBuild supports *hooks* that can be used to customise the behaviour of EasyBuild according to site policies if needed, without having to change the EasyBuild framework or the existing easyblocks.

Contents

6.13.1 What are hooks?

Hooks are user-defined functions that are called by the EasyBuild framework at specific times during the installation procedure. They can be leveraged to alter or augment the installation procedure.

This is usually done to conform with site-specific policies that are difficult to enforce otherwise, but it can also be (ab)used to fix specific problems or inject self-implemented enhancements (before you flesh them out in a proper contribution, for example).

Both the `EasyBlock` instance and the parsed easyconfig file that are being used are fully accessible (and modifiable) from within hook implementations. Hence, this mechanism provides a lot of flexibility to change the EasyBuild functionality should you require it, without having to modify the codebase of EasyBuild itself.

6.13.2 Configuring EasyBuild to use your hook implementations

To instruct EasyBuild to use your hook implementations, you only need to specify the location of the Python module (`*.py`) that implements them.

This is done via the `--hooks` configuration option (or equivalently via the `$EASYBUILD_HOOKS` environment variable, or via `hooks = ...` in an EasyBuild configuration file, see also *Configuring EasyBuild*).

For example:

```
eb --hooks=$HOME/my_eb_hooks.py ...
```

or:

```
export EASYBUILD_HOOKS=$HOME/my_eb_hooks.py
eb ...
```

6.13.3 Available hooks

Currently (since EasyBuild v3.7.0), three types of hooks are supported:

- `start_hook` and `end_hook`, which are triggered *once* before starting software installations, and *once* right after completing all installations, respectively
- `parse_hook`, which is triggered when an easyconfig file is being parsed
- `module_write_hook`, which is triggered right before a module file is written. This includes the temporary module file used when installing extensions and during the sanity check, as well as the devel module.
- "*step*" hooks that are triggered before and after every step of each installation procedure that is performed, also aptly named '`pre`'- and '`post`'-hooks

The list of currently available hooks in order of execution, which can also be consulted using `eb --avail-hooks`, is:

- `start_hook` *(only called once in an EasyBuild session)*
- `parse_hook` *(available since EasyBuild v3.7.0)*
- `pre_fetch_hook`, `post_fetch_hook`
- `pre_ready_hook`, `post_ready_hook`
- `pre_source_hook`, `post_source_hook`
- `pre_patch_hook`, `post_patch_hook`
- `pre_prepare_hook`, `post_prepare_hook`
- `pre_configure_hook`, `post_configure_hook`
- `pre_build_hook`, `post_build_hook`
- `pre_test_hook`, `post_test_hook`
- `pre_install_hook`, `post_install_hook`
- `pre_extensions_hook`, `post_extensions_hook`

- pre_postproc_hook, post_postproc_hook
- pre_sanitycheck_hook, post_sanitycheck_hook
- pre_cleanup_hook, post_cleanup_hook
- pre_module_hook, post_module_hook
- pre_permissions_hook, post_permissions_hook
- pre_package_hook, post_package_hook
- pre_testcases_hook, post_testcases_hook
- end_hook *(only called once in an EasyBuild session)*
- module_write_hook *(called multiple times per installation, available since EasyBuild v4.4.1)*

All functions implemented in the provided Python module for which the name ends with _hook are considered.

If any *_hook functions are encountered that do not match any of the available hooks, an error is reported. EasyBuild will try to provide suggestions for available hooks that closely match the encountered unknown hook.

For example:

```
$ eb --hooks wrong_hooks.py example.eb
== temporary log file in case of crash /tmp/eb-nMawy1/easybuild-Gu2ZP6.log
ERROR: Found one or more unknown hooks:
* stat_hook (did you mean 'start_hook'?)
* this_is_not_a_hook
* install_hook (did you mean 'pre_install_hook', or 'post_install_hook'?)

Run 'eb --avail-hooks' to get an overview of known hooks
```

6.13.4 Implementing hooks

To implement hooks, simply define one or more functions in a Python module (*.py), each named after an available hook.

Do take into account the following:

- for start_hook and end_hook, no arguments are provided
- for parse_hook, one argument is provided: the EasyConfig instance that corresponds to the easyconfig file being parsed (usually referred to as ec)
- **for module_write_hook, 3 arguments are provided:**
  - the EasyBlock instance used to perform the installation (usually referred to as self)
  - the filepath of the module that will be written
  - the module text as a string

  The return value of this hook, when set, will replace the original text that is then written to the module file.
- for the step hooks, one argument is provided: the EasyBlock instance used to perform the installation (usually referred to as self)
- the parsed easyconfig file can be accessed in the step hooks via the EasyBlock instance, i.e., via self.cfg

It is recommended to anticipate possible changes in the provided (named) arguments, using the *args and **kwargs mechanism commonly used in Python. This avoids that your hook implementations may break when updating to future EasyBuild versions. For example:

```
# example pre-configure hook that anticipates changes in provided arguments
def pre_configure_hook(self, *args, **kwargs):
    ...
```

In hooks you have access to the full functionality provided by the EasyBuild framework, so do `import` from `easybuild.tools.*` (or other `easybuild.*` namespaces) to leverage those functions.

Parse hook specifics

`parse_hook` is triggered right *after* reading the easyconfig file, before further parsing of some easyconfig parameters (like `*dependencies`) into custom data structures is done.

This is important since it allows to dynamically modify easyconfig files while they are still "raw", i.e. when the easyconfig parameter values are still basic Python data structures like lists, dictionaries, etc. that are easy to manipulate (see also *Manipulating easyconfig parameters*).

In `parse_hook` easyconfig parameters can be accessed and/or modified in a straightforward way, see *Example hook to inject a custom patch file*.

6.13.5 Caveats

Due to internal details of the EasyBuild framework, you may run into some surprises when implementing hooks. Here are some things to take into account:

Resolving of template values

In all *step* hooks, template values in easyconfig parameters will be resolved whenever they are accessed.

That is, if the `%(version)` template is used in for example the `sources` easyconfig parameter, it will be replaced with the actual value of the `version` easyconfig parameter whenever the `sources` value is used. This can be avoided by temporarily disabling templating by wrapping the code in `with self.cfg.disable_templating:`.

There is one notable exception to this: Templates in easyconfig parameters are *not* resolved in `parse_hook`, because templating has been disabled explicitly before `parse_hook` is called; this helps significantly to simplify manipulating of easyconfig parameter values (see also *Manipulating easyconfig parameters*).

Manipulating easyconfig parameters

You may run into surprises when trying to manipulate easyconfig parameters, for various reasons.

First of all, the original easyconfig parameters may already be processed in another data structure which does not resemble the original format in which the parameter was defined in the easyconfig file.

Moreover, this processing could be done either "in place" by replacing the original easyconfig parameter value, or in a separate variable, which effectively means that any changes to the original easyconfig parameter value are simply ignored.

In addition, because of how the templating mechanism for easyconfig parameter works, changes to easyconfig parameters with non-string values (i.e. lists, dictionaries, etc.) will go up in smoke if not done correctly.

More specifically, the following approach will *not* work in any of the (step) hooks, except for `parse_hook`:

```
def pre_fetch_hook(self):
    "Example of pre-fetch hook to manipulate list of patches."
    # this does NOT have the intended affect in any pre- or post-step hook
    self.cfg['patches'].append('example.patch')
```

The problem here is that the value obtained via `self.cfg['patches']` is not a reference to the actual easyconfig parameter value but a reference to a temporary copy thereof; hence any updates on the copy are effectively lost immediately.

To achieve the intended effect, you can either:

- temporarily disable the templating mechanism:

```
def pre_fetch_hook(self):
    "Example of pre-fetch hook to manipulate list of patches."
    # temporarily disable templating, so changes to 'patches' easyconfig␣
↪parameter are picked up
    with self.cfg.disable_templating:
        # add patch
        self.cfg['patches'].append('example.patch')
    # templating state restored
```

- or replace the original value entirely:

```
def pre_fetch_hook(self):
    "Example of pre-fetch hook to manipulate list of patches."
    self.cfg['patches'] = self.cfg['patches'] + ['example.patch']
```

A better approach for manipulating easyconfig parameters is to use the `parse_hook` that was introduced in EasyBuild v3.7.0 (see *Parse hook specifics*), where these kind of surprises will not occur (because templating is automatically disabled before `parse_hook` is called and restored immediately afterwards). See also *Example hook to inject a custom patch file*.

Archived easyconfig file vs hooks

EasyBuild archives the easyconfig file that was used for a particular installation: A copy is stored both in the `easybuild` subdirectory of the software installation directory and in the easyconfigs repository (see *Easyconfigs repository (–repository, –repositorypath)*).

If any changes were made to the easyconfig file via hooks, these changes will *not* be reflected in these copies. The assumption here is that the hooks will also be in place for future (re-)installations.

EasyBuild does however store an additional copy of the easyconfig file which includes any modifications that were done dynamically, for example by hooks. If subtoolchains were used to resolve dependencies, they will also be hardwired in this copy.

This "*reproducible easyconfig*" is stored in the `easybuild/reprod` subdirectory of the software installation directory.

6.13.6 Examples of hook implementations

Example hook to replace `--with-verbs` with `--without-verbs` in OpenMPI configure options

```
def pre_configure_hook(self, *args, **kwargs):
    """Example pre-configure hook to replace --with-verbs with --without -verbs for␣
↪OpenMPI."""
    if self.name == 'OpenMPI' and '--with-verbs' in self.cfg['configopts']:
        self.log.info("[pre-configure hook] Replacing --with-verbs with --without-
↪verbs")
        self.cfg['configopts'] = self.cfg['configopts'].replace('--with-verbs', '--
↪without-verbs')
```

Example hook to inject a custom patch file

```
def parse_hook(ec, *args, **kwargs):
    """Example parse hook to inject a patch file for a fictive software package named
↪'Example'."""
    if ec.name == 'Example':
        patch_file = 'example.patch'
        ec.log.info("[parse hook] Injecting additional patch file '%s'", patch_file)
        ec['patches'].append(patch_file)
```

Example hook to replace PYTHONPATH by EBPYTHONPREFIXES in (Lua) modules

```
def module_write_hook(self, filepath, module_txt, *args, **kwargs):
    # note: if `self.mod_filepath == filepath` => final module file
    if 'Python' in (dep['name'] for dep in self.cfg.dependencies()):
        search = r'prepend_path\("PYTHONPATH", pathJoin\(root, "lib/python\d.\d/site-
↪packages"\)\)'
        replace = 'prepend_path("EBPYTHONPREFIXES", root)'
        return re.sub(search, replace, module_txt)
```

6.14 Implementing easyblocks

This documentation covers aspects of implementing *easyblocks* and how to use them. For a concise definition of easyblocks, see *Easyblocks*.

Contents

6.14.1 The basics

An *easyblock* is a Python module that implements a software build and installation procedure.

This concept is essentially implemented as a Python script that plugs into the *EasyBuild framework*.

EasyBuild will leverage easyblocks as needed, depending on which software packages it needs to install. Which easyblock is required is determined by the `easyblock` easyconfig parameter, if it is present, or the software name (see *Generic vs software-specific easyblocks*).

Generic vs software-specific easyblocks

Easyblocks can either be *generic* or *software-specific*.

Generic easyblocks implement a 'standard' software build and installation procedure that is used by multiple different software packages. A commonly used example is the ConfigureMake generic easyblock, which implements the standard `configure` - `make` - `make install` installation procedure used by most GNU software packages.

Software-specific easyblocks implement the build and installation procedure for a particular software package (see also *Module name/location*). Typically this involves highly customised substeps, for example specifying dedicated configuration options, creating or adjusting specific files, executing non-standard shell commands, etc. Usually a custom implementation of the sanity check is also included.

Since EasyBuild v2.0, using a generic easyblock requires specifying the `easyblock` parameter in the easyconfig file. If it is not specified, EasyBuild will try and find the software-specific easyblock derived from the software name.

The distinction between generic and software-specific easyblocks can be made based on the naming scheme that is used for an easyblock (see also *Naming scheme for easyblocks*).

See also generic_easyblocks.

6.14.2 Easyblocks vs easyconfigs

Before you start implementing an easyblock, you should determine whether or not you really need an easyblock.

Easyconfig files provide quite a bit of flexibility that alleviate the need for implementing many (software-specific) easyblocks. Examples include easyconfig parameters like `(pre){config,build,installopts}` (available for any easyblock), `install_cmd` (only for `Binary` generic easyblock or derivates), etc. See *Writing easyconfig files: the basics* for an overview of what is supported in easyconfig files, and generic_easyblocks for custom easyconfig parameters for the existing generic easyblocks (see also '`eb -e <easyblock> -a`').

On the other hand, somewhat complex or heavily customised software build and installation procedures may be handled better via a custom easyblock.

Easyblocks are "do once and forget", and can provide a *central* solution for peculiarities of installations. Since they are basically Python scripts, they are very flexible and can take care of the 'heavy lifting' that goes beyond the key-value defining scope of easyconfig files.

Hence, there is a fine line between using 'fat' easyconfigs with a generic easyblock and using a custom software-specific easyblock.

Reasons to consider implementing a custom easyblock include:

- 'critical' values for easyconfig parameters, which are required to make the installation succeed
- compiler- or toolchain-specific aspects of the build and installation procedure (e.g., configure/build/install options)
- interactive commands that need to be run
- custom (configure) options for dependencies
- having to create or adjust specific (configuration) files
- 'hackish' usage of existing (generic) easyblocks and available easyconfig parameters

One important aspect to consider of course is that implementing easyblocks requires some familiarity with Python, while easyconfig files can mostly be seen as a set of pure key-value definitions and hence are easier to create or update.

6.14.3 Naming scheme for easyblocks

Easyblocks need to follow a strict naming scheme, to ensure that EasyBuild can pick them up as needed. This involves two aspects: i) the name of the Python class, and ii) the name and location of the Python module file.

Class name for software-specific easyblocks

The name of the Python class is determined by the *software name* for software-specific easyblocks. It consists of a prefix '`EB_`', followed by the (original) software name.

Because of limitations in Python on characters allowed in names of Python classes, only alphanumeric characters and _ are allowed. Any other characters are replaced following an encoding scheme:

- spaces are replaced by underscores (_)
- dashes - are replaced by `_minus_`
- underscores are replaced by `_underscore_`
- etc.

The `encode_class_name` function provided in `easybuild.tools.filetools` returns the expected class name for a given software name; for example:

```
>>> from easybuild.tools.filetools import encode_class_name
>>> encode_class_name('netCDF-Fortran')
'EB_netCDF_minus_Fortran'
```

Class name for generic easyblocks

For *generic* easyblocks, the class name does *not* include an `EB_` prefix (since there is no need for an escaping mechanism) and hence the name is fully free to choose, taking into account the restriction to alphanumeric characters and underscores.

For code style reasons, the class name should start with a capital letter.

Examples include `Bundle`, `ConfigureMake`, `CMakePythonPackage`.

Module name/location

The *name* of the Python module file is directly related to the name of Python class (i.e., the actual easyblock) that it provides.

It should:

- *not* include the `EB_` prefix of the class name for software-specific easyblocks
- consists only of lower-case alphanumeric characters (`[a-z0-9]`) and underscores (_)
  - dashes (-) are replaced by underscores (_)
  - any other non-alphanumeric characters (incl. spaces) are simply dropped

Examples include:

- `gcc.py` (for *GCC*)
- `netcdf_fortran.py` (for *netCDF-Fortran*)
- `gamess_us.py` (for *GAMESS (US)*)

The `get_module_path` function provided in `easybuild.framework.easyconfig.easyconfig` returns the (full) module location for a particular software name or easyblock class name; for example:

```
>>> from easybuild.framework.easyconfig.easyconfig import get_module_path
>>> get_module_path('netCDF-Fortran')
'easybuild.easyblocks.netcdf_fortran'
>>> get_module_path('EB_netCDF_minus_Fortran')
'easybuild.easyblocks.netcdf_fortran'
```

The location of the Python module is determined by whether the easyblock is generic or software-specific. Generic easyblocks are located in the `easybuid.easyblocks.generic` namespace, while software-specific easyblocks live in the `easybuild.easyblocks` namespace directly. To keep things organised, the actual Python module file for software-specific easyblocks are kept in 'letter' subdirectories, rather than in one large '`easyblocks`' directory (see https://github.com/easybuilders/easybuild-easyblocks/blob/main/easybuild/easyblocks/).

Note that you shouldn't concern yourself too much with getting the location of an easyblock right, as long as you use `--include-easyblocks` to make EasyBuild use additional or customised easyblocks (see *Including additional easyblocks (–include-easyblocks)* for more information).

6.14.4 Structure of an easyblock

The example below shows the overal structure of an easyblock:

```
from easybuild.framework.easyblock import EasyBlock
from easybuild.tools.run import run_cmd

class EB_Example(EasyBlock):
    """Custom easyblock for Example"""

    def configure_step(self):
        """Custom implementation of configure step for Example"""

        # run configure.sh to configure the build
        run_cmd("./configure.sh --install-prefix=%s" % self.installdir)
```

Each easyblock includes an implementation of a `class` that (directly or indirectly) derives from the abstract `EasyBlock` class.

Typically some useful functions provided by the EasyBuild framework are imported at the top of the Python module.

In the class definition, one or more '`*_step`' methods are redefined, to implement the corresponding step in the build and installation procedure.

Each easyblock *must* implement the `configure`, `build` and `install` steps, since these are not implemented in the abstract `EasyBlock` class. This could be done explicitly by redefining the corresponding `*_step` methods, or implicitly by deriving from existing (generic) easyblocks.

6.14.5 Deriving from existing (generic) easyblocks

When implementing an easyblock, it is common to derive from an existing (usually generic) easyblock, and to leverage the functionality provided by it. This approach is typically used when only a specific part of the build and installation procedure needs to be customised.

In the (fictitious) example below, we derive from the generic `ConfigureMake` easyblock to redefine the `configure` step. In this case, we are *extending* the `configure` step as implemented by `ConfigureMake` rather than redefining it entirely, since we call out to the original `configure_step` method at the end.

```
from easybuild.easyblocks.generic.configuremake import ConfigureMake
from easybuild.tools.filetools import copy_file

class EB_Example(ConfigureMake):
    """Custom easyblock for Example"""

    def configure_step(self):
        """Custom implementation of configure step for Example"""

        # use example make.cfg for x86-64
        copy_file('make.cfg.x86', 'make.cfg')

        # call out to original configure_step implementation of ConfigureMake␣
↪easyblock
        super(EB_Example, self).configure_step()
```

6.14.6 Specific aspects of easyblocks

Default easyconfig parameters

All of the parameters which are "set" in an easyconfig file (see vsd_avail_easyconfig_params) become key-value pairs in the `self.cfg` dictionary. For instance, if the easyconfig file specifies

```
name = 'example'
version = '2.5.3'
versionsuffix = '-Python-3.7.4'
```

then these three parameters are accessible within an easyblock via

```
longform = ''.join(self.cfg['name'],'/',self.cfg['version'],self.cfg['versionsuffix'])
```

You can use this notation successfully in your easyblock. A few of the most commonly used parameters can be referenced directly,

- **self.name** = `self.cfg['name']`
- **self.version** = `self.cfg['version']`
- **self.toolchain** = `self.cfg['toolchain']`
- **self.all_dependencies**: combines `builddependencies`, `dependencies`, and `toolchain`, in one dictionary

So in your easyblock code, you may replace the above expression with

```
longform = ''.join(self.name,'/',self.version,self.cfg['versionsuffix'])
```

The other easyconfig parameters, and any additional *custom parameters* which you define for your own easyblock, will not be automatically mapped. You will need to use `self.cfg` to access them in your code.

Custom easyconfig parameters

In an easyblock, additional custom easyconfig parameters can be defined to steer the behaviour of the easyblock. This is done via the `extra_options` static method. Custom parameters can be defined to be mandatory or optional.

The example below shows how this can be implemented:

```
from easybuild.easyblocks.generic.configuremake import ConfigureMake
from easybuild.framework.easyconfig import CUSTOM, MANDATORY

class EB_Example(ConfigureMake):
    """Custom easyblock for Example"""

    @staticmethod
    def extra_options():
        """Custom easyconfig parameters for Example"""
        extra_vars = {
            'required_example_param': [None, "Help text for required example custom␣
↪parameter", MANDATORY],
            'optional_example_param': [None, "Help text for (optional) example custom␣
↪parameter", CUSTOM],
        }
        return ConfigureMake.extra_options(extra_vars)
```

The first element in the list of a defined custom parameter corresponds to the default value for that parameter (both `None` in the example above). The second element provides some informative help text, and the last element indicates whether the parameter is mandatory (`MANDATORY`) or just a custom parameter (`CUSTOM`).

Easyblock constructor

In the `class` constructor of the easyblock, i.e. the `__init__` method, one or more class variables can be initialised. These can be used for sharing information between different `*_step` methods in the easyblock.

For example:

```
from easybuild.framework.easyblock import EasyBlock

class EB_Example(EasyBlock):
    """Custom easyblock for Example"""

    def __init__(self, *args, **kwargs):
        """Constructor for Example easyblock, initialises class variables."""

        # call out to original constructor first, so 'self' (i.e. the class instance)␣
↪is initialised
        super(EB_Example, self).__init__(*args, **kwargs)

        # initialise class variables
        self.var1 = None
        self.var2 = None
```

Reading/writing/copying/patching files

File manipulation is a common use case for implementing easyblocks, hence the EasyBuild framework provides a number of useful functions related to this, including:

- `read_file(<path>)`: read file at a specified location and returns its contents
- `write_file(<path>, <text>)` at a specified location with provided contents; to append to an existing file, use `append=True` as an extra argument
- `copy_file(<src>, <dest>)` to copy an existing file
- `apply_regex_substitutions(<path>, <list of regex substitutions>)` to patch an existing file

All of these functions are provided by the `easybuild.tools.filetools` module.

Executing (interactive) shell commands

For executing shell commands two functions are provided by the `easybuild.tools.run` module:

- `run_cmd(<cmd>)` to run a non-interactive shell command
- `run_cmd_qa(<cmd>, <dict with questions & answers>)` to run an interactive shell command

Both of these accept a number of optional arguments:

- `simple=True` to just return `True` or `False` to indicate a successful execution, rather than the default return value, i.e., a tuple that provides the command output and the exit code (in that order)
- `path=<path>` to run the command in a specific subdirectory

The `run_cmd_qa` function takes two additional specific arguments:

- `no_qa=<list>` to specify a list of patterns to recognize non-questions
- `std_qa=<dict>` to specify patterns for common questions and the matching answer

Manipulating the environment

To (re)define environment variables, the `setvar` function provided by the `easybuild.tools.environment` module should be used.

This makes sure that the changes being made to the specified environment variable are kept track of, and that they are handled correctly under `--extended-dry-run`.

Log statements

It is good practice to include meaningful log messages in the `*_step` methods being customised in the easyblock, to enrich the build log with useful information for later debugging or diagnostics.

For logging, the provided `self.log` logger class should be used, i.e. the `self.log.info` or `self.log.debug` methods should be called.

Custom (default) sanity check

For software-specific easyblocks, a custom sanity check is usually included to verify that the installation was successful or not.

This is done by redefining the `sanity_check_step` method in the easyblock. For example:

```
from easybuild.framework.easyblock import EasyBlock

class EB_Example(EasyBlock):
    """Custom easyblock for Example"""

    def sanity_check_step(self):
        """Custom sanity check for Example."""

        custom_paths = {
            'files': ['bin/example'],
            'dirs': [],
        }
        custom_commands = ['example --version']

        # call out to parent to do the actual sanity checking, pass through custom␣
↪paths and commands
        super(EB_Example, self).sanity_check_step(custom_paths=custom_paths, custom_
↪commands=custom_commands)
```

You can both specify file path and subdirectories to check for, which are specified relative to the installation directory, as well as simple commands that should execute successfully after completing the installation and loading the generated module file.

Note that it is up to you how extensive you make the sanity check, but it is recommended to make the check as complete as possible to catch any potential build or installation problems that may occur.

Version-specific parts

In some case, version-specific actions or checks need to be included in an easyblock. For this, it is recommended to use `LooseVersion` rather than directly comparing version numbers using string values.

For example:

```
from distutils.version import LooseVersion

from easybuild.framework.easyblock import EasyBlock

class EB_Example(EasyBlock):
    """Custom easyblock for Example"""

    def sanity_check_step(self):
        """Custom sanity check for Example."""

        custom_paths = {
            'files': [],
            'dirs': [],
        }

        # in older version, the binary used to be named 'EXAMPLE' rather than 'example
↪'
        if LooseVersion(self.version) < LooseVersion('1.0'):
            custom_paths['files'].append('bin/EXAMPLE')
        else:
            custom_paths['files'].append('bin/example')

        super(EB_Example, self).sanity_check_step(custom_paths=custom_paths)
```

Compatibility with `--extended-dry-run`/`-x` and `--module-only`

Some special care must be taken to ensure that an easyblock is fully compatible with `--extended-dry-run` / `-x` (see *Extended dry run*) and `--module-only` (see *Only (re)generating (additional) module files using –module-only*).

For `--extended-dry-run`/`-x`, this is already well covered at *Detecting dry run mode and enhancing the dry run output*.

For `--module-only`, you should make sure that both the `make_module_step`, including the `make_module_*` submethods, and the `sanity_check_step` methods do not make any assumptions about the presence of certain environment variables or that class variables have been defined already.

This needs to be handled with care since under `--module-only` the large majority of the `*_step` functions is simply skipped entirely. So, if the `configure_step` method is responsible for defining class variables that are picked up in `sanity_check_step`, the latter may run into unexpected initial values like `None`. A possible workaround is to define a separate custom method to define the class variables, and to call out to this method from `configure_step` and `sanity_check_step` (for the latter, conditionally, i.e., only if the class variables still have the initial values).

6.14.7 Using new/custom easyblocks

The best way to make EasyBuild aware of new or customized easyblocks is via `--include-easyblocks`, see *Including additional easyblocks (–include-easyblocks)* for more information.

To verify that your easyblocks are indeed picked up correctly, you can use `--list-easyblocks=detailed`, see also *List of available easyblocks, –list-easyblocks*.

6.14.8 Testing easyblocks

Before testing your easyblock implementation by actually building and installing the software package(s) it was implemented for, it is recommended to:

- study the output produced by `--extended-dry-run`/`-x`
- evaluate the generated module file that is obtained by using `--module-only --force`

For the output of `--extended-dry-run`/`-x`, there should be no ignored errors (cfr. *Errors are ignored (by default) during dry run*), that is the end of the output produced should include this message:

```
(no ignored errors during dry run)
```

With `--module-only --force`, the easyblock complete successfully without crashing, and should generate a module file that includes everything that is expected (except for statements that require that the actual installation was performend).

6.14.9 Use case: an easyblock for Tensorflow

(work in progress)

6.15 Including additional Python modules (`--include-*`)

EasyBuild's capabilities can be extended easily, by including additional Python modules that implement support for building and installing software that is not supported (yet), define additional module naming schemes, or introduce additional toolchains, (optionally) with support for additional compilers, MPI libraries, linear algebra libraries, etc.

Since EasyBuild v2.2.0, dedicated configuration options are available that make it straightforward to get EasyBuild to pick up additional Python modules, and get them registered in the appropriate `easybuild` subnamespace.

- *Including additional easyblocks (–include-easyblocks)*
- *Including additional module naming schemes (–include-module-naming-schemes)*
- *Including additional toolchains (–include-toolchains)*

6.15.1 General aspects of `--include-*` options

Configuration types

The `--include-*` options can be specified via the `eb` command line, using an environment variable (`$EASYBUILD_INCLUDE_*`) or by defining the corresponding `include-*` parameters in an EasyBuild configuration file, just like all other configuration options (see also *Consistentency across supported configuration types*).

Format

The `--include-*` options accept a comma-separated list of paths to Python modules.

These paths can be absolute or relative paths, or so-called *glob patterns*, i.e., paths containing wildcard characters like `*` or `?`. The latter can be used to include several Python modules at once.

For example, to include all Python modules located in the directory `$HOME/myeb`, a path pattern like `$HOME/myeb/*.py` can be specified to the appropriate `--include-*` option.

Note: Shell expansion can get in the way of specifying paths to `eb` that contains wildcards. To avoid problems simply wrap the path in single quotes, or escape the wildcard characters using a backslash. Keep in mind that using single quotes also prevents environment variables (e.g., `$HOME`) from being expanded.

Examples of correct path specifications containing wildcards:

- in a configuration file (no escaping of wildcards required): `include-easyblocks = /home/example/myeb/*.py`
- using an environment variable: `export EASYBUILD_INCLUDE_EASYBLOCKS="$HOME/myeb\*.py"`
- on the command line: `eb --include-easyblocks='/home/example/myeb/*.py' ...`

How it works

For each of the `--include-*` options, EasyBuild will set up a temporary directory providing the corresponding Python package. In this directory, symlinks will be put in place to each of the included Python modules. The parent path is then injected in the Python search path to make the included Python modules available as required.

Order of preference

Python modules that are included via `--include-*` get preference over other Python modules available in the Python search path (e.g., the one that are part of the EasyBuild installation you are using). This may be useful when testing modifications to particular components of EasyBuild, for example easyblocks.

Note: It is recommended to only override existing components during testing. Future EasyBuild versions may include important updates like bug fixes, which may be missed if customised implementations of components were put in place.

6.15.2 Including additional easyblocks (`--include-easyblocks`)

Adding support for building and installing additional software packages can be done by specifying the location of Python modules that implement easyblocks via `--include-easyblocks`.

Generic easyblocks are expected to be located in a directory named `generic`.

To verify that the easyblocks you included are indeed being picked up, `--list-easyblocks=detailed` can be used (see also *List of available easyblocks, –list-easyblocks*).

Since EasyBuild 4.2.0, easyblocks from a pull request on GitHub can also be included, using `--include-easyblocks-from-pr` (see *Using easyblocks from pull requests (–include-easyblocks-from-pr)*).

Example

The example below shows how all self-implemented easyblocks (both software-specific and generic) located in the `$HOME/myeasyblocks` directory can be included:

```
$ export EASYBUILD_INCLUDE_EASYBLOCKS=$HOME/myeasyblocks/\*.py,$HOME/myeasyblocks/
↪generic/\*.py
$ eb --list-easyblocks=detailed
...
|-- EB_mytest (easybuild.easyblocks.mytest @ /tmp/example/eb-Bk3zxb/included-
↪easyblocks/easybuild/easyblocks/mytest.py)
...
|-- foo (easybuild.easyblocks.generic.foo @ /tmp/example/eb-Bk3zxb/included-
↪easyblocks/easybuild/easyblocks/generic/foo.py)
...
```

6.15.3 Including additional module naming schemes (`--include-module-naming-schemes`)

To make EasyBuild aware of one or more custom module naming schemes, the path to the corresponding Python modules can be specified via `--include-module-naming-schemes`.

To verify that EasyBuild is aware of the additional module naming schemes, the `--avail-module-naming-schemes` option can be used.

Example

The example below shows how all custom module naming schemes located in the `$HOME/myebmns` can be included:

```
$ eb --include-module-naming-schemes=$HOME/myebmns/\*.py --avail-module-naming-schemes
List of supported module naming schemes:
    ...
    MyCustomMNS
    MyOtherCustomMNS
    ...
```

6.15.4 Including additional toolchains (`--include-toolchains`)

Plugging in Python modules that add support for additional toolchains, optionally including additional toolchain components (compilers, MPI libraries, BLAS/LAPACK/FFT libraries, ...) can be done via `--include-toolchains`.

EasyBuild will determine whether the Python module is a *toolchain definition* or implements support for an *additional toolchain component* based on the name of the directory in which it is located. Implementations of toolchain components are expected to be located in a directory named according to the type of component (`compiler`, `mpi`, `linalg` or `fft`).

To verify that EasyBuild is aware of the included toolchains, `--list-toolchains` can be used.

Example

The example below shows how the support for additional toolchains and the required additional compiler/MPI toolchain components implemented by the Python modules located in the directory `$HOME/myebtcs` can be included:

```
$ export EASYBUILD_INCLUDE_TOOLCHAINS=$HOME/myebtcs/\*.py,$HOME/myebtcs/compiler/\*.
↪py,$HOME/myebtcs/mpi/\*.py
$ eb --list-toolchains
List of known toolchains (toolchainname: module[,module...]):
```

(continues on next page)

(continued from previous page)

```
...
mytoolchain: MyCompiler, MyMPI
...
```

6.16 Integration with GitHub

EasyBuild provides several features that integrate with GitHub, where the different EasyBuild repositories are located.

From the EasyBuild command line `eb` several options are available to reach out to GitHub, which are documented below.

Contents

        * *Synergy with `--dry-run/-D` and `--extended-dry-run/-x`*

6.16.1 Requirements

Depending on which GitHub integration features you want to use, there are a couple of requirements:

- **a GitHub account**
  - see https://github.com; creating an account is free
- **a GitHub user name**
  - only required for authenticated access to the GitHub API, which can help to avoid rate limitations
  - *not* strictly necessary for read-only operations
    * i.e. *not* required for *Using easyconfigs from pull requests (–from-pr)* and *Reviewing easyconfig pull requests (–review-pr)* (but it can help)
  - see *Providing a GitHub username (–github-user)*
- **a GitHub token +** `keyring` **Python package**
  - install via `pip install keyring` (for Python2: `pip install 'keyring<19.0'`)
  - optionally install potentially unsafe keyrings: `pip install keyrings.alt` (but read and understand the warning)
  - allows accessing the GitHub API with authentication
  - only strictly required for features that require GitHub 'write' permissions
    * i.e. for *Uploading test reports (–upload-test-report)* and *Submitting pull requests (–new-pr)*
  - see *Installing a GitHub token (–install-github-token)*
- `git` **command /** `GitPython` **Python package**
  - install via `pip install GitPython` (for Python2: `pip install 'GitPython<3.0'`)
  - only required when local `git` commands need to be executed, e.g. to manipulate a Git repository
    * i.e. for *Submitting pull requests (–new-pr)* and *Updating existing pull requests (–update-pr)*
- **SSH public key registered on GitHub**
  - only required when `push` access to Git repositories that reside on GitHub is required
    * i.e. for *Submitting pull requests (–new-pr)* and *Updating existing pull requests (–update-pr)*
  - see https://github.com/settings/ssh
- **fork of the EasyBuild repositories on GitHub**
  - only required for submitting/updating pull requests (*Submitting pull requests (–new-pr)* and *Updating existing pull requests (–update-pr)*)
  - see `Fork` button (top right) at https://github.com/easybuilders/easybuild-easyconfigs (for example)

See also *Checking status of GitHub integration (–check-github)*.

6.16.2 Configuration

The following sections discuss the EasyBuild configuration options relevant to the GitHub integration features.

Providing a GitHub username (`--github-user`)

To specify your GitHub username, do one of the following:

- use the `--github-user` configuration option on the `eb` command line
- define the `$EASYBUILD_GITHUB_USER` environment variable
- specify `github-user` in your EasyBuild configuration file

(see also *Configuring EasyBuild*)

Installing a GitHub token (`--install-github-token`)

Note: *requires*: GitHub username + `keyring` Python package

A GitHub token is a string of 40 characters that is tied to your GitHub account, allowing you to access the GitHub API authenticated.

Using a GitHub token is beneficial with respect to rate limitations, and enables write permissions on GitHub (e.g. posting comments, creating gists, opening pull requests).

To obtain a GitHub token:

- visit https://github.com/settings/tokens/new and log in with your GitHub account
- enter a token description, for example: "`EasyBuild`"
- make sure (only) the `gist` and `public_repo` (in the `repo` section) scopes are fully enabled
- click `Generate token`
- *copy-paste* the generated token

Note: You will only be able to copy-paste the generated token right after you have created it. The value corresponding to an existing token can *not* be retrieved later through the GitHub interface.

Please keep your token secret at all times; it allows fully authenticated access to your GitHub account!

You can install the GitHub token in your keyring using EasyBuild, so it can pick it up when it needs to, using `eb --install-github-token`:

```
$ eb --github-user example --install-github-token
Token: <copy-paste-your-40-character-token-here>
Validating token...
Token seems to be valid, installing it.
Token 'e3a..0c2' installed!
```

EasyBuild will validate the provided token, to check that authenticated access to your GitHub account works as expected.

Note: EasyBuild will never print the full token value, to avoid leaking it. For debugging purposes, only the first and last 3 characters will be shown.

Specify location of working directories (`--git-working-dirs-path`)

You can specify the location of your Git working directories with one of the following:

- use the `--git-working-dirs-path` configuration option on the `eb` command line
- define the `$EASYBUILD_GIT_WORKING_DIRS_PATH` environment variable
- specify the `git-working-dirs-path` option in your EasyBuild configuration file

The provided path should be the *parent* directory of the location of the working directories (i.e. clones) of the EasyBuild repositories (`easybuild-easyconfigs`, etc.); the assumption is that you keep them all in a single parent directory.

Although not strictly required, this is useful for speeding up `--new-pr` and `--update-pr`, since it allows that the repository can be copied & updated, rather than being cloned from scratch.

6.16.3 Checking status of GitHub integration (`--check-github`)

To check the status of your setup w.r.t. GitHub integration, the `--check-github` command line option can be used.

Using this will trigger EasyBuild to perform a number of checks, and report back on what the test results mean for the different GitHub integration features.

If all requirements are taken care of in your setup, you should see output like this:

```
$ eb --check-github

== temporary log file in case of crash /tmp/eb-xWCpWl/easybuild-hGnKS5.log

Checking status of GitHub integration...

Making sure we're online... OK

* GitHub user... example => OK
* GitHub token... e3f..0c8 (len: 40) => OK (validated)
* git command... OK ("git version 2.7.4 (Apple Git-66); ")
* GitPython module... OK
* push access to example/easybuild-easyconfigs repo @ GitHub... OK
* creating gists... OK
* location to Git working dirs...  OK (/home/example/git-working-dirs)

All checks PASSed!

Status of GitHub integration:
* --from-pr: OK
* --new-pr: OK
* --review-pr: OK
* --update-pr: OK
* --upload-test-report: OK
```

Note: Checking whether push access to GitHub works may take some time, since a recent clone of the easybuild-easyconfigs GitHub repository will be created in the process (at a temporary location).

See also *Requirements*.

6.16.4 Using easyconfigs from pull requests (`--from-pr`)

(supported since EasyBuild v1.13.0)

Via the `--from-pr` command line option (available since EasyBuild v1.13.0), easyconfig files that are added or modified by a particular pull request to the easybuild-easyconfigs GitHub repository can be used (regardless of whether the pull request is merged or not).

This can be useful to employ easyconfig files that are not available yet in the active EasyBuild installation, or to test new contributions by combining `--from-pr` with `--upload-test-report` (see *Uploading test reports (–upload-test-report)*).

When `--from-pr` is used, EasyBuild will download all modified files (easyconfig files and patches) to a temporary directory before processing them.

For example, to use the GCC v4.9.2 easyconfigs contributed via easyconfigs pull request #1177:

```
$ eb --from-pr 1177 --dry-run
== temporary log file in case of crash /tmp/eb-88quZc/easybuild-62fFdo.log
Dry run: printing build status of easyconfigs and dependencies
 * [ ] /tmp/eb-88quZc/files_pr1177/GCC-4.9.2-CLooG-multilib.eb (module: GCC/4.9.2-
↪CLooG-multilib)
 * [ ] /tmp/eb-88quZc/files_pr1177/GCC-4.9.2-CLooG.eb (module: GCC/4.9.2-CLooG)
 * [ ] /tmp/eb-88quZc/files_pr1177/GCC-4.9.2.eb (module: GCC/4.9.2)
== temporary log file /tmp/eb-88quZc/easybuild-62fFdo.log has been removed.
== temporary directory /tmp/eb-88quZc has been removed.
```

Note: To avoid GitHub rate limiting, let EasyBuild know which GitHub account should be used to query the GitHub API, and provide a matching GitHub token; see also *Installing a GitHub token (–install-github-token)*.

Relation between pull requests and current `develop` branch

Since EasyBuild v2.9.0, the current `develop` branch of the central `easybuild-easyconfigs` GitHub repository is taken into account when applicable with `--from-pr`. Before, only the branch corresponding to the specified pull request itself was being considered, which potentially did not reflect the correct state of things, in particular for pull requests based on an outdated branch in which easyconfigs are changed that have been updated in `develop` as well.

As such, the exact semantics of `--from-pr` depends on the state of the specified pull request, i.e. whether or not the pull request was merged already, whether the pull request is mergeable and stable (as indicated by GitHub Actions), etc.

Open stable pull requests

For *open* pull requests that are *stable* (i.e. tests pass and no merge conflicts), the pull request is effectively treated as a patch to the current `develop` branch. This is done to ensure that contributions that are picked up via `--from-pr` are correctly evaluated.

First, the current `develop` branch of the central `easybuild-easyconfigs` GitHub repository is downloaded to a temporary directory. Afterwards, the patch corresponding to the specified pull request is applied on top of the `develop` branch. This results in a correct reflection of how the easyconfig files would look like if the pull request would be merged, which is particularly important for testing of contributions (see also *Uploading test reports (–upload-test-report)*).

Easyconfig files touched by the pull request that are explicitly specified are then picked up from this location; see also *Specifying particular easyconfig files*.

Merged pull requests

For merged pull requests, the current `develop` branch is considered to be the correct state of the easyconfigs touched by the pull request.

Note that this implies that the easyconfig files being picked up are potentially different from the ones that appear in the specified pull request itself, taking into account that further updates may have been applied in the `develop` branch since the pull request got merged.

Closed or unstable pull requests

For closed and unstable pull requests, only the branch corresponding to the pull request itself is being considered, which aligns with the semantics of `--from-pr` as it was before EasyBuild v2.9.0. In this case, the current `develop` branch is *not* taken into account.

Note: A pull request is considered unstable when GitHub reports merge conflicts or when GitHub Actions reports one or more failing tests.

Synergy with `--robot`

Since EasyBuild v1.15.0, the temporary directory containing the easyconfigs (and patch files) from the specified pull request is included in the robot search path.

Up until EasyBuild v2.9.0, this directory was *prepended* to the robot search path, to ensure that easyconfigs that were modified in the respective pull request are picked up via `--robot` when they are required. Thus, for easyconfig files that were available in the pull request as well as locally, the ones from the specified pull request were preferred.

This was changed in EasyBuild v2.9.0, where the directory containing the easyconfigs touched by the pull request is *appended* to the robot search path. This change was made to ensure that customized easyconfig files that are available in the robot search path are preferred over the (patched) easyconfig files from the `develop` branch (see also *Relation between pull requests and current develop branch*).

For example, to build and install `HPL` with the `intel/2015a` toolchain, both of which are contributed via easyconfigs pull request #1238:

```
$ eb --from-pr 1238 --dry-run --robot $HOME/easyconfigs
== temporary log file in case of crash /tmp/eb-A1fRvw/easybuild-Eqc8Oi.log
Dry run: printing build status of easyconfigs and dependencies
 * [x] /home/example/easyconfigs/g/GCC/GCC-4.9.2.eb (module: GCC/4.9.2)
 * [x] /home/example/easyconfigs/i/icc/icc-2015.1.133-GCC-4.9.2.eb (module: icc/2015.
↪1.133-GCC-4.9.2)
 * [x] /home/example/easyconfigs/i/ifort/ifort-2015.1.133-GCC-4.9.2.eb (module: ifort/
↪2015.1.133-GCC-4.9.2)
 * [x] /home/example/easyconfigs/i/iccifort/iccifort-2015.1.133-GCC-4.9.2.eb (module:␣
↪iccifort/2015.1.133-GCC-4.9.2)
 * [x] /home/example/easyconfigs/i/impi/impi-5.0.2.044-iccifort-2015.1.133-GCC-4.9.2.
↪eb (module: impi/5.0.2.044-iccifort-2015.1.133-GCC-4.9.2)
 * [x] /home/example/easyconfigs/i/iimpi/iimpi-7.2.3-GCC-4.9.2.eb (module: iimpi/7.2.
↪3-GCC-4.9.2)
 * [x] /home/example/easyconfigs/i/imkl/imkl-11.2.1.133-iimpi-7.2.3-GCC-4.9.2.eb␣
↪(module: imkl/11.2.1.133-iimpi-7.2.3-GCC-4.9.2)
 * [ ] /tmp/eb-A1fRvw/files_pr1238/intel-2015a.eb (module: intel/2015a)
 * [ ] /tmp/eb-A1fRvw/files_pr1238/HPL-2.1-intel-2015a.eb (module: HPL/2.1-intel-
↪2015a)
```

(continues on next page)

(continued from previous page)

```
== temporary log file /tmp/eb-A1fRvw/easybuild-Eqc8Oi.log has been removed.
== temporary directory /tmp/eb-A1fRvw has been removed.
```

Note that the easyconfigs that are required to resolve dependencies and are available locally in `$HOME/easyconfigs` are being picked up as needed.

Specifying particular easyconfig files

Since EasyBuid v2.0.0 the particular easyconfigs to be used can be specified, rather than using all easyconfigs that are touched by the pull request (which is the default if no easyconfigs are specified alongside `--from-pr`).

For example, to only use `CMake-3.0.0-intel-2015a.eb` from easyconfigs pull request #1330, and ignore the other easyconfigs being contributed in that same pull request for netCDF, WRF, ...:

```
$ eb --from-pr 1330 CMake-3.0.0-intel-2015a.eb --dry-run --robot $HOME/easyconfigs
== temporary log file in case of crash /tmp/eb-QhM_qc/easybuild-TPvMkJ.log
Dry run: printing build status of easyconfigs and dependencies
 * [x] /home/example/easyconfigs/g/GCC/GCC-4.9.2.eb (module: GCC/4.9.2)
 * [x] /home/example/easyconfigs/i/icc/icc-2015.1.133-GCC-4.9.2.eb (module: icc/2015.
↪1.133-GCC-4.9.2)
 * [x] /home/example/easyconfigs/i/ifort/ifort-2015.1.133-GCC-4.9.2.eb (module: ifort/
↪2015.1.133-GCC-4.9.2)
 * [x] /home/example/easyconfigs/i/iccifort/iccifort-2015.1.133-GCC-4.9.2.eb (module:␣
↪iccifort/2015.1.133-GCC-4.9.2)
 * [x] /home/example/easyconfigs/i/impi/impi-5.0.2.044-iccifort-2015.1.133-GCC-4.9.2.
↪eb (module: impi/5.0.2.044-iccifort-2015.1.133-GCC-4.9.2)
 * [x] /home/example/easyconfigs/i/iimpi/iimpi-7.2.3-GCC-4.9.2.eb (module: iimpi/7.2.
↪3-GCC-4.9.2)
 * [x] /home/example/easyconfigs/i/imkl/imkl-11.2.1.133-iimpi-7.2.3-GCC-4.9.2.eb␣
↪(module: imkl/11.2.1.133-iimpi-7.2.3-GCC-4.9.2)
 * [x] /home/example/easyconfigs/i/intel/intel-2015a.eb (module: intel/2015a)
 * [x] /home/example/easyconfigs/n/ncurses/ncurses-5.9-intel-2015a.eb (module:␣
↪ncurses/5.9-intel-2015a)
 * [ ] /tmp/eb-QhM_qc/files_pr1330/CMake-3.0.0-intel-2015a.eb (module: CMake/3.0.0-
↪intel-2015a)
== temporary log file /tmp/eb-QhM_qc/easybuild-TPvMkJ.log has been removed.
== temporary directory /tmp/eb-QhM_qc has been removed.
```

Again, note that locally available easyconfigs that are required to resolve dependencies are being picked up as needed.

6.16.5 Using easyblocks from pull requests (`--include-easyblocks-from-pr`)

(supported since EasyBuild v4.2.0)

Via the `--include-easyblocks-from-pr` command line option, easyblocks that are added or modified by a particular pull request to the easybuild-easyblocks GitHub repository can be used (regardless of whether the pull request is merged or not).

This can be useful to employ easyblocks that are not available yet in the active EasyBuild installation, or to test new contributions by combining `--include-easyblocks-from-pr` with `--from-pr` and `--upload-test-report` (see *Uploading test reports (–upload-test-report)*).

When `--include-easyblocks-from-pr` is used, EasyBuild will download all modified easyblocks to a temporary directory before processing them. Just like with `--include-easyblocks` (see *Including additional easyblocks (–include-easyblocks)*), the easyblocks that are included are preferred over the ones included in the EasyBuild installation.

For example, to use the LAMMPS easyblock contributed via easyblocks pull request #1964 together with the LAMMPS v7Aug2019 easyconfigs contributed via easyconfigs pull request #9884:

```
$ eb --from-pr 9884 --include-easyblocks-from-pr 1964 --list-easyblocks=detailed
== temporary log file in case of crash /tmp/eb-Eq2zsJ/easybuild-1AaWf8.log
EasyBlock (easybuild.framework.easyblock)
...
|   |   |-- EB_LAMMPS (easybuild.easyblocks.lammps @ /tmp/included-easyblocks-rD2HEQ/
↪easybuild/easyblocks/lammps.py)
...
```

6.16.6 Uploading test reports (`--upload-test-report`)

(supported since EasyBuild v1.13.0)

Note: requires that a GitHub token was required `gist` permissions is available, cfr. *Installing a GitHub token (–install-github-token)*

For every installation performed with EasyBuild, a test report is generated. By default, the test report is copied in the installation directory, right next to the log file (see also *Understanding EasyBuild logs*).

Using `--upload-test-report`, the test report can also be pushed to GitHub (as a *gist*, cfr. https://gist.github.com) to share it with others.

Each test report includes:

- an overview of the easyconfigs being processed
- time & date
- the exact `eb` command line that was used
- the full EasyBuild configuration that was in place
- information about the system on which EasyBuild was used (hostname, OS, architecture, etc.)
- the list of modules that was loaded
- the full environment of the session in which `eb` was run (note: can be filtered, see *Filtering the environment details (–test-report-env-filter)*)

For each easyconfig that *failed* to install a partial log will be uploaded as a separate gist, and a link to this gist will be included in the test report.

If `--upload-test-report` is combined with `--from-pr`, a comment referring to the test report (incl. a brief summary) will be placed in the respective pull request. This makes it a very powerful tool when testing contributions.

Note: If you want to easily access a test report without uploading it to GitHub, use `--dump-test-report`.

Example:

```
$ eb --from-pr 3153 --rebuild --upload-test-report
== temporary log file in case of crash /tmp/eb-aqk20q/easybuild-wuyZBV.log
== processing EasyBuild easyconfig /tmp/eb-aqk20q/files_pr3153/EasyBuild/EasyBuild-2.
↪8.1.eb
== building and installing EasyBuild/2.8.1...
...
```

(continues on next page)

(continued from previous page)

```
== COMPLETED: Installation ended successfully
== Results of the build can be found in the log file /home/example/software/EasyBuild/
↪2.8.1/easybuild/easybuild-EasyBuild-2.8.1-20160603.090702.log
== Test report uploaded to https://gist.github.com/1cb2db8a2913a1b8ddbf1c6fee3ff83c␣
↪and mentioned in a comment in easyconfigs PR#3153
== Build succeeded for 1 out of 1
== Temporary log file(s) /tmp/eb-aqk20q/easybuild-wuyZBV.log* have been removed.
== Temporary directory /tmp/eb-aqk20q has been removed.
```

The resulting test report can be viewed at https://gist.github.com/1cb2db8a2913a1b8ddbf1c6fee3ff83c .

Note: It is common to use `--rebuild` in combination with `--upload-test-report`, to ensure that all easyconfigs in the pull request are rebuilt, resulting in a complete test report.

Filtering the environment details (`--test-report-env-filter`)

Since the environment of the session in which `eb` was used may contain sensitive information, it can be filtered through `--test-report-env-filter`.

This configuration option takes a regular expression that is used to determine which environment variables can be included in the test report (based on their name). Environment variables for which the name *matches* the specified regular expression will *not* be included in the test report.

An example of a typical setting:

```
export EASYBUILD_TEST_REPORT_ENV_FILTER='^SSH|USER|HOSTNAME|UID|.*COOKIE.*'
```

6.16.7 Reviewing easyconfig pull requests (`--review-pr`)

A useful tool when reviewing pull requests for the easybuild-easyconfigs repository that add new or update existing easyconfig files is `--review-pr`.

The 'files' tab in the GitHub interface shows the changes being made to existing files; using `--review-pr` the differences with one or more other *similar* easyconfig files, for example the one(s) with the same toolchain (version) and/or software version, can also be evaluated.

This is very useful to quickly see how easyconfig files in pull requests differ from existing easyconfig files, and to maintain consistency across easyconfig files where desired.

The `--review-pr` output consists of a 'multidiff' view per easyconfig file that is being touched by the specified pull request. The exact format of the output depends on whether EasyBuild is configured to allow colored output (enabled by default, see `--color`).

Search criteria for similar easyconfigs

The set of existing similar easyconfig files is determined by specific search criteria; the first one that results in a non-empty set of easyconfigs is retained.

The search criteria consists of a combination of the *software version criterion* with additional restrictions.

The software version criterion is one of the criterions below (considered in order), with `x.y.z` the software version of the easyconfig file from the pull request:

- exact same software version
- same major/minor software version (same `x` and `y`)
- same major software version (same `x`)
- no (partial) version match (so consider any version)

The addition restrictions are the following (also considered in order):

- matching versionsuffix and toolchain name/version
- matching versionsuffix and toolchain name (any toolchain version)
- matching versionsuffix (any toolchain name/version)
- matching toolchain name/version (any versionsuffix)
- matching toolchain name (any versionsuffix, toolchain version)
- no extra requirements (any versionsuffix, toolchain name/version)

6.16.8 Merging easyconfig pull requests (`--merge-pr`)

(supported since EasyBuild v3.3.1)

EasyBuild maintainers need to take the *Requirements for pull requests* into account.

They can merge a pull request to the `easybuild-easyconfigs` repository via `eb --merge-pr`, which will first verify whether the pull request meets the prescribed requirements (at least the ones that can be verified automatically).

For example, for a pull request that is not eligible for merging yet:

```
$ eb --merge-pr 4725
== temporary log file in case of crash /tmp/eb-ba7rVp/easybuild-fBfcwN.log

easybuilders/easybuild-easyconfigs PR #4725 was submitted by vanzod, you are using␣
↪GitHub account 'example'

Checking eligibility of easybuilders/easybuild-easyconfigs PR #4725 for merging...
* targets develop branch: OK
* test suite passes: FAILED => not eligible for merging!
* last test report is successful: (no test reports found) => not eligible for merging!
* approved review: MISSING => not eligible for merging!
* milestone is set: no milestone found => not eligible for merging!

WARNING: Review indicates this PR should not be merged (use -f/--force to do so␣
↪anyway)
```

When a PR is considered eligible for merging, EasyBuild will go ahead and merge it:

```
$ eb --merge-pr 4829
== temporary log file in case of crash /tmp/eb-F9a3oB/easybuild-3B2wdq.log

easybuilders/easybuild-easyconfigs PR #4829 was submitted by SethosII, you are using␣
↪GitHub account 'example'

Checking eligibility of easybuilders/easybuild-easyconfigs PR #4829 for merging...
* targets develop branch: OK
* test suite passes: OK
* last test report is successful: OK
```

(continues on next page)

(continued from previous page)

```
* approved review: OK (by boegel)
* milestone is set: OK (3.3.1)

Review OK, merging pull request!

Adding comment to easybuild-easyconfigs issue #4829: 'Going in, thanks @SethosII!'
Merged easybuilders/easybuild-easyconfigs pull request #4829
```

Note: `eb --merge-pr` can also be run in dry run mode, by also using one of the following options: `--dry-run`, `-D`, `--extended-dry-run`, `-x`.

This results in the same checks being performed but skips the actual merging of the pull request, resulting in messages like:

```
$ eb --merge-pr 4829 --dry-run

...

Review OK, merging pull request!

[DRY RUN] Adding comment to easybuild-easyconfigs issue #4829: 'Going in, thanks␣
↪@SethosII!'
[DRY RUN] Merged easybuilders/easybuild-easyconfigs pull request #4829
```

6.16.9 Submitting new and updating pull requests (`--new-pr`, `--update-pr`)

(supported since EasyBuild v2.6.0)

EasyBuild provides two simple yet powerful features that make contributing to the central EasyBuild repositories significantly easier and less error-prone, especially for people who are not very familiar with `git` and/or GitHub yet:

- `--new-pr` to create new pull requests
- `--update-pr` to update existing pull requests

Previewing easyconfig pull requests (`--preview-pr`)

(supported since EasyBuild v3.5.0)

It is very useful to quickly see how easyconfig files in pull requests differ from existing easyconfig files, and to maintain consistency across easyconfig files where desired.

Maintainers will use `--review-pr` as part of the review process once the PR is submitted (see *Reviewing easyconfig pull requests (–review-pr)*), but it is now possible to preview that output before submitting a PR, eventually fixing any inconsistencies in advance.

To preview a PR before submitting, simply use `--preview-pr` with the list of files to submit:

```
$ eb --preview-pr example.eb example.patch
```

Besides accepting local files instead of a PR number, `--preview-pr` works the same as `--review-pr`, as described in *Comparing with existing easyconfigs (–review-pr)*.

Submitting pull requests (`--new-pr`)

Note: Submitting pull requests using `--new-pr` only works for the `easybuild-easyconfigs` repository, for now. For other repositories, see the manual procedure documented at *Pull requests*.

To create a new pull request, the `--new-pr` command line option can be used, provided that the necessary requirements are fulfilled (see *Requirements*).

In its simplest form, you just provide the location of the file(s) that you want to include in the pull request:

```
$ eb --new-pr test.eb
```

This takes care of all the steps required to make a contribution, i.e.:

- set up a working copy of the relevant EasyBuild repository (e.g., `easybuild-easyconfigs`)
- create a new 'feature' branch, starting from the up-to-date `develop` branch
- renaming easyconfig files according to their `name`, `version`, `versionsuffix` and `toolchain`
- moving easyconfig files to the right location in the repository (e.g. `easybuild/easyconfigs/e/EasyBuild/`)
- staging and committing the files in the feature branch
- pushing the feature branch to your fork of the relevant EasyBuild repository on GitHub
- creating the pull request, targeting the `develop` branch of the central EasyBuild repository (e.g. `easybuilders/easybuild-easyconfigs`)

It should be clear that automating this whole procedure with a single simple `eb` command greatly lowers the bar for contributing, especially since it even alleviates the need for interacting directly with `git` entirely!

The working copy of the EasyBuild repository is created in a temporary location, and cleaned up once the pull request has been created. EasyBuild does *not* make changes to an existing working copy you may have in place already (cfr. *Specify location of working directories (–git-working-dirs-path)*).

Note: When modifying existing files via `--new-pr`, you *must* specify a (meaningful) commit message using `--pr-commit-msg`, see *Controlling pull request metadata*.

Example

For example, to create a pull request for a new version of, let's say, EasyBuild:

```
$ eb --new-pr example.eb
== temporary log file in case of crash /tmp/eb-mWKR9u/easybuild-cTpf2W.log
== copying /home/example/git-working-dirs/easybuild-easyconfigs...
== fetching branch 'develop' from https://github.com/easybuilders/easybuild-
↪easyconfigs.git...

Opening pull request
* target: easybuilders/easybuild-easyconfigs:develop
* from: boegel/easybuild-easyconfigs:20160530131447_new_pr_EasyBuild281
* title: "{tools}[dummy/dummy] EasyBuild v2.8.1"
* description:
"""
```

(continues on next page)

(continued from previous page)

```
(created using `eb --new-pr`)

"""
* overview of changes:
 .../easyconfigs/e/EasyBuild/EasyBuild-2.8.1.eb     | 35 ++++++++++++++++++++++
 1 file changed, 35 insertions(+)

Opened pull request: https://github.com/easybuilders/easybuild-easyconfigs/pull/3153
```

Yes, it's that easy!

Updating existing pull requests (`--update-pr`)

Note: Updating pull requests using `--update-pr` only works for the `easybuild-easyconfigs` repository, for now. For other repositories, see the manual procedure documented at *Pull requests*.

Similarly to creating new pull requests, existing pull requests can be easily updated using `eb --update-pr` (regardless of whether or not they were created with `--new-pr`).

The usage is equally simple, for example to update pull request `#1234` just list the changed/new file(s):

```
$ eb --update-pr 1234 example.eb
```

Again, this take care of the whole procedure required to update an existing pull request:

- set up a working copy of the relevant EasyBuild repository (e.g., `easybuild-easyconfigs`)
- determining the branch corresponding to the pull request, which should be updated by pushing a new commit to it
- checking out that branch
- renaming easyconfig files according to their `name`, `version`, `versionsuffix` and `toolchain`
- moving easyconfig files to the right location in the repository (e.g. `easybuild/easyconfigs/e/EasyBuild/`)
- staging and committing the (changed/new) files
- pushing the updated branch to GitHub

Again, not a single `git` command to be executed; the only thing that is required is the ID of the pull request that should be updated.

Just like with `--new-pr`, this is done in a temporary working copy of the repository, no changes are made to a possible existing working copy.

Note: When using `--update-pr` you *must* specify a (meaningful) commit message via `--pr-commit-msg`, see *Controlling pull request metadata*.

Example

For example, to update pull request #3153 with a changed easyconfig file:

```
eb --update-pr 3153 example.eb
== temporary log file in case of crash /tmp/eb-gO2wJu/easybuild-37Oo2z.log
== Determined branch name corresponding to easybuilders/easybuild-easyconfigs PR
↪#3153: 20160530131447_new_pr_EasyBuild281
== copying /home/example/git-working-dirs/easybuild-easyconfigs...
== fetching branch '20160530131447_new_pr_EasyBuild281' from https://github.com/
↪boegel/easybuild-easyconfigs.git...
Overview of changes:
 easybuild/easyconfigs/e/EasyBuild/EasyBuild-2.8.1.eb | 3 +++
 1 file changed, 3 insertions(+)

Updated easybuilders/easybuild-easyconfigs PR #3159 by pushing to branch boegel/
↪20160530131447_new_pr_EasyBuild281
```

Including patch files in easyconfigs pull requests

Next to providing one or more easyconfig files to add/update via `--new-pr` or `--update-pr`, you can also include patch files that are required by those easyconfig files.

EasyBuild will try and figure out where each patch file should be located (i.e. in the same directory as the easyconfig files that require that patch file), by scanning the provided easyconfigs (or, if needed, scanning *all* existing easyconfig files).

For example:

```
eb --new-pr example.eb example.patch --pr-commit-msg "just an example"
```

Note: When providing one or more patch files, you *must* specify a (meaningful) commit message via `--pr-commit-msg`, see *Controlling pull request metadata*.

Deleting easyconfig files or patches

Next to adding easyconfigs files or patches, or modifying existing ones, you can also specify to *delete* particular files, by including a colon character `:` before the name of the file.

For example:

```
eb --new-pr :example-1.0.eb --pr-commit-msg "delete example-1.0.eb easyconfig file"
```

Note: When deleting existing files, you *must* specify a custom commit message using `--pr-commit-msg`, see also *Controlling pull request metadata*.

Controlling pull request metadata

You can control the metadata for pull requests using the following configuration options:

- `--pr-branch-name`: branch name for new pull requests
- `--pr-commit-msg`: commit message to use when creating new or updating existing pull requests
- `--pr-descr`: pull request description

- `--pr-title`: pull request title

EasyBuild will use sensible defaults for each of these, see below.

Default branch name for new pull requests

The branch name for new pull requests will be composed from:

- a timestamp, down to the second in an attempt to make it unique
  - example: `20160513141133` for a pull request created on May 13th 2016, 2:11:33 PM
- a label `new_pr`
- the software name and version of the first easyconfig file, with some filtering (e.g. remove `.`'s)
  - example: `GCC530` for GCC v5.3.0

Full example: `20160513141133_new_pr_GCC530`

Although there is usually no reason to change this default, it can be done if desired using `--pr-branch-name` when opening a new pull request with `--new-pr`.

Default commit message

EasyBuild will try to generate an appropriate default commit message when only new easyconfigs are being added via `--new-pr`.

When existing easyconfigs are being modified, patch files are being added/updated or `--update-pr` is used, a custom (meaningful) commit message *must* be provided via `--pr-commit-msg` (see *Controlling pull request metadata*).

Default pull request description

By default, the pull description will only contain the following text:

```
(created using eb --new-pr)
```

It is generally advised to provide more descriptive information, although the changes made by the pull request may be self-explanatory (e.g. when only adding new easyconfig files).

To change this default text, you can either use `--pr-descr` or edit the description via the GitHub interface after the pull request has been opened.

Particularly useful information to specify here is dependencies on other pull requests, by copy-pasting the respective URLs with a short descriptive message like '`depends on PR <URL>`'.

Default pull request title

The pull request title is derived from the easyconfig files being changed/added, taking into account the recommendation for easyconfig pull requests to clearly specify module class, toolchain, software name/version, as follows: `{<module_class>}[<toolchain>] <software_name> v<software_version>`.

For example, when opening a pull request for an easyconfig for Python 2.7.11 with the `intel/2016a` toolchain, the default pull request title will be something like: `{lang}[intel/2016a] Python v2.7.11`.

If multiple easyconfig files are provided, the respective software names/versions will be included separated by a `,`, up until the first 3 easyconfig files (to avoid excessively lengthy pull request titles).

In case (only) existing easyconfig files are being changed, it's advisable to provide a more descriptive title using `--pr-title`.

Configuring `--new-pr` and `--update-pr`

By default, `--new-pr` and `--update-pr` affect pull requests to the central `easybuilders/easybuild-easyconfigs` repository.

However, this can be changed with the following configurations options:

- `--pr-target-account` (default: `easybuilders`): target GitHub account for new pull requests
- `--pr-target-branch` (default: `develop`): target branch for new pull requests
- `--pr-target-repo` (default: `easybuild-easyconfigs`): target repository for new pull requests

Synergy with `--dry-run`/`-D` and `--extended-dry-run`/`-x`

Both `--new-pr` and `--update-pr` are 'dry run-aware', in the sense that you can combine them with either `--dry-run`/`-D-` or `--extended-dry-run`/`-x` to preview the pull request they would create/update without actually doing so.

For example:

```
$ eb --new-pr EasyBuild-2.9.0.eb -D
== temporary log file in case of crash /tmp/eb-1ny69k/easybuild-UR1Wr4.log
== copying /home/example/git-working-dirs/easybuild-easyconfigs...
== fetching branch 'develop' from https://github.com/easybuilders/easybuild-
↪easyconfigs.git...

Opening pull request [DRY RUN]
* target: easybuilders/easybuild-easyconfigs:develop
* from: boegel/easybuild-easyconfigs:20160603105641_new_pr_EasyBuild290
* title: "{tools}[dummy/dummy] EasyBuild v2.9.0"
* description:
"""
(created using `eb --new-pr`)

"""
* overview of changes:
 .../easyconfigs/e/EasyBuild/EasyBuild-2.9.0.eb     | 35 ++++++++++++++++++++++++
 1 file changed, 35 insertions(+)
```

The only difference between using `--dry-run` and `--extended-dry-run` is that the latter will show the full diff of the changes (equivalent to `git diff`), while the former will only show a summary of the changes (equivalent to `git diff --stat`, see example above).

6.17 Locks to prevent duplicate installations running at the same time

Easybuild creates a lock before starting the installation of an easyconfig file, to avoid a collision between multiple installations running at the same time.

If an EasyBuild session tries to install an easyconfig file when a lock for that installation already exists, it will be automatically aborted with an error like "`Lock ... already exists, aborting!`".

Note: Locking of installations was implemented in EasyBuild version 4.2.0 .

Contents

6.17.1 Locking implementation details

Easybuild will create a lock when starting an installation if no corresponding lock exists yet, regardless of whether the installation was requested explicitly or is performed to resolve a dependency.

The name of the lock corresponds to the *full* path of the installation directory, with slashes (`/`) and dashes (`-`) replaced by underscores (_), and with an additional `.lock` added at the end.

Locks are created in the *Locks directory*.

The lock created by EasyBuild is an empty directory (rather than a file), because that can be created more atomically on modern filesystems.

For example, if `OpenFOAM-7-foss-2019b.eb` is being installed to `/apps/easybuild/software`, an empty directory that serves as a lock for this installation will be created at `/apps/easybuild/software/.locks/_apps_easybuild_software_OpenFOAM_7_foss_2019b.lock` (assuming the default *Locks directory* is used).

A lock is automatically removed by Easybuild when the installation ends, regardless of whether the installation was successful or not. Therefore, new installations of the same easyconfig will be aborted in case of:

- another installation for the same easyconfig is in progress;
- a previous installation of the same easyconfig was abruptly interrupted;

6.17.2 Removing locks

If a previous installation was abruptly interrupted and a lock was left in place, it can be easily removed using the `rmdir` command (since the lock is actually an empty directory).

6.17.3 Configuration options related to installation locks

The behaviour of the locking mechanism in Easybuild can be tuned with the following configuration options:

Ignoring locks (`--ignore-locks`)

Using the `--ignore-locks` configuration option, you can instruct EasyBuild to ignore any existing locks. Locks that exist are left untouched, even if the installation completes successfully.

Use this with caution, since installations may be (partially) overwritten if another EasyBuild session is also performing those installations!

Waiting for locks to be removed (`--wait-on-lock`)

Using the `--wait-on-lock` configuration option, you can change how EasyBuild deals with existing locks, by specifying how frequently EasyBuild should check whether an existing lock was removed. By specifying a non-zero value `S`, you can indicate how many seconds EasyBuild should wait before checking again whether the lock is still in place.

Note: EasyBuild will wait indefinitely for an existing lock to be removed if `--wait-on-lock` is set to a non-zero value...

If the lock is never removed, the EasyBuild session will never terminate; it will keep checking every `S` seconds whether the lock is still in place.

By default, EasyBuild will *abort* the installation with an error like "`Lock ... already exists, aborting!`" if a corresponding lock already exists, which is equivalent to setting `--wait-on-lock` to zero (`0`), implying that no waiting should be done at all.

6.17.4 Locks directory

If desired, an alternate location where locks should be created and checked for can be specified via the `--locks-dir` configuration option.

Note: Keep in mind that a path on a *shared* filesystem should be used, to ensure that active EasyBuild sessions running on different systems use the same locks directory.

By default, locks are created in a hidden subdirectory `.locks` in the top-level `software` installation directory; that is, the `software` subdirectory of the `installpath` configuration setting of the active EasyBuild session.

6.18 Manipulating dependencies

A couple of different ways are available to manipulate the list of dependencies that are specified for the software packages being installed.

Contents

6.18.1 Filtering out dependencies using `--filter-deps`

To avoid that certain dependencies are being installed, a list of software names can be specified to `--filter-deps`. Any time a dependency with a name from this list is specified, it will be simply filtered out by EasyBuild, and thus disregarded when resolving dependencies, loading modules for the dependencies in the build environment, and including '`module load`' statements in the generated module files.

This can be useful when particular tools and libraries are already provided by OS packages (or in some other way), and should not be reinstalled as modules by EasyBuild.

For example:

- overview of dependencies of HDF5:

```
$ eb HDF5-1.8.13-intel-2015a.eb -D
 ...
 * [ ] $CFGS/i/intel/intel-2015a.eb (module: intel/2015a)
 * [ ] $CFGS/z/zlib/zlib-1.2.8-intel-2015a.eb (module: zlib/1.2.8-intel-2015a)
 * [ ] $CFGS/s/Szip/Szip-2.1-intel-2015a.eb (module: Szip/2.1-intel-2015a)
 * [ ] $CFGS/h/HDF5/HDF5-1.8.13-intel-2015a.eb (module: HDF5/1.8.13-intel-2015a)
```

- overview of dependencies of HDF5, with zlib and Szip excluded:

```
$ eb HDF5-1.8.13-intel-2015a.eb --filter-deps=zlib,Szip -D
 ...
 * [ ] $CFGS/i/intel/intel-2015a.eb (module: intel/2015a)
 * [ ] $CFGS/h/HDF5/HDF5-1.8.13-intel-2015a.eb (module: HDF5/1.8.13-intel-2015a)
```

Filtering dependencies based on version

Since EasyBuild v3.8.0, filtering dependencies based on their version is also supported.

For each entry in `--filter-deps`, the expected format is either:

- `<name>`: a software name without any version specification, to indicate that this dependency should *always* be filtered out (regardless of its version)
  - examples: `X11`, `zlib`
- `<name>=<version>`: a software name and version, to indicate that a *particular version* of this dependency should be filtered out
  - examples: `zlib=1.2.8`, `ncurses=5.9`
- `<name>=<lower_limit>:<upper_limit>`: a software name and a version range, to indicate that any version of this dependency that falls *within the specified range* should be filtered out

When a version range is specified, a lower limit and/or upper limit *can* be specified (separated by `:`). In other words, the version range can be open ended. The `:` separator is strictly required.

For both limits, you *must* indicate whether it is either:

- *inclusive*, by using `[` for the lower limit, and `]` for the upper limit
- *exclusive*, by using `]` for the lower limit, and `[` for the upper limit

Examples:

- to filter out `zlib` version `1.2` up until `1.2.99` (incl.), use: `zlib=[1.2:1.2.99]`
- to filter out `ncurses` version 5.0 or newer, use: `ncurses=[5.0:`
- to filter out any version of `CMake` that is older than `4.0`, use: `CMake=:4.0[`

6.18.2 Installing dependencies as hidden modules using `--hide-deps`

Selected software packages can be marked to be installed as hidden modules (i.e., modules that are not visible via '`module avail`', but can be loaded) whenever they are included as a dependency, via the `--hide-deps` configuration option.

For example (note the preceding '`.`' in the last part of the module names for zlib and Szip):

```
$ eb HDF5-1.8.13-intel-2015a.eb --hide-deps=zlib,Szip -D
 ...
 * [ ] $CFGS/i/intel/intel-2015a.eb (module: intel/2015a)
 * [ ] $CFGS/z/zlib/zlib-1.2.8-intel-2015a.eb (module: zlib/.1.2.8-intel-2015a)
 * [ ] $CFGS/s/Szip/Szip-2.1-intel-2015a.eb (module: Szip/.2.1-intel-2015a)
 * [ ] $CFGS/h/HDF5/HDF5-1.8.13-intel-2015a.eb (module: HDF5/1.8.13-intel-2015a)
```

Note: Using Lmod (version >= 5.7.5), hidden modules can be made visible in the output of '`module avail`' using the `--show-hidden` option.

For example:

```
$ module avail bzip2

Use "module spider" to find all possible modules.
Use "module keyword key1 key2 ..." to search for all possible modules matching any of␣
↪the "keys".

$ module --show-hidden avail bzip2
----- /home/example/.local/easybuild/modules/all -----
bzip2/.1.0.6

Use "module spider" to find all possible modules.
Use "module keyword key1 key2 ..." to search for all possible modules matching any of␣
↪the "keys".
```

6.18.3 Using minimal toolchains for dependencies

By default, EasyBuild will try to resolve dependencies using the same toolchain as the one that is used for the software being installed, unless a specific toolchain is specified for the dependency itself (see *Dependencies*).

Using the `--minimal-toolchains` configuration option, you can instruct EasyBuild to consider subtoolchains for dependencies in the reverse order (from the bottom of the toolchain hierarchy to the top). This can be useful to refrain from having to frequently hardcode specific toolchains in order to avoid having the same dependency version installed with multiple toolchains that are compatible with each other. Although hardcoding the toolchain for dependencies will work fine, it severely limits the power of other EasyBuild features, like `--try-toolchain` for example.

When instructed to use minimal toolchains, EasyBuild will check whether an easyconfig file is available (in the robot search path, see *Searching for easyconfigs: the robot search path*) for that dependency using the different subtoolchains of the toolchain specified for the 'parent' software. Subtoolchains are considered 'bottom-up', i.e. starting with the most minimal subtoolchain (typically a compiler-only toolchain), and then climbing up towards the toolchain that is specified for the software being installed.

Note that if a specific toolchain is specified for a particular dependency, EasyBuild will stick to using it, even when instructed to use minimal toolchains. Also note that as of v3.0, if no easyconfig exists to resolve a dependency using the default toolchain EasyBuild will search for the dependency using a compatible subtoolchain (the difference being that the search order is from the top of the toolchain hierarchy to the bottom).

Considering `system` as minimal toolchain

The system_toolchain is only considered as the most minimal subtoolchain if the `--add-system-to-minimal-toolchains` configuration option is enabled. By default, this configuration option is *disabled*.

Taking existing modules into account

You can instruct EasyBuild to take existing modules into account when determining which subtoolchain should be used for each of the dependencies, using the `--use-existing-modules` configuration option.

By default existing modules are ignored, meaning that the EasyBuild dependency resolution mechanism will pick a minimal toolchain for each dependency solely based on the available easyconfig files (if the `--minimal-toolchains` configuration option is enabled, that is).

With `--use-existing-modules` enabled, EasyBuild will first check whether modules exist for the dependencies that were built with the toolchain or any of the subtoolchains (searching top-down). If so, the toolchain of the first encountered existing module will determine the toolchain being selected. If not, the toolchain to use will be determined based on the available easyconfig files.

Example

Consider the following (partial) easyconfig file for Python v2.7.9 with the `foss/2015b` toolchain:

```
name = 'Python'
version = '2.7.9'

toolchain = {'name': 'foss', 'version': '2015b'}

dependencies = [
    ('zlib', '1.2.8'),
]
```

When the `--minimal-toolchains` configuration option is enabled, EasyBuild will also consider the subtoolchains `GCC/4.9.3` and `gompi/2015b` of the `foss/2015b` toolchain (in that order) as potential minimal toolchains when determining the toolchain to use for dependencies.

So, for the zlib v1.2.8 dependency included in the example above, the following scenarios are possible:

- without the use of `--minimal-toolchains`, the default behaviour of EasyBuild is to first consider the `foss/2015b` toolchain for zlib v1.2.8, if no such easyconfig file is found, it will continue searching using the `gompi/2015b` toolchain, and finally the `GCC/4.9.3` toolchain

- if (only) `--minimal-toolchains` is enabled, EasyBuild will search for an easyconfig file for zlib v1.2.8 using the `GCC/4.9.3` toolchain; if no such easyconfig file is found, it will continue searching using the `gompi/2015b` toolchain, and finally the `foss/2015b` toolchain
- if `--add-system-to-minimal-toolchains` is also enabled, EasyBuild will try locating an easyconfig file for zlib v1.2.8 that uses the `system` toolchain prior to considering the `GCC/4.9.3` toolchain
- additionally, with `--use-existing-modules` enabled, EasyBuild will first check whether a `zlib` module for version 1.2.8 built with the (sub)toolchains being considered exists; if not, it will search for an easyconfig file for zlib as outlined above

6.19 Packaging support

Contents

Note: Packaging support was added as an experimental feature in EasyBuild v2.2.0 (cfr. *Experimental features*). Since EasyBuild v2.5.0, it is considered stable.

6.19.1 Prerequisites

EasyBuild leverages FPM to create binary packages (RPMs, Debian files, etc.).

Hence, FPM must be available in some way or another. One way is via EasyBuild, for example by installing a module for FPM using one of the available easyconfig files.

EasyBuild will also take care of installing Ruby for you (which is required for FPM itself):

```
$ export EASYBUILD_PREFIX=/home/example

$ eb FPM-1.3.3-Ruby-2.1.6.eb --robot
[...]
== building and installing Ruby/2.1.6...
[...]
== COMPLETED: Installation ended successfully
[...]
== building and installing FPM/1.3.3-Ruby-2.1.6...
[...]
== COMPLETED: Installation ended successfully
== Results of the build can be found in the log file /home/example/software/FPM/1.3.3-
↪Ruby-2.1.6/easybuild/easybuild-FPM-1.3.3-20150524.181859.log
== Build succeeded for 2 out of 2

$ module load FPM/1.3.3-Ruby-2.1.6
```

(continues on next page)

(continued from previous page)

```
$ fpm --version
1.3.3
```

6.19.2 Configuration options

Several configuration options related to packaging support are available.

- `--package`:
  - enables packaging; other options will be void unless this option is enabled
- `--package-tool=<tool>`:
  - specifies which tool you wish to package with; for now, only `fpm` is supported (and is set as default)
- `--package-type=<type>`:
  - specifies which type of package you wish to build, which is passed through to `fpm` (as target type); examples include: `rpm` (default), `deb`, ... (see https://github.com/jordansissel/fpm/wiki#overview)
- `--package-naming-scheme=<PNS>`:
  - specifies which package naming scheme to use; default: `EasyBuildPNS`
- `--packagepath`:
  - specifies destination path of packages being built
- `--package-release`:
  - specifies the package release (default: `1`); typically, this should be an integer value

Note: Changing the package naming scheme should be done with caution. For example, RPM will only allow one package of a particular *name* to be installed, so if you wish multiple versions of a package to be installed at the same time you need to ensure variables like the software version are included in the package name.

6.19.3 Usage

To make EasyBuild generate packages, just use `--package`. By default, this will make EasyBuild leverage FPM to create RPMs:

```
$ export EASYBUILD_PREFIX=/home/example
$ eb --package Perl-5.20.1-GCC-4.9.2-bare.eb --robot
[...]
== building and installing Perl/5.20.1-GCC-4.9.2-bare...
== fetching files...
== creating build dir, resetting environment...
== unpacking...
== patching...
== preparing...
== configuring...
== building...
== testing...
== installing...
== taking care of extensions...
```

(continues on next page)

(continued from previous page)

```
== postprocessing...
== sanity checking...
== cleaning up...
== creating module...
== packaging...
== COMPLETED: Installation ended successfully
== Results of the build can be found in the log file /home/example/software/Perl/5.20.
↪1-GCC-4.9.2-bare/easybuild/easybuild-Perl-5.20.1-20150527.023522.log
== Build succeeded for 1 out of 1
```

Packages will be located in the directory indicated by the `--packagepath` configuration option; by default, this corresponds to `$prefix/packages`.

By defaulit, the package will have the following properties:

```
$ rpm -qip --requires --provides /home/example/packages/Perl-5.20.1-GCC-4.9.2-bare.
↪eb2.2.0-1.x86_64.rpm
Name        : Perl-5.20.1-GCC-4.9.2-bare
Version     : eb2.2.0
Release     : 1
Architecture: x86_64
Install Date: (not installed)
Group       : default
Size        : 64539427
License     : unknown
Signature   : (none)
Source RPM  : Perl-5.20.1-GCC-4.9.2-bare.eb2.2.0-1.x86_64.src.rpm
Build Date  : Tue 07 Jul 2015 11:27:54 PM EDT
Build Host  : 59e46bbf1cd0
Relocations : /
Packager    : <easybuild@59e46bbf1cd0>
Vendor      : easybuild@59e46bbf1cd0
URL         : http://example.com/no-uri-given
Summary     : no description given
Description :
no description given
GCC-4.9.2-dummy-dummy
rpmlib(PartialHardlinkSets) <= 4.0.4-1
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
rpmlib(CompressedFileNames) <= 3.0.4-1
Perl-5.20.1-GCC-4.9.2-bare
Perl-5.20.1-GCC-4.9.2-bare = eb2.2.0-1
Perl-5.20.1-GCC-4.9.2-bare(x86-64) = eb2.2.0-1
```

6.19.4 Packaging existing installations

To create packages for existing software installations (performed using EasyBuild), combine `--package` with `--skip --rebuild`:

```
$ eb --package Perl-5.20.1-GCC-4.9.2-bare.eb --skip --rebuild
[...]
== building and installing Perl/5.20.1-GCC-4.9.2-bare...
== fetching files...
== creating build dir, resetting environment...
== unpacking [skipped]
```

(continues on next page)

(continued from previous page)

```
== patching [skipped]
== preparing...
== configuring [skipped]
== building [skipped]
== testing [skipped]
== installing [skipped]
== taking care of extensions...
== postprocessing [skipped]
== sanity checking...
== cleaning up...
== creating module...
== packaging...
== COMPLETED: Installation ended successfully
== Results of the build can be found in the log file /home/example/software/Perl/5.20.
↪1-GCC-4.9.2-bare/easybuild/easybuild-Perl-5.20.1-20150527.041734.log
== Build succeeded for 1 out of 1
```

6.20 Partial installations

Several ways of performing partial installations are supported. These may be useful when debugging a particular issue with the installation procedure being performed by EasyBuild, updating existing software installations or module files, or after changing the EasyBuild configuration (e.g., switching to module files in Lua syntax or a different module naming scheme).

Contents

6.20.1 Stopping the installation procedure *after* a step using `-s/--stop`

To stop the installation procedure *after* a specific step in the installation procedure, the `-s/--stop` command line option can be used; the name of the step must be supplied as an argument.

The following step names are recognized (listed in execution order): `fetch`, `ready`, `source`, `patch`, `prepare`, `configure`, `build`, `test`, `install`, `extensions`, `package`, `postproc`, `sanitycheck`, `cleanup`, `module`, `testcases`.

Example usage:

```
$ eb GCC-4.9.2.eb --stop configure
== temporary log file in case of crash /tmp/eb-X2Z0b7/easybuild-mGxmNb.log
```

(continues on next page)

(continued from previous page)

```
== processing EasyBuild easyconfig /home/example/GCC-4.9.2.eb
== building and installing GCC/4.9.2...
== fetching files...
== creating build dir, resetting environment...
== unpacking...
== patching...
== preparing...
== configuring...
== COMPLETED: Installation STOPPED successfully
== Results of the build can be found in the log file /dev/shm/example/GCC/4.9.2/dummy-
↪dummy/easybuild/easybuild-GCC-4.9.2-20150430.091644.log
== Build succeeded for 1 out of 1
== temporary log file(s) /tmp/eb-X2Z0b7/easybuild-mGxmNb.log* have been removed.
== temporary directory /tmp/eb-X2Z0b7 has been removed.
```

6.20.2 Fetching sources with `--fetch`

It may be useful to be able to batch-download sources on a machine where no modules tool is installed. The `--fetch` option, which is equivalent with `--stop fetch --ignore-osdeps`, addresses this requirement.

Example usage:

```
$ eb GCCcore-6.2.0.eb --fetch
== temporary log file in case of crash /tmp/eb-1ZZX2b/easybuild-NSmm5P.log
== processing EasyBuild easyconfig /home/example/GCCcore-6.2.0.eb
== building and installing GCCcore/6.2.0...
== fetching files...
== COMPLETED: Installation STOPPED successfully
== Results of the build can be found in the log file(s) /dev/shm/example/GCC/4.9.2/
↪dummy-dummy/easybuild/easybuild-GCCcore-6.2.0-20180330.170523.log
== Build succeeded for 1 out of 1
== Temporary log file(s) /tmp/eb-1ZZX2b/easybuild-NSmm5P.log* have been removed.
== Temporary directory /tmp/eb-1ZZX2b has been removed.
```

Note: `--fetch` can be used in conjunction with the `--robot` and `--robot-path` options to download sources of the whole dependency tree of an easyconfig (see *Enabling dependency resolution, –robot / -r and –robot-paths*).

Note: Sources will be downloaded in the default location (see *Source path (–sourcepath)*), unless EasyBuild is configured via the `--sourcepath` option.

6.20.3 Installing additional extensions using `-k/--skip`

For software applications that may include *Extensions*, it is often required to install one or more additional extensions without having to reinstall the software package (and all extensions) from scratch. For this purpose, the `-k`/`--skip` command line option is available.

To actually skip an existing software installation and all installed extensions, a corresponding module must be available already; if not, the installation procedure will be performed from scratch. To trigger the installation of missing extensions, `--rebuild` (or `--force`, see _force_option) must be used as well; without it, the installation procedure will be skipped as a whole (since the module is already available).

When `--skip` is combined with `--rebuild`, EasyBuild will:

i) ensure that all (extension) sources are available (and try to fetch them if needed);

ii) prepare the build environment;

iii) check which extensions have not been installed yet;

iv) install the missing extensions;

v) run the sanity check (which includes checking that all extensions are available)

vi) regenerate the module file (since it contains a list of installed extensions)

Example usage:

```
$ eb Python-2.7.9-intel-2015a.eb --skip
...
== Python/2.7.9-intel-2015a is already installed (module found), skipping
== No easyconfigs left to be built.
== Build succeeded for 0 out of 0
```

```
$ eb Python-2.7.9-intel-2015a.eb --skip --rebuild
...
== building and installing Python/2.7.9-intel-2015a...
...
== configuring [skipped]
== building [skipped]
== testing [skipped]
== installing [skipped]
== taking care of extensions...
...
== sanity checking...
== cleaning up...
== creating module...
== COMPLETED: Installation ended successfully
```

Note: Upgrading of extensions to a newer version does not work (yet) using `--skip`, because the way in which extensions are checked for availability, i.e. the extensions filter, is (usually) version-agnogstic.

Note: The '`skipsteps`' easyconfig parameter has a different purpose, i.e. to specify which installation steps should *always* be skipped when the installation of a particular software package is performed, no matter whether the software or corresponding module is already available or not.

Note: When `--skip` is used, a backup is created for all existing module files that are regenerated. To disable backing up of module files, use `--disable-backup-modules` (see also *Backing up of existing modules (–backup-modules)*).

6.20.4 Only (re)generating (additional) module files using `--module-only`

Since EasyBuild v2.1, it is possible to only (re)generate the module file that matches the specifications in the easyconfig file, using `--module-only`. Depending on the use case, additional options should be supplied.

Usually `--rebuild` is also required, either to ignore the existing module file (if the module is available under the same name as the one being (re)generated), or to skip the sanity check that verifies the software installation (if no software installation is available).

Combining `--module-only` with `--installpath-modules` is also a common use case, to generate the module file in a (test) location other than the software installation prefix (see *Software and modules install path (–installpath, –installpath-software, –installpath-modules)*).

Note: Although `--module-only` was already supported in EasyBuild v2.1.0, we strongly recommend to use EasyBuild v2.1.1 or a more recent version, because of some critical bug fixes with respect to `--module-only` (see *EasyBuild v2.1.1 (May 18th 2015)*).

Use cases:

- *Only (re)generating (existing) module file*
- *Generating additional module files*

Note: When `--module-only` is used, a backup is created for all existing module files that are regenerated. To disable backing up of module files, use `--disable-backup-modules` (see also *Backing up of existing modules (–backup-modules)*).

Only (re)generating (existing) module file

To only generate a module file (i.e., skip actually building and installing the software), or to regenerate an existing module file, `--module-only` can be used.

In the former case, enabling `--rebuild` is required because the sanity check step that verifies whether the installation produced the expected files and/or directories is not skipped unless forced. In the latter case, `--rebuild` must be used to make EasyBuild ignore that the module is already available according to the modules tool.

Example usage:

- only generate module file:

```
$ module avail GCC
---------------------------- /home/example/.local/modules/all ---------------------
↪--------
GCC/4.8.2

$ eb GCC-5.1.0.eb --module-only --rebuild
...
== building and installing GCC/5.1.0...
== fetching files [skipped]
...
== configuring [skipped]
== building [skipped]
== testing [skipped]
== installing [skipped]
...
== sanity checking [skipped]
== cleaning up [skipped]
== creating module...
== COMPLETED: Installation ended successfully
...
```

(continues on next page)

(continued from previous page)

```
$ module avail GCC

---------------------------- /home/example/.local/modules/all --------------------
↪--------
GCC/4.8.2     GCC/5.1.0
```

- regenerate existing module file:

```
$ module avail GCC/4.8.2

---------------------------- /home/example/.local/modules/all --------------------
↪--------
GCC/4.8.2

$ ls -l /home/example/.local/modules/all/GCC/4.8.2
-rw-rw-r-- 1 example example 1002 Jan 11 17:19 /home/example/.local/modules/all/
↪GCC/4.8.2

$ eb GCC-4.8.2.eb --module-only --rebuild
...
== building and installing GCC/4.8.2...
...
== sanity checking [skipped]
== creating module...
== COMPLETED: Installation ended successfully
...

$ ls -l /home/example/.local/modules/all/GCC/4.8.2
-rw-rw-r-- 1 example example 1064 Apr 30 10:54 /home/example/.local/modules/all/
↪GCC/4.8.2
```

Generating additional module files

Generating an additional module file, next to the one(s) already available, is also supported. This can achieved by combining `--module-only` with additional configuration options that apply to the module generation.

Examples:

- to generate a module file in Lua syntax, next to an already existing module file in Tcl syntax, `--module-only --module-syntax=Lua` can be used:

```
$ module avail GCC/4.8.2

---------------------------- /home/example/.local/modules/all --------------------
↪--------
GCC/4.8.2

$ ls -l /home/example/.local/modules/all/GCC/4.8.2*
-rw-rw-r-- 1 example example 1064 Apr 30 10:54 /home/example/.local/modules/all/
↪GCC/4.8.2

$ eb GCC-4.8.2.eb --modules-tool=Lmod --module-only --module-syntax=Lua --rebuild
...
== building and installing GCC/4.8.2...
...
```

(continues on next page)

(continued from previous page)

```
== sanity checking [skipped]
== creating module...
== COMPLETED: Installation ended successfully
...

$ ls -l /home/example/.local/modules/all/GCC/4.8.2*
-rw-rw-r-- 1 example example 1064 Apr 30 10:54 /home/example/.local/modules/all/
↪GCC/4.8.2
-rw-rw-r-- 1 example example 1249 Apr 30 11:56 /home/example/.local/modules/all/
↪GCC/4.8.2.lua
```

Note: Since only Lmod can consume module files in Lua syntax, it must be used as a modules tool; see also *Module files syntax (–module-syntax)*.

Only changing the syntax of the module file does not affect the module name, so Lmod will report the module as being available. Hence, `--rebuild` must be used here as well.

- to generate a module file using a different naming scheme, `--module-only` can be combined with `--module-naming-scheme`:

```
$ eb --installpath-modules=$HOME/test/modules --module-only --module-naming-
↪scheme=HierarchicalMNS --rebuild
...
== building and installing Core/GCC/4.8.2...
...
== sanity checking [skipped]
== creating module...
== COMPLETED: Installation ended successfully

$ module unuse $HOME/.local/modules/all
$ module use $HOME/test/modules/all
$ module avail

---------------------------- /home/example/test/modules/all -----------------------
↪------
Core/GCC/4.8.2
```

Note: Modules that are generated used different module naming schemes should *never* be mixed, hence the use of `--installpath-modules`, see also *Direct options: –installpath-software and –installpath-modules*.

Note: The modules files generated using the specified module naming scheme will most likely **not** be tied to an existing software installation in this case (unless the software installation was already there somehow), since the name of the subdirectory of the software installation prefix is also governed by the active module naming scheme. This is also why `--rebuild` must be used in the example above (to skip the sanity check that verifies the software installation).

Thus, this is only useful to assess how the module tree would look like under a particular module naming scheme; the modules themselves are useless since they point to empty installation directories.

To tie a module file generated using to an existing software installation that was performed under a different module naming scheme, a simple module naming scheme can be implemented that mixes two modules naming schemes, by providing the name of the software installation subdirectory using one scheme, and the module names (and other metadata for module files) with the other.

An example of such a module naming scheme is `MigrateFromEBToHMNS`, which allows to generate module files using the hierarchical module naming scheme implemented by `HierarchicalMNS` for the software installed in subdirectories following the default EasyBuild module naming scheme `EasyBuildMNS`. The `MigrateFromEBToHMNS` module naming scheme is available since EasyBuild v2.2.0.

6.21 Support for RPATH

Since EasyBuild v3.5.2, (stable) support is available for using RPATH.

Contents

6.21.1 What is RPATH?

RPATH is a mechanism to include a list of directories in a binary where required shared libraries may be available. These locations are considered by the dynamic loader (`ld*.so`) to locate the libraries that are required by a particular binary.

Hence, instructing the dynamic linker (`ld`) to include RPATH entries in a binary is an alternative to specifying library locations through `$LD_LIBRARY_PATH`.

For more information on RPATH, see https://linux.die.net/man/8/ld-linux

6.21.2 Why RPATH?

Using RPATH can be interesting for a number of reasons:

- it can help to avoid a (too) large environment, since:
  - `$LD_LIBRARY_PATH` does not need to be set anymore for all dependencies providing libraries
  - it leads to fewer runtime dependencies (and hence fewer modules need to be loaded)
- binaries can be used without problems w.r.t. resolving required libraries in other environments
- it may result in better startup performance, since `$LD_LIBRARY_PATH` does not have to be iterated over

A minor downside is that it becomes less trivial to move installations of dependencies to a different location (which is something that you should not do without good reason anyway).

6.21.3 Enabling RPATH linking

To instruct EasyBuild to enable RPATH linking, use the `--rpath` configuration option.

6.21.4 Implementation

When EasyBuild is configured to use RPATH, wrapper scripts are put in place for the dynamic linker commands (`ld`, `ld.gold`), as well as for every compiler command that is part of the toolchain being used. This is done during the `prepare` step.

The wrapper scripts will analyze and rewrite the list of arguments supplied to the command they are wrapping as needed, i.e.:

- inject an `-rpath` argument for every `-L` argument that specifies a library directory (with some exceptions, see also *Filtering RPATH entries via –rpath-filter*)
- filter out arguments that affect RPATH (e.g., `--enable-new-dtags`)
- ensure that the library subdirectories (`lib`, `/lib64`) of the installation directory also have an RPATH entry
- include additional arguments related to RPATH (e.g. `--disable-new-dtags`)

As such, `ps` may show something like:

```
\_ /bin/bash /tmp/eb-M3393U/tmpRVJqwr/rpath_wrappers/gcc -O2 example.c -L/example -
↪lexample
|  \_ /example/software/GCCcore/4.9.3/bin/gcc -Wl,-rpath=$ORIGIN/../lib -Wl,-rpath=
↪$ORIGIN/../lib64 -Wl,--disable-new-dtags -Wl,-rpath=/example -O2 example.c -L/
↪example -lexample
```

Here, `/tmp/eb-M3393U/tmpRVJqwr/rpath_wrappers/gcc` is the wrapper script for `gcc`, which tweaks the list of command line arguments for `gcc` before calling out to the real `gcc` command (i.e., `/example/software/GCCcore/4.9.3/bin/gcc` in this example).

RPATH wrapper script log files

When EasyBuild is used in debug mode (`--debug`), the RPATH wrapper script will generate log files in the temporary directory used by EasyBuild, for debugging purposes:

```
$ ls -l /tmp/eb-_hoff5/rpath_wrapper*log | sed 's/vsc40023/example/g'
-rw-rw-r-- 1 example example  739692 Nov 16 15:50 /tmp/eb-_hoff5/rpath_wrapper_gcc.log
-rw-rw-r-- 1 example example   27814 Nov 16 15:50 /tmp/eb-_hoff5/rpath_wrapper_g++.log
-rw-rw-r-- 1 example example 1589626 Nov 16 15:50 /tmp/eb-_hoff5/rpath_wrapper_ld.
↪gold.log
-rw-rw-r-- 1 example example    8870 Nov 16 15:50 /tmp/eb-_hoff5/rpath_wrapper_ld.log
```

These log files include details on every captured compiler/linker command, i.e. the original list of arguments, the tweaked list of arguments that includes the injected `-rpath` arguments, etc., and may be helpful to debug the RPATH support.

Overhead of RPATH wrapper scripts

Wrapping each compiler and linker command being executed comes at a cost, especially since the wrapper (shell) script calls out to a Python script (`rpath_args.py`) to do the heavy lifting.

Some early benchmarking has shown that this overhead is quite limited however, with observed slowdowns of the build and installation procedure of 10-15%.

6.21.5 Filtering RPATH entries via `--rpath-filter`

To avoid that the wrapper scripts inject RPATH entries for particular locations, EasyBuild can be configured with an RPATH filter via `--rpath-filter`.

The specified value should be a comma-separated list of (Python) regular expressions for paths. Only paths that *match* either of the specified patterns will be filtered out.

For example, to filter out locations in either `/opt/lib` or `/apps/lib`, use:

```
eb --rpath-filter='/opt/lib.*,/apps/lib.*'
```

By default, no RPATH entries will be injected for system locations that start with either `/lib` (incl. `/lib64`) or `/usr` (which is equivalent with `--rpath-filter='/lib.*,/usr.*'`).

Note: If you are specifying `--rpath--filter`, the default filter is *overwritten*, so if you want to retain the filtering for system locations you should also include `/lib.*` and `/usr.*`.

For example, to also filter out paths starting with `/example`:

```
eb --rpath-filter='/lib.*,/usr.*,/example.*'
```

6.21.6 Relation to `$LD_LIBRARY_PATH`

As mentioned above (*Why RPATH?*), using RPATH avoids the need to update `$LD_LIBRARY_PATH` for every dependency.

However, there is a chicken-or-egg situation: even though a particular dependency itself can be built and installed using RPATH, it does not mean that software packages that require it *have* to built with RPATH...

Hence, EasyBuild does not automatically exclude `$LD_LIBRARY_PATH` update statements from the generated module files. You need to configure EasyBuild to do so, using the `---filter-env-vars` configuration option.

For example:

```
eb --rpath --filter-env-vars=LD_LIBRARY_PATH example.eb
```

To consistently configure EasyBuild to both use RPATH and not include `$LD_LIBRARY_PATH` update statements in generated module files, you can use either environment variables or a configuration file; see *Configuring EasyBuild*.

6.22 Submitting jobs using `--job`

Topics:

- *Quick introduction to –job*
- *Configuring –job*
- *Usage of –job*
- *Examples*

6.22.1 Quick introduction to `--job`

Using the `--job` command line option, you can instruct EasyBuild to submit jobs for the installations that should be performed, rather than performing the installations locally on the system you are on.

If dependency resolution is enabled using `--robot` (see also *Enabling dependency resolution, –robot / -r and –robot-paths*), EasyBuild will submit separate jobs and set dependencies between them to ensure they are run in the order dictated by the software dependency graph(s).

6.22.2 Configuring `--job`

Selecting the job backend (`--job-backend`)

The job backend to be used can be specified using the `--job-backend` EasyBuild configuration option.

Since EasyBuild 3.8.0, three backends are supported:

- `GC3Pie` *(default)* (supported since EasyBuild 2.2.0)
  - `GC3Pie` version 2.5.0 (or more recent) required (https://gc3pie.readthedocs.org)
  - works with different resource managers and job schedulers, including TORQUE/PBS, Slurm, etc.
  - **note**: requires that a GC3Pie configuration file is provided, see *Configuring the job backend (–job-backend-config)*
- `PbsPython`
  - `pbs_python` version 4.1.0 (or more recent) required (see https://oss.trac.surfsara.nl/pbs_python)
  - **note**: requires TORQUE resource manager (see http://www.adaptivecomputing.com/products/open-source/torque/)
- `Slurm` (supported since EasyBuild 3.8.0)
  - requires Slurm version 17.0 (or more recent), see https://slurm.schedmd.com/

Configuring the job backend (`--job-backend-config`)

To configure the job backend, the path to a configuration file must be specified via `--job-backend-config`.

- for `PbsPython` backend: *(irrelevant, no configuration file required)*
- for `GC3Pie` backend: see https://gc3pie.readthedocs.org/en/latest/users/configuration.html
  - example configuration files are available at *Example configurations for GC3Pie job backend*
- for `Slurm` backend: *(irrelevant, no configuration file required)*

Number of requested cores per job (`--job-cores`)

The number of cores that should be requested for each job that is submitted can be specified using `--job-cores` (default: *not specified*).

The mechanism for determining the number of cores to request in case `--job-cores` was *not* specified depends on which job backend is being used:

- if the `PbsPython` job backend is used, the (most common) number of available cores per workernode in the target resource is determined; this usually results in jobs requesting full workernodes (at least in terms of cores) by default

- if the `GC3Pie` or `Slurm` job backend is used, the requested number of cores is left unspecified, which results in falling back to the default mechanism used by GC3Pie/Slurm to pick a number of cores; most likely, this results in single-core jobs to be submitted by default

Job dependency type (`--job-deps-type`)

The type of dependency that is set by EasyBuild when submitting a job that depends on one or more other jobs can be specified via the `--job-deps-type` configuration setting:

- with `--job-deps-type=abort_on_error`, job dependencies will be set such that a job that depends on other jobs will be *aborted* if one of those jobs completes with an error
  - for both `PbsPython` and `Slurm`, this is equivalent with setting job dependencies using `afterok`
- with `--job-deps-type=always_run`, job dependencies will be set such that a job that depends on other jobs are *always run*, regardless of whether or not those jobs completed successfully
  - for both `PbsPython` and `Slurm`, this is equivalent with setting job dependencies using `afterany`

The default value for `--job-deps-type` depends on the job backend being used (see *Configuring the job backend (–job-backend-config)*):

- for the `GC3Pie` and `Slurm` backends, `--job-deps-type=abort_on_error` is the default;
- for the `PbsPython` backend, `--job-deps-type=always_run` is the default (because of historical reasons, and for the sake of backward compatibility)

Maximum walltime of jobs (`--job-max-walltime`)

An integer value specifying the maximum walltime for jobs (in hours) can be specified via `--job-max-walltime` (default: 24).

For easyconfigs for which a reference required walltime is available via the `build_stats` parameter in a matching easyconfig file from the easyconfig repository (see *Easyconfigs repository (–repository, –repositorypath)*), EasyBuild will set the walltime of the corresponding job to twice that value (unless the resulting value is higher than the maximum walltime for jobs).

If no such reference walltime is available, the maximum walltime is used.

Job output directory (`--job-output-dir`)

The directory where job log files should be placed can be specified via `--job-output-dir` (default: current directory).

Job polling interval (`--job-polling-interval`)

The frequency with which the status of submitted jobs should be checked can be specified via `--job-polling-interval`, using a floating-point value representing the number of seconds between two checks (default: 30 seconds).

Note: This setting is currently only relevant to GC3Pie; see also *Submitting jobs to a GC3Pie backend*.

Target resource for job backend (`--job-target-resource`)

The target resource that should be used by the job backend can be specified using `--job-target-resource`.

- for `PbsPython` backend: hostname of TORQUE PBS server to submit jobs to (default: `$PBS_DEFAULT`)
- for `GC3Pie` backend: name of resource to submit jobs to (default: none, which implies weighted round-robin submission across all available resources)
- for `Slurm` backend: *(not used)*

6.22.3 Usage of `--job`

To make EasyBuild submit jobs to the job backend rather than performing the installations directly, the `--job` command line option can be used.

This following assumes that the required configuration settings w.r.t. the job backend to use are in place, see *Configuring –job*.

Submitting jobs to a `PbsPython` or `Slurm` backend

When using the `PbsPython` or `Slurm` backend, EasyBuild will submit separate jobs for each installation to be performed, and then exit reporting a list of submitted jobs.

To ensure that the installations are performed in the order dictated by the software dependency graph, dependencies between installations are specified via *job dependencies* (see also *Job dependency type (-job-deps-type)*).

See also *Example: submitting installations to TORQUE via pbs_python*.

Note: Submitted jobs will be put on hold until all jobs have been submitted. This is required to ensure that the dependencies between jobs can be specified correctly; if a job would run to completion before other jobs that depend on it were submitted, the submission process may fail.

Submitting jobs to a `GC3Pie` backend

When using the `GC3Pie` backend, EasyBuild will create separate tasks for each installation to be performed and supply them to GC3Pie, which will then take over and pass the installations through as jobs to the available resource(s) (see also *Configuring the job backend (–job-backend-config)*).

To ensure that the installations are performed in the order dictated by the software dependency graph, dependencies between installations are specified to GC3Pie as inter-task dependencies. GC3Pie will then gradually feed the installations to its available resources as their dependencies have been satisfied.

Any log messages produced by GC3Pie are included in the EasyBuild log file, and are tagged with `gc3pie`.

See also *Example: submitting installations to SLURM via GC3Pie*.

Note: The `eb` process will not exit until the full set of tasks that GC3Pie was provided with has been processed. An overall progress report will be printed regularly (see also *Job polling interval (–job-polling-interval)*). As such, it is advised to run the `eb` process in a screen/tmux session when using the GC3Pie backend for `--job`.

6.22.4 Examples

Example configurations for GC3Pie job backend

When using GC3Pie as a job backend, a configuration file must be provided via `--job-backend-config`. This section includes a couple of examples of GC3Pie configuration files (see also https://gc3pie.readthedocs.org/en/latest/users/configuration.html).

Example GC3Pie configuration for local system

```
[resource/localhost]
enabled = yes
type = shellcmd
frontend = localhost
transport = local
max_memory_per_core = 10GiB
max_walltime = 100 hours
# max # jobs ~= max_cores / max_cores_per_job
max_cores_per_job = 1
max_cores = 4
architecture = x86_64
auth = none
override = no
resourcedir = /tmp/gc3pie
```

Example GC3Pie configuration for TORQUE/PBS

```
[resource/pbs]
enabled = yes
type = pbs

# use settings below when running GC3Pie on the cluster front-end node
frontend = localhost
transport = local
auth = none

max_walltime = 2 days
# max # jobs ~= max_cores / max_cores_per_job
max_cores_per_job = 16
max_cores = 1024
max_memory_per_core = 2 GiB
architecture = x86_64

# to add non-std options or use TORQUE/PBS tools located outside of
# the default PATH, use the following:
#qsub = /usr/local/bin/qsub -q my-special-queue
```

Example GC3Pie configuration for SLURM

```
[resource/slurm]
enabled = yes
```

(continues on next page)

(continued from previous page)

```
type = slurm

# use settings below when running GC3Pie on the cluster front-end node
frontend = localhost
transport = local
auth = none

max_walltime = 2 days
# max # jobs ~= max_cores / max_cores_per_job
max_cores_per_job = 16
max_cores = 1024
max_memory_per_core = 2 GiB
architecture = x86_64

# to add non-std options or use SLURM tools located outside of
# the default PATH, use the following:
#sbatch = /usr/bin/sbatch --mail-type=ALL
```

Example: submitting installations to SLURM via GC3Pie

When submitting jobs to the `GC3Pie` job backend, the `eb` process will not exit until all tasks have been completed. A job overview will be printed every N seconds (see *Job polling interval (–job-polling-interval)*).

Jobs are only submitted to the resource manager (SLURM, in this case) when all task dependencies have been resolved.

```
$ export EASYBUILD_JOB_BACKEND=GC3Pie
$ export EASYBUILD_JOB_BACKEND_CONFIG=$PWD/gc3pie.cfg
$ eb GCC-4.6.0.eb OpenMPI-1.8.4-GCC-4.9.2.eb --robot --job --job-cores=16 --job-max-
↪walltime=10
== temporary log file in case of crash /tmp/eb-ivAiwD/easybuild-PCgmCB.log
== resolving dependencies ...
== GC3Pie job overview: 2 submitted (total: 9)
== GC3Pie job overview: 2 running (total: 9)
== GC3Pie job overview: 2 running (total: 9)
...
== GC3Pie job overview: 4 terminated, 4 ok, 1 submitted (total: 9)
== GC3Pie job overview: 4 terminated, 4 ok, 1 running (total: 9)
...
== GC3Pie job overview: 8 terminated, 8 ok, 1 running (total: 9)
== GC3Pie job overview: 9 terminated, 9 ok (total: 9)
== GC3Pie job overview: 9 terminated, 9 ok (total: 9)
== Done processing jobs
== GC3Pie job overview: 9 terminated, 9 ok (total: 9)
== Submitted parallel build jobs, exiting now
== temporary log file(s) /tmp/eb-ivAiwD/easybuild-PCgmCB.log* have been removed.
== temporary directory /tmp/eb-ivAiwD has been removed.
```

Checking which jobs have been submitted to SLURM at regular intervals reveals that indeed only tasks for which all dependencies have been processed are actually submitted as jobs:

```
$ squeue -u $USER
JOBID       USER       ACCOUNT           NAME     REASON    START_TIME     END_TIME ␣
↪TIME_LEFT NODES CPUS   PRIORITY
6161545     easybuild  example      GCC-4.9.2       None 2015-07-01T1 2015-07-01T2   ␣
↪9:58:55     1 16       1242
```

(continues on next page)

(continued from previous page)

```
6161546     easybuild  example      GCC-4.6.0       None 2015-07-01T1 2015-07-01T2
↪9:58:55     1 16       1242

$ squeue -u $USER
JOBID       USER       ACCOUNT           NAME     REASON    START_TIME     END_TIME
↪TIME_LEFT NODES CPUS   PRIORITY
6174527     easybuild  example Automake-1.15-  Resources           N/A          N/A
↪10:00:00     1 16       1120

$ squeue -u $USER
JOBID       USER       ACCOUNT           NAME     REASON    START_TIME     END_TIME
↪TIME_LEFT NODES CPUS   PRIORITY
6174533     easybuild  example OpenMPI-1.8.4-       None 2015-07-03T0 2015-07-03T1
↪9:55:59     1 16       1119
```

Example: submitting installations to TORQUE via pbs_python

Using the `PbsPython` job backend, `eb` submits jobs directly to TORQUE for processing, and exits as soon as all jobs have been submitted:

```
$ eb GCC-4.6.0.eb OpenMPI-1.8.4-GCC-4.9.2.eb --robot --job
== temporary log file in case of crash /tmp/eb-OMNQAV/easybuild-9fTuJA.log
== resolving dependencies ...
== List of submitted jobs (9): GCC-4.6.0 (GCC/4.6.0): 508023.example.pbs; GCC-4.9.2
↪(GCC/4.9.2): 508024.example.pbs;
libtool-2.4.2-GCC-4.9.2 (libtool/2.4.2-GCC-4.9.2): 508025.example.pbs; M4-1.4.17-GCC-
↪4.9.2 (M4/1.4.17-GCC-4.9.2): 50
8026.example.pbs; Autoconf-2.69-GCC-4.9.2 (Autoconf/2.69-GCC-4.9.2): 508027.example.
↪pbs; Automake-1.15-GCC-4.9.2 (Au
tomake/1.15-GCC-4.9.2): 508028.example.pbs; numactl-2.0.10-GCC-4.9.2 (numactl/2.0.10-
↪GCC-4.9.2): 508029.example.pbs;
hwloc-1.10.0-GCC-4.9.2 (hwloc/1.10.0-GCC-4.9.2): 508030.example.pbs; OpenMPI-1.8.4-
↪GCC-4.9.2 (OpenMPI/1.8.4-GCC-4.9.
2): 508031.example.pbs
== Submitted parallel build jobs, exiting now
== temporary log file(s) /tmp/eb-OMNQAV/easybuild-9fTuJA.log* have been removed.
== temporary directory /tmp/eb-OMNQAV has been removed.

$ qstat -a

example.pbs:
                                                                                Req'd
↪ Req'd       Elap
Job ID               Username    Queue    Jobname          SessID  NDS   TSK   Memory
↪ Time    S   Time
------------------- ----------- -------- ---------------- ------ ----- ------ ------ -
↪-------- - ---------
508023.example.pbs  easybuild   batch    GCC-4.6.0          --      1     16    --
↪24:00:00 R  00:02:16
508024.example.pbs  easybuild   batch    GCC-4.9.2          --      1     16    --
↪24:00:00 Q       --
508025.example.pbs  easybuild   batch    libtool-2.4.2-GC   --      1     16    --
↪24:00:00 H       --
508026.example.pbs  easybuild   batch    M4-1.4.17-GCC-4.   --      1     16    --
↪24:00:00 H       --
```

(continues on next page)

(continued from previous page)

```
508027.example.pbs  easybuild   batch    Autoconf-2.69-GC     --      1      16     --  ␣
↪24:00:00 H       --
508028.example.pbs  easybuild   batch    Automake-1.15-GC     --      1      16     --  ␣
↪24:00:00 H       --
508029.example.pbs  easybuild   batch    numactl-2.0.10-G     --      1      16     --  ␣
↪24:00:00 H       --
508030.example.pbs  easybuild   batch    hwloc-1.10.0-GCC     --      1      16     --  ␣
↪24:00:00 H       --
508031.example.pbs  easybuild   batch    OpenMPI-1.8.4-GC     --      1      16     --  ␣
↪24:00:00 H       --
```

Holds are put in place to ensure that the jobs run in the order dictated by the dependency graph(s). These holds are released by the TORQUE server as soon as they jobs on which they depend have completed:

```
$ qstat -a

example.pbs:
                                                                                  Req'd  ␣
↪ Req'd       Elap
Job ID               Username    Queue    Jobname          SessID  NDS   TSK   Memory ␣
↪ Time    S   Time
-------------------- ----------- -------- ---------------- ------ ----- ------ ------ -
↪-------- - ---------
508025.example.pbs  easybuild   batch    libtool-2.4.2-GC     --      1      16     --  ␣
↪24:00:00 Q       --
508026.example.pbs  easybuild   batch    M4-1.4.17-GCC-4.     --      1      16     --  ␣
↪24:00:00 Q       --
508027.example.pbs  easybuild   batch    Autoconf-2.69-GC     --      1      16     --  ␣
↪24:00:00 H       --
508028.example.pbs  easybuild   batch    Automake-1.15-GC     --      1      16     --  ␣
↪24:00:00 H       --
508029.example.pbs  easybuild   batch    numactl-2.0.10-G     --      1      16     --  ␣
↪24:00:00 H       --
508030.example.pbs  easybuild   batch    hwloc-1.10.0-GCC     --      1      16     --  ␣
↪24:00:00 H       --
508031.example.pbs  easybuild   batch    OpenMPI-1.8.4-GC     --      1      16     --  ␣
↪24:00:00 H       --

...

$ qstat -a

example.pbs:
                                                                                  Req'd  ␣
↪ Req'd       Elap
Job ID               Username    Queue    Jobname          SessID  NDS   TSK   Memory ␣
↪ Time    S   Time
-------------------- ----------- -------- ---------------- ------ ----- ------ ------ -
↪-------- - ---------
508031.example.pbs  easybuild   batch    OpenMPI-1.8.4-GC     --      1      16     --  ␣
↪24:00:00 R  00:03:46
```

6.23 Tracing progress

To trace the progress of EasyBuild while it is installing software, you can use `eb --trace`.

Note: Tracing support was added as an experimental feature in EasyBuild v3.4.0, and thus required using `--experimental`. Since EasyBuild v3.4.1, `--trace` is considered stable and no longer requires the use of `--experimental`.

Contents

6.23.1 Trace output

When `eb --trace` is used, EasyBuild will print additional output on top of the standard output, which only mentions which step of the installation procedure is being executed (without any further details).

This output includes:

- location of build and install directories
- list of sources and patches
- modules being loaded
- executed commands
- results of the sanity check
- location of generated module file

For each (non-trivial) executed command, the location to a temporary log file will be provided so the output of that command can be monitored while it is running. In addition, the start time of the command is printed, to allow determining how long the command has been running.

6.23.2 Example

```
$ eb HDF5-1.10.1-intel-2017a.eb -df --trace --experimental
== temporary log file in case of crash /tmp/eb-ieEeg3/easybuild-Ouw3jV.log
== processing EasyBuild easyconfig /home/example/HDF5/HDF5-1.10.1-intel-2017a.eb
== building and installing HDF5/1.10.1-intel-2017a...
  >> installation prefix: /prefix/software/HDF5/1.10.1-intel-2017a
== fetching files...
  >> sources:
  >> /prefix/sources/h/HDF5/hdf5-1.10.1.tar.gz [SHA256:␣
↪048a9d149fb99aaa1680a712963f5a78e9c43b588d0e79d55e06760ec377c172]
== creating build dir, resetting environment...
  >> build dir: /tmp/HDF5/1.10.1/intel-2017a
== unpacking...
  >> running command 'tar xzf /prefix/sources/h/HDF5/hdf5-1.10.1.tar.gz' (output in /
↪tmp/eb-ieEeg3/easybuild-run_cmd-P9kf6c.log) [started at: 2017-09-06 08:28:42]
== patching...
== preparing...
  >> loading toolchain module: intel/2017a
```

(continues on next page)

(continued from previous page)

```
  >> (no build dependencies specified)
  >> loading modules for (runtime) dependencies:
  >>  * zlib/1.2.11-GCCcore-6.3.0
  >>  * Szip/2.1-intel-2017a
  >> defining build environment for intel/2017a toolchain
== configuring...
  >> running command './configure --prefix=/prefix/software/HDF5/1.10.1-intel-2017a  -
↪-with-szlib=/prefix/software/Szip/2.1-intel-2017a  --with-zlib=/prefix/software/
↪zlib/1.2.11-GCCcore-6.3.0  --with-pic --with-pthread --enable-shared  --enable-cxx -
↪-enable-fortran FC="mpiifort"  --enable-unsupported --enable-parallel' (output in /
↪tmp/eb-ieEeg3/easybuild-run_cmd-dPat3D.log) [started at: 2017-09-06 08:28:44]
== building...
  >> running command 'make -j 24  CXXFLAGS="$CXXFLAGS -DMPICH_IGNORE_CXX_SEEK"  FC=
↪"mpiifort"' (output in /tmp/eb-ieEeg3/easybuild-run_cmd-25vKdK.log) [started at:␣
↪2017-09-06 08:31:01]
== testing...
== installing...
  >> running command 'make install' (output in /tmp/eb-ieEeg3/easybuild-run_cmd-
↪BepE8P.log) [started at: 2017-09-06 08:34:09]
== taking care of extensions...
== postprocessing...
== sanity checking...
  >> file 'bin/h52gif' found: OK
  >> file 'bin/h5c++' found: OK
  >> file 'bin/h5copy' found: OK
  >> file 'bin/h5debug' found: OK
  >> file 'bin/h5diff' found: OK
  >> file 'bin/h5dump' found: OK
  >> file 'bin/h5import' found: OK
  >> file 'bin/h5jam' found: OK
  >> file 'bin/h5ls' found: OK
  >> file 'bin/h5mkgrp' found: OK
  >> file 'bin/h5perf_serial' found: OK
  >> file 'bin/h5redeploy' found: OK
  >> file 'bin/h5repack' found: OK
  >> file 'bin/h5repart' found: OK
  >> file 'bin/h5stat' found: OK
  >> file 'bin/h5unjam' found: OK
  >> file 'bin/gif2h5' found: OK
  >> file 'bin/h5perf' found: OK
  >> file 'bin/h5pcc' found: OK
  >> file 'bin/h5pfc' found: OK
  >> file 'bin/ph5diff' found: OK
  >> file 'lib/libhdf5.so' found: OK
  >> file 'lib/libhdf5_cpp.so' found: OK
  >> file 'lib/libhdf5_fortran.so' found: OK
  >> file 'lib/libhdf5_hl_cpp.so' found: OK
  >> file 'lib/libhdf5_hl.so' found: OK
  >> file 'lib/libhdf5hl_fortran.so' found: OK
  >> (non-empty) directory 'include' found: OK
== cleaning up...
== creating module...
  >> generating module file @ /prefix/modules/all/HDF5/1.10.1-intel-2017a.lua
== permissions...
== packaging...
== COMPLETED: Installation ended successfully
== Results of the build can be found in the log file(s) /prefix/software/HDF5/1.10.1-
↪intel-2017a/easybuild/easybuild-HDF5-1.10.1-20170906.083425.log
```

(continues on next page)

(continued from previous page)

```
== Build succeeded for 1 out of 1
== Temporary log file(s) /tmp/eb-ieEeg3/easybuild-Ouw3jV.log* have been removed.
== Temporary directory /tmp/eb-ieEeg3 has been removed.
```

6.24 Using external modules

Since EasyBuild v2.1, support for using modules that were not installed via EasyBuild is available. We refer to such modules as *external modules*.

This can be useful to reuse vendor-supplied modules, for example on Cray systems.

Contents

6.24.1 Using external modules as dependencies

External modules can be used as dependencies, by including the module name in the `dependencies` list (see *Dependencies*), along with the `EXTERNAL_MODULE` constant marker.

For example, to specify the readily available module `fftw/3.3.4.2` as a dependency:

```
dependencies = [('fftw/3.3.4.2', EXTERNAL_MODULE)]
```

For such dependencies, EasyBuild will:

- load the module before initiating the software build and install procedure
- include a '`module load`' statement in the generated module file (for non-build dependencies)

If the specified module is not available, EasyBuild will exit with an error message stating that the dependency can not be resolved because the module could not be found. It will *not* search for a matching easyconfig file in order to try and install the module to resolve the dependency.

6.24.2 Metadata for external modules

Since very little information is available for external modules based on the dependency specification alone (i.e., only the module name), metadata can be supplied to EasyBuild for external modules.

Using the `--external-modules-metadata` configuration option, the location of one or more files can be specified that provide such metadata. The files are expected to be in INI format, with a section per module name and key-value assignments denoting the metadata specific to that module.

Format:

```
[modulename]
key1 = value1
key2 = value2, value3
```

Default

Up until EasyBuild v2.6.0, no metadata for external modules was available by default.

Since EasyBuild v2.7.0, a file providing metadata for Cray-provided modules on Cray systems is included, and is active by default (i.e., unless other files are specified via `--external-modules-metadata`); see https://github.com/easybuilders/easybuild-framework/blob/main/etc/cray_external_modules_metadata.cfg.

Using wildcards

Since EasyBuild v4.1.0, you can use so-called *glob patterns* to specify a list of paths to `--external-modules-metadata`, using wildcard characters like `*` and `?`.

For example, to specify that the metadata for external modules in all `*.cfg` files in the directory `/example` should be taken into account, you can use:

```
export EASYBUILD_EXTERNAL_MODULES_METADATA='/example/*.cfg'
```

Example

For example, the following file provides metadata for a handful of modules that may be provided on Cray systems:

```
[cray-hdf5/1.8.13]
name = HDF5
version = 1.8.13
prefix = HDF5_DIR

[cray-hdf5-parallel/1.8.13]
name = HDF5
version = 1.8.13
prefix = /opt/cray/hdf5-parallel/1.8.13/GNU/49

[cray-netcdf/4.3.2]
name = netCDF, netCDF-Fortran
version = 4.3.2, 4.3.2
prefix = NETCDF_DIR

[fftw/3.3.4.5]
name = FFTW
version = 3.3.4.5
prefix = FFTW_INC/..
```

Supported metadata values

The following keys are supported:

- `name`: specifies the software name(s) that is (are) provided by the module
- `version`: specifies the software version(s) that is (are) provided by the module
- `prefix`: specifies the installation prefix of the software that is provided by the module

For `prefix`, a couple of different options are available, i.e.:

- specifying an absolute path (cfr. `cray-hdf5-parallel/1.8.13` in the example above)
- specifying the environment variable that is defined by the module and provides the installation prefix (cfr. `cray-netcdf/4.3.2` in the example above)
  - this can be optionally combined with a relative path that serves as a 'correction' (cfr. `fftw/3.3.4.5` in the example above) *[supported since EasyBuild v2.7.0]*

Any other keys are simply ignored.

Note: When both `name` and `version` are specified, they must provide an *equal number of values* (see for example the `cray-netcdf` example above).

Handling of provided metadata

Using the provided metadata, EasyBuild will define environment variables that are also defined by modules that are generated by EasyBuild itself, if an external module for which metadata is available is loaded as a dependency.

In particular, for each software name that is specified:

- the corresponding environment variable `$EBROOT<NAME>` is defined to the specified `prefix` value (if any)
- the corresponding environment variable `$EBVERSION<NAME>` is defined to the corresponding `version` value (if any)

For example, for the external modules for which metadata is provided in the example above, the following environment variables are set in the build environment when the module is used as a dependency:

- for `cray-hdf5/1.8.1.13`:
  - `$EBROOTHDF5` = `$HDF5_DIR`
  - `$EBVERSIONHDF5` = `1.8.13`
- for `cray-hdf5-parallel/1.8.13`:
  - `$EBROOTHDF5` = `/opt/cray/hdf5-parallel/1.8.13/GNU/49`
  - `$EBVERSIONHDF5` = `1.8.13`
- for `cray-netcdf/4.3.2`:
  - `$EBROOTNETCDF` = `$NETCDF_DIR`
  - `$EBROOTNETCDFMINFORTRAN` = `$NETCDF_DIR`
  - `$EBVERSIONNETCDF` = `4.3.2`
  - `$EBVERSIONNETCDFMINFORTRAN` = `4.3.2`
- for `fftw/3.3.4.5`:

- $EBROOTFFTW = $FFTW_INC/../
- $EBVERSIONFFTW = 3.3.4.5

The get_software_root and get_software_version functions that are commonly used occasionally in easyblocks pick up the $EBROOT* and $EBVERSION* environment variables, respectively.

6.25 Wrapping dependencies

Since EasyBuild v3.7.0 a special-purpose generic easyblock named ModuleRC is available, which can be used to generate a (software-specific) .modulerc file (as opposed to generating an actual module file).

Note: For compatibility with the different modules tools supported by EasyBuild (see *Modules tool (–modules-tool)*), the .modulerc file is always generated in Tcl syntax (for now), regardless of the module syntax that is used for module files. Only Lmod 7.8 (or later) supports .modulerc.lua files in Lua syntax.

The ModuleRC easyblock supports including module-version in the generated .modulerc file, which defines a so-called *"symbolic version"*.

This can be used to install a *"wrapper"* for a particular module, which can be useful in the context of dependencies. For example, it can be used to avoid depending on a specific version of a particular software package by specifying the dependency only on the <major>.<minor> version instead (i.e., without including the "subminor" version, which usually corresponds to a bugfix release).

One particular example where this is employed is for Java. Since the 2018b generation of the *Common toolchains*, we use a wrapper for Java (e.g., Java/1.8), rather than depending on a specific version (e.g., Java/1.8.0_181):

```
# specify dependency on Java/1.8 "wrapper", rather than a specific Java version
dependencies = [('Java', '1.8', '', True)]
```

This has a couple of advantages:

- it helps with avoiding version conflicts on the Java dependency used by two (or more) otherwise independent module files;
- it allows to perform an in-place update of the Java wrapper to a more recent Java (bugfix) release in the blink of an eye, since it only involves re-generating a .modulerc file (as opposed to performing actual installation and generating a module file); in addition, it doesn't require updating/re-generating existing module files that depend on the Java wrapper (since the version of the wrapper does not change)

Since ModuleRC is a generic easyblock , it can also be employed to install module "wrappers" for dependencies other than Java.

CHAPTER 7

Other topics

7.1 Code style

The code style we follow in the EasyBuild code repository is mainly dictated by the Python standard PEP8.

Highlighted PEP8 code style rules:

- **use 4 spaces for indentation, do not use tabs** for example, use `:set tabstop 4` and `:set expandtab` in Vim
- **indent items in a list at an extra 4 spaces** nested lists can be indented at the same indentation as the first item in the list if it is on the first line, closing brackets must match visual indentation
- use *optional underscores*, not camelCase, for variable, function and method names (i.e. `poll.get_unique_voters()`, **not** `poll.getUniqueVoters`)
- use `InitialCaps` for class names
- in docstrings, don't use "action words"

The only (major) exception to PEP8 is our preference for longer line lengths: line lengths **must be limited to 120 characters**, and should by preference be *shorter than 100 characters* (as opposed to the 80-character limit in PEP8).

7.1.1 Notes

Code style in easyconfig files can be **automatically checked** using `--check-contrib`, for example: `eb --check-contrib sympy-1.3-intel-2018a-Python-2.7.14.eb` (see *Code style review* for more details).

Style guides that go a step beyond PEP8:

- http://www.gramps-project.org/wiki/index.php?title=Programming_guidelines
- http://code.google.com/p/volatility/wiki/StyleGuide

Automatic rewriting of Python code: http://pypi.python.org/pypi/PythonTidy/1.22

`pep8` might be a useful tool to check PEP8 compliance: https://github.com/jcrocholl/pep8

7.2 Unit tests

Since EasyBuild v1.0, an extensive suite of unit tests has been implemented. The unit tests have become an indispensable factor in keeping EasyBuild stable and backward-compatible during development.

We refer to the available tests as unit tests, even though they may not be *unit* tests in the strict sense of the word. Some tests are actually end-to-end tests or integration tests, see also http://en.wikipedia.org/wiki/Software_testing#Testing_levels.

Following the test-driven development paradigm, additional unit tests are implemented when new features are added or when bugs are uncovered (and fixed).

7.2.1 What the unit tests are *not*

Each of the EasyBuild unit tests aim to test a specific characteristic of EasyBuild, e.g., a configuration option, a particular function or method in the EasyBuild framework, some specific feature, how EasyBuild handles a particular type of input, etc.

The unit tests do *not* test the build and installation process of particular supported software packages (other than a handful of toy ones included in the tests themselves), let alone test the software installations obtained using EasyBuild themselves.

Each stable EasyBuild release is subjected to a (time- and resource-consuming) *regression test*, however, which is out of scope here.

7.2.2 Available unit test suites

Three different unit test suites are available for EasyBuild, each of which tied to one of the EasyBuild code repositories on GitHub: *EasyBuild framework*, *easyblocks*, *easyconfigs*.

EasyBuild framework unit tests

The unit test suite for the EasyBuild framework is by far the most extensive one, in terms of code size, coverage and the amount of effort that is put into it.

These tests probe the functionality provided by the EasyBuild framework, ranging from standard operation (building and installing software, and generating module files) to specific configuration options, selected functional aspects, optional features and edge cases.

At the time of writing (EasyBuild v2.0), more than 250 tests were implemented, organised in 28 subgroups.

Running the full EasyBuild framework unit test suite takes about 15-40 minutes, depending on your system resources and testing configuration (see for example https://jenkins1.ugent.be/view/EasyBuild/).

easyblocks unit tests

The easyblocks unit test suite consists of a coupe of light-weight tests that are run per easyblock:

- initialising the easyblock, to check for Python syntax errors and faulty imports

- checking for the use of deprecated (or no longer supported) functionality

Running the full easyblocks unit test suite takes less than one minute.

easyconfigs unit tests

The easyconfigs unit test suite consists a couple of tests for each of the available easyconfig files, followed by two large-scale overall tests.

For each of the available easyconfig files, the following aspects are tested:

- parsing the easyconfig file, to make sure no syntax errors are present
- verifying that the filename matches the contents of the easyconfig file (software name/version, version suffix and toolchain name/version)
- creating an `EasyBlock` instance using the parsed easyconfig, to check whether mandatory easyconfig parameters are defined
- ensuring that all required patch files are available (right next to the easyconfig file)
- making sure that the specified sanity check paths adhere to the requirements, i.e. only (and both) the `files`/`dirs` keys are listed, with the value for either one being non-empty
- checking for the use of deprecated (or no longer supported) functionality
- verifying whether `eb --check-style` on the easyconfig file passes
- ensuring SHA256 checksums are included in new or changed easyconfig files

If these tests pass for each and every available easyconfig file, two additional overall tests are run, i.e.:

- ensuring that an easyconfig file is available for each specified dependency
- checking whether any version conflicts occur in the dependency graph for each easyconfig file

Running the full easyconfigs unit test suite should only take a couple of minutes.

7.2.3 Applications

The unit test suites are automatically triggered by Jenkins for:

- each pull request (update), see https://jenkins1.ugent.be/view/pull-request-builder/
- each (set of) change(s) that is merged in (usually via a pull request)

The status of the different unit test suites is tracked separately for the `main` and `develop` branches of the EasyBuild code repositories:

- `main` (stable, latest EasyBuild release): https://jenkins1.ugent.be/view/EasyBuild/
- `develop` (development, potentially unstable): https://jenkins1.ugent.be/view/EasyBuild%20(develop)/

We strictly adher to the policy of only merging pull requests for which the corresponding (latest) run of the *full* unit test suite passes successfully.

7.2.4 Usage

Using the unit tests is deliberately kept very simple.

Configuration

Since EasyBuild v2.0, the unit test suites are fully isolated from any (system- or user-level) EasyBuild configuration which is in place in the environment where the tests are being run.

The easyblocks and easyconfigs unit test suite are oblivious to any defined configuration options.

For the EasyBuild framework unit tests, all configuration files and `EASYBUILD_`-prefixed environment variables are ignored (see also configuration_types:).

To enable running the EasyBuild framework unit test suite under a specific configuration that differs from the default, environment variables can be defined for each of the configuration options supported by EasyBuild.

Before starting a set of EasyBuild framework tests, all defined environment variables for which the name is prefixed by `TEST_EASYBUILD_` will be injected into the test environment as environment variables prefixed with `EASYBUILD_` instead. Thus, to set a particular configuration option `--foo`, you should define the environment variable `$TEST_EASYBUILD_FOO`.

Modules tool to use for running tests

One particular configuration option worth mentioning explicitly is the modules tool that is to be used by the EasyBuild framework, which is by default the traditional Tcl/C environment modules, referred to as `EnvironmentModulesC` in EasyBuild configuration terms (see `eb --help` and `eb --avail-modules-tools`).

To run the EasyBuild framework unit tests with a particular modules tool, simply define the `$TEST_EASYBUILD_MODULES_TOOL` environment variable with the corresponding value. For example:

```
export TEST_EASYBUILD_MODULES_TOOL=Lmod
```

Just like for EasyBuild itself, the EasyBuild framework unit test suite expects that the modules tool binary/script (`modulecmd`, `modulecmd.tcl` or `lmod`) is available via `$PATH`, see *Required modules tool*.

Installing a GitHub token for the unit tests

Some of the EasyBuild framework unit tests require that a GitHub token is in place for the `easybuild_test` user, in a non-encrypted keyring (so it can be obtained without having to provide a password).

This can be done as follows (copy-paste the GitHub token at the `Password:` prompt):

```
$ python
>>> import getpass, keyring
>>> keyring.set_keyring(keyring.backends.file.PlaintextKeyring())
>>> keyring.set_password('github_token', 'easybuild_test', getpass.getpass())
Password:
>>> exit()
```

More details about obtaining and installing a GitHub token in your keyring are available at https://github.com/easybuilders/easybuild/wiki/Review-process-for-contributions#setting-things-up.

Running

To run a full unit test suite, simply run the respective `suite` Python module.

- EasyBuild framework: `python -m test.framework.suite`
- easyblocks: `python -m test.easyblocks.suite`

- easyconfigs: `python -m test.easyconfigs.suite`

For the EasyBuild framework unit tests, each of the test subgroups can be run separately via a dedicated Python module other than `suite`, to focus on testing a particular aspect. See https://github.com/easybuilders/easybuild-framework/tree/main/test/framework for an overview of the available Python modules corresponding to subgroups of tests (note: `__init__.py` and `utilities.py` are *not* such modules).

For example, to run the full EasyBuild framework unit test suite using Lmod as a modules tool:

```
$ export TEST_EASYBUILD_MODULES_TOOL=Lmod
$ python -m test.framework.suite
```

To only run the subgroup of tests for `filetools`:

```
$ python -m test.framework.filetools
```

Since EasyBuild v2.8.2, tests can be *filtered* by name. Simply add the string to filter with to the test command.

For example, to run only the tests containing the word `load` in the subgroup `modules`, run:

```
$ python -m test.framework.modules load

Filtered ModulesTest tests using 'load', retained 2/19 tests: test_load, test_load_in_
↪hierarchy
..
----------------------------------------------------------------------
Ran 2 tests in 2.688s

OK
```

This works with as many filter words as you want to use. For example, to run every test method in `modules` containing the words `load` or `bash`:

```
$ python -m test.framework.modules load bash
```

Results

The test results will be printed as the unit test suite progresses, potentially producing a lot of information for failing tests to help determine the cause of the failure. It may thus be useful to capture the output for later inspection, for example:

```
python -m test.framework.suite 2>&1 | tee eb_test.log
```

Note: Some tests will be skipped deliberately, because of missing optional dependencies or other provisions, for example a GitHub token. The output of the unit tests will clearly indicate which tests were skipped.

Examples

A successful run of the EasyBuild framework test suite, without skipped tests:

```
$ python -m test.framework.suite
Running tests...
----------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
.....................................................................................
↪.....................................................................................
↪...................................................................................
----------------------------------------------------------------------
Ran 250 tests in 1404.897s

OK
```

A run with a couple of deliberately skipped tests and a single failed test (denoted by `F`), along with the corresponding traceback:

```
$ python -m test.framework.suite
Running tests...
----------------------------------------------------------------------
................Skipping test_from_pr, no GitHub token available?
.Skipping test_from_pr, no GitHub token available?
......F.............(skipping GitRepository test)
..(skipping SvnRepository test)
.....................................................................................
↪.........................Skipping test_fetch_easyconfigs_from_pr, no GitHub␣
↪token available?
.Skipping test_read, no GitHub token available?
.Skipping test_read_api, no GitHub token available?
.Skipping test_walk, no GitHub token available?
.....................................................................................
↪.......
======================================================================
FAIL: Test listing easyblock hierarchy.
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/tmp/example/easybuild-framework/test/framework/options.py", line 544, in␣
↪test_list_easyblocks
    self.assertTrue(re.search(pat, outtxt), "Pattern '%s' is found in output of --
↪list-easyblocks: %s" % (pat, outtxt))
AssertionError: Pattern 'EasyBlock\n' is found in output of --list-easyblocks:

----------------------------------------------------------------------
Ran 250 tests in 2641.200s

FAILED (failures=1)
ERROR: Not all tests were successful.
Log available at /tmp/example/easybuild-dy2ZTx/easybuild-tests-l0doQ2.log
```

7.3 Useful scripts

A couple of useful stand-alone scripts are provided along with the EasyBuild framework.

The latest stable version of these scripts is available in the `easybuild-framework` GitHub repository at https://github.com/easybuilders/easybuild-framework/tree/main/easybuild/scripts.

This documentation provides some information on how to use these scripts.

7.3.1 fix_broken_easyconfigs.py

download from: https://github.com/easybuilders/easybuild-framework/blob/main/easybuild/scripts/fix_broken_easyconfigs.py

To fix easyconfig files that are broken due to relying on functionality that is no longer supported in EasyBuild v2.x, the `fix_broken_easyconfigs.py` script can be used.

This script rewrites easyconfig files that are broken because they still:

- rely on the automagic fallback to the generic `ConfigureMake` easyblock (see *Automagic fallback to ConfigureMake*)
- define the `premakeopts` and/or `makeopts` eaysconfig parameters (see *Options for build command*)
- use the `shared_lib_ext` 'constant' (see *Shared library extension*)
- incorrectly use the `license` easyconfig parameter (see *Software license*)

The script accepts a list of easyconfig files or directories containing easyconfig files (`.eb`) as arguments, and will consider all easyconfig files it can find.

Only easyconfig files that are considered broken (according to one or more of the aspects listed above) are patched; other easyconfigs will be left untouched.

To determine whether `easyblock = 'ConfigureMake'` should be added in an easyconfig file that does not include any `easyblock` specification yet, the easyblocks available in the active Python search path (i.e., the ones listed in the output of `eb --list-easyblocks`, see also *List of available easyblocks, –list-easyblocks*) are taken into account.

A backup copy is created for each easyconfig file that is being patched.

Example usage:

```
$ python easybuild/scripts/fix_broken_easyconfigs.py broken.eb myeasyconfigs GCC-4.9.
↪2.eb
== 2015-03-05 17:02:22,438 fix_broken_easyconfigs.FIX_BROKEN_EASYCONFIGS INFO␣
↪Processing 3 easyconfigs
== 2015-03-05 17:02:22,454 fix_broken_easyconfigs.FIX_BROKEN_EASYCONFIGS INFO Backed␣
↪up broken.eb to broken.eb.bk
== 2015-03-05 17:02:22,454 fix_broken_easyconfigs.FIX_BROKEN_EASYCONFIGS INFO broken.
↪eb: fixed
== 2015-03-05 17:02:22,458 fix_broken_easyconfigs.FIX_BROKEN_EASYCONFIGS INFO /home/
↪example/myeasyconfigs/HPL-2.1-intel-2015a.eb: nothing to fix
== 2015-03-05 17:02:22,461 fix_broken_easyconfigs.FIX_BROKEN_EASYCONFIGS INFO GCC-4.9.
↪2.eb: nothing to fix
```

This results in `broken.eb` being rewritten/fixed, after creating a backup copy `broken.eb.bk`:

```
$ diff -u broken.eb.bk broken.eb
--- broken.eb.bk    2015-02-25 14:45:52.000000000 +0100
+++ broken.eb       2015-03-05 16:51:44.000000000 +0100
@@ -1,4 +1,6 @@
 # licenseheader
+easyblock = 'ConfigureMake'
+
 name = 'foo'
 version = '1.2.3'

@@ -7,7 +9,7 @@
```

(continues on next page)

(continued from previous page)

```
 toolchain = {'name': 'bar', 'version': '3.2.1'}

-premakeopts = 'FOO=libfoo.%s' % shared_lib_ext
-makeopts = 'CC=gcc'
+prebuildopts = 'FOO=libfoo.%s' % SHLIB_EXT
+buildopts = 'CC=gcc'

-license = 'foo.lic'
+license_file = 'foo.lic'
```

7.3.2 install-EasyBuild-develop.sh

download from: https://github.com/easybuilders/easybuild-framework/blob/main/easybuild/scripts/install-EasyBuild-develop.sh

A script to set up an EasyBuild development environment. For more information, see *Installation of latest development version using provided script*.

7.3.3 clean_gists.py

download from: https://github.com/easybuilders/easybuild-framework/blob/main/easybuild/scripts/clean_gists.py

When using `--from-pr` in combination with `--upload-test-report` (see https://github.com/easybuilders/easybuild/wiki/Review-process-for-contributions#automated-testing-of-easyconfigs-pull-requests), you can end up with a bunch of gists in your GitHub account containing test reports, that may no longer be relevant.

To help with that the `clean_gists.py` script is available, to clean up gists containing test reports:

- `clean_gists.py -p`: delete all gists from closed pull requests (default action if no other action is specified)
- `clean_gists.py -a`: delete all gists generated by Easybuild
- `clean_gists.py -o`: delete all gists without a matching pull request (created by using `--upload-test-report` without `--from-pr`)

By default, the script will use the same GitHub account that Easybuild uses (see `--github-user`); to specify a different GitHub account, use `-g`.

The script expects that a valid GitHub token for the used GitHub account username is available, see *Installing a GitHub token (–install-github-token)*.

7.4 Deprecated easyconfigs

Since EasyBuild v3.8.0, individual easyconfig files or particular (versions of) toolchains can be marked as deprecated.

Contents

7.4.1 Symptoms

Using an easyconfig file or toolchain that was marked as deprecated results in a warning message like:

```
WARNING: Deprecated functionality, will no longer work in v4.0: easyconfig file '/
↪home/example/test.eb' is marked as deprecated:
This is an example message explaining why the easyconfig file was deprecated.
(see also http://easybuild.readthedocs.org/en/latest/Deprecated-easyconfigs.html)
```

Or, in case you are trying to use an easyconfig file or toolchain that was marked deprecated in a previous major version of EasyBuild:

```
ERROR: Failed to process easyconfig /home/example/test.eb: DEPRECATED (since v4.0)␣
↪functionality used: easyconfig file '/home/example/test.eb' is marked as deprecated:
This is an example message explaining why the easyconfig file was deprecated.
(see also http://easybuild.readthedocs.org/en/latest/Deprecated-easyconfigs.html)
```

7.4.2 Reasons for deprecation

There are several possible reasons why a particular easyconfig file or toolchain was deprecated, some of which include:

- old/obsolete versions of software or toolchain components
- toolchains that are superseded by other toolchains
- known installation problems that are hard to resolve (or not worth the effort to resolve)

7.4.3 Implications

Easyconfig files that are deprecated or which use a deprecated toolchain are *not actively maintained*, and are *no longer included in the EasyBuild regression test* (which means they may be broken by recent changes to the EasyBuild framework or relevant easyblocks).

In a future major version of EasyBuild, these easyconfig files will be archived (see also *Archived easyconfigs*).

7.4.4 Deprecated toolchains

Overview of deprecated toolchains:

- *foss toolchain*
- *gompi toolchain*
- *goolf and goolfc toolchains*

- *ictce toolchain*
- *Oldest versions of the iccifort, iimpi and intel toolchains*

`foss` toolchain

- *deprecated since:* EasyBuild v3.9.4
- *will be archived in:* EasyBuild v4.0.0

The oldest versions of the `foss` toolchain have been deprecated, which currently includes any version older than `foss/2016a`.

`gompi` toolchain

- *deprecated since:* EasyBuild v3.8.0
- *will be archived in:* EasyBuild v4.0.0

Versions of the `gompi` toolchain that were used as a subtoolchain for a deprecated toolchain have also been deprecated; this includes `gompi` toolchain versions that match `1.*`, and any version older than `gompi/2016a` (the latter only since EasyBuild v3.9.4).

`goolf` and `goolfc` toolchains

- *deprecated since:* EasyBuild v3.8.0
- *will be archived in:* EasyBuild v4.0.0

The `goolf` and `goolfc` toolchains have been deprecated, since they are superseded by the *foss toolchain* and `fosscuda` toolchains, respectively.

The `foss*` toolchains are equivalent to the `goolf*` toolchains, except that `binutils` is also included as a companion to `GCC(core)` in the `foss*` toolchains.

`ictce` toolchain

- *deprecated since:* EasyBuild v3.8.0
- *will be archived in:* EasyBuild v4.0.0

The `ictce` toolchain has been deprecated, since it is superseded by the *intel toolchain*.

The `ictce` toolchain is equivalent to `intel` w.r.t. toolchain components, except that `binutils` is also included as a companion to `GCC(core)` (which serves as a base for the Intel compilers) in the `intel` toolchain.

Oldest versions of the `iccifort`, `iimpi` and `intel` toolchains

- *deprecated since:* EasyBuild v3.8.0
- *will be archived in:* EasyBuild v4.0.0

The oldest versions of the `iccifort`, `iimpi` and *intel toolchain* have been deprecated, since they are no longer considered relevant for recent systems.

More specifically, deprecated versions include:

- `iccifort` versions older than `2016.1.150`

- `iimpi` versions older than `2016.01`, except version `8.1.5-*`
- `intel` versions older than `2016a`

7.5 Deprecated functionality

Some of the functionality that was available in previous EasyBuild versions is *deprecated*, and should no longer be used.

This functionality will be removed in a future version of EasyBuild (see *Removed functionality* for more information about functionality that has been removed already).

This page:

- provides an *overview of currently deprecated functionality* together with available alternatives
- describes the *EasyBuild deprecation policy*
- explains how to easily *check whether you are still relying on deprecated functionality*

7.5.1 Overview of deprecated functionality in EasyBuild version 4.4.2

The different section below document the functionality that is deprecated in EasyBuild version 4.4.2, for which support will be removed in EasyBuild version 5.0.

For EasyBuild users:

- *Support for using Lmod 6.x*

For authors of easyconfig files:

- *dummy toolchain*

For developers of easyblocks:

(nothing yet)

For EasyBuild framework developers:

(nothing yet)

Support for using Lmod 6.x

- *deprecated since:* EasyBuild v4.1.0 (Nov'19)
- *removed in:* EasyBuild v5.0
- *alternative(s)*: **use Lmod 7.x or more recent**

Support for using Lmod 6.x as modules tool with EasyBuild has been deprecated, and will be removed in a future version of EasyBuild.

`dummy` toolchain

- *deprecated since:* EasyBuild v4.0.0 (Sept'19)
- *removed in:* EasyBuild v5.0
- *alternative(s)*: **use** `system` **toolchain instead**

The `dummy` toolchain is has been deprecated, and is replaced with the `system` toolchain.

For more information, see system_toolchain.

7.5.2 Deprecation policy

With every EasyBuild release, we try very hard to maintain *backward compatibility*. That is, EasyBuild version `X.Y` should still build software packages that could be built with EasyBuild version `X.(Y-1)`, without requiring modifications to the used easyconfig file or easyblock.

However, every now and then a breaking change needs to be made to EasyBuild, because of design decisions or to resolve mistakes that were made in the past. These changes involve *deprecating* the behaviour or functionality we want to get rid of, together with supporting a better alternative.

Deprecating functionaliy:

- using a `log.deprecated("msg", 'X.Y')` statement in EasyBuild version `X.(Y-n)` a certain block of code is tagged as *deprecated*, indicating that the corresponding functionality will no longer be supported in EasyBuild version `X.Y`; for example, to deprecate the use of the `makeopts` easyconfig parameter with EasyBuild v2.0:

```
if self.cfg['makeopts']:
    self.log.deprecated("Easyconfig parameter 'makeopts' is deprecated, use
↪'buildopts' instead", '2.0')
```

- until EasyBuild version `X.Y`, the deprecation log message will manifest itself as a *warning*, highlighting the use of deprecated functionality; for example:

```
== 2014-12-16 16:29:07,906 main.easyconfig.easyconfig WARNING Deprecated␣
↪functionality, will no longer work in v2.0:
Easyconfig parameter 'makeopts' is deprecated, use 'buildopts' instead;
see http://easybuild.readthedocs.org/en/latest/Deprecated-functionality.html for␣
↪more information
```

Removing support for deprecated behavior:

- starting with EasyBuild version `X.Y`, the deprecation log message will result in an *error*, indicating that the deprecated behavior is no longer supported; for example:

```
ERROR: EasyBuild encountered an exception (at easybuild/framework/easyconfig/
↪easyconfig.py:937 in process_easyconfig):
Failed to process easyconfig /home/example/gzip-1.5-goolf-1.4.10.eb:
DEPRECATED (since v2.0) functionality used: Easyconfig parameter 'makeopts' is␣
↪deprecated, use 'buildopts' instead;
see http://easybuild.readthedocs.org/en/latest/Deprecated-functionality.html for␣
↪more information.
```

- the code supporting the deprecated functionality is *removed* in EasyBuild version `X.(Y+1)` (i.e., the first non-bugfix-only release after version `X.Y`), see also *Removed functionality*
- until EasyBuild version `X.(Y+1)`, the code supporting the deprecated functionality will still be available; using the `--deprecated` command line option (or, equivalently, the `$EASYBUILD_DEPRECATED` environment variable), the deprecated functionality can be reactivated by specifying a *lower* version; for example, to avoid running into an error with EasyBuild v2.0 for functionality that was deprecated for EasyBuild v2.0:

```
eb gzip-1.5-goolf-1.4.10.eb --deprecated=1.0
```

7.5.3 How to check for use of deprecated functionality

Since EasyBuild v1.16.0, the `--deprecated` command line option can be used to check whether deprecated behavior is still being triggered in your EasyBuild setup.

For example, using `--deprecated=5.0` with EasyBuild v4.x will transform any deprecation warning for functionality that will no longer be supported in EasyBuild v5.0 into an error message. For example:

```
$ eb test.eb --deprecated=5.0
== temporary log file in case of crash /tmp/easybuild-WWalWX/easybuild-aoL9Nd.log
ERROR: Failed to process easyconfig /home/example/test.eb:
DEPRECATED (since v5.0) functionality used: Use of 'dummy' toolchain is deprecated,␣
↪use 'system' toolchain instead;
see http://easybuild.readthedocs.org/en/latest/Deprecated-functionality.html for more␣
↪information
be used; see http://easybuild.readthedocs.org/en/latest/Deprecated-functionality.html␣
↪for more information
```

Tip: Define `deprecated` to the next major EasyBuild version in one of your EasyBuild configuration files (see *Configuration file(s)*) or by defining `$EASYBUILD_DEPRECATED=5.0`, to ensure you are made aware of deprecated behavior as early as possible.

You can (temporarily) still rely on the deprecated functionality by specifying a *lower* version via `--deprecated` to overrule that setting, until the functionality is actually disabled.

7.6 Removed functionality

Some of the functionality that was available in previous EasyBuild versions is now *removed*, after it was deprecated first in an earlier EasyBuild version (see *Deprecation policy*).

7.6.1 Overview of removed functionality since EasyBuild v4.0

In EasyBuild v4.0, some intrusive changes were made that break backward compatibility with earlier versions.

Note: In addition, please take into account the additional changes in EasyBuild v4.0, which are documented here.

For authors of easyconfig files:

- *use_fma custom easyconfig parameter for FFTW*
- *Specifying source files as 2-element tuples to provide a custom extraction command*
- *use_easy_install and use_setup_py_develop custom easyconfig parameters for PythonPackage easyblock*

For developers of easyblocks:

- *copytree function*
- *skip_symlinks named argument for adjust_permissions*

For EasyBuild framework developers:

- *default_fallback named argument for get_easyblock_class*
- *add_dependencies method in Toolchain class*

`use_fma` custom easyconfig parameter for FFTW

- *deprecated since:* EasyBuild v3.2.0 (May 5th 2017)
- *removed in:* EasyBuild v4.0
- *alternatives*: **use** `use_fma4` **easyconfig parameter instead**

The `use_fma` easyconfig parameter is no longer supported, and was replaced by the equivalent easyconfig parameter `use_fma4`.

`use_fma` was introduced in EasyBuild v3.1.0 allow configuring FFTW with `--enable-avx-128-fma`. Since it is only supported on systems with AMD processors that have the `FMA4` feature, it was replaced by the more fittingly named `use_fma4` parameter in EasyBuild v3.2.0.

Specifying source files as 2-element tuples to provide a custom extraction command

- *deprecated since:* EasyBuild v3.3.0 (June 22nd 2017)
- *removed in:* EasyBuild v4.0
- *alternatives*: **use** `extract_cmd` **key in Python dictionary format instead**

Specyfing a custom extraction command for a particular source file by using a 2-element tuple in `sources` is no longer supported.

Instead, a Python dictionary containing the `filename` and `extract_cmd` keys should be used instead, see *Alternative formats for sources*.

So, this:

```
# source file is actually a gzipped tarball (filename should be .tar.gz)
# DEPRECATED FORMAT, don't use this anymore!
sources = [('example.gz', "tar xfvz %s")]
```

should be replaced with:

```
sources = [{
  'filename': 'example-%(version)s.gz',
  'extract_cmd': "tar xfvz %s",  # source file is actually a gzipped tarball␣
↪(filename should be .tar.gz)
}]
```

`use_easy_install` and `use_setup_py_develop` custom easyconfig parameters for `PythonPackage` easyblock

- *deprecated since:* EasyBuild v3.5.1 (Jan 17th 2018)
- *removed in:* EasyBuild v4.0
- *alternatives*: **use** `install_target` **easyconfig parameter instead**

The custom easyconfig parameters `use_easy_install` and `use_setup_py_develop` for the `PythonPackage` easyblock are no longer supported. They are obsolete since the `install_target` custom easyconfig parameter was added in https://github.com/easybuilders/easybuild-easyblocks/pull/1341.

Rather than using `use_easy_install = True`, you should now use `install_target = 'easy_install'` instead.

Rather than using `use_setup_py_develop = True`, you should now use `install_target = 'develop'` instead.

`copytree` function

- *deprecated since:* EasyBuild v3.2.0 (May 5th 2017)
- *removed in:* EasyBuild v4.0
- *alternatives*: **use** `copy_dir` **instead**

The `copytree` function, which was a copy of the `shutil.copytree` function (introduced when Python 2.4 was still supported) is no longer supported. It has been replaced by the superior `copy_dir` function in the `easybuild.tools.filetools` module.

`copy_dir` graciously handles any exceptions that occur, and is aware of the EasyBuild *dry run* mode.

`skip_symlinks` named argument for `adjust_permissions`

- *deprecated since:* EasyBuild v3.8.0 (Nov 2018)
- *removed in:* EasyBuild v4.0
- *alternatives*: *(none required)*

The `skip_symlinks` argument for the `adjust_permissions` function is no longer supported since `adjust_permissions` has been changed to *always* skip symbolic links (this was already the default behaviour); see also https://github.com/easybuilders/easybuild-framework/pull/2644 .

`default_fallback` named argument for `get_easyblock_class`

- *deprecated since:* EasyBuild v3.2.0 (May 5th 2017)
- *removed in:* EasyBuild v4.0
- *alternatives*: **use** `error_on_missing_easyblock` **named parameter instead**

The `get_easyblock_class` implementation was cleaned up to remove the support for falling back to the generic `ConfigureMake` easyblock in EasyBuild v3.2.0 (see https://github.com/easybuilders/easybuild-framework/pull/2178), following the disabling of the *Automagic fallback to ConfigureMake* in EasyBuild v2.0.

The `default_fallback` named argument for `get_easyblock_class` was replaced by `error_on_missing_easyblock`, to retain support for ignoring a missing matching easyblock rather than raising an error.

`add_dependencies` method in `Toolchain` class

- *deprecated since:* EasyBuild v3.8.0
- *removed in:* EasyBuild v4.0
- *alternatives*: **pass list of dependencies to** `deps` **named argument of** `prepare` **method instead**

The `add_dependencies` method in the `Toolchain` class is no longer supported, to provide more flexibility in the EasyBuild framework w.r.t. handling of dependencies (see https://github.com/easybuilders/easybuild-framework/pull/2674).

Instead, the list of dependencies should be passed to the `Toolchain.prepare` method, via the `deps` named argument.

7.6.2 Overview of removed functionality since EasyBuild v3.0

In EasyBuild v3.0, some intrusive changes were made that break backward compatibility with earlier versions.

For EasyBuild users & authors of easyconfig files:

- *Archived easyconfigs*

For developers of easyblocks:

- *error and exception log methods no longer raise an exception*
- *get_blas_lib function provided by LAPACK easyblock has been removed*
- *get_netcdf_module_set_cmds function provided by netCDF easyblock was removed*

For EasyBuild framework developers:

- *error and exception log methods no longer raise an exception*

`error` and `exception` log methods no longer raise an exception

- *deprecated since:* EasyBuild v2.1.0 (April'15)
- *removed in:* EasyBuild v3.0
- *alternative(s)*: **use** `raise EasyBuildError(...)` **instead**

The `error()` and `exception()` log methods defined by EasyBuild (in the `easybuild.tools.build_log` module) did not match the semantics of the standard Python log methods, in the sense that they used to also raise an exception next to logging messages.

This caused problems when 3rd party libraries (e.g., gc3pie) were being used by EasyBuild, since they may be using these log methods without expecting an exception being raised.

The custom definitions for the `error()` and `exception()` log methods was removed in EasyBuild v3.0.

Hence, these log methods should no longer be used to report errors since they will not raise an exception anymore once. Note that this applies both to the EasyBuild framework and to (custom) easyblocks.

To report errors, an `EasyBuildError` should be raised instead. For example:

```
# make sure config.sh script is there
if not os.path.exists(os.path.join(self.builddir, 'config.sh')):
    raise EasyBuildError("config.sh script is missing in %s", self.builddir)
```

`get_blas_lib` function provided by `LAPACK` easyblock has been removed

- *deprecated since:* EasyBuild v1.3.0 (April'13); see https://github.com/easybuilders/easybuild-easyblocks/pull/150
- *removed in:* EasyBuild v3.0
- *alternative(s)*: **leverage modules from** `easybuild.toolchain.linalg`

The `get_blas_lib` function provided by the `LAPACK` easyblock was removed, mainly because it included a hard-coded list of BLAS libraries.

It was replaced by 'inlining' similar code into the easyblocks that rely on it (e.g. ScaLAPACK, cfr. https://github.com/easybuilders/easybuild-easyblocks/pull/1014), which only refers to the BLAS libraries that are relevant in that context.

`get_netcdf_module_set_cmds` function provided by `netCDF` easyblock was removed

- *deprecated since:* EasyBuild v2.1.0 (April'15); see https://github.com/easybuilders/easybuild-easyblocks/pull/590
- *removed in:* EasyBuild v3.0
- *alternative(s)*: **rely on** `set_netcdf_env_vars` **and use** `self.module_generator.set_environment`

The `get_netcdf_module_set_cmds` function provided by the `netCDF` easyblock was removed, because it returned `setenv` statements to be included in module files that are only compatible with module files in Tcl syntax; i.e. it did not take into account the `--module-syntax` configuration option.

The use of `get_netcdf_module_set_cmds` should be replaced by using `set_netcdf_env_vars` to define the `NETCDF*` environment variables, in combination with `self.module_generator.set_environment` to obtain `setenv` statements that are compatible with the module syntax (`Tcl` or `Lua`) being used.

See for example the changes made to the `WRF` and `WPS` easyblocks in https://github.com/easybuilders/easybuild-easyblocks/commit/7a05cbd823769e343b951002b4735dc7632e19c0.

7.6.3 Overview of removed functionality since EasyBuild v2.0

In EasyBuild v2.0, some intrusive changes were made that break backward compatibility with earlier versions.

For EasyBuild users:

- *Python version compatibility*
- *EasyBuild configuration*
- *$SOFTX environment variables in generated module files*

For authors of easyconfig files:

- *Automagic fallback to ConfigureMake*
- *Easyconfig parameters*
- *BEAGLE dependency in MrBayes easyblock replaced by beagle-lib*

For developers of easyblocks:

- *Easyblocks API (EasyBlock class from easybuild.framework.easyblock)*
- *Renamed/relocated functions*
- *Changes in (generic) easyblocks*

For EasyBuild framework developers:

- *easybuild.tools.modules Python module*

Note: A script `fix-broken-easyconfigs.py` is provided to fix easyconfig files that were broken by the backward-incompatible changes documented at *Automagic fallback to ConfigureMake* and *Easyconfig parameters*. See *fix_broken_easyconfigs.py* for more information.

Python version compatibility

Compatibility with Python 2.6 is removed.

- *deprecated since:* EasyBuild v4.1.0 (Nov'19)
- *removed in:* EasyBuild v4.4.0
- *alternative(s)*: **upgrade to Python v2.7.x or v3.5+**

Support for running EasyBuild on top of Python 2.6 was removed in EasyBuild version 4.4.0.

You should upgrade to a newer version of Python (see also py2_py3_compatibility).

Compatibility with Python 2.4 is removed.

- *deprecated since:* EasyBuild v1.14.0 (July'14)
- *removed in:* EasyBuild v2.0
- *alternative(s)*: **upgrade to Python v2.6.x or v2.7.x**

Ever since EasyBuild v1.0, the codebase has been Python 2.4 compatible. One reason for this is that EasyBuild was being used on a daily basis on Scientific Linux 5, in which the Python 2.4.x is the system default.

Starting with EasyBuild v2.0 support for Python 2.4 is removed, and only ensure compatibility with Python 2.6.x or a more recent Python 2.x.

This will enable us to gradually also make the codebase compatible with Python 3.x, which is difficult to do without removing support for Python 2.4.

EasyBuild configuration

Old-style EasyBuild configuration is removed.

- *deprecated since:* EasyBuild v1.3.0 (Apr'13)
- *removed in*: EasyBuild v2.0
- *alternatives:* **new-style configuration** (see *Configuring EasyBuild*)

Early versions of EasyBuild v1.x provided support for configuring EasyBuild via a *Python module* that was automagically executed when available.

Since EasyBuild v1.3 a safer and more consistent way of configuring EasyBuild is supported, which aligns the EasyBuild command line options, `$EASYBUILD_X` environment variables and key-value style configuration files.

More information about the new(er) and recommended configuration style is available *here*.

For detailed information with respect to porting from the old to the new configuration style, see *Configuration Legacy*.

Location of default configuration file

The default configuration file location `$HOME/.easybuild/config.cfg` is no longer considered.

- *deprecated since:* EasyBuild v1.11.0 (Feb'14)
- *removed in*: EasyBuild v2.0
- *alternatives:* `$XDG_CONFIG_HOME/easybuild/config.cfg` (equivalent to `$HOME/.config/easybuild/config.cfg`)

The default path for the new-style configuration path is `$XDG_CONFIG_HOME/easybuild/config.cfg` (or `$HOME/.config/easybuild/config.cfg` if `$XDG_CONFIG_HOME` is not set), see *List of used configuration files*.

The previous default path `$HOME/.easybuild/config.cfg` that was in place since EasyBuild v1.3.0 is no longer considered.

Automagic fallback to `ConfigureMake`

The automagic fallback to the `ConfigureMake` easyblock is removed.

- *deprecated since:* EasyBuild v1.16.0 (Dec'14)
- *removed in:* EasyBuild v2.0
- *alternative(s)*: specify `easyblock = 'ConfigureMake'` in easyconfig file

If the `easyblock` easyconfig was not specified, EasyBuild tries to find a matching easyblock based on the software name. In EasyBuild v1.x, the generic `ConfigureMake` easyblock was used if no matching easyblock could be found.

This behavior is now removed; instead, easyconfigs that require using the `ConfigureMake` easyblock *must* include the following:

```
easyblock = 'ConfigureMake'
```

Easyconfig parameters

Some easyconfig parameters are removed.

Note: A script is available to fix easyconfig files that are broken because they still rely on this functionality, see *fix_broken_easyconfigs.py*.

Options for build command

The `premakeopts` and `makeopts` easyconfig parameters are removed.

- *deprecated since:* EasyBuild v1.13.0 (May'14)
- *removed in:* EasyBuild v2.0
- *alternative(s)*: use `prebuildopts`/`buildopts` instead

For consistency in terminology, the `premakeopts` and `makeopts` generic easyconfig parameters are removed, in favor of their alternative parameters, `prebuildopts` and `buildopts`, resp.

(see also *Configure/build/install command options*)

Note: Since EasyBuild v1.13.0, `buildopts` is automatically defined with the value of `makeopts`, unless `buildopts` was specified by itself. When both values are specified, `buildopts` takes precedence of `makeopts` (analogous for `prebuildopts`/`premakeopts`).

Shared library extension

The `shared_lib_ext` 'constant' in easyconfigs is no longer defined.

- *deprecated since:* EasyBuild v1.5.0 (June'13)
- *removed in:* EasyBuild v2.0
- *alternative(s)*: use `SHLIB_EXT` instead

The `shared_lib_ext` "magic" variable representing the extension for shared libraries (`.so` on Linux, `.dylib` on OS X) is no longer defined; the easyconfig constant `SHLIB_EXT` should be using instead.

Software license

The `license` easyconfig parameter is removed.

- *deprecated since:* EasyBuild v1.11.0 (Feb'14)
- *removed in:* EasyBuild v2.0
- *alternative(s)*: use `license_file` or `software_license` instead

The `license` easyconfig parameter, which was specific to the `IntelBase` generic easyblock and thus relevant for Intel tools, is removed. The generic `license_file` easyconfig parameter should be used instead, to specify the location of the license file (or server).

This change was made to avoid confusion with the `software_license` generic easyconfig parameter, which can be used to specify the license under which the software was released (e.g., GPLv2, BSD, etc.). Here, the specified value *must* be a known license type (see `eb --avail-easyconfig-licenses`).

Note: The *software_license* easyconfig parameter will become **mandatory** at some point.

`BEAGLE` dependency in `MrBayes` easyblock replaced by `beagle-lib`

The `MrBayes` easyblock no longer considers `BEAGLE` as a valid dependency.

- *deprecated since:* EasyBuild v1.6.0 (Jul'14)
- *removed in:* EasyBuild v2.0
- *alternative(s)*: use `beagle-lib` instead

Due to a misnomer in the easyconfig files for `beagle-lib` (formerly named `BEAGLE`), the custom easyblock for `MrBayes` now no longer considers `BEAGLE` as a dependency.

The library required by `MrBayes` must now be provided as a dependency named `beagle-lib`.

EasyBuild API changes

Some changes in the EasyBuild API were made, which potentiallty affects easyblocks and the EasyBuild framework itself.

Easyblocks API (`EasyBlock` class from `easybuild.framework.easyblock`)

The API for easyblocks was modified slightly, to correct for a couple of historic mistakes.

Return type of `extra_options` method

The list-of-tuples return type of the `extra_options` method must now be a `dict` instead.

- *deprecated since:* EasyBuild v1.12.0 (Apr'14)
- *removed in:* EasyBuild v2.0
- *alternative(s)*: ensure/assume `dict` return type

The return type of the `extra_options` static method in the `EasyBlock` class has been changed to a *dictionary* (`dict`), rather than a list of key-value tuples.

Custom easyconfig parameters should be added via a *dict*-typed value to the `extra_options` function of parent easyblock.

For example (taken from the generic easyblock `Binary`):

```
@staticmethod
def extra_options(extra_vars=None):
    """Extra easyconfig parameters specific to Binary easyblock."""
    extra_vars = EasyBlock.extra_options(extra_vars)
    extra_vars.update({
        'install_cmd': [None, "Install command to be used.", CUSTOM],
    })
    return extra_vars
```

Extension filter template

The `name` and `version` templates in `exts_filter` are removed.

- *deprecated since:* EasyBuild v1.2.0 (Feb'13)
- *removed in:* EasyBuild v2.0
- *alternative(s)*: use `ext_name` and `ext_version` instead

Only the `ext_name`, `ext_version` and `src` template strings can be used in the `exts_filter` extension filter easyconfig parameter; the `name` and `version` template strings are removed.

For example (default extension filter for Python packages):

```
exts_filter = ("python -c 'import %(ext_name)s'", "")
```

Module path of default class for extensions

Specifying the module path in `exts_defaultclass` is no longer possible.

- *deprecated since:* EasyBuild v0.5 (Apr'12)
- *removed in:* EasyBuild v2.0
- *alternative(s)*: *(none required, module path is derived from specified class name)*

Explicitly specifying the module path for the default class to use for extensions (via `exts_defaultclass`) is no longer possible. Only the class name should be specified, the corresponding module path is derived from it.

Module path for easyblocks

Deriving the module path for easyblocks from the software name is removed.

- *deprecated since:* EasyBuild v1.4.0 (May'13)
- *removed in:* EasyBuild v2.0
- *alternative(s)*: use easyblock class name according to encoding scheme (e.g., `EB_Foo`)

Determining the *location* of Python modules representing easyblocks based on the software name (`name`) is removed.

EasyBuild *must* be able to determine the easyblock module path solely based on the name of the easyblock Python class.

Easyblocks with a class name that is already honoring the encoding scheme implemented by the `encode_class_name` function will not be affected.

`easybuild.tools.modules` Python module

The API of the `easybuild.tools.modules` module has been updated, certain aspects of the old API are removed.

- *deprecated since:* EasyBuild v1.8.0 (Oct'13) & v1.15.0 (Sept'15)
- *removed in:* EasyBuild v2.0
- *alternative(s)*: use equivalents available in new API (see below)

The API of the `easybuild.tools.modules` Python module has been changed extensively when implementing support for alternative module naming schemes:

- the `modules` class variable and the `add_module`/`remove_module` methods are removed; modules should be (un)loaded using the `load` and `unload` methods instead
- the `mod_paths` and `modulePath` named arguments for the `run_module` method are removed; the class instance should be created with a specific list of module paths instead
- the `Modules` class to obtain a class instance representing a modules tool interface is removed; the `modules_tool` function should be used instead

Additionally, the `exists` method which only takes a single module name is removed; it is replaced by the `exist` method, which takes a list of module names *(since EasyBuild v1.15.0 (Sept'15))*.

Easyblocks should not be using `easybuild.tools.modules` directly, and hence should be unaffected.

`$SOFTX` environment variables in generated module files

`$SOFTX` **environment variables set by module files generated with EasyBuild v0.x will no longer be taken into account.**

- *deprecated since:* EasyBuild v1.3.0 (Apr'13)
- *removed in:* EasyBuild v2.0
- *alternative(s)*: reinstall (ancient) module files which are only defining the `$SOFTX` environment variables

The `get_software_root` and `get_software_version` functions will only take `$EBROOTFOO` and `$EBVERSIONFOO` environment variables into account, as opposed to also considering the `$SOFTROOTFOO` and `$SOFTVERSIONFOO` environment variables (which were set in modules generated by EasyBuild v0.x). Likewise, adhering to the `$SOFTDEVELFOO` environment variables is removed.

This is only relevant to early adopters who are still using module files generated by EasyBuild v0.x.

Renamed/relocated functions

Some functions/methods have been renamed or relocated, their equivalents under a previous location/name are removed.

- *deprecated since: (depends on function/method, see below)*
- *removed in:* EasyBuild v2.0
- *alternative(s)*: use new location/name

A number of functions and methods that are part of the EasyBuild framework API have been renamed, mainly for consistency reasons.

- the `moduleGenerator` handle to the `ModuleGenerator` object instance has been renamed to `module_generator`; hence, easyblock should be using `self.module_generator` rather than `self.moduleGenerator` *(since EasyBuild v1.16.0 (Dec'14))*
- `source_paths()` (in `easybuild.tools.config`) replaces the removed `source_path()` *(since EasyBuild v1.8.0 (Oct'13))*
- `get_avail_core_count()` (in `easybuild.tools.systemtools`) replaces the removed `get_core_count()` *(since EasyBuild v1.9.0 (Nov'13))*
- `get_os_type()` (in `easybuild.tools.systemtools`) replaces the removed `get_kernel_name` *(since EasyBuild v1.3.0 (Apr'13))*
- the `det_full_ec_version` function available from `easybuild.tools.module_generator` replaces the removed `det_installversion` function that was available from `easybuild.framework.easyconfig.*` *(since EasyBuild v1.8.0 (Oct'13))*

Some functions have moved to a different location:

- the `read_environment` function is now provided by the `easybuild.tools.environment` module, rather than by `easybuild.tools.config` or `easybuild.tools.utilities` *(since EasyBuild v1.7.0 (Sept'13))*
- the `modify_env` function is now provided by the `easybuild.tools.environment` module, rather than by `easybuild.tools.filetools` *(since EasyBuild v1.7.0 (Sep'13))*
- the `run_cmd`, `run_cmd_qa` and `parse_log_for_error` functions are now provided by the `easybuild.tools.run` module, rather than by `easybuild.tools.filetools` *(since EasyBuild v1.11.0 (Feb'14))*

The `get_log` function provided by the `easybuild.tools.build_log` module has been removed entirely, no alternatives are provided (since none are needed). *(since EasyBuild v1.3.0 (Apr'13))*

Changes in (generic) easyblocks

`srcdir` replaces `builddir` as named argument in `CMakeMake.configure_step`

The named argument `builddir` in the `configure_step` method of the generic `CMakeMake` easyblock was replaced by `srcdir`.

- *deprecated since:* EasyBuild v1.4.0 (May'13)
- *removed in:* EasyBuild v2.0

- *alternative(s)*: equivalent `srcdir` named argument

Since the `builddir` named argument in the `configure_step` method of the generic `CMakeMake` easyblock was a misnomer (it specifies the location of the *source* directory that should be provided to `cmake`), it was replaced with an equivalent named argument `srcdir`.

`VersionIndependentPythonPackage` replaces `VersionIndependendPythonPackage`

The generic easyblock `VersionIndependendPythonPackage` was replaced with the equivalent generic easyblock `VersionIndependentPythonPackage`.

- *deprecated since:* EasyBuild v1.11.0 (Feb'14)
- *removed in:* EasyBuild v2.0
- *alternative(s)*: `VersionIndependentPythonPackage`

Because of to a typo in the name, the `VersionIndependendPythonPackage` generic easyblock was replaced by the equivalent `VersionIndependentPythonPackage` generic easyblock.

`get_sitearch_suffix` function in `Perl` easyblock is removed

The `get_sitearch_suffix` function in the `Perl` easyblock was replaced in favor of the more generic `get_site_suffix` function.

- *deprecated since:* EasyBuild v1.7.0 (Sept'13)
- *removed in:* EasyBuild v2.0
- *alternative(s)*: `get_site_suffix('sitearch')`

The `get_sitearch_suffix` function provided by the `Perl` easyblock, which can be (and is) imported in/used by other easyblocks, has been replaced by the more generic `get_site_suffix` function.

To obtain the same functionality as was provided by `get_sitearch_suffix`, use `get_site_suffix('sitearch')` instead.

7.7 EasyBuild maintainers

This page provides an overview of the current maintainers of the EasyBuild repositories.

Contents

7.7.1 Criteria

EasyBuild maintainers should meet the following criteria:

- being sufficiently familiar with the contribution procedure (see *Contributing*)
- frequently contributing to the EasyBuild GitHub repositories
- being an active member of the EasyBuild community for a significant amount of time, i.e.:
  - following the EasyBuild mailing list and participating in discussions (see *Getting help*)
  - actively participating in the EasyBuild IRC or Slack channel (see *Getting help*);
  - (occasionally) joining in on the EasyBuild conference calls (see https://github.com/easybuilders/easybuild/wiki/Conference-calls)
- having access to a working setup for submitting test reports (see *Test reports for easyconfig contributions (upload-test-report)*)

7.7.2 Roles

Release manager

The release manager is responsible for releasing new stable EasyBuild versions on a regular basis, with the help of the other EasyBuild maintainers.

Miguel & Kenneth are working together to get EasyBuild releases out the door on a regular basis (about once every 6 weeks, see https://pypi.org/project/easybuild/#history).

- **Miguel Dias Costa** (National University of Singapore)
  - GitHub: @migueldiascosta - e-mail: `migueldiascosta (at) nus.edu.sg` - IRC/Slack: `migueldiascosta`
- **Kenneth Hoste** (HPC-UGent)
  - GitHub: @boegel - e-mail: `kenneth.hoste (at) ugent.be` - IRC/Slack: `boegel`

Maintainers

The EasyBuild maintainers all have admin access to the different EasyBuild GitHub repositories:

- https://github.com/easybuilders/easybuild (documentation (http://easybuild.readthedocs.io), EasyBuild website, *easybuild* metapackage)
- https://github.com/easybuilders/easybuild-framework (EasyBuild framework)
- https://github.com/easybuilders/easybuild-easyblocks (central repository for easyblocks)
- https://github.com/easybuilders/easybuild-easyconfigs (central repository for easyconfigs)

(maintainers are listed in alphabetical order, by last name)

- **Sebastian Achilles** (JSC)
  - GitHub: @SebastianAchilles - e-mail: `s.achilles (at) fz-juelich.de` - IRC/Slack: `Sebastian Achilles`
- **Damian Alvarez** (JSC)
  - GitHub: @damianam - e-mail: `d.alvarez (at) fz-juelich.de` - IRC/Slack: `dalvarez`

- **Simon Branford** (BEAR - University of Birmingham)
  - GitHub: @branfosj - e-mail: s.j.branford (at) bham.ac.uk - IRC/Slack: Simon
- **Miguel Dias Costa** (National University of Singapore)
  - GitHub: @migueldiascosta - e-mail: migueldiascosta (at) nus.edu.sg - IRC/Slack: migueldiascosta
- **Alex Domingo** (Vrije Universiteit Brussel)
  - GitHub: @lexming - e-mail: alex.domingo.toro (at) vub.be - IRC/Slack: lexming
- **Bob Dröge** (University of Groningen)
  - GitHub: @bedroge - e-mail: b.e.droge (at) rug.nl - IRC/Slack: Bob Dröge
- **Pablo Escobar** (sciCORE, University of Basel)
  - GitHub: @pescobar - e-mail: pablo.escobarlopez (at) unibas.ch - IRC/Slack: pescobar
- **Fotis Georgatos** (SDSC)
  - GitHub: @fgeorgatos - e-mail: kefalonia (at) gmail.com - IRC/Slack: fotis
- **Balázs Hajgató** (HPC-UGent)
  - GitHub: @hajgato - e-mail: balazs.hajgato (at) ugent.be - IRC/Slack: hajgato
- **Kenneth Hoste** (HPC-UGent)
  - GitHub: @boegel - e-mail: kenneth.hoste (at) ugent.be - IRC/Slack: boegel
- **Adam Huffman** (Big Data Institute, University of Oxford)
  - GitHub: @verdurin - e-mail: adam.huffman (at) gmail.com - IRC/Slack: verdurin
- **Samuel Moors** (Vrije Universiteit Brussel)
  - GitHub: @smoors - e-mail: samuel.moors (at) vub.be - IRC/Slack: smoors
- **Alan O'Cais** (JSC)
  - GitHub: @ocaisa - e-mail: a.ocais (at) fz-juelich.de - IRC/Slack: ocaisa
- **Mikael Öhman** (Chalmers University of Technology)
  - GitHub: @Micket - e-mail: micketeer (at) gmail.com - IRC/Slack: micketeer
- **Bart Oldeman** (ComputeCanada)
  - GitHub: @bartoldeman - e-mail: bart.oldeman (at) calculquebec.ca - IRC/Slack: bartoldeman
- **Ward Poelmans** (Vrije Universiteit Brussel)
  - GitHub: @wpoely86 - e-mail: wpoely86 (at) gmail.com - IRC/Slack: wpoely86
- **Åke Sandgren** (Umeå University, Sweden)
  - GitHub: @akesandgren - e-mail: ake.sandgren (at) hpc2n.umu.se - IRC/Slack: ake_s
- **Caspar Van Leeuwen** (SURFsara, Netherlands)
  - GitHub: @casparvl - e-mail: caspar.vanleeuwen (at) surfsara.nl - IRC/Slack: Caspar Van Leeuwen
- **Davide Vanzo** (Microsoft)
  - GitHub: @vanzod - e-mail: davide.vanzo (at) microsoft.com - IRC/Slack: vanzod

- **Lars Viklund** (Umeå University, Sweden)
  - GitHub: @zao - e-mail: `lars.viklund (at) umu.se` - IRC/Slack: `zao`

CHAPTER 8

Overview of version specific (auto-generated) documentation pages

- Available config file constants
- Available easyconfig parameters
- Constants available for easyconfig files
- demos
- EasyBuild framework API
- License constants available for easyconfig files
- List of available easyblocks
- List of available toolchain options
- List of known toolchains
- List of supported software
- Overview of configuration options (eb –help)
- Overview of generic easyblocks
- Templates available for easyconfig files

CHAPTER 9

Getting help

Having trouble? We'd like to help!

- Search this documentation collection
- Search for information in the archives of the easybuild@lists.ugent.be mailing list or subscribe to post a question.
- Did you try `eb --help`?
- Ask a question in the #easybuild IRC channel on the Freenode network, or in the EasyBuild Slack channel https://easybuild.slack.com/ (subscribe via https://easybuild-slack.herokuapp.com)
- Consider participating to an EasyBuild conference call
- Report issues with EasyBuild framework in our framework ticket tracker.
- Report issues with EasyBuild easyblocks in our easyblocks ticket tracker.
- Report issues with EasyBuild easyconfigs in our easyconfigs ticket tracker.
- Report issues with EasyBuild documentation or other aspects in our general ticket tracker.

CHAPTER 10

Lists and tables

As of version EasyBuild version 4.4.2:

- The complete table of available toolchains is visible at *List of known toolchains*
- The list of available easyblocks is visible at *List of easyblocks*
- The list of available (generic) easyconfig parameters is visible at easyconfigs_parameters

CHAPTER 11

Appendices

11.1 Changelog for EasyBuild documentation

(for EasyBuild release notes, see *EasyBuild release notes*)

- **release 20210907.01** (*Sep 7th 2021*): update release notes for EasyBuild v4.4.2 (see *EasyBuild v4.4.2 (September 7th 2021)*)
- **release 20210706.01** (*Jul 6th 2021*): update release notes for EasyBuild v4.4.1 (see *EasyBuild v4.4.1 (July 6th 2021)*)
- **release 20210602.01** (*Jun 2nd 2021*): update release notes for EasyBuild v4.4.0 (see *EasyBuild v4.4.0 (June 2nd 2021)*)
- **release 20210409.01** (*Apr 9th 2021*): update release notes for EasyBuild v4.3.4 (see *EasyBuild v4.3.4 (Apr 9th 2021)*)
- **release 20210223.01** (*Feb 23rd 2021*): update release notes for EasyBuild v4.3.3 (see *EasyBuild v4.3.3 (Feb 23rd 2021)*)
- **release 20201210.01** (*Dec 10th 2020*): update release notes for EasyBuild v4.3.2 (see *EasyBuild v4.3.2 (December 10th 2020)*)
- **release 20201029.01** (*Oct 29th 2020*): update release notes for EasyBuild v4.3.1 (see *EasyBuild v4.3.1 (October 29th 2020)*)
- **release 20200913.01** (*Sep 13th 2020*): update release notes for EasyBuild v4.3.0 (see *EasyBuild v4.3.0 (September 13th 2020)*)

- **release 20200708.01** (*July 8th 2020*): update release notes for EasyBuild v4.2.2 (see *EasyBuild v4.2.2 (July 8th 2020)*)
- **release 20200520.01** (*May 20th 2020*): update release notes for EasyBuild v4.2.1 (see *EasyBuild v4.2.1 (May 20th 2020)*)
- **release 20200414.01** (*Apr 14th 2020*):
  - document new EasyBuild locking mechanism (see *Locks to prevent duplicate installations running at the same time*)
  - document support for creating index files (see *Using an index to speed up searching for easyconfigs*)
  - update release notes for EasyBuild v4.2.0 (see *EasyBuild v4.2.0 (April 14th 2020)*)
- **release 20200316.01** (*Mar 16th 2020*): update release notes for EasyBuild v4.1.2 (see *EasyBuild v4.1.2 (March 16th 2020)*)
- **release 20200116.01** (*Jan 16th 2020*): update release notes for EasyBuild v4.1.1 (see *EasyBuild v4.1.1 (January 16th 2020)*)
- **release 20191204.01** (*Dec 4th 2019*): update release notes for EasyBuild v4.1.0 (see *EasyBuild v4.1.0 (December 4th 2019)*)
- **release 20191015.01** (*Oct 15th 2019*): update release notes for EasyBuild v4.0.1 (see *EasyBuild v4.0.1 (October 15th 2019)*)
- **release 20190920.01** (*Sep 20th 2019*): update release notes for EasyBuild v4.0.0 (see *EasyBuild v4.0.0 (September 20th 2019)*); see also eb4_changes_overview
- **release 20190823.01** (*Aug 23rd 2019*): update release notes for EasyBuild v3.9.4 (see *EasyBuild v3.9.4 (August 23rd 2019)*)
- **release 20190708.01** (*Jul 8th 2019*): update release notes for EasyBuild v3.9.3 (see *EasyBuild v3.9.3 (July 8th 2019)*)
- **release 20190609.01** (*Jun 9th 2019*): update release notes for EasyBuild v3.9.2 (see *EasyBuild v3.9.2 (June 9th 2019)*)
- **release 20190908.01** (*Jun 8th 2019*): update documentation on support for generating (Singularity) container recipes/images (see *Generating container recipes & images*)
- **release 20190520.01** (*May 20th 2019*): update release notes for EasyBuild v3.9.1 (see *EasyBuild v3.9.1 (May 20th 2019)*)
- **release 20190412.01** (*Apr 12th 2019*): update release notes for EasyBuild v3.9.0 (see *EasyBuild v3.9.0 (April 12th 2019)*)
- **release 20190129.01** (*Jan 29th 2019*): update release notes for EasyBuild v3.8.1 (see *EasyBuild v3.8.1 (January 29th 2019)*)
- **release 20190124.01** (*Jan 24th 2019*): add definitions for `2019a` common toolchains (see *Common toolchains*)
- **release 20181218.01** (*Dec 18th 2018*): update release notes for EasyBuild v3.8.0 (see *EasyBuild v3.8.0 (December 18th 2018)*)
- **release 20181114.01** (*Nov 14th 2018*): add documentation on deprecated easyconfigs & toolchains (*Deprecated easyconfigs*)
- **release 20181104.01** (*Nov 4th 2018*): update `--job` documentation to also cover `Slurm` job backend (see *Submitting jobs using –job*)
- **release 20181018.01** (*Oct 18th 2018*): update release notes for EasyBuild v3.7.1 (see *EasyBuild v3.7.1 (October 18th 2018)*)

- **release 20180925.01** (*Sep 25th 2018*): update release notes for EasyBuild v3.7.0 (see *EasyBuild v3.7.0 (September 25th 2018)*)
- **release 20180921.01** (*Sep 21st 2018*): document support for wrapping dependencies (see *Wrapping dependencies*)
- **release 20180920.01** (*Sep 20th 2018*): document support for downloading sources directly from a Git repository (see *Downloading from a Git repository*)
- **release 20180711.01** (*Jul 11th 2018*): update release notes for EasyBuild v3.6.2 (see *EasyBuild v3.6.2 (July 11th 2018)*)
- **release 20180710.01** (*Jul 10th 2018*): add definitions for `2018b` common toolchains (see *Common toolchains*)
- **release 20180528.01** (*May 28th 2018*): update release notes for EasyBuild v3.6.1 (see *EasyBuild v3.6.1 (May 26th 2018)*)
- **release 20180427.01** (*Apr 27th 2018*): update release notes for EasyBuild v3.6.0 (see *EasyBuild v3.6.0 (April 26th 2018)*)
- **release 20180425.01** (*Apr 25th 2018*): add documentation on support for generating container recipes & images (see *Generating container recipes & images*)
- **release 20180307.01** (*Mar 7th 2018*): update release notes for EasyBuild v3.5.3 (see *EasyBuild v3.5.3 (March 7th 2018)*)
- **release 20180302.01** (*Mar 2nd 2018*): update release notes for EasyBuild v3.5.2 (see *EasyBuild v3.5.2 (March 2nd 2018)*)
- **release 20180116.01** (*Jan 16th 2018*): update release notes for EasyBuild v3.5.1 (see *EasyBuild v3.5.1 (January 16th 2018)*)
- **release 20180112.01** (*Jan 12th 2018*): add definitions for `2018a` common toolchains (see *Common toolchains*)
- **release 20171215.01** (*Dec 15th 2017*): update release notes for EasyBuild v3.5.0 (see *EasyBuild v3.5.0 (December 15th 2017)*)
- **release 20171208.01** (*Dec 8th 2017*): document support for user-defined hooks (see *Hooks*)
- **release 20171017.01** (*Oct 17th 2017*): update release notes for EasyBuild v3.4.1 (see *EasyBuild v3.4.1 (October 17th 2017)*)
- **release 20170910.01** (*Sept 10th 2017*): update release notes for EasyBuild v3.4.0 (see *EasyBuild v3.4.0 (September 10th 2017)*)
- **release 20170906.01** (*Sept 6th 2017*): document `--trace` (see *Tracing progress*)
- **release 20170824.02** (*Aug 24th 2017*): document `--inject-checksums` (see *Adding or replacing checksums using –inject-checksums*)
- **release 20170824.01** (*Aug 24th 2017*): document `--backup-modules` (see *Backing up of existing modules (–backup-modules)*)
- **release 20170712.01** (*July 12th 2017*): update release notes for EasyBuild v3.3.1 (see *EasyBuild v3.3.1 (July 12th 2017)*)
- **release 20170708.01** (*July 8th 2017*): add documentation on *Merging easyconfig pull requests (–merge-pr)*
- **release 20170705.01** (*July 5th 2017*): clarify *Requirements for pull requests*, add page listing *EasyBuild maintainers*
- **release 20170626.01** (*June 26th 2017*): update release notes for EasyBuild v3.3.0 (see *EasyBuild v3.3.0 (June 26th 2017)*)

- **release 20170623.01** (*June 23rd 2017*): document use of `checksums` & alternative formats for `sources` (see *Source files, patches and checksums*)
- **release 20170622.01** (*June 22nd 2017*): document support for detecting loaded modules (see detect_loaded_modules)
- **release 20170522.01** (*May 22nd 2017*): document deprecated behaviour in EasyBuild v3.2.0 (see *Overview of deprecated functionality in EasyBuild version 4.4.2*)
- **release 20170512.01** (*May 12th 2017*): update release notes for EasyBuild v3.2.1 (see *EasyBuild v3.2.1 (May 12th 2017)*)
- **release 20170505.01** (*May 5th 2017*): update release notes for EasyBuild v3.2.0 (see *EasyBuild v3.2.0 (May 5th 2017)*)
- **release 20170320.01** (*Mar 20th 2017*): update release notes for EasyBuild v3.1.2 (see *EasyBuild v3.1.2 (March 20th 2017)*)
- **release 20170307.01** (*Mar 7th 2017*): update release notes for EasyBuild v3.1.1 (see *EasyBuild v3.1.1 (March 7th 2017)*)
- **release 20170221.01** (*Feb 21st 2017*): add documentation on *Contributing*
- **release 20170209.01** (*Feb 9th 2017*): add documentation on implementing easyblocks (see *Implementing easyblocks*)
- **release 20170203.01** (*Feb 3rd 2017*): update release notes for EasyBuild v3.1.0 (see *EasyBuild v3.1.0 (February 3rd 2017)*)
- **release 20170129.01** (*Jan 29th 2017*): update `--optarch` documentation (see *Setting architecture flags for different compilers via –optarch=<compiler:flags>;<compiler:flags>*)
- **release 20170109.01** (*Jan 9th 2017*): add documentation on *Common toolchains*
- **release 20161222.01** (*Dec 22nd 2016*): update documentation and release notes for EasyBuild v3.0.2 (see *EasyBuild v3.0.2 (December 22nd 2016)*)
- **release 20161218.01** (*Dec 18th 2016*): document need to download `vsc-*` source tarballs from PyPI (see bootstrap_offline)
- **release 20161202.01** (*Dec 2nd 2016*): add documentation on Cray support (see *EasyBuild on Cray*)
- **release 20161130.01** (*Nov 16th 2016*): update release notes for EasyBuild v3.0.1 (see *EasyBuild v3.0.1 (November 30th 2016)*)
- **release 20161117.01** (*Nov 17th 2016*): update mentions of default configuration according to updated default in EasyBuild v3.0.0
- **release 20161116.01** (*Nov 16th 2016*): update documentation and release notes for EasyBuild v3.0.0 (see *EasyBuild v3.0.0 (November 16th 2016)*)
  - *Archived easyconfigs*
  - *Support for RPATH*
- **release 20161028.01** (*Oct 28th 2016*): recommend using `--rebuild` rather than `--force` (see *Rebuild installation, –rebuild*)
- **release 20161023.01** (*Oct 24th 2016*): add section on iterating over configure/build/install options (see *List of configure/build/install options*)
- **release 20161014.01** (*Oct 14th 2016*): update documentation on deprecated functionality (see *Deprecated functionality*)
- **release 20161010.01** (*Oct 10th 2016*): add page for EasyBuild demos (see demos)

- **release 20160923.02** (*Sept 23rd 2016*): update release notes for EasyBuild v2.9.0 (see *EasyBuild v2.9.0 (September 23rd 2016)*)
- **release 20160923.01** (*Sept 23rd 2016*): add generated list of supported software (see list_software)
- **release 20160713.01** (*July 13th 2016*): update release notes for EasyBuild v2.8.2 (see *EasyBuild v2.8.2 (July 13th 2016)*)
- **release 20160613.01** (*June 13th 2016*): clarify required dependencies (setuptools, vsc-install) (see *Required Python packages*)
- **release 20160607.01** (*June 7th 2016*): update/complete documentation on GitHub integration (see *Integration with GitHub*)
- **release 20160530.01** (*May 30th 2016*): update release notes for EasyBuild v2.8.1 (see *EasyBuild v2.8.1 (May 30th 2016)*)
- **release 20160518.01** (*May 18th 2016*): update release notes for EasyBuild v2.8.0 (see *EasyBuild v2.8.0 (May 18th 2016)*)
- **release 20160429.01** (*April 29th 2016*): add section on updating EasyBuild, see *Updating an existing EasyBuild installation*
- **release 20160320.01** (*March 20th 2016*): update release notes for EasyBuild v2.7.0 (see *EasyBuild v2.7.0 (March 20th 2016)*)
- **release 20160228.01** (*February 28th 2016*):
  - update documentation on external modules metadata (see *Metadata for external modules*)
- **release 20160214.01** (*February 14th 2016*):
- add section on `--show-config` (see *Overview of current configuration (–show-config, –show-full-config)*)
- **release 20160126.02** (*January 26th 2016*): packaging support is stable since EasyBuild v2.5.0 (see *Packaging support*)
- **release 20160126.01** (*January 26th 2016*): update release notes for EasyBuild v2.6.0 (see *EasyBuild v2.6.0 (January 26th 2016)*)
- **release 20151217.01** (*December 17th 2015*): update release notes for EasyBuild v2.5.0 (see *EasyBuild v2.5.0 (December 17th 2015)*)
- **release 20151209.01** (*December 9th 2015*):
  - add documentation on controlling compiler optimizations flags, see *Controlling compiler optimization flags*
- **release 20151110.01** (*November 10th 2015*): update release notes for EasyBuild v2.4.0 (see *EasyBuild v2.4.0 (November 10th 2015)*)
- **release 20151108.01** (*November 8th 2015*):
  - document (experimental) support for using minimal toolchains (see *Using minimal toolchains for dependencies*)
- **release 20151028.01** (*October 28th 2015*): document extended dry run mechanism (see *Extended dry run*)
- **release 20151021.01** (*October 21st 2015*):
  - include initial documentation on experimental support for easyconfig files in YAML syntax (`.yeb`), see easyconfig_yeb_format
- **release 20150902.01** (*September 2nd 2015*): update release notes for EasyBuild v2.3.0 (see *EasyBuild v2.3.0 (September 2nd 2015)*)

- **release 20150715.01** (*July 15th 2015*): update release notes for EasyBuild v2.2.0 (see *EasyBuild v2.2.0 (July 15th 2015)*)
- **release 20150714.01** (*July 14th 2015*): add documentation on *Packaging support*
- **release 20150709.01** (*July 9th 2015*): add documentation on *Submitting jobs using –job*
- **release 20150708.01** (*July 8th 2015*):
  - add documentation on `--include-*` options (see *Including additional Python modules (–include-*)*)
- **release 20150703.01** (*July 3rd 2015*):
  - fix outdated documentation on `easyblock` parameter (see *Easyblock specification*)
- **release 20150624.01 (*June 24th 2015*): mention `MigrateFromEBToHMNS` module naming scheme in section on** `--module-only` (see *Generating additional module files*)
- **release 20150610.01** (*June 10th 2015*): update *Installing Lmod without root permissions* for Lmod v6.0
- **release 20150518.01** (*May 18th 2015*):
  - update section on `--search`: better examples + highlight ability to search via regular expression (see *Searching for easyconfigs, –search / -S*)
  - update release notes for EasyBuild v2.1.1 (see *EasyBuild v2.1.1 (May 18th 2015)*)
- **release 20150506.01** (*May 6th 2015*): updated documentation for EasyBuild v2.1.1
  - add note on `$LMOD_CMD` fallback to find full path to `lmod` binary (see *Required modules tool*)
- **release 20150430.01** (*Apr 30th 2015*): updated documentation for EasyBuild v2.1.0
  - also cover extensions in page on concepts and terminology (see *Extensions*)
  - add documentation on *Partial installations*, covering `--stop`, `--skip` and `--module-only`
  - add documentation on *Manipulating dependencies*, covering `--filter-deps` and `--hide-deps`
  - document `-module-syntax` configuration option (see *Module files syntax (–module-syntax)*)
  - add note on detection of unknown `$EASYBUILD`-prefixed environment variables (see *Environment variables*)
  - mention support for prepending/appending to `--robot-paths` (see *Prepending and/or appending to the default robot search path*)
  - update release notes for EasyBuild v2.1.0 (see *EasyBuild release notes*)
- **release 20150425.01** (*Apr 25th 2015*):
  - add documentation on *Using external modules*
- **release 20150407.01** (*Apr 7th 2015*):
  - add link to *Unit tests* page in dedicated section at *Installing EasyBuild* page (see install_running_unit_tests)
  - clarify relation between `--installpath`, `--prefix`, `-subdir-*` and `--installpath-*` configuration options (see *Software and modules install path (–installpath, –installpath-software, –installpath-modules)*)
  - mention `--show-default-configfiles` command line option in relevant section (see *Default configuration files*)
- **release 20150327.01** (*Mar 27th 2015*):
  - documented deprecated functionality w.r.t. error reporting (see *error and exception log methods no longer raise an exception*)

- **release 20150316.01** (*Mar 16th 2015*):
  - include list of EasyBuild repositories cloned by `install-EasyBuild-develop.sh` script (see *Installation of latest development version using provided script*)
- **release 20150312.01** (*Mar 12th 2015*):
  - enhance documentation w.r.t. template values in configuration files (see *Templates and constants supported in configuration files*)
  - improve documentation on `--robot` and `--robot-paths` (see *Controlling the robot search path*)
- **release 20150310.01** (*Mar 10th 2015*):
  - document peculiarties w.r.t. dependencies in combination with a `dummy` toolchain (see *Dependencies*)
  - document `clean_gists.py` script (see *clean_gists.py*)
  - mention taking into account of proxy settings for downloading sources (see *Source files, patches and checksums*)
- **release 20150306.03** (*Mar 6th 2015*): add release notes for EasyBuild v2.0.0 (see *EasyBuild release notes*)
- **release 20150306.02** (*Mar 6th 2015*):
  - add documentation on GitHub integration features (see *Integration with GitHub*), mainly `--from-pr` (see from_pr)
  - document locations where (specified) easyconfig files are being searched for (see *Specifying builds*)
- **release 20150306.01** (*Mar 6th 2015*):
  - add documentation on removed functionality (see *Removed functionality*)
  - clean up documentation on deprecated functionality (see *Deprecated functionality*)
  - add documentation on provided scripts, in particular `fix-broken-easyconfigs.py` (see *Useful scripts*)
- **release 20150302.01** (*Mar 2nd 2015*): update/cleanup documentation on *Alternative installation methods*
- **release 20150227.02** (*Feb 27th 2015*): add documentation on the EasyBuild unit test suites, see *Unit tests*
- **release 20150227.01** (*Feb 27th 2015*): enhance documentation w.r.t. to (optional dependencies), see *Installing EasyBuild*
- **release 20150220.01** (*Feb 20th 2015*):
  - document new advanced bootstrapping options: skipping stage 0 and providing source tarballs (see *Installing EasyBuild*)
- **release 20150219.01** (*Feb 19th 2015*): first updates for EasyBuild v2.0.0
  - extend section on (default) EasyBuild configuration files to also cover `$XDG_CONFIG_DIRS` (see configuration_file:)
- **release 20150205.01** (*Feb 5th 2015*): include information on deprecated functionality in (generic) easyblocks (see *Deprecated functionality*)
- **release 20150126.01** (*Jan 26th 2015*):
  - fix `pip` installation prefix option (*Alternative installation methods*)
  - clarify need to have modules tool binary available in `$PATH` (*Installing EasyBuild*)
- **release 20150112.01** (*Jan 12th 2015*): mention need to escape `%` when setting log file format via config file (see *Logfile format (–logfile-format)*)

- **release 20150107.01** (*Jan 7th 2015*): document behaviour of *dummy* toolchain (*dummy toolchain (DEPRECATED)*)
- **release 20141219.01** (*Dec 19th 2014*): add release notes for EasyBuild v1.16.1 (see *EasyBuild release notes*)
- **release 20141218.01** (*Dec 18th 2014*): add release notes for EasyBuild v1.16.0 (see *EasyBuild release notes*)
- **release 20141217.01** (*Dec 17th 2014*): document deprecated functionality in EasyBuild v1.x (*Deprecated functionality*)
- **release 20141204.02** (*Dec 4th 2014*): add EasyBuild release notes (see *EasyBuild release notes*)
- **release 20141204.01** (*Dec 4th 2014*): updates for EasyBuild v1.16.0
  - document details w.r.t. (controlling of) robot search path, incl. `--robot-paths` (*Using the EasyBuild command line*)
  - document use of templates and constants in EasyBuild configuration files (*Configuring EasyBuild*)
  - bump EasyBuild version to 1.16.0
  - changed release number scheme for documentation (based on datestamp)
- **release 1.0.3** (*Dec 3rd 2014*): add page on *Code style*
- **release 1.0.2** (*Nov 6th 2014*): typo and grammar fixes, update Lmod installation instructions for Lmod v5.8
- **release 1.0.1** (*Nov 4th 2014*): fix issues with Changelog
- **release 1.0.0** *(Nov 4th 2014)*: initial release of revamped EasyBuild documentation @ http://easybuild.readthedocs.org, covering basic topics:
  - introductory topics:
    - *What is EasyBuild?*
    - *Concepts and terminology*
    - *Typical workflow example: building and installing WRF*
  - getting started:
    - *Installing EasyBuild*
    - *Configuring EasyBuild*
  - basic usage topics:
    - *Using the EasyBuild command line*
    - *Writing easyconfig files: the basics*
    - *Understanding EasyBuild logs*

11.2 Configuration Legacy

Legacy configuration is currently **deprecated!**

If you are a new user of EasyBuild you can safely ignore everything below this line, refer instead to *Configuring EasyBuild*.

11.2.1 Porting from legacy configuration style

In EasyBuild v1.x, a couple of configuration options, other than the standard ones aligned with variables, are available that follow the **legacy configuration style**, including:

- the `-C` and `--config` command line arguments (**use** `--configfiles` **instead**)
- the `$EASYBUILDCONFIG` environment variable (**use** `$EASYBUILD_CONFIGFILES` **instead**)
- the default path `$HOME/.easybuild/config.py` (**new-style default path is** `$XDG_CONFIG_HOME/easybuild/config.cfg`)
- the legacy fallback path `<installpath>/easybuild/easybuild_config.py` (**only default/fallback path is** `$XDG_CONFIG_HOME/easybuild/config.cfg`)

Likewise, the following legacy environment variables allowed to override selected configuration settings:

- `$EASYBUILDBUILDPATH`: build path to be used by EasyBuild (**use** `$EASYBUILD_BUILDPATH` **instead**)
- `$EASYBUILDINSTALLPATH`: install path to be used by EasyBuild (**use** `$EASYBUILD_INSTALLPATH` **instead**)
- `$EASYBUILDSOURCEPATH`: source path to be used by EasyBuild (**use** `$EASYBUILD_SOURCEPATH` **instead**)
- `$EASYBUILDPREFIX`: build/install/source path prefix to be used (**use** `$EASYBUILD_PREFIX` **instead**)

We *strongly* advise to switch to the new way of configuring EasyBuild as soon as possible, since the legacy configuration style will no longer be supported in EasyBuild v2.x.

11.2.2 How EasyBuild used to be configured in the early days

Configuring *EasyBuild* is done by providing a configuration file.

EasyBuild expects the configuration file to contain valid Python code, because it executes its contents (using `exec`). The rationale is that this approach provides a lot of flexibility for configuring EasyBuild.

EasyBuild will use the file that is provided by the path/filename in the following order of preference:

- path/filename specified on the EasyBuild command line (using `--config`),
- path/filename obtained from the environment variable `EASYBUILDCONFIG` (if it is defined)
- `$HOME/.easybuild/config.py` (as of EasyBuild v1.1)
- the (default) configuration file at `<path where EasyBuild was installed>/easybuild/easybuild_config.py`

Configuration variables

The configuration file must define the following five variables: `build_path`, `install_path`, `source_path`, `repository`, and `log_format`. If one of them is not defined, EasyBuild will complain and exit.

Build path (required)

The `build_path` variable specifies the directory in which EasyBuild builds its software packages.

Each software package is (by default) built in a subdirectory of the `build_path` under `<name>/<version>/<toolchain><versionsuffix>`.

Note that the build directories are emptied by EasyBuild when the installation is completed (by default).

Install path (required)

The `install_path` variable specifies the directory in which EasyBuild installs software packages and the corresponding module files.

The packages themselves are installed under `install_path/software` in their own subdirectory aptly named `<name>/<version>-<toolchain><versionsuffix>` (by default), where name is the package name. The corresponding module files are installed under `install_path/modules`.

Setting $MODULEPATH

After the configuration, you need to make sure that `$MODULEPATH` environment variable is extended with the `modules/all` subdirectory of the `install_path`, i.e.:

```
export MODULEPATH=<install_path>/modules/all:$MODULEPATH
```

It is probably a good idea to add this to your (favourite) shell .rc file, e.g., `.bashrc`, and/or the `.profile` login scripts, so you do not need to adjust the `$MODULEPATH` variable every time you start a new session.

Source path (required)

The `source_path` variable specifies the directory in which EasyBuild looks for software source and install files.

Similarly to the configuration file lookup, EasyBuild looks for the installation files as given by the `sources` variable in the .eb easyconfig file, in the following order of preference:

- `<source_path>/<name>`: a subdirectory determined by the name of the software package
- `<source_path>/<letter>/<name>`: in the style of the `easyblocks`/`easyconfigs` directories: in a subdirectory determined by the first letter (in lower case) of the software package and by its full `name`
- `<source_path>`: directly in the source path

Note that these locations are also used when EasyBuild looks for patch files in addition to the various `easybuild/easyconfigs` directories that are listed in the `$PYTHONPATH`.

(replaced by All available easyconfig parameters, –avail-easyconfig-params / -a)

11.3 Available easyconfig parameters for EB_WRF

Overview of easyconfig parameters, including those specific to the easyblock for WRF (indicated with `(*)`):

```
$ eb -a --easyblock EB_WRF

Available easyconfig parameters (* indicates specific for the EB_WRF EasyBlock)
MANDATORY
---------
buildtype(*):          Specify the type of build (serial, smpar (OpenMP), dmpar (MPI),␣
↪dm+sm (hybrid OpenMP/MPI)). (default: None)
description:           A short description of the software (default: None)
docurls:               List of urls with documentation of the software (not␣
↪necessarily on homepage) (default: None)
homepage:              The homepage of the software (default: None)
```

(continues on next page)

(continued from previous page)

```
name:                  Name of software (default: None)
software_license:      Software license (default: None)
software_license_urls:         List of software license locations (default: None)
toolchain:             Name and version of toolchain (default: None)
version:               Version of software (default: None)

EASYBLOCK-SPECIFIC
------------------
rewriteopts(*):                Replace -O3 with CFLAGS/FFLAGS (default: True)
runtest(*):            Build and run WRF tests (default: True)

TOOLCHAIN
---------
onlytcmod:             Boolean/string to indicate if the toolchain should only load␣
↪the environment with module (True) or also set all other variables (False) like␣
↪compiler CC etc (if string: comma separated list of variables that will be ignored).
↪ (default: False)
toolchainopts:                 Extra options for compilers (default: None)

BUILD
-----
buildopts:             Extra options passed to make step (default already has -j X)␣
↪(default: )
checksums:             Checksums for sources and patches (default: [])
configopts:            Extra options passed to configure (default already has --
↪prefix) (default: )
easyblock:             EasyBlock to use for building (default: ConfigureMake)
easybuild_version:     EasyBuild-version this spec-file was written for (default: None)
installopts:           Extra options for installation (default: )
maxparallel:           Max degree of parallelism (default: None)
parallel:              Degree of parallelism for e.g. make (default: based on the␣
↪number of cores, active cpuset and restrictions in ulimit) (default: None)
patches:               List of patches to apply (default: [])
postinstallcmds:       Commands to run after the install step. (default: [])
prebuildopts:          Extra options pre-passed to build command. (default: )
preconfigopts:                 Extra options pre-passed to configure. (default: )
preinstallopts:                Extra prefix options for installation. (default: )
runtest(*):            Indicates if a test should be run after make; should specify␣
↪argument after make (for e.g.,"test" for make test) (default: None)
sanity_check_commands:         format: [(name, options)] e.g. [('gzip','-h')]. Using a␣
↪non-tuple is equivalent to (name, '-h') (default: [])
sanity_check_paths:    List of files and directories to check (format: {'files':<list>,
↪ 'dirs':<list>}) (default: {})
skip:                  Skip existing software (default: False)
skipsteps:             Skip these steps (default: [])
source_urls:           List of URLs for source files (default: [])
sources:               List of source files (default: [])
stop:                  Keyword to halt the build process after a certain step.␣
↪(default: None)
tests:                         List of test-scripts to run after install. A test␣
↪script should return a non-zero exit status to fail (default: [])
unpack_options:                Extra options for unpacking source (default: None)
unwanted_env_vars:     List of environment variables that shouldn't be set during␣
↪build (default: [])
versionprefix:                 Additional prefix for software version (placed before␣
↪version and toolchain name) (default: )
versionsuffix:                 Additional suffix for software version (placed after␣
↪toolchain name) (default: )
```

(continues on next page)

(continued from previous page)

```
FILE-MANAGEMENT
---------------
buildininstalldir:    Boolean to build (True) or not build (False) in the␣
↪installation directory (default: False)
cleanupoldbuild:      Boolean to remove (True) or backup (False) the previous build␣
↪directory with identical name or not. (default: True)
cleanupoldinstall:    Boolean to remove (True) or backup (False) the previous install␣
↪directory with identical name or not. (default: True)
dontcreateinstalldir: Boolean to create (False) or not create (True) the install␣
↪directory (default: False)
keeppreviousinstall:  Boolean to keep the previous installation with identical name.␣
↪Experts only! (default: False)
keepsymlinks:         Boolean to determine whether symlinks are to be kept during␣
↪copying or if the content of the files pointed to should be copied (default: False)
start_dir:            Path to start the make in. If the path is absolute, use that␣
↪path. If not, this is added to the guessed path. (default: None)

DEPENDENCIES
------------
allow_system_deps:    Allow listed system dependencies (format: (<name>, <version>))␣
↪(default: [])
builddependencies:    List of build dependencies (default: [])
dependencies:         List of dependencies (default: [])
hiddendependencies:   List of dependencies available as hidden modules (default: [])
osdependencies:               OS dependencies that should be present on the system␣
↪(default: [])

LICENSE
-------
group:                        Name of the user group for which the software should be␣
↪available (default: None)
key:                  Key for installing software (default: None)
license_file:         License file for software (default: None)
license_server:               License server for software (default: None)
license_server_port:  Port for license server (default: None)

EXTENSIONS
----------
exts_classmap:                Map of extension name to class for handling build and␣
↪installation. (default: {})
exts_defaultclass:    List of module for and name of the default extension class␣
↪(default: None)
exts_filter:          Extension filter details: template for cmd and input to cmd␣
↪(templates for name, version and src). (default: None)
exts_list:            List with extensions added to the base installation (default:␣
↪[])

MODULES
-------
include_modpath_extensions:Include $MODULEPATH extensions specified by module naming␣
↪scheme. (default: True)
modaliases:           Aliases to be defined in module file (default: {})
modextrapaths:                Extra paths to be prepended in module file (default: {})
modextravars:         Extra environment variables to be added to module file␣
↪(default: {})
modloadmsg:           Message that should be printed when generated module is loaded␣
↪(default: {})
```

(continues on next page)

(continued from previous page)

```
modtclfooter:         Footer to include in generated module file (Tcl syntax)␣
↪(default: )
moduleclass:          Module class to be used for this software (default: base)
moduleforceunload:    Force unload of all modules when loading the extension␣
↪(default: False)
moduleloadnoconflict: Don't check for conflicts, unload other versions instead ␣
↪(default: False)

OTHER
-----
buildstats:           A list of dicts with build statistics (default: None)
```

(replaced by eb_help *)*

11.4 List of easyblocks

The list of available easyblocks can easily be obtained as follows:

```
$ eb --list-easyblocks

EasyBlock
|-- Binary
|   |-- EB_ABAQUS
|   |-- EB_Allinea
|   |-- EB_CPLEX
|   |-- EB_CUDA
|   |-- EB_EPD
|   |-- PackedBinary
|   |   |-- EB_Java
|   |   |-- EB_Tornado
|   |-- EB_Mathematica
|   |-- Rpm
|   |   |-- EB_QLogicMPI
|   |-- JAR
|-- EB_ACML
|-- EB_ALADIN
|-- EB_ANSYS
|-- EB_ant
|-- ConfigureMake
|   |-- EB_ARB
|   |-- CMakeMake
|   |   |-- EB_Armadillo
|   |   |-- EB_BamTools
|   |   |-- EB_CGAL
|   |   |-- EB_Clang
|   |   |-- CMakePythonPackage
|   |   |   |-- EB_DOLFIN
|   |   |   |-- EB_UFC
|   |   |-- EB_Geant4
|   |   |-- EB_GROMACS
|   |   |-- EB_netCDF
|   |   |-- EB_OpenBabel
|   |   |-- EB_Trilinos
|   |-- EB_ATLAS
```

(continues on next page)

(continued from previous page)

```
|    |-- MakeCp
|    |    |-- EB_BamTools
|    |    |-- EB_BLAT
|    |    |-- EB_NAMD
|    |    |-- CmdCp
|    |-- EB_R
|    |-- EB_BLACS
|    |-- EB_Bowtie
|    |-- EB_Bowtie2
|    |-- EB_BWA
|    |-- EB_bzip2
|    |-- EB_CBLAS
|    |-- EB_Chapel
|    |-- EB_Cufflinks
|    |-- EB_DB
|    |-- EB_DL_underscore_POLY_underscore_Classic
|    |-- EB_Python
|    |-- EB_Doxygen
|    |-- EB_ESMF
|    |-- EB_ESPResSo
|    |-- EB_Ferret
|    |-- EB_flex
|    |-- EB_freetype
|    |-- EB_g2clib
|    |-- EB_g2lib
|    |-- EB_GCC
|    |-- EB_GHC
|    |-- EB_Go
|    |-- EB_HDF5
|    |-- EB_HPCG
|    |-- EB_HPL
|    |-- EB_Hypre
|    |-- EB_LAPACK
|    |-- EB_libint2
|    |-- EB_libxml2
|    |-- EB_MetaVelvet
|    |-- EB_METIS
|    |-- EB_Mothur
|    |-- EB_MrBayes
|    |-- EB_Perl
|    |-- EB_MUMmer
|    |-- EB_MUMPS
|    |-- EB_MVAPICH2
|    |-- EB_ncurses
|    |-- EB_netCDF_minus_Fortran
|    |-- EB_NEURON
|    |-- EB_NWChem
|    |-- EB_OpenSSL
|    |-- EB_Pasha
|    |-- EB_PETSc
|    |-- EB_Primer3
|    |-- EB_PSI
|    |-- EB_Qt
|    |-- EB_QuantumESPRESSO
|    |-- EB_ROOT
|    |-- EB_SAMtools
|    |-- EB_ScaLAPACK
```

(continues on next page)

(continued from previous page)

```
|   |-- EB_Scalasca1
|   |-- EB_Score_minus_P
|   |-- EB_SHRiMP
|   |-- EB_SLEPc
|   |-- EB_SOAPdenovo
|   |-- EB_SuiteSparse
|   |-- EB_SWIG
|   |-- EB_Velvet
|   |-- EB_XCrySDen
|   |-- PerlModule
|-- ExtensionEasyBlock
|   |-- RPackage
|   |   |-- EB_Bioconductor
|   |   |-- EB_Rmpi
|   |   |-- EB_Rserve
|   |   |-- EB_XML
|   |-- PythonPackage
|   |   |-- CMakePythonPackage
|   |   |   |-- EB_DOLFIN
|   |   |   |-- EB_UFC
|   |   |-- EB_EasyBuildMeta
|   |   |-- EB_libxml2
|   |   |-- EB_netcdf4_minus_python
|   |   |-- EB_nose
|   |   |-- FortranPythonPackage
|   |   |   |-- EB_numpy
|   |   |   |-- EB_scipy
|   |   |-- EB_PyQuante
|   |   |-- EB_python_minus_meep
|   |   |-- EB_PyZMQ
|   |   |-- EB_VSC_minus_tools
|   |   |-- VersionIndependentPythonPackage
|   |   |   |-- VersionIndependendPythonPackage
|   |   |   |   |-- VSCPythonPackage
|   |-- PerlModule
|-- EB_BiSearch
|-- EB_Boost
|-- EB_CHARMM
|-- EB_CP2K
|-- EB_Eigen
|-- EB_FDTD_underscore_Solutions
|-- EB_FLUENT
|-- Tarball
|   |-- EB_FoldX
|   |-- EB_FreeSurfer
|   |-- EB_MTL4
|   |-- BinariesTarball
|-- EB_FSL
|-- EB_GenomeAnalysisTK
|-- IntelBase
|   |-- EB_icc
|   |   |-- EB_ifort
|   |-- EB_ifort
|   |-- EB_imkl
|   |-- EB_impi
|   |-- EB_Inspector
|   |-- EB_ipp
```

(continues on next page)

(continued from previous page)

```
|   |-- EB_itac
|   |-- EB_tbb
|   |-- EB_VTune
|-- PackedBinary
|   |-- EB_Java
|   |-- EB_Tornado
|-- EB_libsmm
|-- EB_Maple
|-- EB_MATLAB
|-- EB_Modeller
|-- EB_NCL
|-- EB_OpenFOAM
|-- EB_OpenIFS
|-- EB_ParMETIS
|-- EB_picard
|-- EB_Rosetta
|-- EB_SCOTCH
|-- EB_TotalView
|-- EB_Trinity
|-- EB_WIEN2k
|-- EB_WPS
|-- EB_WRF
|-- Toolchain

Extension
|-- ExtensionEasyBlock
|   |-- RPackage
|   |   |-- EB_Bioconductor
|   |   |-- EB_Rmpi
|   |   |-- EB_Rserve
|   |   |-- EB_XML
|   |-- PythonPackage
|   |   |-- CMakePythonPackage
|   |   |   |-- EB_DOLFIN
|   |   |   |-- EB_UFC
|   |   |-- EB_EasyBuildMeta
|   |   |-- EB_libxml2
|   |   |-- EB_netcdf4_minus_python
|   |   |-- EB_nose
|   |   |-- FortranPythonPackage
|   |   |   |-- EB_numpy
|   |   |   |-- EB_scipy
|   |   |-- EB_PyQuante
|   |   |-- EB_python_minus_meep
|   |   |-- EB_PyZMQ
|   |   |-- EB_VSC_minus_tools
|   |   |-- VersionIndependentPythonPackage
|   |   |   |-- VersionIndependendPythonPackage
|   |   |   |   |-- VSCPythonPackage
|   |-- PerlModule
```

11.5 List of known toolchains

The list of known toolchains can easily be obtained with:

```
$ eb --list-toolchains
List of known toolchains (toolchainname: module[,module...]):
[...]
```

An up-to-date overview of known toolchains is available at vsd_list_toolchains.

Note: The *system* toolchain is a special case, see system_toolchain.

11.6 Alternative installation methods

We warmly recommend installing EasyBuild via the bootstrap procedure, see bootstrapping.

This page describes the alternative installation methods:

- *Standard installation of latest release*
- *Installation from downloaded sources*
- *Installation of latest release from GitHub*
- *Installation of latest development version*

Do take into account the list of (required) dependencies (see *Dependencies*).

11.6.1 Standard installation of latest release

Usually, you just want to install the latest (stable) version of each of the EasyBuild packages (framework, easyblocks, easyconfigs).

Python provides a couple of ways to do that. Every version of the EasyBuild packages is released via PyPi.

Installing EasyBuild without admin rights

If you do not have EasyBuild installed yet, or if you just want to install the most recent version of each of the EasyBuild packages, you can use one of the following simple commands:

- using `easy_install` (old tool, but still works):

```
easy_install --prefix $HOME/EasyBuild easybuild
```

Note: If you already have *easybuild* installed, you may need to instruct `easy_install` to install a newer version, using `--upgrade` or `-U`.

- using `pip` (more recent and better installation tool for Python software):

```
pip install --install-option "--prefix=$HOME/EasyBuild" easybuild
```

The `--prefix $HOME/EasyBuild` part in these commands allows you to install EasyBuild without admin rights into `$HOME/EasyBuild`.

Note: For pip v8.0 and newer, `pip install --prefix=$HOME/EasyBuild easybuild` works too.

Adjusting `$PATH` and `$PYTHONPATH` environment variables

After installing EasyBuild with either `easy_install` or `pip` like this, you will need to update the `$PATH` and `$PYTHONPATH` environment variable to make sure the system can find the main EasyBuild command `eb`. On (most) Linux distributions, the command for doing this is:

```
export PATH=$HOME/EasyBuild/bin:$PATH
export PYTHONPATH=$HOME/EasyBuild/lib/python2.7/site-packages:$PYTHONPATH
```

Tip: To determine the path that should be added to the `$PYTHONPATH` environment variable for a given installation prefix, you can use the following command:

```
python -c "import distutils.sysconfig; print distutils.sysconfig.get_python_
↪lib(prefix='$HOME/EasyBuild/');"
```

Install with admin rights

If you do have admin rights on the system where you want to install EasyBuild, you can simply omit the `--prefix $HOME/EasyBuild/` to have EasyBuild installed system-wide. In that case, you do not need to touch the `$PATH` or `$PYTHONPATH` environment variables since the `eb` command will be installed in one of the default paths.

Alternatives to –prefix

As an alternative to `--prefix` when you do not have admin rights, you can specify that EasyBuild should be installed in your `$HOME` directory using the `--user` option.

The full list of commands to install EasyBuild in your `$HOME` directory using `pip` would be:

```
pip install --user easybuild
export PATH=$HOME/.local/bin:$PATH
```

Warning: In our experience, using `--user` creates more problems than it solves. We have run into unexpected behavior with Python software installed in your home directory using `--user`, for example it **always** being preferred over versions installed somewhere else. Hence, we strongly discourage using `--user` to install EasyBuild (or other Python software).

Installing the EasyBuild packages separately

Each of the EasyBuild packages can also be installed separately:

```
pip install --install-option "--prefix=$HOME/EasyBuild" easybuild-framework
pip install --install-option "--prefix=$HOME/EasyBuild" easybuild-easyblocks
pip install --install-option "--prefix=$HOME/EasyBuild" easybuild-easyconfigs
```

This is the exact same sequence of steps as they will be performed when running `pip install --install-option "--prefix=$HOME/EasyBuild" easybuild`.

11.6.2 Installation from downloaded sources

To install one of the EasyBuild packages from a downloaded source tarball, use the following steps:

```
tar xfvz easybuild-framework-1.0.tar.gz
cd easybuild-framework-1.0
pip install --install-option "--prefix=$HOME/EasyBuild" .
```

Do note that when an EasyBuild package is being installed without having the EasyBuild packages that it depends upon available, both `easy_install` and `pip` will try and pull in the latest available version of those packages from PyPi.

Thus, to have full control over the EasyBuild installation, you need to respect the following installation order: easybuild-framework, easybuild-easyblocks, easybuild-easyconfigs. The easyblocks package depends on the framework package; the easyconfigs package depends on both the framework and easyblocks packages.

If you do not have `pip` or `easy_install` available, you can also fall back to using the `setup.py` script directly:

```
python setup.py --prefix $HOME/EasyBuild install
```

11.6.3 Installation of latest release from GitHub

To install the latest (stable) release of an EasyBuild package directly from GitHub, use the following command:

```
pip install --install-option "--prefix=$HOME/EasyBuild" https://github.com/
↪easybuilders/easybuild-framework/archive/main.tar.gz
```

Again, the order in which the EasyBuild packages are installed is important to have full control over the installation process, see previous section.

11.6.4 Installation of latest development version

To install the latest development version of an EasyBuild package from GitHub, you can simply adjust the command from the previous section to install from the `develop` branch (or any of the available feature branches in any EasyBuild repository for that matter):

```
pip install --install-option "--prefix=$HOME/EasyBuild" https://github.com/
↪easybuilders/easybuild-framework/archive/develop.tar.gz
```

Note: You should use this only if you are interested in developing for EasyBuild. Although it is well tested, the development version of the EasyBuild repositories may be unstable at a given point in time.

11.6.5 Installation of latest development version using provided script

After you have forked each of the EasyBuild repositories on GitHub (+ vsc-base), you can set up a development version of EasyBuild using the `install-EasyBuild-develop.sh` script.

This script will clone the different EasyBuild repositories from GitHub:

- easybuild: EasyBuild metapackage & documentation sources for http://easybuild.readthedocs.org
- vsc-base: dependency for EasyBuild framework (logging, command line interface, . . .)
- easybuild-framework: EasyBuild framework
- easybuild-easyblocks: collection of easyblocks
- easybuild-easyconfigs: collection of easyconfig files
- easybuild-wiki: EasyBuild wiki pages

It can be used as follows:

```
# pick an installation prefix (adjust as you like)
INSTALL_PREFIX=$(mktemp -d $HOME/EasyBuild-XXXXXX)
# download script
curl -O https://raw.githubusercontent.com/easybuilders/easybuild-framework/main/
↪easybuild/scripts/install-EasyBuild-develop.sh
# run downloaded script, specifying *your* GitHub username and the installation prefix
bash install-EasyBuild-develop.sh GITHUB_USERNAME $INSTALL_PREFIX
# update $MODULEPATH via 'module use', and load the module
module use $INSTALL_PREFIX/modules
module load EasyBuild-develop
eb --version  ## This should ensure you have a reasonable instance of EasyBuild
```

Note: The above creates a module file which you can load/inspect at will. The interesting aspect about it is that it is pointing to an EasyBuild installation directly on local git repositories, which allows you to customise it easily. Remember to commit/push or otherwise save your changes, if you intend to use them later.

11.7 Installing environment modules without root permissions

This short guide will explain how to install the standard environment modules Tcl/C software package without root permissions on a Linux or Mac OS X system, together with Tcl on which it depends.

11.7.1 Tcl

1. Go to http://www.tcl.tk and download the latest Tcl sources. At the time of writing, the latest available Tcl version was 8.5.15, which can be downloaded from here. The remainder of these commands will assume Tcl v8.5.15 is being installed, you may need to adjust them accordingly. **Note**: Stick to Tcl v8.5.x, don't consider using the more recent v8.6.x or higher, since the environment modules package is not compatible with those Tcl versions.
2. Unpack the Tcl source tarball:

```
tar xfvz tcl8.5.15-src.tar.gz
```

3. Pick a location where you will install Tcl. It should be a directory you have write permissions on. My suggestion would be to use something like `$HOME/.local/Tcl`.
4. Go to the `unix` subdirectory of the unpacked Tcl directory, and run the `configure` script using the `--prefix` option:

```
cd tcl8.5.15/unix
./configure --prefix=$HOME/.local/Tcl
```

   If you're building Tcl and environment modules on Mac, you should run `configure` in the `tcl8.5.15/macosx` directory instead.

5. Next, build Tcl using the `make` command. If the system you are building on has multiple cores, running make in parallel will speed up the build. Just use the `-j` option, and pass it a degree of parallelism (just use the number of cores your system has available), e.g.:

```
make -j 4
```

6. The final step consists of installing Tcl to the directory specified in step 4. To do this, simply run:

```
make install
```

All done! Now you are ready to build the environment modules package, which requires Tcl.

11.7.2 Environment Modules

1. Download the latest source tarball for the environment modules tools from http://modules.sourceforge.net/. At the time of writing, the latest available version is 3.2.10 which can be downloaded from here.
2. Unpack the downloaded source tarball:

```
tar xfvz modules-3.2.10.tar.gz
```

3. Configure the build, again use `--prefix` to specify where to install the environment modules tool in the end. If you needed to install Tcl by hand as outlined in the previous section, you'll also need to specify where it was installed using the `--with-tcl` option:

```
cd modules-3.2.10
./configure --prefix=$HOME/.local/environment-modules --with-tcl=$HOME/.local/Tcl/
↪lib
```

4. Build with `make`, consider parallel build if your system is recent enough:

```
make -j 4
```

5. Install:

```
make install
```

Alright, now just one more thing...

11.7.3 Set up your environment

Because you've installed environment modules and Tcl in a non-default location, you need to make sure your environment is setup up correctly to use them.

To make a long story short, these are the commands you need to execute:

```
export PATH=$HOME/.local/environment-modules/Modules/3.2.10/bin:$PATH
export LD_LIBRARY_PATH=$HOME/.local/Tcl/lib:$LD_LIBRARY_PATH
# adjust line below if you're using a shell other than bash, check with 'echo $SHELL'
source $HOME/.local/environment-modules/Modules/3.2.10/init/bash
```

Tip: Add these three lines in your `.bashrc` file, that way they'll be executed every time you log in.

11.8 Installing Lmod without root permissions

This short guide will show how to install Lmod (and Lua, on which it depends) on Linux, without requiring root permissions.

11.8.1 Lua

Build and install Lua using the source tarball available in the Lmod SourceForge repository (http://sourceforge.net/projects/lmod/files/). This version is a lot easier to build, and already includes the required extra Lua modules. At the time of writing this relates to the `lua-5.1.4.8.tar.gz` tarball.

Step 1: Download and unpack lua-5.1.4.8.tar.gz.

Step 2: Configure the Lua build, provide a custom installation prefix (e.g. `$HOME/lua`) and specify to statically link libraries (i.e. `libreadline` and `ncurses`), to avoid problems when modules that provide these libraries are being loaded. Then build and install via `make`:

```
./configure --with-static=yes --prefix=$HOME/lua && make && make install
```

Step 3: Make sure the `lua` binary is available in your `$PATH` (only required when building Lmod, see below):

```
export PATH=$HOME/lua/bin:$PATH
```

Optionally, check whether the `lua` binary indeed doesn't link to any unexpected `readline` or `ncurses` libraries:

```
$ ldd $HOME/lua/bin/lua
        linux-vdso.so.1 (0x00007fffad7ff000)
        libm.so.6 => /lib64/libm.so.6 (0x00007ff9914db000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00007ff9912d7000)
        libc.so.6 => /lib64/libc.so.6 (0x00007ff990f2a000)
        /lib64/ld-linux-x86-64.so.2 (0x00007ff9917d9000)
```

11.8.2 Lmod

Step 1: Download and unpack the latest available Lmod version, Lmod-6.1.tar.bz2 at the time of writing.

```
tar xfvj Lmod-6.1.tar.bz2 && cd Lmod-6.1
```

Step 2: Configure, build and install Lmod build, in a custom prefix:

```
./configure --prefix=$HOME && make install
```

Step 3: Update `$PATH` so `lmod` is available (put this in your `.bashrc`):

```
export PATH=$HOME/lmod/6.1/libexec:$PATH
```

Optionally, give it a spin:

```
$ lmod --version

Modules based on Lua: Version 6.1  2016-02-05 16:31
    by Robert McLay mclay@tacc.utexas.edu
```

Step 4: Define `module` function to use `lmod` (optional for use with EasyBuild):

```
source $HOME/lmod/6.1/init/bash
export LMOD_CMD=$HOME/lmod/6.1/libexec/lmod
```

11.9 Useful links

Tools used to convert from other wiki formats and prepare .rst when in pre-production mode:

- http://johnmacfarlane.net/pandoc/try/?text=%0A&from=markdown_github&to=rst
- http://rst.ninjs.org

11.10 Sphinx and Read the Docs reference documents online

- https://github.com/rtfd/readthedocs.org
- http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html
- http://matplotlib.org/sampledoc/cheatsheet.html
- http://openalea.gforge.inria.fr/doc/openalea/doc/_build/html/source/sphinx/rest_syntax.html
- http://thomas-cokelaer.info/tutorials/sphinx/rest_syntax.html
- http://docutils.sourceforge.net/docs/user/rst/quickref.html
- http://nipy.org/devel/guidelines/sphinx_helpers.html
- https://pythonhosted.org/an_example_pypi_project/sphinx.html
- http://openmdao.org/docs/documenting/sphinx.html

11.11 Third party Sphinx examples, themes and possible extensions

- http://sphinx.readthedocs.org/en/latest/examples.html
- http://django.readthedocs.org/en/latest/
- http://docs.cython.org/index.html
- http://docs.writethedocs.org/tools/sphinx-themes/
- http://ryan-roemer.github.io/sphinx-bootstrap-theme/examples.html
- http://django-profiletools.readthedocs.org/en/latest/
- http://pythonhosted.org/korean/

- https://pythonhosted.org/cloud_sptheme/cloud_theme.html#features
- http://openmdao.org/docs/documenting/sphinx.html
- http://sphinx-doc.org/genindex.html

11.12 EasyBuild release notes

The latest version of EasyBuild provides support for building and installing **2,467** different software packages, including 36 different (compiler) toolchains. It contains 241 software-specific easyblocks and 37 generic easyblocks, alongside 13,265 easyconfig files.

Note: See also the concise overview of major changes in EasyBuild v4.0.

11.12.1 EasyBuild v4.4.2 (September 7th 2021)

bugfix/update release

framework

- various enhancements, including:
  - add per-extension timing in output produced by `eb` command (#3734)
  - add definition for new toolchain `nvpsmpic` (NVHPC + ParaStationMPI + CUDA) (#3736)
  - include list of missing libraries in warning about missing FFTW libraries in `imkl` toolchain component (#3776)
  - check for recursive symlinks by default before copying a folder (#3784)
  - add `--filter-ecs` configuration option to filter out easyconfigs from set of easyconfigs to install (#3796)
  - check type of `source_tmpl` value for extensions, ensure it's a string value (not a list) (#3799)
  - also define `$BLAS_SHARED_LIBS` & co in build environment (analogous to `$BLAS_STATIC_LIBS`) (#3800)
  - report use of `--ignore-test-failure` in success message in output (#3806)
  - add `get_cuda_cc_template_value` method to `EasyConfig` class (#3807)
  - add support for `fix_bash_shebang_for` (#3808)
  - pick up `$MODULES_CMD` to facilitate using Environment Modules 4.x as modules tool (#3816)
  - use more sensible branch name for creating easyblocks PR with `--new-pr` (#3817)
- various bug fixes, including:
  - remove Python 2.6 from list of supported Python versions in ‘‘setup.py ‘‘(#3767)
  - don't add directory that doesn't include any files to `$PATH` or `$LD_LIBRARY_PATH` (#3769)
  - make logdir writable also when `--stop`/`--fetch` is used and `--read-only-installdir` is enabled (#3771)
  - fix forgotten renaming of `l` to `char` in `__init__.py` that is created for included Python modules (#3773)

  - fix `verify_imports` by deleting all imported modules before re-importing them one by one (#3780)
  - fix `ignore_test_failure` not set for Extension instances (#3782)
  - update `iompi` toolchain to `intel-compiler` subtoolchain for oneAPI versions (>= `iompi` 2020.12) (#3785)
  - don't parse patch files as easyconfigs when searching for where patch file is used (#3786)
  - make sure git clone with a tag argument actually downloads a tag (#3795)
  - fix CI by excluding GC3Pie 2.6.7 (which is broken with Python 2) and improve error reporting for option parsing (#3798)
  - correctly resolve templates for patches in extensions when uploading to GitHub (#3805)
  - add `--easystack` to ignored options when submitting job (#3813)
- other changes:
  - speed up tests by caching checked paths in `set_tmpdir` + less test cases for `test_compiler_dependent_optarch` (#3802)
  - speed up `set_parallel` method in `EasyBlock` class (#3812)

easyblocks

- 4 new software-specific easyblocks:
  - CRISPR-DAV (#2487), jaxlib (#2545), ORCA (#2504), RepeatModeler (#2470)
- minor enhancements, including:
  - update ABAQUS easyblock to support installation of v2020 (#2034)
  - enable make check and sanity check exec for OpenMPI (#2444)
  - fixed sanity check for later versions of Molpro (#2499)
  - add Ampere to known GPU architectures for Kokkos in LAMMPS easyblock (#2510)
  - update sanity check in ESMF easyblock to take into account new binary names from version 8.1.0 onwards (#2512)
  - use `report_test_failure` in TensorFlow easyblock (#2519)
  - add support for imkl 2021.x in easyblocks for
    - Amber (#2527), BerkeleyGW (#2528), CP2K (#2529), FreeFEM (#2530), GROMACS (#2531), numexpr (#2532)
  - add `install_src` option and enhance `buildcmd` option in PythonPackage easyblock (#2534)
  - make sure AOCC commands like clang/flang pick up `GCCcore` as GCC toolchain (#2538)
  - create `CMakeConfig` files for tbb >= 2020 (#2539)
  - find source dir for libunwind when building ldd for Clang versions >= 12.0.1 (#2540)
  - fix motorBike test in sanity check for OpenFOAM 9 (#2544)
  - generate and install pkg-config files for OpenSSL wrapper (#2549)
  - don't run `setup.py config` for numpy >= 1.21, since it's no longer supported (#2554)
  - update WIEK2k easyblock to handle new Intel versions, WIEN2k v21, Perl dependency, and tuning of dimension parameters (#2558)
  - remove PCRE from list of system libraries for TensorFlow 2.6.0 (#2559)

  - add support for also installing Tosca component in ABAQUS (#2560)
  - update ABAQUS easyblock to support installing of version 2021 with hot fixes (#2562)
  - update sanity check in VTune easyblock for 2021.x versions (#2563)
- various bug fixes, including:
  - handle failure of running `nvidia-smi` in TensorFlow tests (#2506)
  - remove Python 2.6 from list of supported Python versions in `setup.py` (#2507)
  - clean up installation of Tkinter (#2509)
  - remove redundant `$CPATH` entry of tbb modules + fix `--module-only` (#2511)
  - honor `--ignore-test-failure` in PythonPackage (#2516)
  - use correct GTK+ version for OpenCV based on dependencies (#2520)
  - ensure Python prep is done in sanity check + support debug builds in TensorFlow easyblock (#2522)
  - correct cleanup of single-letter local variable in `__init__` of `easybuild.easyblocks` (#2524)
  - reset runtest to `None` in CMakePythonPackage (#2536)
  - set `$I_MPI_ROOT` correctly for 2021.x in impi easyblock (#2537)
  - let Bundle easyblock pick up custom easyblock for components based on name if no easyblock is specified explicitly (#2543, #2547)
  - explicitly use only OpenBLAS for PyTorch if MKL is not used (#2448)
  - keep symlinks in cuDNN installations (#2550)
  - use `${BLAS,LAPACK}_SHARED_LIBS` in GROMACS easyblock for FlexiBLAS-based toolchains (#2552)
  - copy the list to avoid changing the original lists in `DEFAULT_TARGETS_MAP` for Clang and LLVM (#2556)
  - fix `pathsep join` bug that breaks installation of CUDA versions < 10.1 if `$PERL5LIB` is defined (#2561)
- other changes:
  - simplify Boost easyblock (#2513)
  - remove deprecated options from PythonPackage (#2535)

easyconfigs

- added easyconfig for foss/2021.07 (candidate for foss/2021b) (#13788) and iomkl/2021a (#13521) toolchains
- added example easyconfig files for 79 new software packages:
  - AlphaFold (#13867), andi (#13768), aria2 (#13709), AUTO-07p (#13831), babl (#13216), BAli-Phy (#13484), biogeme (#13735), Bio-SearchIO-hmmer (#13419), BuDDy (#13907), c-ares (#13709), CellRank (#13408), Coin (#13818), Co-phylog (#13769), cppy (#13823), CREST (#13648), CRISPR-DAV (#13139), CSBLAST (#13794), DGL (#13793), DistributedStream (#13659), Excel-Writer-XLSX (#13139), FBPIC (#13500), freebayes (#12859), freud-analysis (#13354), fumi_tools (#13517), GEGL (#13216), GIMP (#13216), glib-networking (#13216), GOATOOLS (#13364, #13629), gofasta (#13651), GTK2 (#13900), GTK3 (#13900), Highway (#13453), hipSYCL (#13837), ISA-L (#13663), jax (#13760), JSON-GLib (#13216), Kalign (#13463), LADR (#13351), lDDT (#13794), libdivsufsort (#13768), libjxl (#13453, #13829), libmypaint (#13216), libtecla (#13908), lie_learn (#13793), longestrunsubsequence (#13800), Maude (#13909), mimalloc (#13726), MINPACK (#13802), Mish-Cuda (#13759, #13784), muparserx (#13779), nlohmann_json (#13777), OpenStackClient (#13644), OSPRay (#12852), pandapower

(#13687), pangolin (#13733, #13848), PSIPRED (#13794), PYPOWER (#13689), PyPSA (#13673), PyRosetta (#13793), pysamstats (#13139), pytest-benchmark (#13622), python-isal (#13663), QCxMS (#13650), RepeatModeler (#13126), Schrodinger (#11698), sklearn-som (#13699), SoQt (#13818), spdlog (#13778), SpectrA (#13743), SymEngine-python (#13652), SyRI (#13800), torchinfo (#13662), TRAVIS-Analyzer (#13503), UShER (#13708), Vampire (#13352), VBZ-Compression (#13536), VisPy (#13468), wgsim (#13475), Yices (#13906)

- added additional easyconfigs for various supported software packages, including:
  - ABAQUS 2020.eb, ANTs 2.3.5, AOCC 3.1.0, Arb 2.19.0, archspec 0.1.2, Armadillo 10.5.3, ASE 3.22.0, Autotools 20210726, BamTools 2.5.2, bgen 4.1.3, Biopython 1.79, bitarray 1.5.3, Boost 1.76.0, Bowtie2 2.4.4, bpytop 1.0.67, Cartopy 0.19.0.post1, Centrifuge 1.0.4, Clang 12.0.1, CMake 3.21.1, CRISPResso2 2.2.1, CUDA 11.4.1, cuDNN 8.2.2.26, cURL 7.78.0, cutadapt 3.4, DIAMOND 2.0.11, DROP 1.1.0, Dsuite 20210718, dtcmp 1.1.2, ecCodes 2.22.1, Elk 7.2.42, ESMF 8.1.1, expat 2.4.1, fastq-tools 0.8.3, Fiona 1.8.20, FLINT 2.7.1, g2clib 1.6.3, g2lib 3.2.0, GCC(core) 11.2.0, GDCM 3.0.8, GDRCopy 2.3, GenomeTools 1.6.2, GffCompare 0.12.2, gffread 0.12.7, GLib 2.69.1, GMT 6.2.0, Go 1.16.6, GPAW 21.6.0, Gradle 6.9.1.eb, GROMACS 2021.3, HarfBuzz 2.8.2, HDF5 1.12.1, Horovod 0.22.1, hwloc 2.5.0, hypothesis 6.14.6, igraph 0.9.4, ImageMagick 7.1.0, IMOD 4.11.5, IPython 7.26.0, Java 16, Julia 1.6.2, JupyterHub 1.4.1, JupyterLab 3.1.6, Kent_tools 418, lavaan 0.6-9, libfabric 1.13.0, libffcall 2.4, libffi 3.4.2, libgeotiff 1.7.0, libRmath 4.1.0, LIBSVM 3.25, LibTIFF 4.3.0, likwid 5.2.0, LLVM 12.0.1, LMDB 0.9.29, magma 2.6.1, MariaDB 10.6.4, MATIO 1.5.21, matplotlib 3.4.2, MEGA 10.0.5, Mesa 21.1.7, Meson 0.59.1, Metal 2020-05-05, Mini-XML 3.2, molmod 1.4.8, MPICH 3.4.2, nanopolish 0.13.3, NCCL 2.10.3, ncdu 1.16, NCO 5.0.1, neptune-client 0.10.5, netcdf4-python 1.5.7, networkx 2.5.1, nodejs 14.17.6, NSPR 4.32, NSS 3.69, NVHPC 21.7, OpenBLAS 0.3.17, OpenEXR 3.1.1, OpenFOAM 9, OpenFOAM v2106, OpenMM 7.5.1, OpenMolcas 21.06, openpyxl 3.0.7, Pango 1.48.8, parallel 20210722, Paraver 4.9.2, ParaView 5.9.1, patchelf 0.13, PCRE2 10.37, PCRE 8.45, Perl 5.34.0, PETSc 3.15.1, petsc4py 3.15.0, Pillow 8.3.1, Pillow-SIMD 8.3.1, plotly.py 5.1.0, PLUMED 2.7.2, PMIx 4.1.0, poppler 21.06.1, PostgreSQL 13.3, preseq 3.1.2, pybind11 2.7.1, pyFFTW 0.12.0, PyGEOS 0.10.2, Pyomo 6.0.1, PyQt5 5.15.4, pyshp 2.1.3, Python 3.9.6, python-igraph 0.9.6, QTLtools 1.3.1, RAxML-NG 1.0.3, R-bundle-Bioconductor 3.13, re2c 2.2, ReFrame 3.8.0., RStudio-Server 1.4.1717, Rust 1.54.0, Scalasca 2.6, scVelo 0.2.3, Shapely 1.8a1, shrinkwrap 1.1.0, SLEPc 3.15.1, slepc4py 3.15.1, snakemake 6.6.1, snappy 1.1.9, snippy 4.6.0, snpEff 5.0e, SOCI 4.0.2, SpaceRanger 1.3.0, Spack 0.16.2, spatialreg 1.1-8, spglib-python 1.16.1, SQLite 3.36, STAR 2.7.9a, statsmodels 0.12.2, StringTie 2.1.7, Subread 2.0.3, SymEngine 0.7.0, TensorFlow 2.6.0, TetGen 1.6.0, Tkinter 3.9.6, torchtext 0.9.1, tqdm 4.61.2, UCX 1.11.0, UCX-CUDA 1.11.0, unrar 6.0.2, util-linux 2.37, VTune 2021.6.0, WIEN2k 21.1, WPS 4.2, WRF 4.3, X11 20210802, x264 20210613, xarray 0.19.0, XlsxWriter 1.4.4, XML-Parser 2.46, zstd 1.5.0
- minor enhancements, including:
  - add extensions to recent R v4.x easyconfigs: PCAmatchR (#13448), hal9001 (#13451), cobalt (#13544), CBPS (#13549), SBdecomp (#13565), lwgeom (#13674), naturalsort (#13762), finalfit + gtsummary (#13766)
  - add VBZ compress capability to nanopolish 0.13.3 (for fast5 files) (#13536)
  - add CMake build dependency to easyconfig for tbb 2020.03 (#13704)
  - also build shared libs for MUMPS 5.3.5 (#13702) and 5.4.0 (#13705)
  - add dependency on pkg-config to OpenSSL wrappers (#13765)
- various bug fixes, including:
  - fix LAMMPS 3Mar2020 easyconfigs using intel toolchain on AMD CPUs by patching out hardcoded `-xHost` (#11577)
  - fix sanity check error for OpenCV v4.5.1 by adding (and using) freetype and HarfBuzz dependencies (#12517)
  - add patches for TensorFlow 1.13.1 to fix installation (#13326)

- remove Python 2.6 from list of supported Python versions in `setup.py` (#13349)
- switch back to Bundle easyblock for PyQt5 5.15.1 + add back source URLs for components (#13371)
- add pkg-config build dependency for GDAL 3.2.1 (#13383, #13808), Rust (#13538), GenomeTools (#13805), x264 (#13834)
- add missing Python 3 build dependency for Rust v1.52.1 (#13399)
- fix order of cffi extension in old Python easyconfigs (#13400)
- patch out `__asm__` use in ctffind to avoid build failure on ppc64le (#13409)
- disable building man pages in Gdk-Pixbuf (#13410)
- add missing Bio-Search-hmmer dependency for prokka (#13419)
- avoid downloading old numpy version during install for h5py (#13428)
- add patch for Qt5 5.8.0 to fix compilation failure in webkit (#13434)
- avoid downloading and building freetype and qhull for matplotlib 3.4.2 (#13437)
- fix order of components in recent GTK+ easyconfigs to avoid dependency on system GTK+ in GTK+ themes (#13449)
- fix Python 2.7.13 easyconfigs by adding missing extensions (#13456)
- add TensorFlow patch to fix crash on shutdown (#13462)
- allow for non-x86_64 in `postinstallcmds` for CuPy (#13501)
- disable default EULA acceptance in easyconfig for NVHPC v21.5 (#13516)
- fix error with p7zip's 7z command not finding `7z.dll` (#13542)
- patch MaxBin2 Perl script to use provided Perl dependency (#13551)
- add missing zlib dependency to libarchive (#13579)
- fix pkgconfig version in patch for bzip 1.0.8 (+ add easyconfig with GCCcore/11.2.0 toolchain) (#13581)
- fix build of DIRAC 19.0 easyconfig with high compiler optimizations (#13613)
- fix source URLs and add alternative checksum for Hypre 2.15.1 (since it moved to a different GitHub repo) (#13616)
- add new source URL for Mesa easyconfigs using 2017b toolchain (#13617)
- avoid `MPICXX` dependency in SimpleElastics ITK (#13623)
- add upstream patch for GCC 9.x, 10.x, 11.x to avoid spurious FPE on avx512 (affects UCX) (#13628)
- add patch for ctffind 4.1.14 to declare functions without return as void (fixes segfault) (#13665)
- enable EGL in recent libepoxy ECs (#13684)
- remove unused `buildcmd` from libgpuarray easyconfigs (#13720)
- add patch to fix building Qt5.10.1 against newer glibc (#13730)
- add Python 3 as a builddependency to recent fontconfig easyconfigs (#13731)
- update source_URL in BLAST 2.2.26 easyconfig (#13732)
- restore error message on failing easyconfigs test suite, required by bot to determine end of output of test suite (#13745, #13770)
- add Perl build dependency to recent Clang easyconfigs (#13746)

  - disable using system Valgrind in the json-c tests (#13750)
  - use Archive source URL for spatstat.geom extension in Seurat (#13761)
  - add ld.gold relocts patch to binutils 2.30 (#13785)
  - add missing Python + Zip build dependencies for old Bazel versions used as build dep for TensorFlow 1.12.0 + 1.13.1 (#13786)
  - add alternative checksum for AFNI 18.3.00 (#13790)
  - update tensorflow-probability easyconfigs to include `--release` flag in `installopts` (#13810)
  - fix source URL for DB 18.1.32 (#13813)
  - add missing cppy build dependency for matplotlib 3.4.2 (#13823)
  - disable bash completion in recent x264 easyconfigs (#13834)
  - add Perl as build dependency for ELPA 2021.05.001 + patch to fix hardcoded `/usr/bin/perl` (#13835)
  - add archive source URL to Spark 3.x easyconfigs to fix broken download (#13842)
  - fix PCRE dependency for GDAL 3.2.1 and 3.3.0, don't use PCRE2 (#13861)
  - add Perl build dependency for HMMER 3.3.2 + patch to ensure it is used (#13870)
  - stick to intel-compilers toolchain for DFT-D3 v3.2.0 (#13878)
  - add missing flex build dependency for leidenalg (#13884)
  - update CMake build dependency for Eigen 3.3.4 + 3.3.5 (#13889)
  - fix source URL for Yasm 1.3.0 (+ add missing checksum) (#13901)
  - update to more recent Meson build dependency for GCCcore/8.3.0 easyconfigs to fix failing RPATH sanity check (#13910)
  - fix shebang for RepeatMasker 4.1.2-p1 Perl script (#13911)

- other changes:
  - use custom ORCA easyblock in easyconfigs for ORCA v4.x (#13348)
  - move archived easyconfig files to correct `__archive__` folder (#13422)
  - rename NINJA to TWL-NINJA to avoid nameclash (#13529)
  - remove superfluous `configopts` and add sanity checks/checksums in CLHEP easyconfigs (#13614)
  - remove superfluous `-DCMAKE_BUILD_TYPE` configuration option in Arrow easyconfigs (#13615)
  - switch ITK easyconfigs to CMakePythonPackage easyblock, also enable `-DITK_USE_SYSTEM_HDF5` configuration option, fix download URLs (#13619)
  - use `build_type` instead of `-DCMAKE_BUILD_TYPE=RELEASE` in `configopts` in MMseqs2 easyconfigs (#13620)
  - avoid using system GTK+3 and remove `-DCMAKE_BUILD_TYPE=RELEASE` configure option in OpenCV easyconfigs (#13621)
  - remove `-DCMAKE_BUILD_TYPE=RELEASE` configure option and add checksums in LLVM easyconfigs (#13624)
  - create libtinfo symlinks in easyconfigs for ncurses with system toolchain (#13658), 6.1 (#13661) and 6.2 (#13660)
  - increase test timeouts for PyTorch 1.8.1 and 1.9.0 (#13700)

  - rename CUDAcore to CUDA for v11.3.1 and v11.4.1 after merging foss/fosscuda, to ensure that `get_software_root('CUDA')` used in easyblocks works (#13874)
  - use CUDA instead of CUDAcore in recent NVHPC comments (#13875)

11.12.2 EasyBuild v4.4.1 (July 6th 2021)

bugfix/update release

framework

- various enhancements, including:
  - enhance detection of patch files supplied to `eb` command with better error messages (#3709)
  - add per-step timing information (#3716)
  - add `module-write hook` (#3728)
  - add `ignore-test-failure` configuration option to ignore failing test step (#3732)
  - add toolchain definition for nvompic (NVHPC + OpenMPI) (#3735)
  - warn about generic milestone in `--review-pr` and `--merge-pr` (#3751)
- various bug fixes, including:
  - don't override `COMPILER_MODULE_NAME`, inherited from `Ffmpi`, in Fujitsu toolchain definition (#3721)
  - avoid overwriting `pr_nr` in `post_pr_test_report` for reports with `--include-easyblocks-from-pr` (#3724, #3726)
  - fix crash in `get_config_dict` when copying modules that were imported in easyconfig file (like `import os`) (#3729)
  - parse C standard flags in `CFLAGS` for Fujitsu compiler (#3731)
  - fix error message for `--use-ccache` (#3733)
  - error out when passing a list to `run_cmd*` to avoid running wrong command unintended, unless `shell=True` is used (#3737)
  - minor fixes to output of test reports when using multiple PRs (#3741)
  - fix location for modules installed with intel-compilers toolchain in HierarchicalMNS by always checking toolchain compiler name against template map (#3745)
  - avoid checking `msg` attribute of `GitCommandError` (#3756)
  - consider sources provided as dict in `EasyBlock.check_checksums_for` (#3758)
  - don't make changes to software installation directory when using `--sanity-check-only` (#3761)
  - honor specified easyblock for extensions (#3762)
- other changes:
  - make sure that tests requiring a github token have `github` in the test name so that they can be easily filtered (#3730)
  - deprecate EasyBuild bootstrap script (#3742)
  - use temporary download folder in `test_http_header_fields_urlpat` (#3755)

easyblocks

- 5 new software-specific easyblocks:
  - AOMP (#2435, #2462, #2464), FreeFEM (#1969), NCCL (built from source) (#2337, #2460), torchvision (#2467), UCX plugins (#2491)
- minor enhancements, including:
  - enhance Amber easyblock to support installing Amber via CMake (#2445)
  - enhance ConfigureMake generic easyblock to add support for building multiple build targets (#2449, #2479, #2480)
  - enhance sanity check for Clang to verify if CUDA offload library was produced (#2454)
  - update custom easyblock for Boost to always build single and multi threaded versions (#2456)
  - enhance `sitecustomize.py` in Python easyblock to support overriding core Python packages, and allowing overriding in virtualenv (#2458, #2483)
  - update CMakeMake to handle old and new Boost/Boost.Python builds using custom easyblock for Boost (#2461)
  - add file prefix option to XALT easyblock (#2463)
  - enhance Java easyblock to define `%(jdkarch)s` template (#2472)
  - adjust Perl easyblock to only check for a `man` subdirectory if groff is a dependency (#2474)
  - support arbitrary version strings in OpenSSL wrapper and add `minimum_openssl_version` option (#2475)
  - enhance Python easyblock to explicitly disable installing core-pip when `install_pip` is not `True` (#2476)
  - enhance intel-compiler easyblock to include multipath include dir in `$CPATH` (#2477)
  - enhance Hypre easyblock to enable CUDA support (#2482)
  - update Xmipp easyblock for new version (v3.20.07) (#2486)
  - enhance FFTW easyblock to skip tests if `--mpi-tests` EasyBuild configuration option is disabled (#2490)
  - use `PYPI_SOURCE` as the default for `source_urls` of extensions of Python easyconfigs (#2493)
- various bug fixes, including:
  - only use siterc fix for NVHPC < 21.3 (#2453)
  - fix CPU-only runtime for dpcpp-generated executables in custom easyblock for intel-compilers (#2457)
  - always add `distinct_host_configuration=false` to build command for TensorFlow (#2459)
  - disable installation of bundled BioPerl and HTSLib if they are dependencies of VEP (#2468)
  - don't use list value for command to detect installed Python packages in TensorFlow easyblock (#2469)
  - change Bundle easyblock to also collect `altroot` and `altversion` in the module step so they are set when running `--module-only` (#2485)
  - always strip output from `gcc -print-multiarch` in intel-compilers easyblock (#2489)
  - don't overwrite all of `exts_default_options` in TensorFlow easyblock (#2494)
  - enhance GCC easyblock to make sure that system GCC provides LTO support, and disable LTO when building MPFR if not (#2498)

  - rework the dependency handling of OpenMPI to use explicit configure options to disable features if required dependency is not provided (#2500, #2501)

easyconfigs

- added example easyconfig files for 39 new software packages:
  - AMPHORA2 (#13003), AOMP (#12909), CAMPARI (#13152), CSB (#12877), dijitso (#10719), DosageConvertor (#13278), dSFMT (#12971), exiv2 (#13204), FEniCS (#10719), FreeFEM (#9902), futhark (#12921), futile (#12864), gappa (#13186), GenomeWorks (#13083, #13092), gexiv2 (#13204), HAL (#13267), hipify-clang (#12961), inline (#12029), libcint (#13214), LTR_retriever (#13125), mapDamage (#13172), MbedTLS (#13061), microctools (#13010), Nektar++ (#12664), NINJA (#13125), OBITools3 (#12969), ont-fast5-api (#13033), OpenMM-PLUMED (#13268), phototonic (#13241), PyFoam (#12675), RagTag (#13247), RcppGSL (#13172), RECON (#13123), RepeatScout (#13124), retworkx (#13228), UCX-CUDA (#13260), unimap (#13247), Vala (#13204), WhatsHap (#12989)
- added additional easyconfigs for various supported software packages, including:
  - Amber 20.11, BBMap 38.90, Beast 2.6.4, BLIS 3.0 (AMDo fork), Bonito 0.4.0, CellRanger-ARC 2.0.0, CellRanger-ATAC 2.0.0, Check 0.15.2, CubeGUI 4.6, CubeLib 4.6, CubeWriter 4.6, cuDNN 8.2.1.32, CVXOPT 1.2.6, DOLFIN 2019.1.0.post0, eggnog-mapper 2.1.4, ELPA 2021.05.001, FDS 6.7.6, FFC 2019.1.0.post0, FFmpeg 4.3.2, FIAT 2019.1.0, flatbuffers 2.0.0, flatbuffers-python 2.0, FLTK 1.3.6, gc 8.0.4, GCC 8.4.0 + 9.4.0, GDAL 3.3.0, Gdk-Pixbuf 2.42.6, geopy 2.1.0, Ghostscript 9.54.0, git 2.32.0, GitPython 3.1.18, GLFW 3.3.4, gmsh 4.8.4, gnuplot 5.4.2, GnuTLS 3.7.2, Go 1.16.5, GObject-Introspection 1.68.0, gperftools 2.9.1, GraphicsMagick 1.3.36, Graphviz 2.47.2, GSL 2.7, GTK+ 2.24.33, Guile 2.2.7, h5py 3.2.1, HarfBuzz 2.8.1, Horovod 0.22.0, Hypre 2.21.0, ICU 69.1, ImageMagick 7.0.11-14, IOR 3.3.0, IPython 7.25.0, IRkernel 1.2, ispc 1.16.0, JupyterLab 3.0.16, LDC 1.26.0, libcerf 1.17, libepoxy 1.5.8, libgd 2.3.1, libStatGen 1.0.15, libxc 5.1.5, LittleCMS 2.12, LMfit 1.0.2, Lua 5.4.3, M4 1.4.19, MCR R2021a.3, medaka 1.4.3, Mercurial 5.8, minimap2 2.20, MMseqs2 13, MrBayes 3.2.7, MUMPS 5.4.0, NCCL 2.9.9, netCDF 4.8.0, Nim 1.4.8, nodejs 14.17.0, Nsight-Compute 2021.2.0, numba 0.53.1, NVHPC 21.5.eb, nvtop 1.2.1, Octave 6.2.0, OPARI2 2.0.6, openkim-models 20210128, OpenMPI 4.0.6, OTF2 2.3, p11-kit 0.24.0, Pango 1.48.5, parallel 20210622, petsc4py 3.12.0, picard 2.25.5, Pillow 8.2.0, PROJ 8.0.1, protobuf 3.17.3, protobuf-python 3.17.3, PSolver 1.8.3, PySCF 1.7.6, pyspoa 0.0.8, pytest-xdist 2.3.0, PyTorch 1.9.0, PyYAML 5.4.1, Qhull 2020.2, Quandl 3.6.1, R 4.0.5 + R 4.1.0, Ragout 2.3, RE2 2021-06-01, ReFrame 3.6.2, RepeatMasker 4.1.2, rgdal 1.5-23, RMBlast 2.11.0, Ruby 3.0.1, scikit-learn 0.24.2, Score-P 7.0, Seurat 4.0.3, slepc4py 3.12.0, spoa 4.0.7, Subread 2.0.2, Subversion 1.14.1, SuiteSparse 5.10.1, sympy 1.8, tensorboardX 2.2, TensorFlow 2.5.0, Tk 8.6.11, tmux 3.2a, torchtext 0.8.1, torchvision 0.9.1, UDUNITS 2.2.28, UFL 2019.1.0, utf8proc 2.6.1, VEP 103.1, VMD 1.9.4a51, vsc-mympirun 5.2.6, XCFun 2.1.1, Xvfb 1.20.11, Z3 4.8.11, ZeroMQ 4.3.4
- minor enhancements, including:
  - update cuDNN 8.0.x easyconfigs with a download location (#12368)
  - add extensions to recent R easyconfigs: GxEScanR (#13039), cSEM (#13208), cubelyr + broom.mixed (#13252), grf (#13261), twang + xgboost (#13284), neuralnet (#13330)
  - add check to easyconfigs test suite to ensure OpenSSL wrapper is used in easyconfigs using a recent toolchain (#13079)
  - add gipaw to QuantumESPRESSO/6.7 (#13087)
  - add checksum for aarch64 installation file for CUDAcore easyconfigs (#13014, #13097)
  - add Perl dependency to makeinfo easyconfigs (#13166)
  - set minimal OpenSSL version to 1.1.1 for OpenSSL v1.1 wrapper (#13188)
  - add JIT support for CUDA 11 to PyTorch 1.7.1 (#13207)
  - add `ninja --version` as sanity check command in Ninja easyconfigs (#13222)

  - code cleanup + improvements for easyconfigs test suite (#13257)
- various bug fixes, including:
  - download sources via git for XGBoost 20171120 easyconfig due to use of submodules (#6880)
  - add elfutils as build dependency for Clang 8.0.x and 9.0.1 (#13015) and Clang 11.0.1 (#13008) easyconfigs that have a CUDA dependency
  - add missing CMake option to Geant4 v10.7.1 in order to actually use external CLHEP (#13019)
  - add new checksum for mvabund extension to R v4.0.4 eaysconfigs (#13020, #13021)
  - add patch to fix numpy `test_ccompiler_opt` in SciPy-bundle v2021.05 (#13042)
  - add patch to fix installation of HDF 4.2.15 on aarch64 (#13059)
  - make sure that MbedTLS' Makefile uses `python` rather than `python2` (#13061)
  - fix checksum for snpEff 5.0 (#13062)
  - fix source URL for LIBSVM 3.24 by downloading from GitHub (#13076)
  - add `preinstallopts` for minimap2 to fix installation on aarch64 (#13080)
  - add patch for GCC 10.2.0 to fix internal compiler error on aarch64 (#13086)
  - switch to NCCL 2.8.3 built from source for CuPy, Horovod, libgpuarray, PyTorch and TensorFlow using fosscuda/2020b (#13103)
  - add astor to dependencies of TensorFlow with 2019b (#13111) and 2020b (#13103, #13112) toolchains
  - break cyclic dependency between groff, makeinfo and Perl by adding Perl-minimal and makeinfo-minimal easyconfigs (#13163 - #13165)
  - add missing rgdal dep to R-INLA (#13179)
  - add patch to fix `uniq` operation in TensorFlow 2.4.1 (#13181)
  - add Perl build dependency for PnetCDF 1.12.1 (#13183)
  - disable own avx detection of libfabric v1.12.1 (#13215)
  - add GCCcore-[1-9][0-9].x to `test_dep_versions_per_toolchain_generation` (#13243, #13249, #13251)
  - add patch to fix installation of TensorFlow 2.0.x (#13248)
  - add patch for recent GCCcore versions to fix compatibility with CUDA 11 (#13290)
  - fix install step for glew (#13297)
  - update Boost 1.74.0 easyconfigs to use `tagged_layout` rather than `boost_multi_thread` (#13300)
  - avoid using PMIx from system instead of PMIx dependency specified via `--with-pmix` for OpenMPI 4.x (#13307)
  - add patch for GCCcore 10.3.0 to prevent issues when compiling with nvcc (#13310)
  - add patch for OpenMPI 4.0.x to fix building against (system) UCX version > 1.7 (#13329)
  - fix build of manta 1.6.0 on top of Boost 1.74.0 (#13338)
  - fix recent taxator-tk easyconfigs by adding (back) `-DBoost_NO_BOOST_CMAKE=ON` configure option (#13342)
- other changes:

- disable `debuginfod` for elfutils to minimize required dependencies (#13034)
- add bare Python 3.9.5 easyconfig and use that as builddep for LLVM 11.1.0 (#13035)
- remove misleading comment from Python >= 3.7 easyconfigs, libffi is now also required for Python itself (no longer bundled) (#13041)
- use new custom easyblock in torchvision easyconfigs (#13102)
- remove superfluous TensorFlow patch (#13223)
- cleanup of easyconfigs for (bundles) of Python packages: remove default `PYPI_SOURCE` source URLs + use `PythonBundle` rather than `Bundle` easyblock (#13253, #13295, #13336)
- update `README` with instructions for MATLAB easyconfigs (#12597)

11.12.3 EasyBuild v4.4.0 (June 2nd 2021)

feature release

framework

- various enhancements, including:
  - enhance `apply_regex_substitutions` to allow specifying action to take in case there are no matches (#3440)
  - performance improvements for easyconfig parsing and eb startup (#3555)
  - add support for downloading easyconfigs from multiple PRs with `--from-pr` (#3605, #3707, #3708)
  - add support for prepending custom library paths in RPATH section via `--rpath-override-dirs` (#3650)
  - allow amending easyconfig parameters which are not the default (#3651)
  - update HierarchicalMNS for Intel OneAPI compilers (#3653)
  - add support for `--sanity-check-only` (#3655)
  - add support for running commands asynchronously via `run_cmd` + `complete_cmd` functions (#3656)
  - add support for using oneAPI versions of 'intel' toolchain components (#3665)
  - add toolchain definition for gofbf (foss with FlexiBLAS rather than OpenBLAS) (#3666)
  - add support for using `ARCH` constant in easyconfig files (#3670)
  - honor keyboard interrupt in `eb` command (#3674)
  - add toolchain definition for Fujitsu toolchain (#3677, #3704, #3712, #3713, #3714, #3717)
  - extend sanity check step to check whether specific libraries are not linked into installed binaries/libraries (#3678)
    - via `banned-linked-shared-libs` and `required-linked-shared-libs` EasyBuild configuration options
    - via `banned_linked_shared_libs` and `required_linked_shared_libs` methods in toolchain support
    - via `banned_linked_shared_libs` and `required_linked_shared_libs` methods in easyblock
    - via `banned_linked_shared_libs` and `required_linked_shared_libs` easyconfig parameters

  - add `locate_solib` function to locate Linux shared libraries (#3682)
  - add system agnostic function to locate shared libraries (#3683)
  - add `update_build_option` function to update specific build options after initializing the EasyBuild configuration (#3684)
  - allow opting out of recursively unloaded of modules via recursive_module_unload easyconfig parameter (#3689)
  - check for correct version values when parsing easystack file (#3693)
  - run post-install commands specified for a specific extension (#3696)
  - add support for `--skip-extensions` (#3702)
  - include PR title in output produced by `--merge-pr` (#3706)
- various bug fixes, including:
  - avoid metadata greedy behaviour when probing for external module metadata (mostly relevant for integration with Cray Programming Environment) (#3559)
  - catch problems early on if `--github-user` is not specified for `--new-pr` & co (#3644)
  - re-enable write permissions when installing with `read-only-installdir` (#3649)
  - also run sanity check for extensions when using `--module-only` (#3655)
  - improve logging when failing to load class from `exts_classmap` (#3657)
  - fix use of `--module-only` on existing installations without write permissions (#3659)
  - correct help text for `subdir-user-modules` (#3660)
  - avoid picking up easyblocks outside of sandbox in framework tests (#3680)
  - use unload/load in `ModuleGeneratorLua.swap_module`, since `swap` is not supported by Lmod (#3685)
  - update HierarchicalMNS to also return 'Toolchain/<name>/<version>' as `$MODULEPATH` extension for cpe* Cray toolchains (#3686)
  - make `EasyConfigParser.get_config_dict` return a copy rather than a reference (#3692)
  - make sure that `$TAPE` is unset when using piped tar (#3698)
  - fix extending message for changed files in `new_pr_from_branch` (#3699)
  - enhance `sched_getaffinity` function to avoid early crash when counting available cores on systems with more than 1024 cores (#3701)
  - correctly strip extension from filename in `extract_cmd` and `back_up_file` functions (#3705)
- other changes:
  - deprecate adding a non-existing path to `$MODULEPATH` (#3637)
  - bump cryptography requirement from 3.2.1 to 3.3.2 (#3643, #3648)
  - test bootstrap script in separate workflow, and limit test configurations a bit (#3646)
  - update `setup.py` to indicate compatibility with Python 3.8 and 3.9 (#3647)
  - replace `log_error` parameter of `which()` by `on_error` (#3661, #3664)
  - don't skip sanity check for `--module-only --rebuild` (#3645)
  - move `disable_templating` function into the EasyConfig class (#3668)

  - pin GitPython version for Python<3.6, don't test bootstrap script against Python 3.8/3.9 (#3675)
  - tweak foss toolchain definition to switch from OpenBLAS to FlexiBLAS in foss/2021a (#3679)
  - suggest missing SSH key when not able to read from remote repository in `--check-github` (#3681)
  - drop support for Python 2.6 (#3715)

easyblocks

- 3 new software-specific easyblocks:
  - FlexiBLAS (#2369, #2422, #2424, #2426)
  - dm-reverb (#2413)
  - custom easyblock to install OpenSSL wrapper for OpenSSL installed in OS, with fallback to build and install OpenSSL from source if not available in OS (#2429)
- minor enhancements, including:
  - also add `-pthread` to `prebuildopts` of cryptography (#2270)
  - don't unpack Python wheel (*.whl) files by default in generic PythonPackage easyblock (#2366, #2442)
  - enable installation of samples for CUDA > 10.1 (#2374)
  - add option to disable pip connecting to PyPi (enable use of `--no-index`) (#2390)
  - update MotionCor2 easyblock to handle new version and be aware of CUDAcore (#2394)
  - make it possible to force disabling kernel features in Qt easyblock (#2403)
  - update imkl easyblock to support oneAPI versions (>= 2021.x) (#2407)
  - add‘‘”gurobi_cl –help‘‘ as default sanity check command for Gurobi (#2411)
  - enhance BWA easyblock to copy includes and libraries (#2417)
  - allow `default` versions to be defined by ModuleRC easyblock (#2418)
  - enhance MesonNinja and CMakeMake easyblocks to create unused build dir when `separate_build_dir` is set (#2419)
  - enable `sanity_pip_check` by default for Python easyconfigs if pip >= 9.0 will be installed (#2423)
  - enhance FFTW easyblock to support SVE CPU feature and building with Fujitsu compiler (#2425)
  - make ScaLAPACK easyblock aware of FlexiBLAS (#2427)
  - update imkl easyblock to unpack example tarballs and set `$MKL_EXAMPLES` (+ some code cleanup) (#2430)
  - update list of system libs for TensorFlow 2.5 (#2432)
  - allow disabling MPI tests when installing Intel MPI (impi) via `--disable-mpi-tests` (#2440)
  - exclude bottleneck tests in PyTorch (#2450)
- various bug fixes, including:
  - ensure lib subdirectory is found in stage 2 of GCC installation + fall back to lib64 (#2339)
  - fix permission on MATLAB installer config file so it can be written to (#2385)
  - fix problem with installing older CUDA versions that uses the Perl based installer (#2387)
  - enhance Python easyblock: add option to install pip with core Python, tweak defaults, create unversioned pip symlink (#2388)

  - fix `installopts` before installing the extension in GROMACS easyblock (#2391)
  - updated numpy easyblock to use `read_file` for patch (#2395)
  - use explicit build toolset and compiler path in Boost easyblock (#2402)
  - replace hardcoded `2021.1.1` with `self.version` in impi easyblock (#2405)
  - set `$SANDCASTLE` when running PyTorch tests to disable some tests as-if we are on Facebook's CI (#2412)
  - make GROMACS easyblock work with `--module-only` (#2414)
  - make sure OpenFOAM sanity checks don't require builddir write permissions (#2415)
  - make Tkinter easyblock work with `--module-only` (#2416)
  - also disable altivec with FFTW 3.3.9 on POWER (#2437)
  - make sure that `self.python_cmd` is set before using it in `PythonPackage.sanity_check_step` (#2447)
- other changes, including:
  - update `setup.py` to indicate compatibility with Python 3.8 and 3.9 (#2384)
  - use `on_error` rather than deprecated `log_error` named argument for `which` function (#2406)
  - remove `cuda_compute_capabilities` from custom easyconfig parameters for Clang, LAMMPS and TensorFlow (now supported as general easyconfig parameter) (#2433)

easyconfigs

- added easyconfigs for new Fujitsu toolchain: FCC/4.5.0 (#12999, #12995, #13007), ffmpi/4.5.0 (#13000) and Fujitsu/21.05 (#13001, #13007)
- add easyconfigs for updates of common toolchains: foss/2021a (#12867, #12975), intel/2021a (#12885, #12975)
  - see also https://easybuild.readthedocs.io/en/latest/Common-toolchains.html
- added easyconfig for gomkl/2021a toolchain (#12987)
- added example easyconfig files for 58 new software packages:
  - Archive-Zip (#12651), BirdNET (#12737, #12712, #12737), cell2location (#12448), cryoDRGN (#12704), dm-reverb (#12824), DROP (#12559, #12801, #12950), fastahack (#12841), fermi-lite (#12856), filevercmp (#12841), FlexiBLAS (#12476), freetype-py (#12918), fsom (#12841), garnett (#12529), gawk (#12716), gemmi (#12855), georges (#12570), hifiasm (#12897), intervaltree (#12838), LDC (#12671), libidn2 (#12670), librosa (#10477), librsb (#12780), line_profiler (#12556), loompy (#12804), mmtf-cpp (#12580), mongolite (#12632), msgpack-c (#12580), multichoose (#12839), networkTools (#12810), NewHybrids (#12528), Octopus-vcf (#12598), onedrive (#12671), p4-phylogenetics (#12549), pagmo (#12678), pyfaidx (#12872), pyfasta (#12952), pygmo (#12678), pysheds (#12791), R-INLA (#12955, #12927, #12955), RegTools (#12874), request (#12448), rMATS-turbo (#12982), scanpy (#12731), SeqLib (#12856), SeuratData (#12993), SeuratWrappers (#12994), smithwaterman (#12841), snp-sites (#12900), SSW (#12856), tabixpp (#12837), TagDust (#11599), TALON (#12963), tMAE (#12559), TranscriptClean (#12952), umap-learn (#12448), vcflib (#12841), xESMF (#12799), XlsxWriter (#12820)
- added additional easyconfigs for various supported software packages, including:
  - ABINIT 9.4.1, apex 20210420, astropy 4.2.1, binutils 2.36.1, BLIS 0.8.1, Bonito 0.3.8, BUSCO 5.1.2, canu 2.1.1, carputils 20210513, CastXML 0.4.3, ccache 4.2.1, CDO 1.9.10, CIRCexplorer2 2.3.8, CLHEP 2.4.4.0, CMake 3.20.1, CNVkit 0.9.8, CUDA(core) 11.3.0, CuPy 8.5.0, cURL 7.76.0, DFA 2.1.2, Doxygen 1.9.1, Eigen 3.3.9, FastANI 1.33, FFTW 3.3.9, GATK 4.2.0.0, GCC 10.3.0 + 11.1.0, GDB 10.2, gdc-client 1.6.0, GDRCopy 2.2, Geant4 10.7.1, Geant4-data 20210510, GeneMark-ET 4.65, glew 2.2.0,

GLib 2.68.2, GLPK 5.0, GMP 6.2.1, Go 1.16.3, Graphviz 2.47.0, GROMACS 2021.2, GStreamer 1.18.4, GTDB-Tk 1.5.0, Gurobi 9.1.2, HMMER 3.3.2, Horovod 0.21.3, hwloc 2.4.1, hypothesis 6.13.1, IGV 2.9.4, impi 2021.2.0, imkl 2021.2.0, intel-compilers 2021.2.0, JasPer 2.0.28, Julia 1.6.1, Kraken2 2.1.1, KronaTools 2.8, libarchive 3.5.1, libdeflate 1.7, libdrm 2.4.106, libfabric 1.12.1, libreadline 8.1, libRmath 4.0.0, libsndfile 1.0.31, LIBSVM 3.24, LibTIFF 4.2.0, libunwind 1.5.0, libwebp 1.2.0, LLVM 11.1.0, LMDB 0.9.28, LUMPY 0.3.1, lz4 1.9.3, Mako 1.1.4, MATLAB 2021a, Mesa 21.1.1, meshalyzer 2.2, Meson 0.58.0, MetaBAT 2.15, metaWRAP 1.3, minimap2 2.18, Molden 6.8, MotionCor2 1.4.2, ncview 2.1.8, NetPIPE 5.1.4, nettle 3.7.2, NiBabel 3.2.1, Nilearn 0.7.1, Nim 1.4.6, Ninja 1.10.2, NLopt 2.7.0, NSS 3.65, numactl 2.0.14, NWChem 7.0.2, OpenBLAS 0.3.15, openCARP 6.0, OpenEXR 3.0.1, OpenMM 7.5.0, OpenMPI 4.1.1, OpenSSL 1.1 (wrapper), OSU-Micro-Benchmarks 5.7.1, Pandoc 2.13, parallel 20210322, parasail 2.4.3, PAUP 4.0a168, PCRE2 10.36, Perl 5.32.1, pigz 2.6, PMIx 3.2.3, Primer3 2.5.0, PRSice 2.3.3, py-aiger 6.1.14, pybind11 2.6.2, PyCharm 2021.1.1, pydot 1.4.2, pyproj 3.0.1, PySAT 0.1.7.dev1, PyTorch 1.8.1, python-parasail 1.2.4, Pylint 2.7.4, Python 3.9.5, Qt5 5.15.2, R-keras 2.4.0, rasterio 1.2.3, RAxML-NG 1.0.2, rclone 1.54.1, re2c 2.1.1, ReFrame 3.5.2, Rmath 4.0.4, Rust 1.52.1, Sambamba 0.8.0, samblaster 0.1.26, samclip 0.4.0, scikit-allel 1.3.2, SciPy-bundle 2021.05, SCons 4.1.0.post1, snakemake 6.1.0, SQLite 3.35.4, SRPRISM 3.1.2, Tcl 8.6.11, TCLAP 1.2.4, tmux 3.2, tqdm 4.60.0, UCX 1.10.0, Valgrind 3.17.0, WannierTools 2.5.1, wget 1.21.1, wxWidgets 3.1.4, X11 20210518, x264 20210414, x265 3.5, xorg-macros 1.19.3, YAXT 0.9.0, zarr 2.8.1, zstd 1.4.9

- minor enhancements, including:
  - add additional extensions to R v4.0.3 and v4.0.4: miceadds, visdat, UpSetR, naniar, stringdist, image.binarization (#12735), lassosum (#12821), lslx, regsem, semPLS (#13005)
  - add GenABEL to R-bundle-Bioconductor (#12822)
- various bug fixes, including:
  - disable automatic acceptance of NVHPC EULA (#12014)
  - enhance RStudio-Server, add patch to inherit environment variables, add sanity check commands to verify installation (#12544)
  - add fix to scipy to handle `NaN` arguments to `vi` function (#12551)
  - copy all binaries + examples & co for PAML v4.9j (#12567)
  - add patch to fix hardcoded `num_cores` in DMCfun extension included with R 4.0.x (#12583)
  - backport Charm++ patch for NAMD/2.14 on intel/2020a to handle newer glibc (#12594)
  - add setuptools_scm and pytest-runner extensions to Pylint easyconfigs (#12599)
  - fix typo in Delly easyconfig to actually do parallel build (#12633)
  - fix potential memory leak in OpenBLAS 0.3.12 (#12649)
  - consistently use pip to install Python packages in recent Python easyconfigs (#12650)
  - replace bintray source url for Groovy (#12652)
  - add missing Python & Perl dependencies to AUGUSTUS v3.4.0 with foss/2020b (#12662)
  - fix wget dependency: use libidn2 rather than libidn (#12670)
  - fix source URLs for recent ELPA versions (#12700)
  - override host compiler check in CUDAcore (#12701)
  - add Python build dependency to libpsl 0.21.0 easyconfigs (#12715)
  - fix error in libgpg-error 1.36 with gawk builtin (#12716)
  - add libtool build dependency to leidenalg (#12741)

  - fix source URLs for Boost 1.71.0 - 1.74.0 easyconfigs (#12743) and Boost.Python (#12744)
  - add patches for PyTorch 1.7.1 avoiding failures on POWER and A100 (#12753)
  - add patch for OpenPGM 5.2.122 eaysconfigs to fix non-existent directory in *.pc (pkgconfig) file (#12774)
  - add missing Perl build dependency to recent wget easyconfigs (#12787)
  - make sure Python dependency is used in preCICE 2.x easyconfigs (#12793)
  - don't skip source step in FastTree easyconfigs + fix SHA256 checksum for FastTree 2.1.11 (#12794)
  - add patch for rhdf5filters extension in Bioconductor 3.12 bundle to fix installation on aarch64 (#12836)
  - add IceLake detection to OpenBLAS 0.3.12 and 0.3.15 (#12865)
  - fix checksum for MaxBin 2.2.7 (#12869)
  - run `make clean` before building FragGeneScan, to clean up object files included in source tarball (#12870)
  - clean up install of KronaTools (#12871)
  - add UCX dependency to OneAPI versions of impi (#12873)
  - correct description in libdeflate easyconfig (#12886)
  - override Makefile with hardcoded `CC=cc` in UnZip easyconfigs (#12887)
  - fix compatibility of FLAIR v1.5.1-20200630 with rpy2 v3.x (#12899)
  - fix test step for libxc 4.3.4 and 5.1.x when using RPATH linking (#12912)
  - fix source URLs in BLAST 2.10.x easyconfigs (#12914)
  - add missing xproto build dependency to imake easyconfig (#12930)
  - add patch to fix GCC 10.2.0 rejecting valid code on PPC (#12948)
  - in easyconfig tests, check version of dependencies named Python, not if dependencies with certain versions are named Python (#12962)
  - update easyconfigs for binutils 2.35 to use binutils 2.35.2 source tarball instead to pick up bug fixes (#12967, #12988)
  - fix download URL for DB 18.1.40 (#12974)
  - fix test failure in TensorFlow 2.4.1 on recent CUDA drivers (#12979)
  - fix error in `configopts` for netCDF and HDF5 and add missing dependencies of ABINIT 9.x (#12981)
  - add patch to fix buffer overflow in OpenMPI 4.1.x (#12983)

- other changes:
  - update `setup.py` to indicate compatibility with Python 3.8 and 3.9 (#12565)
  - add `-Java` versionsuffix for Hadoop easyconfig using GCCcore/10.2.0 toolchain, since it depends on Java 1.8 (#12709)
  - remove unnecessary workaround for missing version of OpenDataCube and Spyder for 2020+ toolchains (#12757)
  - remove `unpack_sources = False` from recent easyconfigs that use a *.whl file as source (#12783)
  - whitelist Seaborn 0.10.1 for NanoComp 1.13.1 and NanoPlot 1.33.0 (#12790)
  - add check to make sure that source step is not being skipped (#12807)
  - use `EasyConfig.disable_templating` method in test suite (#12848)

  - disable usNIC by default in libfabric to avoid problems due to linking to both libnl and libnl-3 (#12854)
  - use OpenSSL wrapper as dependency in easyconfigs using GCCcore/10.3.0 toolchain (#12922, #12944)
  - dump easyconfig before initializing easyblock in order to compare it with original easyconfig (#12925)
  - remove postinstallcmds from imkl 2020.x easyconfigs, easyblock now installs examples (#12937)

11.12.4 EasyBuild v4.3.4 (Apr 9th 2021)

bugfix/update release

framework

- various enhancements, including:
  - add support for filtering dependencies by using `False` as version (#3506)
  - add `create_unused_dir` function to create a directory which does not yet exist (#3551)
  - avoid running expensive `module use` and `module unuse` commands when using Lmod as modules tool, update `$MODULEPATH` directly instead (#3557, #3633)
  - create CUDA cache (for JIT compiled PTX code) in build dir instead of `$HOME` (#3569)
  - add "Citing" section to module files (#3596)
  - add support for using fallback `arch=*` key in dependency version specified as arch->version mapping (#3600)
  - also check for pending change requests and mergeable_state in `check_pr_eligible_to_merge` (#3604)
  - ignore undismissed `changes requested` review if there is an `approved` review by the same user (#3607, #3608)
  - sort output of `eb --search` in natural order (respecting numbers) (#3609)
  - enhance `eb` command to ensure that `easybuild.main` can be imported before settling on python* command to use (#3610)
  - add `--env-for-shebang` configuration option to define the `env` command to use for shebangs (#3613)
  - add templates for architecture independent Python wheels (#3618)
  - mention easyblocks PR in gist when uploading test report for it + fix `clean_gists.py` script (#3622)
  - also accept regular expression value for `--accept-eula-for` (#3630)
  - update `validate_github_token` function to accept GitHub token in new format (#3632)
- various bug fixes, including:
  - fix `$BLAS_LIB_MT` for OpenBLAS, ensure `-lpthread` is included (#3584)
  - use `--opt=val` for passing settings from config file to option parser to avoid error for values starting with - or -- (#3594)
  - avoid raised exception when getting output from interactive command in `run_cmd_qa` (#3599)
  - add option to write file from file-like object and use in `download_file` (#3614)
  - make sure that path to `eb` is always found by tests (#3617)
- other changes:

  - add `pick_default_branch` function to clean up duplicate code in `tools/github.py` (#3592)
  - refactor the CI configuration to use inclusion instead of exclusion (#3616)
  - use develop branch when testing push access in `--check-github` (#3629)
  - deprecate `--accept-eula`, rename to `--accept-eula-for` (#3630)

easyblocks

- minor enhancements, including:
  - make OpenCV easyblock aware of protobuf, libwebp and OpenEXR dependencies provided via EasyBuild (#2346)
  - update CP2K easyblock w.r.t. running regtest for CP2K v8.1 (#2350)
  - update GROMACS easyblock for GROMACS/2021 with CUDA (#2353)
  - adjust call to `python-config` for Python >= 3.8 in VMD easyblock (#2355)
  - enhance cuDNN and CUDA easyblocks to support `aarch64` (#2356)
  - pass down compiler flags provided by EasyBuild in g2clib easyblock (#2357)
  - update VTune easyblock for version 2020 (#2359)
  - make WRF and WPS easyblocks aware of `(pre)configopts` (#2361)
  - add Clang version 12.0.0 for AOCC 3.0.0 to mapping in custom easyblock for AOCC (#2362)
  - use `PYPI_SOURCE` as the default for source_urls of PythonPackage (#2364, #2370)
  - enhance PythonPackage easyblock to catch faulty version (0.0.0) for installed Python packages (#2367, #2377)
  - enhance BWA easyblock: pass compiler flags + use `filetools` functions (#2368)
- various bug fixes, including:
  - set `$R_LIBS_SITE` rather than `$R_LIBS` when installing R packages (#2326)
  - update PETSc easyblock to take into account that ScaLAPACK installation may not have header files + fix building in parallel (#2348)
  - disable CMake user package repository in CMakeMake generic easyblock (#2351)
  - update LAPACK easyblock to keep control of compiler options for versions >= 3.9.0 (#2358)
  - also set `$TORCH_CUDA_ARCH_LIST` for PyTorch tests (#2363)
  - enhance Hadoop easyblock to avoid copying same native library twice (#2371)
  - fix pip extension download pattern for PythonPackage easyblock (#2372)
  - make the CUDA stub libs take preference over system libs when linking (#2373)
  - improve Python package version check and add `unversioned_packages` easyconfig parameter (#2377)

easyconfigs

- added easyconfig for goblf/2020b toolchain (#12381, #12535)
- added example easyconfig files for 41 new software packages:
  - amplimap (#12205), BEEF (#12104), bpp (#12036), Brotli (#11651), CDAT (#12322), cicero (#12252), CIF2Cell (#12258), CompareM (#9377), DL_POLY_4 (#12324), DMCfun (#12412), ESMPy (#12339), FLAC (#12300), gdbm (#12322), GPyOpt (#12524), json-c (#12344), libcdms (#12322), libdrs (#12322),

libogg (#12285), libvorbis (#12300), LncLOOM (#12287), LPJmL (#12344), maze (#12354), MetaEuk (#12188), mrcfile (#12497), Myokit (#12261), NCCL-tests (#12415), pyABC (#12329), PyCifRW (#12258), PyOD (#12507), pyro-api (#12447), pyro-ppl (#12447), R-opencv (#12226), Ratatosk (#12443), RevBayes (#12419), sansa (#12354), scikit-cuda (#12352), Seeder (#9057), suave (#12354), voltools (#12497), vorbis-tools (#12300), YACS (#12309)

- added additional easyconfigs for various supported software packages, including:
  - AOCC 3.0.0, Arriba 2.1.0, ArviZ 0.11.1, arpack-ng 3.8.0 BCFtools 1.12, BEDTools 2.30.0, BUSCO 5.0.0, BioPerl 1.7.8, Blosc 1.21.0, Boost.Python 1.74.0, bitarray 1.2.1, bokeh 2.2.3, CP2K 8.1, CUDAcore 11.2.2, CellRanger 6.0.0, Clang 11.0.1, DIAMOND 2.0.7, Delly 0.8.7, dask 2021.2.0, dm-tree 0.1.5, Elk 7.0.12, Extrae 3.8.0, FLUENT 2021R1, FabIO 0.11.0, Fiji 20201104, Flye 2.8.3, FreeSurfer 7.1.1, GDAL 3.2.1, GEOS 3.9.1, GLFW 3.3.3, GMAP-GSNAP 2020-12-17, GROMACS 2021, GetOrganelle 1.7.4, gmsh 4.7.1, HTSeq 0.11.3, HTSlib 1.12, Hypre 2.20.0, hyperopt 0.2.5, iVar 1.3.1, igraph 0.9.1, Jansson 2.13.1, Kent_tools 411, LAPACK 3.9.1, LAST 1179, LibSoup 2.72.0, libxc 5.1.3, MAFFT 7.475, MCR R2020a.6 + R2020b.5 + R2021a.0.eb, MDTraj 1.9.5, MUMPS 5.3.5, MaSuRCA 4.0.1, Mercurial 5.7.1, Monocle3 0.2.3, NGS 2.10.9, NVHPC 21.2, NetLogo 6.2.0, Nextflow 21.03.0, ncbi-vdb 2.10.9, OSU-Micro-Benchmarks 5.7, OpenCV 4.5.1, OpenEXR 2.5.5, OptiX 7.2.0, PETSc 3.14.4, PLUMED 2.7.0, PROJ 7.2.1, PyAMG 4.0.0, PyCUDA 2020.1, PyCairo 1.20.0, PyOpenCL 2021.1.2, PyTorch-Geometric 1.6.3, p7zip 17.03, pFUnit 4.2.0, picard 2.25.1, pocl 1.6, preCICE 2.2.0, protobuf 2.5.0, pymatgen 2022.0.4, python-igraph 0.9.0, Qtconsole 5.0.2, R 4.0.4, RASPA2 2.0.41, RDFlib 5.0.0, ReFrame 3.5.1, Ruby 2.7.2, rnaQUAST 2.2.0, SAMtools 1.12, SDL2 2.0.14, SIMPLE 3.0.0, SPAdes 3.15.2, SUNDIALS 5.7.0, Seurat 4.0.1, Spark 3.1.1, scikit-image 0.18.1, silx 0.14.0, spglib 1.16.1, sympy 1.7.1, tensorboardX 2.1, tensorflow-probability 0.12.1, tmux 3.1c, USEARCH 11.0.667, VTK 9.0.1, VTune 2020_update3
- minor enhancements, including:
  - verify checksum of all patch files in easyconfigs test suite (#12221)
  - add libwebp and OpenEXR dependencies for OpenCV 4.2.0 easyconfig with foss/2020a (#12227)
  - add sanity check commands for vorbis-tools (#12304)
  - add extensions to R-bundle-Bioconductor 3.12: motifmatchr (#12390), OUTRIDER + FRASER (#12510)
  - add DMCfun extension for R v4.0.3 + v4.0.4 (#12409)
- various bug fixes, including:
  - add missing GNU time dependency to WRF 4.0.2 built with foss/2018b (#12179)
  - add archive `source_urls` for Hadoop and Spark (#12220)
  - add missing Python build dep for SeqAn 2.4.0 (#12222)
  - add missing pkg-config build dependency to VCFtools 0.1.16 (#12245), GObject-Introspection-1.64.0 (#12298), libsndfile (#12303)
  - set `$HTSLIB_DIR` in HTSlib for use by EnsEMBLCoreAPI & Bio-DB-HTS (#12253)
  - bump JasPer version to latest 2.0.24 for 2020b generation of easyconfigs + remove easyconfigs for ancient JasPer 2.0.1.4 with GCCcore/10.2.0 (#12277, #12288)
  - bump pip to 20.3.4 in Python 2.7.18 easyconfig to fix unicode error (#12293)
  - skip sanity check test in IPython-7.18.1-GCCcore-10.2.0.eb (#12294)
  - add missing Perl and Autotools build dependencies in recent WRF easyconfigs (#12301)
  - add missing groff build dependency for Perl 5.30.0 and 5.32.0 (#12307)
  - add linkcomm + rnetcarto extensions to R v4.0.3 (#12311)

  - add dependency on Flask in all ASE v3.21.1 easyconfigs (#12312)
  - remove hard-coded `-xHost` from MMseqs2-11-e1a1c (#12317)
  - revised outdated easyconfigs for libcerf + replaced obsolete homepages and source urls (#12323)
  - consistently include `new_archive` source URL in Qt5 easyconfigs + add missing checksums (#12325, #12426)
  - fix source URL and add alternate checksum for Hypre 2.14.0 (#12337)
  - add patch for recent netCDF easyconfigs to fix MPI_Info_f2c issue with OpenMPI (#12340)
  - add SourceForge fallback source URL for recent freetype easyconfigs (#12341)
  - revert to Seaborn 0.10.1 as dependency for NanoPlot 1.33.0 (#12345)
  - use `-D_USE_METIS_5p1` in OpenSees v3.2.0 patch to correctly build on top of METIS v5.1.0 (#12403)
  - make sure that path to `eb` is always found by tests (#12436)
  - stop tests changing the EasyBuild easyconfigs (#12454)
  - use pip to install pkgconfig 1.5.1 (#12455)
  - add CI test checking if the Python default `source_urls` are used and fix CI check where `use_pip=False` was ignored (#12456, #12471)
  - fix source URLs in Arrow 0.x easyconfigs (#12475)
  - fix numpy tests for recent SciPy-bundle easyconfig on POWER (#12481)
  - don't download hwloc during FIRESTARTER build (#12482)
  - avoid 0.0.0 install version for various Python apps (#12519, #12522)
  - enable USER_SDPD package and disable building docs in LAMMPS 3Mar2020 (#12527)
  - update `$R_LIBS_SITE` rather than `$R_LIBS` in easyconfigs installing R packages (#12534)
  - add Python 3 build dep for HMMER 3.3.x test step (#12536)
- other changes:
  - rename opencv_contrib and update to OpenCV v3.4.1 with contrib `versionsuffix` (#12229)
  - fix minor style issues in POV-Ray v3.7.0.8 easyconfigs (#12342)
  - enable `-fPIC` for g2clib (#12349)
  - consistently include Keras-Applications and Keras-Preprocessing extensions in Keras 2.3.1 easyconfigs (#12375)
  - remove `PYPI_SOURCE` source URL from easyconfigs using PythonPackage or PythonBundle (#12541, #12452, #12453)
  - require `sanity_pip_check` for all Python package/bundles (#12464)

11.12.5 EasyBuild v4.3.3 (Feb 23rd 2021)

bugfix/update release

framework

- various enhancements, including:
  - advise PR labels in `--review-pr` and add support for `--add-pr-labels` (#3177)

  - add support for using customized HTTP headers in `download_file` (#3472, #3583)
  - also take toolchain dependencies into account when defining template values (#3541, #3560)
  - add support for `--accept-eula` configuration option + `accept_eula` easyconfig parameter (#3535, #3536, #3546)
  - detect `SYSTEM` toolchain as special case in easystack files (#3543)
  - enhance `extract_cmd` function to use `cp -a` for shell scripts (.sh) (#3545)
  - allow use of alternate envvar(s) to `$HOME` for user modules (#3558)
  - use https://sources.easybuild.io as fallback source URL (#3572, #3576)
  - add toolchain definition for `iibff` toolchain (#3574)
  - add `%(cuda_cc_space_sep)s` and `%(cuda_cc_semicolon_sep)s` templates (#3578)
  - add support for intel-compiler toolchain (>= 2021.x versions, oneAPI) (#3581, #3582)
- various bug fixes, including:
  - add `--init` and `--recursive` options to `git submodule update` command that is used when creating source tarball for specific commit (#3537)
  - filter out duplicate paths in RPATH wrapper script (#3538)
  - don't clean up imported modules after verifying imports of included Python modules (#3544)
  - avoid no-op changes to `$LD_*` environment variables in ModulesTool (#3553)
  - fix UTF-8 encoding errors when running EasyBuild with Python 3.0.x-3.6.x (#3565)
  - create `lib64` symlink as a relative symlink (#3566)
  - don't reuse variable name in the loop to fix adding extra compiler flags via toolchainopts (#3571)
  - symlink `lib` to `lib64` if it doesn't exist (#3580)
  - include `%(mpi_cmd_prefix)s` and `%(cuda_*)s` templates in output of `--avail-easyconfig-templates` (#3586)
- other changes:
  - rename `EasyBlock._skip_step` to `EasyBlock.skip_step`, to make it part of the public API (#3561)
  - make symlinking of `posix_c.so` to `posix.so` in test suite configuration conditional (#3570)

easyblocks

- 2 new software-specific easyblocks:
  - AOCC (#2295), Intel compilers (v2021.x, oneAPI) (#2305)
- minor enhancements, including:
  - run motorBike tutorial case as sanity check for recent (community) OpenFOAM versions (#2201)
  - add foamMonitor to sanity checks of OpenFOAM (#2256)
  - create versioned symlinks for CMake commands + create symlink for `cmake3` in PyTorch easyblock if `cmake3` command is not found (#2259)
  - improve Bazel easyblock: add support for running tests, enable static linking, use build dir rather than `tmpdir`, verbose output (#2285)

  - add support for skipping steps in Python packages installed as extension + print progress on individual steps for installing Python packages as extensions (#2290)
  - update BerkeleyGW easyblock to support GCC 10 and fftlib (#2297)
  - update QuantumESPRESSO easyblock to support GCC 10 (#2298)
  - update Clang easyblock to add support for building extra tools + leveraging hwloc and Z3 as optional dependencies (#2310)
  - add support for running TensorFlow CPU and GPU tests (#2263, #2292, #2312)
  - update impi easyblock for impi 2021.x (oneAPI) (#2313)
  - update QuantumESPRESSO easyblock to handle v6.7 (#2319)
  - update OpenFOAM easyblock for changes in v2012 (#2321)
  - add sanity check commands to GCC (including LTO support) (#2322)
  - update FLUENT sanity check for v2021R1 (#2334)
- various bug fixes, including:
  - filter out user packages in LAMMPS easyblock if corresponding dependency isn't included + only set `-DUSER-INTEL` on x86_64 systems (#2254)
  - unify handling of `pylibdirs` and don't add duplicated `$PYTHONPATH` in PythonBundle (#2281)
  - enhance Amber easyblock to fix running of `update_amber` script when `python` command is not available in OS (#2282)
  - guard `module unload` statements in modules for Cray* toolchains (#2286)
  - set `$PYTHONNOUSERSITE` in `PythonBundle.extensions_step` to avoid picking up on Python packages installed in $HOME (#2289)
  - create less temporary directories for TensorFlow by (only) using `--output_user_root` (#2293)
  - fix logic w.r.t. enabling Python support in PETSc (#2299)
  - make `builddeps` a list of names in SLEPc easyblock (#2300)
  - make `builddeps` a list of names in Trilinos easyblock (#2301)
  - make sure the installation of `libiberty.a` in the binutils easyblock goes into a populated directory (#2308)
  - fix for building GCC with `--sysroot` on ppc64le (#2315)
  - fix OpenFOAM sanity check on POWER (#2320)
  - use library search paths of compiler for RPATH when building binutils with system compiler + enhance sanity check by running `--version` for binutils commands (#2323, #2327)
  - pass `$CXXFLAGS` to PDT's configure script via `-useropt` (#2324)
  - pass down compilation flags from build environment for ESMF (#2325)
  - update URLs for test data for WRF to `https` (#2335)
  - read MATLAB configuration file in binary mode to avoid UTF-8 encoding errors when using Python 3.6 (#2340)
  - fix Boost sanity check on POWER (#2291) and aarch64 (#2341)
- other changes, including:
  - rework module-only tests to use unique software name (rather than 'foo') (#2287)

  - prefer default value for extra options in easyblock tests (#2280, #2302)
  - add check for accepted EULA in custom easyblock for NVHPC (#2311)
  - update optional feature support of TensorFlow (#2314)
  - make symlinking of `posix_c.so` to `posix.so` in test suite configuration conditional (#2330)

easyconfigs

- add easyconfig for new iibff toolchain: iibff/2020b (#12185)
- added easyconfigs for 6 new toolchains:
  - gobff/2020b (#12098), goblf/2018b (#6615), gomkl/2020b (#12198), iimkl/2018a (#6092), iomkl/2019b (#11981), iomkl/2020b (#12009)
- added example easyconfig files for 72 new software packages:
  - AOCC (#11868), CHERAB (#7141), CaDiCaL (#11966), CellRanger-ARC (#12114), CuPy (#11749), DFA (#11979), FIGARO (#11924), FIRESTARTER (#12160), FIt-SNE (#8630), GPyTorch (#12010), GSEA (#10395), GetOrganelle (#11948), Glucose (#11965), GraPhlAn (#10707), ITSx (#10558), Kaleido (#11998), LAPACK (#6615), LSD2 (#11903), Lingeling (#11964), MPB (#7026), MiniCARD (#11963), MiniSat (#11962), NGSpeciesID (#11918), NanopolishComp (#11823), Nsight-Compute (#12043), Nsight-Systems (#12042), PHANOTATE (#8667), PIPITS (#10558), PyClone (#11940), PySAT (#12000), RNAmmer (#7262), Raysect (#7141), SICER2 (#12200), SOCI (#12045), SeisSol (#7194), SignalP (#11862), Stack (#11310), SuperLU_DIST (#11693), Teneto (#12056), Transformers (#12032), YANK (#11742), Z3 (#12013), bgen (#7456, #11867), bgen-reader (#7456, #11867), bpytop (#12040), byobu (#11932), chi2comb (#11867), cuTENSOR (#11914), dd (#11978), decona (#11891), dicom2nifti (#11955), eccodes-python (#12083), fftlib (#11944), flatbuffers-python (#12148), fpocket (#11980), gh (#11851), intel-compilers (oneAPI) (#11982), libGDSII (#7026), libpci (#11871), liknorm (#7456, #11867), limix (#7456, #11867), logaddexp (#11867), neptune-client (#11985), plinkQC (#12068), preCICE (#11886), py-aiger (#11999), py-aiger-bdd (#11999), pytest-xdist (#11883, #11893), samblaster (#7378), terastructure (#12197), typing-extensions (#11636), webin-cli (#8674)
- added additional easyconfigs for various supported software packages, including:
  - ack 3.4.0, AmberTools 20, AMD-LibM 3.6.0-4, AMD-RNG 2.2, AMD-SecureRNG 2.2, annovar 20191024, ASE 3.21.1, AUGUSTUS 3.4.0, Bazel 3.7.2, bcgTree 1.1.0, BLAST+ 2.11.0, Bonito 0.3.5, Bowtie2 2.4.2, causalml 0.8.0-20200909, CGAL 5.2, ConnectomeWorkbench 1.4.2, CUDAcore 11.2.1, cuDNN 8.0.5.39, dcm2niix 1.0.20201102, DendroPy 4.5.2, DIAMOND 0.9.36 + 2.0.6, ecCodes 2.20.0, ELPA 2020.11.001, Emacs 27.1, FusionCatcher 1.30, gensim 3.8.3, GHC 8.6.5, gnuplot 5.4.1, GPAW 21.1.0, Graphviz 2.44.1, GROMACS 2020.5, Gurobi 9.1.0, HH-suite 3.3.0, HMMER 3.3.2, HTSlib 1.11, Horovod 0.21.1, imbalanced-learn 0.7.0, impi 2021.1.1, inferCNV 1.3.3, ITK 5.1.2, IQ-TREE 2.1.2, JasPer 2.0.24, JUBE 2.4.1, libgit2 1.1.0, libzip 1.7.3, likwid 5.1.0, MariaDB 10.5.8, medaka 1.2.0, Meep 1.6.0, mkl-service 2.3.0, MPICH 3.3.2, muParser 2.3.2, NanoComp 1.13.1, NanoPlot 1.33.0, networkx 2.5, NLTK 3.5, numba 0.52.0, NVHPC 20.11, nvtop 1.1.0, OpenCoarrays 2.9.2, OpenFOAM v2012, OpenJPEG 2.4.0, OpenMPI 4.1.0, parasail 2.4.2, PLUMED 2.6.2, PostgreSQL 13.2, pydicom 2.1.2, PyMC3 3.11.0, python-parasail 1.2.2, PyTorch 1.7.1, QIIME2 2020.11, QuantumESPRESSO 6.7, QuickFF 2.2.7, R 4.0.3, R-bundle-Bioconductor 3.12, ReFrame 3.4.1, RMBlast 2.10.0, RSEM 1.3.3, Salmon 1.4.0, scikit-build 0.11.1, SciPy-bundle 2020.03 w/ Python 2.7.18, Seaborn 0.11.1, SEPP 4.4.0, SHAPEIT4 4.2.0, SpaceRanger 1.2.2, Stacks 2.54, STAR 2.7.7.a, statsmodels 0.12.1, SuiteSparse 5.8.1, tbb 2020.3, TensorFlow 1.15.5 + 2.4.1, Theano 1.1.2, torchvision 0.8.2, V8 3.4.0, Wannier90 3.1.0
- minor enhancements, including:
  - enable building of QtWebEngine in Qt5 easyconfig using foss/2017b or intel/2017b (#7302)
  - enable NVPTX offload in GCCcore 9.3.0 easyconfig (#11839)
  - also build shared library in recent HDF easyconfigs (#11847)

  - add support for HDF4 to GDAL v3.0.2 and v3.0.4 (#11855)
  - add patch for magma 2.5.4 with fosscuda/2019b to allow any `sm_*` value to be passed via `GPU_TARGET` (#11861)
  - add missing M4 build dependency to recent SuiteSparse versions (#11869)
  - enable tests for most recent Bazel versions (3.x) (#11894)
  - enable gipaw in QuantumESPRESSO 6.6 easyconfig (#11905)
  - add RCAL + sensemakr extensions to R 4.0.0 (#11921)
  - add additional extensions to R v4.0.3 easyconfig (#11922, #12057)
  - improve sanity_check_paths for AMD-LibM (#11933)
  - symlink `include`/`lib` subdirs + enhance `sanity_check_paths` for AMD-RNG v2.2-4 (#11934)
  - improve `sanity_check_paths` for AMD-SecureRNG (#11935)
  - update Clang 10+ dependencies & build extra tools (#12013)
  - add Inline extension to recent Perl easyconfigs (5.30+) (#12029)
  - add EnsDb.Hsapiens.v86 to Bioconductor 3.11 (#12078)
  - add sanity check command for bam-readcount v0.8.0 to check `--version` output (#12092)
  - add EnsDb.Hsapiens.v75 + Signac extensions to R-bundle-Bioconductor 3.12 (#12174)
  - add tensorboard profile plugin to recent TensorFlow 2.x easyconfigs (#12136, #12137)
- various bug fixes, including:
  - added missing space in `configopts` in ParaView 5.8.0 easyconfigs using 2020a toolchain (#10989)
  - use bfd linker for glibc 2.30 (#11331)
  - add missing moduleclass in UCLUST easyconfig (#11477)
  - don't disable optarch for Clang 11.0.0 (#11814)
  - add patch to fix miscompilation bug on POWER for GCC 8.x and 9.x (#11837)
  - fix compilation of TensorFlow 2.3.1 with CUDA and glibc 2.26 on POWER (#11859)
  - disable building of manpages for GDCM to fix installation problem with docbook (#11866)
  - add patch for LLVM 6.0.0 to fix missing exported symbol `LLVMInitializeInstCombine` (#11873)
  - fix name of source file for GDRCopy v2.1 (#11887)
  - fix Tombo to work with rpy2 v3 when creating DataFrames (#11892)
  - fix GCCcore 8.1.0 w.r.t. removed `sys/ustat.h` in glibc 2.28 (#11896)
  - add git as a dependency to GitPython version 3.x (#11902)
  - fix undefined `__ieee128` on ppc64le with glibc 2.26 for magma (#11930) and PyTorch (#11936)
  - update Hypre git location for PETSc 3.11.0 using downloaded dependencies (#11947)
  - fix source URL in expat easyconfigs (and consistently add custom `sanity_check_paths`) (#11960)
  - use libpng provided by EasyBuild in VTK to fix build issue on ppc64le (#11990)
  - add missing pkg-config build dependency for GObject-Introspection v1.66.1 (#11949)
  - add missing plotly-orca dependency for NanoPlot (#11967, #12015)

  - add patch for Boost 1.74.0 to fix missing include file (#12007)
  - fix for `error: 'runtime_error' is not a member of 'std'` in qtlocation for Qt5 v5.14.2 (#12012)
  - fix homepage for gnuplot 5.2.8, use `http://` since homepage is not reachable via `https://` (#12047)
  - fix OS dependencies for libfabric (#12058)
  - fix vector `mul` and `div` with broadcasts in `-masm=intel` mode in GCCcore v9.3.0 (#12065)
  - add missing ESMF dependency in NCO easyconfigs (#12071, #12072)
  - add patch to fix version for bam-readcount 0.8.0 (#12075)
  - add missing Perl build dep for (recent) libcerf versions, required for `pod2html` command (#12085)
  - add missing bzip2 dependency to recent ncbi-vdb easyconfigs (#12120)
  - add missing Perl build dependency for BLIS 0.8.0 (#12146)
  - use `https://sources.easybuild.io` as fallback source URL for UDUNITS (#12049, #12156, #12182)
  - correct the GCC version check to allow IBM VSX builds of GROMACS 2020.4 and 2020.5 (#12159)
  - add missing mkl-service dependency for Theano built with intel/2019b + enhance sanity check (#12172)
  - add patches to fix test problems with p4est 2.2 (#12028)
  - fix build of Bison using older system GCC (v4.x) (#12203)
  - fix lack of optimisation for SHAPEIT4 v4.1.3 (#12206)
  - add missing gnuplot dependency for OpenFOAM from v2.4.0 to v6 (#11801, #12208)
  - fix source URL for libspatialite (#12213)
  - add `archive` fallback source URL to MAGMA easyconfigs (#12214)
  - fix checksum for patch in make 4.2.1 easyconfig using GCC/7.3.0-2.30 (#12223)
- other changes:
  - replace easyconfigs for bpp-core/bpp-phyl/bpp-seq v2.4.1 with a single easyconfig for BioPP v2.4.1 (using Bundle easyblock) (#11609)
  - add CESM-deps to whitelist in check for custom `sanity_check_paths` (#11916)
  - include '-4' in version for AMD-SecureRNG v2.2-4 (#11934, #11935)
  - switch to using python-parasail and tqdm dependencies in Bonito easyconfigs (#11937)
  - rename orca to plotly-orca (#12015)
  - remove duplicate extensions in R 3.5.x easyconfigs, and add test to detect such issues (#12059)
  - remove Python dependency from ecCodes v2.17.0 since it doesn't provide Python bindings (#12084)
  - update Java/1.8 wrapper to Java 1.8.0_281 (#11928, #12088)
  - update Bison (build) dependency for flex built with system compiler to v3.5.3 (#12111)
  - make symlinking of `posix.so` in test suite configuration conditional (#12123)
  - move make 4.3 easyconfigs to GCCcore toolchain (#12166)
  - move most recent BLIS and libFLAME easyconfigs from GCC to GCCcore (#12168)
  - rename SNAP to SNAP-HMM and update easyconfig (#12218)

11.12.6 EasyBuild v4.3.2 (December 10th 2020)

bugfix/update release

framework

- add (experimental) support for specifying easyconfig files via an "easystack" file (#3479, #3511, #3515, #3517, #3520, #3521)
  - see also https://easybuild.readthedocs.io/en/latest/Easystack-files.html
- add definition for new `gobff` toolchain using BLIS and LibFLAME (#3505)
- various enhancements, including:
  - add support for toolchain options like `extra_cxxflags` to specify extra compiler options (#2193)
  - fix combination of `--copy-ec` and `--from-pr` (#3482)
  - enhance `copy_files` function: support single file target, error on empty input list, support verbose mode (#3483)
  - cache result of `fetch_files_from_pr` function (mainly to speed up framework test suite) (#3484)
  - add `locate_files` function to filetools module (#3485)
  - add support for `%(module_name)s` template value (#3497)
  - clarify input format for `--cuda-compute-capabilities` in `eb --help` output (#3509)
  - add support for skipping unit tests (test step) via `--skip-test-step` (#3524)
- various bug fixes, including:
  - also ignore `vsc.*` imports coming from from `pkg_resources/__init__.py` (setuptools) in fake vsc namespace (#3491)
  - don't pass username in `github_api_get_request` when no GitHub token is available (#3494)
  - also inject `-rpath` options for all entries in `$LIBRARY_PATH` in `RPATH` wrappers (#3495)
  - avoid `TypeError` being raised by `list_toolchains` (#3499)
  - check if PR is already merged in `--merge-pr` (#3502)
  - graciously handle wrong PR id in `fetch_pr_data` (#3503)
  - fix regression in `apply_regex_substitutions`: also accept list of paths to patch (#3507)
  - update installation procedure for EasyBuild in generated Singularity container recipes (#3510)
  - fix GitHub Actions workflow for test suite: run outside of repo checkout + also test bootstrap script with Python 3.9 (#3518)
  - bump cryptography from 2.9.2 to 3.2 for Python 2 in `requirements.txt` (#3519)
  - fix `eb --help=rst` when running with Python 3 (#3525)
- other changes:
  - exclude test configurations with Lmod 7 and Python 3, except for Python 3.6 (#3496)
  - significantly speed up parsing of easyconfig files by only extracting comments from an easyconfig file when they're actually needed (#3498)
  - don't include `file`/`ldd`/`readelf` commands run during RPATH sanity check in `--trace` output (#3508)

easyblocks

- 2 new software-specific easyblocks:
  - code-server (#2255), Metagenome-Atlas (#2219)
- minor enhancements, including:
  - add `-fallow-argument-mismatch` option when building CP2K 7.1 or older with GCC 10.x (#2223)
  - update TensorFlow easyblock for upcoming TensorFlow 2.4 (#2225)
  - add support for building Clang with OpenMP offload support (#2229)
  - enhance OpenMPI easyblock to catch any `--with-ucx*` configure options (#2230)
  - take into account `preinstallopts` and `installopts` in custom easyblock for NCL (#2234)
  - add support for `withnvptx` easyconfig parameter, to enable GPU offloading, in GCC easyblock (#2235)
  - take into account versions like `4.x` in OpenFOAM easyblock (#2239)
  - also add `bin` subdir to `$PATH` when installing a Python package (#2244)
- various bug fixes, including:
  - fix two bugs in GROMACS easyblock when using GCC & MKL for FFT and BLAS/LAPACK (#2212)
  - fix version check in Qt5 easyblock w.r.t. disabling features on old Linux kernel versions (#2220)
  - always define `$FCCPP` in QuantumESPRESSO easyblock (not just when using Intel compilers) (#2221)
  - allow wxPython to be installed as an extension (#2227, #2275)
  - only configure Python with `--enable-optimizations` when compiling Python with (recent) GCC compiler (#2228)
  - fix sanity check for Boost MT libraries (#2231)
  - fix hardcoded path in NVHPC easyblock to support multiple architectures (#2233)
  - fix `CPASSERT` test faults on RHEL8 in CP2K easyblock (#2236)
  - stop silently ignoring failing numpy tests, but include support for ignoring (failing) numpy tests (#2238, #2271)
  - append to module guesses in easyblocks for Chapel, icc, imkl and impi (rather than overwriting guesses from parent easyblock) (#2242)
  - weed out duplicates when determining paths to `include-fixed` subdirectory in GCC easyblock (#2245)
  - prepend all hardcoded `/usr/*` paths with sysroot in Python's setup.py installation script (#2246)
  - don't try to patch newer versions of Bazel where the patches won't apply (#2249)
  - fix setting of `$RUNPARALLEL` in HDF5 easyblock (#2250)
  - move `--build` and `--host` logic to `run_configure_cmd` in GCC easyblock (#2252)
  - set `$UCX_TLS` in module for impi installation on top of UCX, and allow it to be overwritten in impi easyconfig file (#2253, #2258)
  - enhance PyTorch easyblock to ensure it finds MKL (via `$MKLROOT`) (#2257)
  - use integer division to determine number MPI ranks to use in WRF test step (#2266)
  - also specify `locincpth` and `glibpth` configure options for Perl based on `--sysroot` (#2268)
- other changes, including:

  - add link to GCC mailing list thread confirming that binutils should not be configured with `--with-sysroot=$EASYBUILD_SYSROOT` when GCC is being configured like that (#2215)
  - pass paths to patch one by one to `apply_regex_substitutions` in GCC easyblock when `--sysroot` is set (#2217)
    * workaround for regression in `apply_regex_substitutions` introduced in EasyBuild v4.3.1 (which was fixed for v4.3.2)

easyconfigs

- add easyconfigs for new *gobff* toolchain: *gobff/2020.11* + *gobff/2020.06-amd* (#11761)
- add easyconfigs for updates of common toolchains: *foss/2020b*, *fosscuda/2020b*, *intel/2020b*, *intelcuda/2020b*
  - see also https://easybuild.readthedocs.io/en/latest/Common-toolchains.html
- added example easyconfig files for 41 new software packages:
  - alsa-lib (#11658), assimp (#11759), BioServices (#11602), carputils (#11270), cctools (#11799), code-server (#11778), CRISPResso2 (#11775), elfutils (#11783), EMU (#11641), fgbio (#11519), Flink (#11747), FreeSASA (#11699), Geant4-data (#11610), geocube (#11714), IPM (#11768, #11772), libmicrohttpd (#11783), libStatGen (#7982), libzeep (#11687), LLDB (#11822), meshalyzer (#11270), meshtool (#11270), Metagenome-Atlas (#11620), MLxtend (#11601), MRChem (#11604), MRCPP (#11579), mxml (#11769), mxmlplus (#11771), nanocompore (#11690), neon (#11797), Open-Data-Cube-Core (#11713), OpenSees (#11613), PEST++ (#11565), pyFAI (#11849), pymca (#11848), RE2 (#11718), scikit-bio (#11660), SpaceRanger (#11776), SplAdder (#11607), SWAT+ (#11615), velocyto (#11744), WCT (#11779)
- added additional easyconfigs for various supported software packages, including:
  - BBMap 38.87, Boost 1.74.0, CUDA 11.1.1, CellRanger 5.0.0.eb, CheckM 1.1.3, Clang 11.0.0, cuDNN 8.0.4.30, DMTCP 2.6.0, FDS 6.7.5, FFmpeg 4.3.1, GROMACS 2020.4, Geant4 10.6.2, Ghostscript 9.53.3, GitPython 3.1.9, GlobalArrays 5.8, HDF5 1.10.7, h5py 3.1.0, hypothesis 5.41.5, JasPer 2.0.16, LittleCMS 2.11, libedit 20191231, libyaml 0.2.5, MPFR 4.1.0, magma 2.5.4, matplotlib 3.3.3, NCCL 2.8.3, NLopt 2.6.2, Nextflow 20.10.0, netCDF-Fortran 4.5.3, OpenBLAS 0.3.12, OpenMolcas 20.10, Pillow 8.0.1, PnetCDF 1.12.1, PyYAML 5.3.1, packmol 20.2.2, protobuf 3.14.0, psycopg2 2.8.6, pybind11 2.6.0, pycocotools 2.0.2, Qt5 5.14.2, RAxML-NG 1.0.1, RSeQC 4.0.0, rioxarray 0.1.1, SCOTCH 6.1.0, SciPy-bundle 2020.11, SentencePiece 0.1.94, StringTie 2.1.4, SuperLU 5.2.2, scikit-learn 0.23.2, snakemake 5.26.1, tqdm 4.51.0, vsc-mympirun 5.2.5, x264 20201026
- minor enhancements, including:
  - add extension to R-bundle-Bioconductor 3.11: snpStats (#11586), SCANVIS (#11638)
  - add extensions to R 4.0.0: coloc, Exact, lmom, gld, DescTools (#11587); nlsem (#11733); mitools, survey, tableone, jstable (#11841)
  - add gxmapi extension (Python bindings) to GROMACS 2020.4 (#11640)
  - add missing extensions for QIIME2 to Perl 5.30.2 and 5.32.0 easyconfigs (#11654)
  - enable NVPTX offload support in GCCcore 10.2.0 easyconfig (#11720)
- various bug fixes, including:
  - prevent Minimac4 easyconfig from downloading libStatGen from GitHub (#7982)
  - limit build parallelism for RAxML-NG to avoid build failure (#10363)
  - fix `source_urls` in Geant4 easyconfigs (#11596)
  - add alternate checksum for extensions in R easyconfigs: KernSmooth (#11600), codetools (#11616, #11736)

  - fix BBMap version check in FusionCatcher v1.20 (#11608)
  - add patch for TensorFlow 2.3.1 to fix installation on Arm64 (#11614)
  - add patch for GCCcore 10.2 to fix `__has_include` regression (#11627)
  - change default `bitmaptype` for IRkernel to cairo (#11645)
  - set `$JUPYTER_PATH` to make Jupyter find the ipywidgets extension in recent IPython easyconfigs (#11649)
  - add missing dependencies + fix moduleclass in CheckM easyconfigs (should be bio) (#11662)
  - use Python 2 build dep for nodejs 12.19.0 (#11679)
  - use correct OS deps in UCX (#11702)
  - in-place update to magma 2.5.4 for PyTorch 1.2.0 (#11723)
  - in-place update to magma 2.5.4 for PyTorch easyconfigs using fosscuda/2019b toolchain (#11726)
  - add missing Java dep to Prokka (#11732)
  - add patch to fix possible memory leak in OpenBLAS 0.3.3+ (#11745)
  - do not treat warnings as errors in Java v1.8_191-b26-OpenJDK (POWER) (#11755)
  - add missing PEAR dependency in easyconfig for GBprocesS v2.3 + enhance sanity check (#11767)
  - add missing gnuplot dependency in OpenFOAM easyconfigs (#11770, #11800)
  - consistently add Java as a dep for prokka 1.14.5 (#11782)
  - stick to http download URL in fetchMG easyconfig (https doesn't work) (#11788)
- other changes:
  - rename XCfun to XCFun (#11603)
  - rename ambiguous Ray easyconfigs to Ray-assembler and Ray-project (#11727)
  - change 'rstudio' name to 'RStudio-Server' to agree with official name and better reflect what it provides (#11764)
  - rename Sumo to SUMO for consistency (#11791)

11.12.7 EasyBuild v4.3.1 (October 29th 2020)

bugfix/update release

framework

- various enhancements, including:
  - further GCC toolchain fixes for aarch64 (#3433)
  - take into account `--include-easyblocks-from-pr` when uploading test reports (#3446)
  - add path to pkg-config files in sysroot to `$PKG_CONFIG_PATH` when `--sysroot` is specified (#3451)
  - add support for NVHPC compiler + toolchain (based on PGI) (#3454)
  - check for _VERSION and _PREFIX Cray environment variables with both software and module name (#3460)
  - allow including easyblocks from multiple PRs (#3480, #3481)
- various bug fixes, including:

  - avoid `UnicodeDecodeError` in `apply_regex_substitutions` when patching files that include non-UTF-8 characters (#3450)
  - avoid appending lib stubs pattern to RPATH filter over and over again (#3452)
  - fix missing string template on error for incorrect extension `sources` value (#3461)
  - fix compatibility with Python 3.9 by renaming fancy root logger (#3465)
  - also remove empty checksums list specified in easyconfig file when using `--inject-checksums` (#3466)
  - avoid confusing error log message when determining impi version while trying to define value for `%(mpi_cmd_prefix)s` template (#3474)
  - unset `$LD_LIBRARY_PATH` when checking for OS dependencies with `rpm` & co (#3477)
  - don't change directory in `download_repo` function in `tools.github` (#3486)
  - take `source_urls`, `checksums`, `patches` into account when extension source is specified via `sources` (#3487)
- other changes:
  - consider `$EB_INSTALLPYTHON` in `eb` command to specify `python` command to use for running EasyBuild (#3428)
  - use only the sub folder name for `createSubmoduleDeps` script (#3464)

easyblocks

- 2 new software-specific easyblocks:
  - CFDEMcoupling (#1439), NVHPC (#2190)
- minor enhancements, including:
  - support having PyQt5 installed as part of Qt5 in QScintilla easyblock (#2040)
  - update TensorFlow easyblock to put Bazel build files in build directory + avoid unnecessary runtime patching (#2166)
  - update CBLAS easyblock for toolchains that include imkl (#2175)
  - add workaround for duplicate prefix path in Eigen CMake config (#2176)
  - enable `CTEST_OUTPUT_ON_FAILURE` for CMakeMake test step (#2181)
  - add XLA build support to TensorFlow easyblock (#2182)
  - also consider libfabric dep (`--with-ofi`) when setting default OpenMPI configure options (#2184)
  - make easyblock for installing CMake aware of `--sysroot` (#2187)
  - make OpenBLAS respect the parallelism set by easybuild (#2191)
  - handle option of building Michigan State University CCT3 & CCSD3A methods in GAMESS_US easyblock (#2194)
  - add support to RubyGem easyblock for installing zipped gems (#2203)
  - update ScaLAPACK easyblock to support installation with CMake for recent versions (>= v2.1.0) (#2205)
  - update Score-P easyblock to add support for NVHPC toolchain (#2206)
- various bug fixes, including:
  - add `torch/lib` subdirectory in Python lib dir to `$LD_LIBRARY_PATH` for PyTorch installations (#2183)

  - update MUMmer easyblock to use `apply_regex_substitutions` and `copy_file` functions (#2185)
  - configure OpenMPI 4.x with `--without-verbs` when using UCX (#2188)
  - take into account that zlib may be listed in `--filter-deps` in custom easyblock for XML R package (#2189)
  - add sanity check commands to Clang easyblock and print warning for missing ncurses (#2193)
  - enhance OpenFOAM easyblock to add symlinks for libraries to ensure `mpi` versions have preference over `dummy` versions (#2196)
  - leverage installed OpenSSL/BoringSSL when cURL is used as a dependency for TensorFlow (#2197)
  - add `-ffree-line-length-none` to gfortran flags in Siesta easyblock (#2204)
  - respect `--disable-mpi-tests` in Siesta easyblock (#2207)
- other changes, including:
  - set `$EB_INSTALLPYTHON` in module generated for EasyBuild rather than setting `$EB_PYTHON`, to allow overriding Python command to be used for running EasyBuild with `$EB_PYTHON` (#2109)
  - remove unused custom easyblock for DIRAC (#2192, #2198)

easyconfigs

- added easyconfigs for 2 new toolchains:
  - fosscuda/2020a (#11424) and intelcuda/2020a (#11425)
- added example easyconfig files for 75 new software packages:
  - Bsoft (#6551, #11537), causallift (#11436), CAVIAR (#11158), CCfits (#11505), Cereal (#11506), CFDEMcoupling (#6465), Check (#11295), chewBBACA (#11418), COMSOL (#11513), CUDAcore (#11295), Dalton (#5808), DIRAC (#11414), ESMValTool (#6329), eSpeak-NG (#11236), ExifTool (#11521), FabIO (#11517), festival (#11236), fetchMG (#11283), FHI-aims (#11198), Flexbar (#11305), FRUIT (#6613), FRUIT_processor (#6631), Gaussian (#4247), GBprocesS (#11512), GDRCopy (#11295), Genome_Profiler (#6066), groff (#11200), ILAMB (#11309), kedro (#11436), leidenalg (#11407), libav (#6194), libmo_unpack (#6329), libobjcryst (#11321), makeinfo (#11368), mauveAligner (#11395), MBROLA (#11236), MDSplus-Java (#10705), MDSplus-Python (#10705), MyCC (#11283), nanomax-analysis-utils (#11517), Nek5000 (#6408), NVHPC (#11391), OPERA-MS (#11410), ORFfinder (#7031), phonemizer (#11236), Pingouin (#11280), Pint (#11151), pydantic (#11151), pyobjcryst (#11321), PyOpenCL (#11517), pyspoa (#11438), QtPy (#11517), Qtconsole (#11517), rasterio (#11468), rasterstats (#11468), ReMatCh (#6067), Ruby-Tk (#6613), SBCL (#11413), scikit-uplift (#11432), SDL (#6202), SDL2 (#6203), SDL2_image (#6203), SDL_image (#6202), SeqKit (#11538), silx (#11517), SiNVICT (#11404), sonic (#11236), speech_tools (#11236), Statistics-R (#11274), SUMO (#11435), tidybayes (#11335), treatSens (#11431), tsne (#11283), wandb (#11450), WisecondorX (#11399)
- added additional easyconfigs for various supported software packages, including:
  - ABINIT 9.2.1, ASE 3.20.1, Autotools 20200321, Bazel 3.6.0, Biopython 1.78, CCL 1.12, CMake 3.18.4, CUDA 11.0.2, cURL 7.72.0, deepdiff 5.0.2, fmt 7.0.3, GATK 4.1.8.1, GLib 2.66.1, GROMACS 2020.3, GTK+ 3.24.23, git 2.28.0, HMMER 3.3.1, Horovod 0.20.3, ICU 67.1, IPython 7.18.1, ichorCNA 0.3.2, JupyterHub 1.1.0, JupyterLab 2.2.8, LLVM 11.0.0, libarchive 3.4.3, libcircle 0.3, libevent 2.1.12, libfabric 1.11.0, libglvnd 1.3.2, libunwind 1.4.0, libxcb 1.13, MAFFT 7.471, MDSplus 7.96.12, MEGAHIT 1.2.9, MMseqs2 11-e1a1c, Mako 1.1.3, Mesa 20.2.1, Meson 0.55.3, medaka 1.1.3, mpifileutils 0.10.1, NASM 2.15.05, NSPR 4.29, NSS 3.57, Ninja 1.10.1, nettle 3.6, nglview 2.7.7, nodejs 12.19.0, OpenImageIO 2.1.12.0, OpenRefine 3.4.1, OpenSSL 1.1.1h, PCRE2 10.35, PSI4 1.3.2, Pango 1.47.0, PyGEOS 0.8, PyOpenGL 3.1.5, PyQt5 5.15.1, PyQtGraph 0.11.0, PyRETIS 2.5.0, Python 3.8.6, pandas 1.1.2, phonopy

2.7.1, picard 2.22.1, pixman 0.40.0, protobuf 3.13.0, pyEGA3 3.4.0, pytest 6.0.1, ReFrame 3.2. re2c 2.0.3, SAMtools 1.11, SCons 4.0.1, SQLite 3.33.0, Spyder 4.1.5, Subversion 1.14.0, sbt 1.3.13, spglib-python 1.16.0, spoa 4.0.0, TINKER 8.8.1, TRIQS 3.0.0, Taiyaki 5.1.0, TensorFlow 2.3.1, Tkinter 2.7.18 + 3.8.6, torchvision 0.7.0, UCX 1.9.0, V8 3.2.0, VirtualGL 2.6.2, vsc mympirun 5.2.0 X11 20201008, XGBoost 1.2.0, XZ 5.2.5, Xerces-C++ 3.2.3, xarray 0.16.1, ZeroMQ 4.3.3, zstd 1.4.5

- minor enhancements, including:
  - use more EasyBuild installed dependencies for TensorFlow 2.2.0 (#11224)
  - add additional extensions to R 4.0.0 easyconfig (#11340, #11430, #11487)
  - add additional extensions to Bioconductor 3.11 bundle (#11341, #11488)
  - make libtirpc easier to use as replacement of rpc in glibc (for RHEL8) (#11355)
  - add libevent, libfabric and PMIx dependencies to OpenMPI 4.0.3 (for foss/2020a & iomkl/2020a) (#11387, #11568)
  - build LibTIFF with `-fPIC` (#11527)
  - update Java/1.8 wrapper to also support aarch64 (#11545)
- various bug fixes, including:
  - add Python 3 as build dep for Xvfb 1.20.8 easyconfigs (#10745)
  - patch out bug in collective primitive in TensorFlow 2.2.0 (#11175)
  - add missing groff build dependency for Perl (provides nroff tool requires to install man pages) (#11200)
  - add pocl dependency to GDAL easyconfig using fosscuda/2019b toolchain to make sure it builds on POWER (#11273)
  - add libtirpc as a build dep for HDF to fix installation on RHEL8 (#11279)
  - fix build problems with make 4.2.1 on RHEL8 (#11282, #11371)
  - change `$LIBS` in Ghostscript 9.27 easyconfig to include location of zlib library to link with (#11291)
  - remove `--no-dist-info` configuration option for SIP in recent PyQt5 easyconfigs (#11307)
  - make sure correct zlib is used in recent Ghostscript easyconfigs (#11319)
  - remove GLog and GFlags from PyTorch 1.2.0 (#11327), 1.3.1 (#11325), 1.4.0 (#11322) and 1.6.0 (#11323)
  - use PyTorch easyblock for PyTorch 1.3.1 (#11325)
  - add missing git build dependency for OpenImageIO 2.0.12 (#11328)
  - fix HDF 4.2.14 easyconfigs for RHEL8 (#11330)
  - add patch to fix installation of LAMMPS 7Aug2019 on AMD Epyc systems (#11334)
  - fix dependency on Bowtie of v0.x in seq2HLA easyconfigs (#11339)
  - fix installation of Hadoop v2.10.0 on RHEL8 (#11358)
  - add patch to fix installation of Sailfish v0.10.1 on RHEL8 (#11364)
  - add patch to fix installation of FuSeq v1.1.2 on RHEL8 (#11365)
  - fix installation of Qt5 5.14.1 on top of zlib provided by Gentoo + stick to Python 2 as build dep (#11385, #11386)
  - add patch for LAME configure script to make it check for correct ncurses symbol (#11388)
  - add protobuf-python as a dependency and corresponding sanity check to PyTorch 1.6 (#11390)

  - fix Multiwfn 3.6 installation on RHEL8 (#11402)
  - fix OpenFOAM 2.3.1 installation with intel/2019b on RHEL8 (#11409)
  - fix broken worker easyconfigs (#11412)
  - add ncurses runtime dependency to Clang easyconfigs (#11415, #11416, #11419, #11472)
  - fix installation of Bioconductor 3.11 bundle on aarch64 (#11417)
  - add patch to libunwind fixing a failure on POWER (#11421)
  - fix infinite loop build bug on ppc64le for R 4.0.0 (#11428)
  - fix compilation of Qt5 v5.12.3 and v5.13.1 on Ubuntu 20.04 (#11434)
  - fix PyVCF easyconfig, only supports Python 2 (#11437)
  - add patches to fix miscomputation (on POWER) and performance issues for OpenBLAS (#11443, #11444, #11445)
  - add missing DB dependency (required for DB_File) to easyconfigs for Perl (#11451, #11452)
  - fix dbarts extension in R v4.0.0 easyconfigs for non-x86_64 (#11453)
  - use Homebrew source mirror to auto-download sources for DB v18.1.25 and v18.1.32 (#11454)
  - add missing UnZip dependency for Python 3.8.2 (#11458)
  - add support for building OpenBLAS on ARM TSV110 with GCC 8.3 (#11464)
  - allow Kent_tools to build when MySQL installed at the OS level (#11471)
  - add missing `moduleclass` to easyconfigs for:
    * Arlequin 3.5.2.2 (#11473), MEGAHIT 1.2.8 (#11474) and 1.2.9 (#11475), PyCUDA 2019.1.2 (#11476)
  - fix source for SQLite 3.31.1 (#11483)
  - fix installation of worker 1.6.11 with intel/2019b on RHEL 8.2 (#11498)
  - fix checksum in NAMD 2.12 easyconfigs + add source URL (#11515)
  - add BLAS/LAPACK check to GPAW patch adding EasyBuild configuration files (#11523)
  - backport fixes for Score-P v6.0 (#11540)
  - add XZ build dep to easyconfigs for libarchive v3.4.0 and v3.4.2 (#11561)
  - add patch for OpenMPI 3.1.4 adding device parameters for ConnectX-6 (#11575)
- other changes:
  - update README on constructing source file for MATLAB (#6341)
  - remove easyconfigs for Taiyaki that depend on PyTorch 1.3.1, since latest version still requires PyTorch 1.2.0 (#11301)
  - make CI error messages less confusing ("is" -> "should be") (#11314)
  - remove extensions from R-bundle-Bioconductor 3.11 easyconfigs that are also included in R 4.0.0 (#11429)
  - cache M4 source tarball to avoid test failures because of download problems (#11469)
  - rename jupyterhub to JupyterHub (#11571)

11.12.8 EasyBuild v4.3.0 (September 13th 2020)

feature release

framework

- various enhancements, including:
  - add script to create `sources` entries for git submodules (#3369, #3436)
  - add templates for CUDA compute capabilities (#3382)
    - `%(cuda_compute_capabilities)s`, `%(cuda_sm_comma_sep)s`, `%(cuda_sm_space_sep)s`
  - add EasyBuild configuration option `--generate-devel-module` (#3388)
    - to allow disabling generating of "devel" modules via `--disable-generate-devel-module`
  - set up a minimal build environment when using system compiler (#3399)
    - by default, `$CC` is set to `gcc` and `$CXX` is set to `g++` when using `system` toolchain
    - minimal build environment can be customized via `--minimal-build-env` configuration option
  - add `--sysroot` configuration option to specify alternative location of system root (#3419)
    - this is useful when installing software in a Gentoo Prefix environment (for example)
- various bug fixes, including:
  - allow including easyblocks from multiple locations, by combining `--include-easyblocks` and `--include-easyblocks-from-pr` (#3311)
  - also escape backslashes in `quote_py_str()` (#3386)
  - use one argument `module swap` statements in Tcl modulefiles (required by Modules 4.2.3+) (#3397)
  - fix copying of (non-existing) file with `apply_patch` (#3400)
  - create symlink from `lib64` to `lib` subdir in installation directories to avoid that GCC prefers `/lib64` system directories (#3401)
  - fix default value for `lib64_fallback_sanity_check` build option (#3402)
  - correctly determine commit status in `--merge-pr` (#3406)
  - stop installing ccache wrapper for Fortran compiler (#3409)
  - fix issues with applying PR patch in `--from-pr` (#3414)
  - make `RPATH` wrapper script more robust by using `python -E -s -S` to run `rpath_args.py` (#3422)
  - don't inject `-Wl,-rpath` options when `-x c++-header` compiler option is used (#3424)
  - fix lack of `-mno-recip` on aarch64 by tweaking GCC options used when `precise` toolchain option is enabled (#3425)
  - make sure `self.start_dir` is set in `ExtensionEasyBlock` (#3426, #3435)
  - exclude local variables from typo check in easyconfig files (#3427)
- other changes:
  - stick to cryptography 2.9.2 when using Python 2.7 to avoid broken test suite in CI (#3392)
  - automatically enable `--ignore-locks` with `--fetch` (#3404)

  - switch to status badge based on tests run in GitHub Actions CI in README (#3415)
  - make `flake8` code style checks pass (#3416, #3417)
  - limit Travis to only test with Python 2.6 + Lmod 7 (#3418)
  - ignore deprecation warning raised for cryptography when using Python 3.5 produced by output of test suite (#3423)

easyblocks

- 2 new software-specific easyblocks:
  - pybind11 (#2115), PyTorch (#2104)
- minor enhancements, including:
  - update MATLAB easyblock to use new binary installer for versions >= 2020a (#2058)
  - add `use_pip_requirement` custom easyconfig parameter to PythonPackage easyblock to allow providing a requirements file to the `pip` command, as a source (#2064)
  - update CP2K easyblock for recent versions (>= 7.0) (#2069)
  - add `multi_deps` support into custom easyblock for QScintilla (#2077)
  - update Mothur easyblock for v1.44.0 and higher (#2084)
  - update MotionCor2 easyblock to add support for v1.3.2 (#2100)
  - update Tinker easyblock to handle skipping of tests depending on version and link with `fftw_omp` (#2102)
  - launch test & sanity check commands through `mpirun` for netcdf4-python if MPI support is enabled (#2106)
  - check for installation of pip & setuptools in Python 3.4+ (#2108)
  - update QuantumESPRESSO easyblock to support version 6.6 (#2112)
  - allow NAMD to be built on POWER, and also use the CUDA arch info (#2113, #2123)
  - update Libint easyblock for versions >= 2.6.0 + add custom easyconfig parameter to enable Fortran support (#2116)
  - update TensorFlow easyblock to use system/EasyBuild installed libraries (#2117, #2165, #2163, #2172)
  - add support in NAMD easyblock to pass additional C++ compiler options for building Charm++ component with + wrap them in single quotes (#2118)
  - automatically add required `-G Ninja` option when using CMakeNinja easyblock + add `generator` custom easyconfig parameter for CMakeMake (#2120)
  - make several easyblocks aware of `--sysroot` EasyBuild configuration option: binutils (#2147, #2159), CMakeMake (#2152), GCC (#2143), Perl (#2142), Python (#2148, #2149)
  - make Mesa easyblock aware of aarch64 (#2153)
  - enhance sanity check in custom Doxygen easyblock to catch broken installation (#2171)
- various bug fixes, including:
  - make TensorFlow easyblock ignore the `PKG_REVISION` identifier if NCCL version if it exists (#2085)
  - remove version check against the `version.txt` file in CUDA easyblock (#2097)
  - add `lib` symlink in tbb installation directory when building from source (#2103)
  - handle GNUInstallDirs special cases in CMakeMake easyblock (#2105, #2124)

  - patch ELPA's `manual_cpp` script to fix hardcoded `/usr/bin/python` (#2107)
  - make TensorFlow easyblock also set `$GCC_HOST_COMPILER_PREFIX` to specify binutils location (#2110)
  - ensure stand-alone Python package being installed is in view when running `pip check` by loading fake module first (#2114)
  - be more patient when running interactive configure script for WRF (#2119)
  - make sure `self.start_dir` is set to a full path before constructing installation command in RPackage easyblock (#2125)
  - correctly check whether `modinc` easyconfig parameter is set to `True` in CP2K easyblock (#2138)
  - update PSI easyblock to correctly find Python and enable PCMSolver/CheMPS2 (#2141)
  - fix running GROMACS tests when using `eb --rpath` (#2144, #2154)
  - remove existing Python installation directory if both `eb --rpath` and `--enable-optimizations` configuration option are used (#2146)
  - ensure `libQt5Core.so` is compatible with older Linux kernels by disabling `renameat2` and `getentropy` features (#2151)
  - solve issue where pybind11 picks up on the system Python instead of one provided by a module (#2158)
  - fix Qt5 easyblock to support installation on Arm/aarch64 (requires platform to be `linux-g++`) (#2160)
  - fix OpenFOAM easyblock to support installation on Arm/aarch64 systems (#2162)
  - explicitly enable/disable CUDA options in PyTorch easyblock + fix download check in sanity check + fix for disabling of *NNPACK on POWER systems (#2164)
  - fix sanity check in Mathematica easyblock for 11.x versions older than 11.3 (#2168)
  - unset `$COLUMNS` if it is set to `0` before running Perl's configure script (#2169)
- other changes:
  - filter out Python in SLEPc configure (#2101)
  - switch to status badge based on tests run in GitHub Actions CI in README (#2126)
  - fix code style issues to make `flake8` checks pass (#2128 - #2137, #2140, #2145, #2155)
  - limit test configurations in Travis CI to only Python 2.6 (#2139)

easyconfigs

- added easyconfigs for 2 new toolchains:
  - gomkl/2020a and iomkl/2020a (#11036)
- added example easyconfig files for 49 new software packages:
  - almosthere (#11152), arcasHLA (#10867), BioPP (#11113), Bracken (#10829), BUFRLIB (#11140), Calib (#11111), CellRanger-ATAC (#11186), edlib (#10470, #11246), flatbuffers (#11109), gengetopt (#11117), graphite2 (#11168), HeFFTe (#10990), hierfstat (#11249), immunedeconv (#11136), ioapi (#10959), itpp (#10958), LiBis (#11059), libosmium (#11024), limix-bgen (#11152), minibar (#10470, #11246), misha (#11127), MOABS (#10747), moonjit (#11163), NGLess (#11128), nsync (#11109), openCARP (#11117), OpenForceField (#11048), OpenMMTools (#11046), OpenMS (#10994), PennCNV (#10986), plantcv (#10968), PlasmaPy (#10732), Portcullis (#11038), PycURL (#11169), PyGEOS (#11110), pySCENIC (#11115), Reapr (#9296), RnBeads (#11142), sf (#11248), SLiM (#11172), stars (#11215, #11248), Sumo (#11071), Telescope (#10943), tensorflow-probability (#10312), texlive (#11168), tidymodels (#11010), Trycycler (#11207), umi4cPackage (#11127), variant_tools (#11169)

- added additional easyconfigs for various supported software packages, including:
  - Bazel 3.4.1, Bonito 0.2.2, binutils 2.35, CP2K 7.1, Clang 10.0.1, ccache 3.7.11, ctffind 4.1.14, datamash 1.5, ELPA 2020.05.001, Emacs 26.3, Flye 2.8.1, GCC(core) 10.2.0, googletest 1.10.0, HDF 4.2.15, Horovod 0.19.5, imageio 2.9.0, JUBE 2.4.0, Julia 1.5.1, Kent_tools 401, Libint 2.6.0, libxsmm 1.16.1, MDSplus 7.96.8, MDTraj 1.9.4, MariaDB 10.4.13, Meson 0.55.1, MotionCor2 1.3.2, NAMD 2.14, NCO 4.9.3, OpenFOAM 8, OpenMPI 4.0.5, OptiX 6.5.0, Pandoc 2.10, Perl 5.32.0, PostgreSQL 12.4, PyCUDA 2019.1.2, PyFR 1.9.0, PyTorch 1.6.0, PyZMQ 18.1.1, patchelf 0.12, pocl 1.5, pycocotools 2.0.1, pyproj 2.6.1.post1, QuantumESPRESSO 6.6, RDKit 2020.03.3, rgdal 1.5, Salmon 1.3.0, Spark 3.0.0, scikit-image 0.17.1, scipy 1.4.1, snpEff 5.0, sympy 1.6.2, TINKER 8.7.2, tqdm 4.47.0, Unicycler 0.4.8, VMD 1.9.4a43
- minor enhancements, including:
  - also build Python libraries for ParaView 5.8.0 (#10927)
  - add extensions to recent Bioconductor easyconfigs: FlowSorted.Blood.EPIC (#11021), DRIMSeq + stageR (#11053)
  - add extensions to recent R easyconfigs: AICcmodavg + biomod2 (#11030), qqman (#11052), poLCA (#11081), coxed (#11094), testit + data.tree (#11135), celestial + fasterize (#11206)
  - add Config::Simple extension to Perl 5.30.x easyconfigs (#11051)
  - update TensorFlow v2.1.0 (#11109) and v2.0.0 (#11233) easyconfigs to provide more dependencies via EasyBuild
  - add CUDA compute capabilities to torchvision-0.5.0 (#11241)
- various bug fixes, including:
  - patch to fix exporting images with PyQtGraph v0.10.0 (#10848)
  - add missing Python dependency to PETSc 3.11.1 (#10907) and PETSc 3.12.x (#10908) easyconfigs
  - use CMakeMake easyblock for installing magma to avoid C++11 related failures on POWER (#10929)
  - fix source URL for Graphviz v2.40.1 (#10944)
  - fix `lapack.h` for use with C++ in OpenBLAS 0.3.9 easyconfigs (#10960)
  - add missing build dep on M4 for 2019b versions of netCDF-Fortran (#10972)
  - update easyconfig for PyTorch 1.4 to use custom easyblock and run on POWER (#11000)
  - create symlinks to incorrectly named directories in OpenBabel-3.1.1 so `$BABEL_LIBDIR` and `$BABEL_DATADIR` work properly (#11004)
  - add missing Python build dep to recent ELPA easyconfigs (#11011)
  - use `is_generic_easyblock` from filetools in easyconfigs test suite (#11020)
  - fix sources in Portcullis v1.2.2 easyconfigs (#11038)
  - add patch for FFTW 3.3.8 to avoid use of `-no-gcc` when building with Intel compilers, to fix installation on CentOS 8 (cfr. #10932) (#11050)
  - add missing Python build dep for BEDTools 2.29.x (#11054)
  - add missing SciPy-bundle and Kent_tools (for bedPartition command) dependencies to FLAIR (#11057)
  - add patch to fix bug in LiBiS v20200428 easyconfig (#11059)
  - use FFTW provided via EasyBuild for ScaFaCoS (#11060)
  - fix undefined reference to `qfloat16::mantissatable` in Qt5.14.1 (#11063)
  - add alternative checksum for rstantools 2.0.0 extension (#11081)

  - update checksums for R 3.6.0 packages and add patch for ppc (#11088)
  - install scikit-learn 0.23.1 as a bundle and include required threadpoolctl extension (#11089)
  - update pybind11 easyconfigs to use custom easyblock to install with pip (#11091)
  - add recent six as extension to archspec installed on top of Python 3.7.4 (#11092)
  - add missing Seaborn dependency to LiBis easyconfig (#11095)
  - add missing dependencies for OpenPIV + switch to PythonBundle easyblock and include progressbar2 as extension (#11096)
  - add missing argparse dep to TEtranscripts easyconfig (#11097)
  - add missing pkg-config build dep in Octave 5.1.0 easyconfigs (#11100)
  - don't statically link MUSCLE, to avoid requiring that glibc-static is installed in OS (#11102)
  - add missing pkg-config build dependency in recent R-bundle-Bioconductor easyconfigs (#11104)
  - add patch to h5py 2.8.0 (#11119) and 2.9.0 (#11118) easyconfigs to avoid MPI_Init on `import h5py`
  - add patch to support libbfd 2.34 API change in Score-P 6.0 (#11123)
  - use pip to install protobuf-python in 2019b toolchain (requires re-downloading source tarball!) (#11143, #11260)
  - add missing Keras-Applications extension to TensorFlow 2.2.0 easyconfigs with `foss*/2019b` toolchain (#11156)
  - add missing pocl dependency in R 3.6.0 (#11157)
  - update Meson build dep to 0.55.1 for GLib, X11, Mesa & co to fix aggressive RPATH stripping (#11178)
  - disable generating of man pages in recent libdrm easyconfigs to avoid installation failure if docbook-xsl is not installed (#11182)
  - add fontconfig and bzip2 as direct dependencies for Qt5 to fix installation with `--rpath` (#11183)
  - fix failing `make check` for MPFR 4.x (#11187) and GMP (#11188) when installing with `eb --rpath`
  - add pkg-config as a build dependency for libglvnd (#11189)
  - add missing libiconv dep in recent Doxygen easyconfigs (#11191, #11257)
  - enhance Java/11 wrapper to also support for aarch64/Arm (#11192)
  - add pkg-config as a build dependency to Ghostscript 9.52, needed on aarch64/arm (#11194)
  - add patch for pycrypto extension in recent Python easyconfigs to remove hardcoded `/usr/include` which causes problems when `eb --sysroot` is used (#11202)
  - fix installation of R v3.6.3 and v4.0.0 with foss/2020a on Arm (aarch64) (#11213)
  - update PyTorch 1.4 easyconfigs to use custom easyblock (and do less downloading during installations) (#11219)

- other changes:
  - don't require custom `sanity_check_paths` for CUDA bundle easyconfigs (#10936)
  - move intervaltree and sortedcontainers to main Python easyconfigs (#10969, #10970)
  - disable qtwayland in Qt5 v5.14.1 (#11107)
  - remove mklml (small MKL) from PyTorch 1.3.1 and 1.4.0 easyconfigs (#11019)
  - use pip to install h5py 2.10.0 (#11044)

- stop testing easyconfig PRs with Travis, only use GitHub Actions from now on (#11008, #11055)
- switch to Kent_tools built from source as dep for FusionCatcher (#11057)
- fall back to using PR target branch when determining "merge base" between PR branch & target branch fails in test suite (#11069)
- rename gtest to googletest (#11082)
- rename sdsl-lite to SDSL (to use one single name) (#11084)
- stop using `remove_usr_bin` patch in TensorFlow easyconfig, no longer required with updated TensorFlow easyblock (#11087)
- extend timeout for libxc-4.3.4 to avoid failing tests (#11126)
- move GitHub Actions status badge to top of README (#11138)
- fix code style issues in test (#11146)
- enable `flake8` on CI and fix issues (#11147)
- prefer gc GitHub site for source downloads (#11208)
- prefer `https` over `ftp` in `source_urls` of IgBLAST easyconfig (#11244)

11.12.9 EasyBuild v4.2.2 (July 8th 2020)

bugfix/update release

framework

- various enhancements, including:
  - add support for using `sources` and `git_config` for extensions in `exts_list` (#3294)
  - add support for software minver template (like `%(pyminver)s`, etc.) (#3344, #3345)
  - add support for updating dictionary or tuple easyconfig parameters with `self.cfg.update` (#3356)
- various bug fixes, including:
  - fix crash in `--avail-easyconfig-constants` when using `--output-format=rst` + ensure sorted output (#3341)
  - always take into account `builddependencies` when generating template values, also when we're not iterating over `builddependencies` (#3346)
  - fix running command as `easybuild` user in generated Singularity definition file (#3347)
  - allow ignoring `versionsuffix` in `--try-update-deps` (#3350, #3353)
  - retain order of paths when generating `prepend_path` statements for module file (don't sort them alphabetically) (#3367)
  - also put easyblocks used by extensions in `reprod` directory (#3375)
  - also copy template values in `EasyConfig.copy` method to ensure they are defined when installing extensions (#3377)
  - skip lines that start with `module-version` when determining whether a module exists in `ModulesTool.exist` (#3379)

easyblocks

- 3 new software-specific easyblocks:

  - LLVM (#2065), Scipion (#1847), XALT (#1942)
- minor enhancements, including:
  - use `wclean -platform` instead of `wcleanPlatform` for OpenFOAM v2006 & newer (#2088)
- various bug fixes, including:
  - only build OpenCV with IPP support on x86_64 systems (#2056)
  - make sure `CFLAGS` and `MYCFLAGS` are used in Lua easyblock (#2062)
  - also accept `exist_lic` as valid value for `license_file` in IntelBase easyblock (#2067)
  - fix minor bug in shared libs handling and correctly setup SuiteSparse variables in Trilinos easyblock (#2071)
  - make sure `PythonBundle` uses correct `python` command in extensions filter + also add `lib64/python*/site-packages` to `$PYTHONPATH` (if it exists) (#2075, #2081)
  - fix build environment for WRF by patching shebang in compile script (#2079)
  - fix order of `intel64/lib` paths in generated module for impi (and drop `intel64/lib/release_mt`) (#2080)
  - set `$XDG_CACHE_HOME` to avoid abuse of `$HOME/.cache/pip` when installing DOLFIN (#2082)
  - make VMD easyblock Python 3 compatible (#2083)
  - rename include-fixed subdirectory in GCC installation to avoid problems after OS upgrade (#2087, #2093, #2094)
  - use glob pattern to determine name of Mathematica installation script (#2089)
  - use `mpirun` for sanity check commands in LAMMPS easyblock (#2096)

easyconfigs

- added easyconfigs for 2 new toolchains:
  - golf/2020a (#10672)
  - intel/2020.06, based on intel/2020a but with impi 2018 update 5 (#10864)
- added example easyconfig files for 36 new software packages:
  - AmrPlusPlus (#9919), arrow (#10882), attrdict (#10827), Cartopy (#10685), dm-tree (#10775), dotNET-SDK (#10661), FLAIR (#10860), fastqsplitter (#10706), GLFW (#10709), git-extras (#10440), hl7apy (#10728), hyperopt (#10735), IgBLAST (#10889), IntelDAAL (#9848), ichorCNA (#10595), MARS (#10691), MITObim (#10897), MUST (#10849), mhcnuggets (#9743), PAUP (#10830), Percolator (#10665), PyBerny (#10885), poetry (#10884), pycocotools (#10806), python-hl7 (#10673), SHAPEIT4 (#10814), SYMPHONY (#10058), SunPy (#10676), sdsl-lite (#10691), TEtranscripts (#10909, #10921), Tombo (#10646), tensorboardX (#10774), Unidecode (#10828), WHAM (#10736), XALT (#9792), Xvfb (#10512)
- added additional easyconfigs for various supported software packages, including:
  - Armadillo 9.900.1, astropy 4.0.1, BLAST+ 2.10.1, BioPerl 1.7.7, Blitz++ 1.0.2, Bonito 0.2.0, bokeh 2.0.2, CFITSIO 3.48, CGAL 4.14.3, CLHEP 2.4.1.3, cutadapt 2.10, dask 2.18.1, Elk 6.3.2, GATE 9.0, GLM 0.9.9.8, GRASS 7.8.3, Geant4 10.6, geopandas 0.8.0, giflib 5.2.1, HDF5 1.12.0, HISAT2 2.2.0, Hadoop 2.10.0, IPython 7.15.0, inferCNV 1.2.1, Julia 1.4.2, LibSoup 2.70.0, MATIO 1.5.17, MRtrix 3.0.0, Mathematica 12.1.1 MultiQC 1.9, NGS 2.10.5, NiBabel 3.1.0, Nipype 1.4.2, ncbi-vdb 20.10.7, numba 0.50.0, OpenBabel 3.1.1, OpenFOAM v2006, PDT 3.25.1, PRSice 2.3.1, ParaView 5.8.0, ParmEd 3.2.0, PyQt5 5.13.2, Pysam 0.16.0.1, parallel 20200522, plotly.py 4.8.1, pybedtools 0.8.1, ROOT 6.20.04, ReFrame 3.0, Roary 3.13.0, rjags 4-10, rstudio 1.3.959, SPAdes 3.14.1, SRA-Toolkit 2.10.8, Saxon-HE 9.9.1.7,

Seaborn 0.10.1, Seurat 3.1.5, SimPEG 0.14.1, Spark 2.4.5, StringTie 2.1.3, scikit-allel 1.2.1, scikit-learn 0.23.1, snpEff 4.3t, Valgrind 3.16.1, VarScan 2.4.4, vsc-mympirun 5.1.0, WebKitGTK+ 2.27.4, wxPython 4.0.7.post2, zarr 2.4.0

- minor enhancements, including:
  - add extensions to R 4.0.0 easyconfig: drgee + stdReg (#10833), copCAR (#10911), ngspatial (#10913), drugCombo (#10914)
  - add extensions to R-bundle-Bioconductor 3.11 easyconfig: BSgenome.Cfamiliaris.UCSC.canFam3 (#10840), SingleR (#10904)
  - include extra extension and support for MPI in MAFFT v7.453 and v7.470 (#10853)
  - add Array::Transpose to Perl (#10878)
- various bug fixes, including:
  - fix dependencies and configuration of CoinUtils ecosystem (#10450)
  - fix Togl configure patch so that it completely ignores `TCL_SRC_DIR` and `TK_SRC_DIR` (#10662)
  - change Xerces-C++ to official CMakeMake build (fixes missing links to curl) (#10664)
  - add `-fPIC` to `toolchainopts` in easyconfig for Lua 5.3.5 with system toolchain (#10671)
  - remove hardcoded X86 target in LLVM easyconfigs (#10677)
  - switch to https `homepage` and `source_urls` in libGLU easyconfigs (#10686)
  - add patch for h5py 2.10.0 to avoid triggering `MPI_Init` at import (#10687)
  - add alternative checksum for pkgmaker, doRNG, cobs extension in R 3.6.0 easyconfigs (#10692)
  - update UDUNITS `source_urls` to https sources (#10693)
  - add missing M4 to netCDF-Fortran easyconfigs using 2020a toolchain (#10695, #10697)
  - add Rgraphviz patch to R-bundle-Bioconductor easyconfigs (#10710)
  - add M4 builddependency to SuiteSparse 2019b easyconfigs (#10720)
  - make sure WHAM is built with correct compiler (#10736)
  - define `MAX_JOBS` in easyconfigs of PyTorch from v1.1.0 to v1.4.0 (#10772)
  - add dm-tree and lz4 as dependencies to Ray v0.8.4 (#10775)
  - fix source URL in recent ant easyconfigs (#10790)
  - bump versions of GO.gb/org.Hs.eg.db/PFAM.db extensions in Bioconductor 3.11 bundle, since sources for old versions have disappeared (#10791)
  - add patch to Armadillo easyconfigs using the foss toolchain to prevent it from picking up MKL if installed system-wide (#10812)
  - add missing kerneltree extension in FLAIR easyconfig (#10860)
  - fix source URLs in Eigen 3.2.* (#10872) and 3.3.* (#10869) easyconfigs
  - refer to issue in GitHub repo in configparser patch, since Bitbucket repo is disappearing (#10873)
  - switch to github.com source URL for MetaPhlAn (#10874)
  - switch to GitHub sources for ScientificPython 2.9.4 (#10875)
  - switch to GitHub source URL for HPDBSCAN (#10876)
  - download x265 3.2/3.3 sources from `bitbucket.org/multicoreware/x265_git` (#10905)

  - fix download URL for libatomic in gc 7.6.12 easyconfigs (#10915)
  - fix `sanity_check_paths` in CFITSIO-3.48-GCCcore-9.3.0.eb easyconfig (#10917)
  - add missing dependencies for geopandas 0.7.0 (#10923)
- other changes:
  - rename OpenPyXL 2.6.4 easyconfig to openpyxl (#10916)

11.12.10 EasyBuild v4.2.1 (May 20th 2020)

bugfix/update release

framework

- various enhancements, including:
  - also mention CPU architecture (x86_64, POWER) in comment for test reports (#3281)
  - add support for enhancing existing sanity check in easyconfigs, as opposed to overwriting paths/commands (#3288)
  - clean up locks when EasyBuild session is cancelled with a signal like `SIGTERM` (#3291, #3321)
  - add `find_glob_pattern` function to `filetools` module (#3297)
  - add constants for common OS dependencies (`OS_PKG_IBVERBS_DEV`, `OS_PKG_OPENSSL_DEV`, ...) (#3309, #3334)
  - flesh out `get_mpi_cmd_template` function from `Mpi.mpi_cmd_for` method (#3312)
  - add variable `moddependpaths` to specify extra `$MODULEPATH` entry to consider for loading dependency modules (#3324)
  - allow copying of tweaked easyconfigs when using `--try-*` with `--copy-ec` (#3332)
- various bug fixes, including:
  - make `ModulesTool.exist` more robust w.r.t. module wrappers, aliases, defaults, etc. (#3216, #3337)
  - clean up rst output of `--list-toolchains` (#3246)
  - cast CPU arch name provided by archspec to a regular string (#3286)
  - get `pr_title` and `pr_descr` build_options in `new_pr_from_branch` instead of `new_pr` (and commit_msg in both) (#3298)
  - make `pypi_source_urls` more robust by using `HTMLParser` rather than `xml.etree.ElementTree` (#3303, #3329)
  - fix broken test for `--include-easyblocks-from-pr` (#3304)
  - don't use `distutils.dir_util` in `copy_dir` (#3310)
  - print trace message for sanity check command before running it (#3316)
  - fix problems with processing of easyconfigs using a Cray* toolchain when there are no actual external modules (#3319)
  - make `test_find_eb_script` more robust in case `$EB_SCRIPT_PATH` is already set (#3320)
  - fix several small problems with `--try-update-deps` (experimental feature) (#3325, #3326, #3330)
  - add `--disable-job` in `eb` command used in jobs, to prevent infinite job cycle (#3328)
  - avoid empty entries in `$LD_LIBRARY_PATH` and other path-like environment variables (#3333)

- other changes:
  - fix code style issues in `easybuild.tools` + add `flake8` linting test (#3282)
  - introduce `contextmanager` for disabling templating and reduce resolving errors (#3287)
  - add `change_into_dir` named argument to `extract_file` + print deprecation warning if it's not specified (#3292)
  - improve `install_eb_dep.sh` script to install EasyBuild dependencies in CI environment (#3314)

easyblocks

- add generic easyblock for installing Go packages: `GoPackage` (#2042)
- minor enhancements, including:
  - update `config.guess` for all R packages, required installing R + extensions on Linux/POWER systems (#1949)
  - add support for `preinstallopts` and install in subdirectory to `Tarball` generic easyblock (#1989, #2049)
  - rewrite `GROMACS` easyblock to install all four variations (single/double precision, with/without MPI) in the same directory (#1991)
  - add custom `subdir_version` easyconfig parameter for FLUENT (#2021)
  - use updated `config.guess` in `GCC` easyblock (#2033, #2059)
  - add extra question pattern to support installing CPLEX 12.10 (#2038)
  - add support for building NWChem on top of external GlobalArrays + also define `$LAPACK_LIB` (required for NWChem 7.x) (#2043)
  - execute minimal test in sanity check commands in `impi` easyblock (#2045)
  - update for `MotionCor2` easyblock for v1.3.1 (#2046)
  - fixes + enhancements for `PETSc` easyblock (#2048)
- various bug fixes, including:
  - use correct version in `ELPA` preprocessor flag used for QuantumESPRESSO v6.x (#2027)
  - generically find the arch folder and add top-level bin/lib symlinks in `PDT` & `TAU` easyblocks (#2029)
  - fix typo in `LAMMPS` easyblock preventing disabling the `USER-INTEL` package via `configopts` (#2031)
  - let `CMakeMake` remove `easybuild_obj` build directory if it already exists (#2032)
  - include own lib dir in `RPATH` before system lib dirs in `binutils` easyblock (#2044)
  - clean `$CPATH` before building ROOT to avoid clash between external LLVM loaded as a dependency and internal LLVM used by ROOT (#2047)
  - don't let `extract_file` change directory in various easyblocks (#2051)
  - make `postinstallcmds` available to Mathematica easyconfigs + add bin/Executables to `$PATH` (#2052)
  - fix the extra dirs added to PATH in `FreeSurfer` easyblock (#2053)
  - correct download url for test data in `WPS` easyblock (#2055)
  - fix typo in warning in `TensorFlow` easyblock (#2057)
  - fix failing sanity check under `--module-only` in `GCC` easyblock (#2059)

- other changes:
  - add configure options for CubeLib/CubeWriter dependencies in `Score-P` easyblock (#2030)
  - remove local `find_glob_pattern` in `Blender` & `ROOT` easyblocks, use function provided by `filetools` instead (#2037)

easyconfigs

- added easyconfigs for new common toolchains: foss/2020a (#10483, #10492), intel/2020a (#10494)
- added example easyconfig files for 66 new software packages:
  - Alpha (#9994), antiSMASH (#10589), Arlequin (#10620), artic-ncov2019 (#10459), augur (#10405), AutoMap (#10419), Bio-EUtilities (#10037), CaSpER (#10593), cdbfasta (#10547), cddlib (#10429), CoCoALib (#10429) dftd3-lib (#10351), DoubletFinder (#10603), E-ANTIC (#10429), FastViromeExplorer (#10571), FIX (#8870), FusionCatcher (#10134), geopandas (#10322), goalign (#10469), gotree (#10459), gretl (#10413), harmony (#10604), HDF-EOS (#10534), HDF-EOS5 (#10536), HMMER2 (#10588), HyPo (#10642), king (#10365), libdeflate (#10459), libfabric (#10616), libgit2 (#10453), libuv (#10444), mbuffer (#10524), MDAnalysis (#10545), MEM (#10605), MESS (#10597), metaerg (#10037), MinCED (#10037), MitoZ (#7735), nauty (#10429), nifti2dicom (#10598), NLMpy (#10029), ntCard (#10502), NTL (#10431), pIRS (#10508), popscle (#10550), ProtHint (#10549), protozero (#10495), pysndfx (#10452), PyVCF (#10564), PyWavelets (#10501), rampart (#10459), rickflow (#10641), RNA-Bloom (#10502), root_numpy (#10424), rstudio (#10619), ScaFaCoS (#10537), Scythe (#10524), SDSL (#10642), SHAP (#10379), SNPomatic (#10524), SoX (#10452), swissknife (#10037),) Taiyaki (#10573), TCLAP (#10598), torchaudio (#10516), wtdbg2 (#10524)
- added additional easyconfigs for various supported software packages, including:
  - ADF 2019.303, BRAKER 2.1.5, Bazel 2.0.0, Bonito 0.1.4, Boost.Python 1.72.0, Bowtie2 2.4.1, CMake 3.16.4, CPLEX 12.10, CVXOPT 1.2.4, Coreutils 8.32, cURL 7.69.1, DFTB+ 19.1, ecCodes 2.17.0, expat 2.2.9, FFmpeg 4.2.2, FriBidi 1.0.9, GATK 4.1.5.0, GCC(core) 10.1.0, GDAL 3.0.4, GEOS 3.8.1, GLib 2.64.1, GMP 6.2.0, GROMACS 2020.1, GTK+ 3.24.17, GenomeThreader 1.7.3, GffCompare 0.11.6, Ghostscript 9.52, GlimmerHMM 3.0.4c, GlobalArrays 5.7.2, gmpy2 2.1.0b4, gmsh 4.5.6, gpustat 0.6.0, gradunwarp 1.2.0, HDF5 1.10.6, hwloc 2.2.0, hypothesis 5.6.0, ICU 66.1, IPython 7.13.0, ImageMagick 7.0.10, Julia 1.4.1, KMC 3.1.2rc1, Kraken2 2.0.9, LMfit 1.0.0, Longshot 0.4.1, libarchive 3.4.2, libffcall 2.2, libffi 3.3, libgd 2.3.0, libjpeg-turbo 2.0.4, librsvg 2.48.4, libsigsegv 2.12, lrslib 7.0a, MEME 5.1.1, MPC 1.1.0, Mako 1.1.2, Mesa 20.0.2, Meson 0.53.2, MotionCor2 1.3.1, MultiQC 1.8, matplotlib 3.2.1, NGS-Python-2.10.4, NGS 2.10.4, NSPR 4.25, NSS 3.51, NWChem 7.0.0, Nextflow 20.04.1, Ninja 1.10.0, Normaliz 3.7.4, nanopolish 0.13.1, ncbi-vdb 2.10.4 netCDF 4.7.4, OpenBLAS 0.3.9, OpenEXR 2.4.1, OpenMPI 4.0.3, OpenSSL 1.1.1e, openpyxl 3.0.3, PAPI 6.0.0, PCRE 8.44, PCRE2 10.34, PLUMED 2.6.0, PMIx 3.1.5, PROJ 7.0.0, Perl 5.30.2, Pillow 7.0.0, PyYAML 5.3, Python 2.7.18 + 3.8.2, parallel 20200422, Qt5 5.14.1, R-bundle-Bioconductor 3.11, R 3.6.3 + 4.0.0, RMBlast 2.9.0, Racon 1.4.13, Ray 0.8.4, Rust 1.42.0, re2c 1.3, rioxarray 0.0.24, rootpy 1.0.1, rstudio 1.2.5042, SCons 3.1.2, SDL2 2.0.10, SIONlib 1.7.6, SQLite 3.31.1, SRA-Toolkit 2.10.4, Salmon 1.2.0, ScaLAPACK 2.1.0, SciPy-bundle 2020.03, Stacks 2.53, StringTie 2.1.1, SuiteSparse 5.7.1, snappy 1.1.8, spaln 2.4.03, sympy 1.5.1, Tcl 8.6.10, TensorFlow 2.2.0, Tk 8.6.10, Tkinter 3.8.2, tbl2asn 20200302, torchvision 0.5.0, UCX 1.8.0, UMI-tools 1.0.1, utf8proc 2.5.0, util-linux 2.35, worker 1.6.12, wxWidgets 3.1.3, X11 20200222, XZ 5.2.5, x264 20191217, x265 3.3, zsh 5.8, zstd 1.4.4
- minor enhancements, including:
  - add additional extensions for recent versions of R (#10359, #10585, #10586, #10621) and R-bundle-Bioconductor (#10585, #10596, #10621)
  - add additional extensions for recent versions of Perl (#10412, #10546, #10623)
  - include LLVM linker in Clang 7.0.1 on GCC 7.3.0-2.30(#10458)
  - include static lib and header in nimimap2 easyconfigs for foss-2018b and GCC-8.2.0* (#10464)

  - add alternate checksum for EMBOSS (#10607)
- various bug fixes, including:
  - fix incorrect checksums for Amber patches (#8870)
  - add patches to fix installation of R 3.6.2 on POWER (#9830)
  - add missing build dep. pkg-config in GObject-Introspection 1.63.1 w/ Python 3.7.4 easyconfig (#10380)
  - add Perl as a build dependency for recent Autoconf/Automake easyconfigs (#10408, #10426)
  - fix source_urls in ICU easyconfigs (#10417)
  - disable `USER-INTEL` package in LAMMPS easyconfigs using intel/2019b, since it results in an installation that produces incorrect results (#10418)
  - fix undefined reference to `qfloat16::mantissatable` in Qt5-5.13.1-GCCcore-8.3.0 (#10425)
  - get rid of double '-' in versionsuffix of torchtext easyconfig (#10472)
  - fix broken Mako easyconfigs (#10480, #10627)
  - move builddependencies to dependencies in OTF2 and Score-P easyconfigs (#10496)
  - add missing cURL dep for LAMMPS (#10527)
  - add Python 2 build dependency for OpenPGM (#10539)
  - use `OS_PKG_IBVERBS_DEV` constant for OS dependency in PyTorch 1.4.0 easyconfigs (#10540)
  - add missing Bison build dep in Graphviz easyconfigs (#10541)
  - use https in homepage for Mathematica 12.0.0 + clean up sanity check commands (now done by easyblock) (#10559)
  - add missing PyVCF dependency for BAMSurgeon (#10564)
  - remove FFTW dependencies from LAMMPS easyconfigs, no longer needed (MKL can be used too now) (#10565)
  - fix ITK v5.0.1 easyconfig w.r.t. locale and location of libjpeg-turbo library (#10592)
  - add patch to fix missing `const` qualifiers for ncurses (#10606)
  - fix OS dependency for rstudio for Debian-based systems (#10608)
  - fix Rmath paths in easyconfig for FastQTL v2.184 (#10612)
  - add symlink for Arlequin commands + fix sanity check commands (#10620)
  - fix recent binutils easyconfigs using system toolchain for Fedora 32 / GCC 10 (#10633)
  - replace '/path/to' with actual installation prefix in FuSeq scripts (#10640)
- other changes:
  - check sdist with different Python versions in CI (#10388)
  - use Bison 3.3.2 as build dep for flex 2.6.4 (#10403)
  - mention `http://` Pfam website rather than `ftp://` in load message of BiG-SCAPE easyconfig (#10439)
  - archive old Singularity configs (#10591) * Singularity is not well suited to install via EasyBuild because it requires admin privileges to enable setuid
  - stop using old hpcugent URL for cloning framework/easyblocks repos in CI (#10635)
  - use `SYSTEM` constant for toolchain in easyconfigs already use `system` toolchain (#10638)

11.12.11 EasyBuild v4.2.0 (April 14th 2020)

feature release

framework

- add support for `--try-update-deps` (experimental feature), to upgrade dependencies based on available easyconfigs (#2599)
- adding locking to prevent two parallel builds of the same installation directory (#3009)
  - for more information, see https://easybuild.readthedocs.io/en/latest/Locks.html
- significantly speed up `-D`/`--dry-run` by avoiding useless '`module show`' (#3203)
- add support for creating an index & using it when searching for easyconfigs (#3210)
  - for more information, see https://easybuild.readthedocs.io/en/latest/Easyconfigs_index.html
- additional GitHub integration features:
  - add support for targeting easyblocks and framework repositories in `--new-pr` (#1876, #3189)
  - add support for `--include-easyblocks-from-pr` (#3206)
  - for more information, https://easybuild.readthedocs.io/en/latest/Integration_with_GitHub.html
- various other enhancements, including:
  - add a contrib/hooks dir with some examples of hooks used (#2777)
  - also mention working directory + input passed via stdin (if any) in trace output of `run_cmd` (#3168)
  - probe external modules for missing metadata that is not provided via external module metadata file (#3174)
  - also update `$CMAKE_PREFIX_PATH` and `$CMAKE_LIBRARY_PATH` in generated module file (#3176)
  - optionally call `log.warning` in `print_warning` (#3195)
  - add an option to `git_config` to retain the `.git` directory (#3197)
  - allow use of `SYSTEM` as `--try-toolchain` option (#3213)
  - mention CPU arch name in comment for uploaded test report, if it's known by `archspec` (#3227)
  - make `--merge-pr` take into account `--pr-target-branch` (#3236)
  - make `--check-contrib` print a warning when `None` is used for checksums (#3244)
  - update `install-EasyBuild-develop.sh` script and create script for 2020a merge sprint (#3245)
  - add template for `mpi_cmd_prefix` (#3264)
  - update `copy_dir` to include option to merge directories (#3270)
  - support template name for CUDA version (#3274)
- various bug fixes, including:
  - use correct module for `errors_found_in_log` (#3119)
  - fix `EasyConfig.update` code to handle both strings and lists as input (#3170)
  - fix removing temporary branch on `--check-github` (#3182)
  - fix shebang even if first line doesn't start with '`#!`' (#3183)
  - make bootstrap script work with Python 3 (#3186)

  - read patch files as bytestring to avoid `UnicodeDecodeError` for patches that include funky characters (#3191)
  - fix online check in `--check-github`: try repeatedly and with different URLs to cater for HTTP issues (#3194)
  - don't crash when GitPython is not installed when using Python 3 (#3198)
  - fix `os_name_map` for RHEL8 (#3201)
  - don't add shebang to binary files (#3208)
  - use `checkout@v2` in GitHub Actions to fix broken re-triggered tests (#3219)
  - don't filter out `None` values in to_checksums, leave them in place (#3225)
  - fix defining of `$MPI_INC_*` and `$MPI_LIB_*` environment variables for impi version 2019 and later (#3229)
  - use correct target account/repo when creating test report & posting comment in PR (#3234)
  - reorganize `EasyBlock.extensions_step` to ensure correct filtering of extensions (#3235)
  - also support `%(installdir)s` and `%(builddir)s` templates for extensions (#3237)
  - unset `$GITHUB_TOKEN` in Travis after installing token, to avoid failing `test_from_pr_token_log` (#3252)
  - fix reporting when skipping extensions (#3254)
  - avoid API rate limit errors on online check in `--check-github` (#3257)
  - show easyconfig filenames for parallel build (#3258)
  - make `EasyConfig.dump` aware of toolchain hierarchy, to avoid hardcoded subtoolchains in dependencies easyconfig parameters (#3261)
  - fix use of `--copy-ec` with a single argument, assume copy to current working directory (#3262)
  - fix posting of comment in PR with `--upload-test-report` (#3272)
  - take into account dependencies marked as external modules when composing template values like `%(pyver)s` (#3273)
- other changes:
  - increase timeout & use `api.github.com` for connectivity check in `check_github` (#3192)
  - restore `flake8` default ignores (#3193)
  - enable tracking of test suite coverage using coveralls (#3204)
  - make tests use `easybuilders/testrepository` rather than `hpcugent/testrepository` after it was moved (#3238)
  - improve raised error in `remove_dir` and deprecate `rmtree2` (#3228)

easyblocks

- 7 new software-specific easyblocks:
  - BerkeleyGW (#1951), CMake (#1936), ELSI (#1883), LAMMPS (#1964, #1975, #1978, #1982, #1997), libdrm (#1983), Mesa (#1892, #2006), SEPP (#1998)
- minor enhancements, including:
  - make ParMETIS easyblock compatible with custom `configopts` (#1774)
  - update Trinity easyblock for v2.9.0 (#1906)

  - disable ROCM and Android workspace explicitly in TensorFlow easyblock (#1944)
  - update QuantumESPRESSO easyblock for version 6.5 (#1947)
  - update Siesta easyblock to support MaX-1.0 release (#1954)
  - update Ferret easyblock for versions >= 7.5.0 (#1956)
  - update XCrySDen easyblock to support Togl dependency (#1959)
  - pass value for `TARGET` specified in `buildopts` to `testopts` and `installopts` in OpenBLAS easyblock (#1960)
  - fix netCDF(-Fortran) support in Siesta easyblock (#1967, 1971)
  - add `NCDF_PARALLEL` and METIS support to Siesta easyblock (#1973)
  - add support for `use_pip_extras` custom easyconfig parameter in `PythonPackage` easyblock (#1980)
  - update Open Babel easyblock for version 3.0.0 (#1992)
  - allow differing names for TensorFlow wheel (#1995)
  - make MATLAB runtime available from MATLAB (#2008)
- various bug fixes, including:
  - fix linking of Hypre to external BLAS/LAPACK + support building (only) static/shared library (#1885)
  - fix MPI-CXX dependency of PETSc (#1917)
  - limit amount of parallelism in TensorFlow easyblock (#1934)
  - support GCCcore and gcc4.8 (if that dir exists) in function `get_tbb_gccprefix` in tbb easyblock (#1943)
  - restore default `flake8` warnings (#1950)
  - remove tests from `build_step` and raise error on failed tests in OpenBLAS easyblock (#1952, #1955, #1962)
  - add optional `runtest` to the catch for `FATAL ERRORS` to OpenBLAS easyblock (#1958)
  - ensure right '`python`' command is used to determine Python lib dir when system Python is used with PythonBundle easyblock (#1961)
  - make sure `lib/python*/lib-dynload` exists in Python installation (#1966)
  - in version 6.0 of TensorRT, `libnvinfer.a` is renamed `libnvinfer_static.a` (#1970)
  - handle `configopts` without configure args in CMake easyblock (#1974)
  - use `checkout@v2` in GitHub Actions to fix broken re-triggered tests (#1976)
  - don't insist that pylibdir always exists in OpenCV easyblock (#1977)
  - also set `$TRINITY_HOME` environment variable in Trinity easyblock (#1979)
  - fix sanity check for OpenFOAM-Extend >= 4.1 (#1981)
  - fix pattern matching in regex subst for `I_MPI_ROOT` in impi easyblock (#1986)
  - use `remove_dir` instead of deprecated `rmtree2` in various easyblocks (#1993)
  - fix "`AttributeError: module 'git' has no attribute 'Git'`" that may occur when using Python 3 (#1994)
  - don't sanity check for `QtWebEngine` in Qt easyblock when building for POWER (#2000)

  - fix installation of TensorFlow in some environments, by setting `$PYTHONNOUSERSITE` (& more) (#2002, #2020)
  - make sure libxml2 is built with XZ provided as dependency (#2005)
  - look for Python version directories with suffixes in ROOT easyblock (#2010)
  - enable TK, FLTK and OPENGL configure options in VMD easyblock (#2013)
  - update the contributing docs (#2014)
  - fix numexpr easyblock to allow (correctly) installing it as extension (#2022)
- other changes:
  - simplify various CMakeMake-based easyblocks by enhancing `CMakeMake` (w.r.t. `CMAKE_BUILD_TYPE`, shared vs static libs, `-fPIC`) (#1929)
  - enable out of tree build by default in `CMakeMake` easyblock (#1933)
  - force building Clang without CUDA when it is not found as a proper dependency (#1968)

easyconfigs

- added example easyconfig files for 114 new software packages:
  - ABRA2 (#10272), ABRicate (#10310), ADIOS (#10036), aNCI (#9929), any2fasta (#10310), apex (#10269), archspec (#9898), ArviZ (#10366), autopep8 (#9626), BAMSurgeon (#10330), BatMeth2 (#10323), BiG-SCAPE (#10352), BinSanity (#10001), Bonito (#10269), BSMAPz (#10283), BSseeker2 (#10039), BUStools (#9838), Cbc (#10052), Cgl (#10048), CGmapTools (#10288), Clp (#10033), CoinUtils (#9937), dtcwt (#9695), ELSI (#9857), EnsEMBLCoreAPI (#8734), fastq-pair (#9894), FigureGen (#10076), Fiona (#10321), FuSeq (#10004), GenomeTools (#9797), GraphMap2 (#10299), GRASP (#9896), Groovy (#9809), gsport (#9821), gubbins (#9689), igv-reports (#9977), inferCNV (#9686), iVar (#10291), joypy (#10212), JupyterLab (#9752), kma (#10259), LAMMPS (#10371), lancet (#10271), libBigWig (#10006), libGridXC (#9858), libPSML (#5859), LtrDetector (#10343), manta (#5104), medImgProc (#10228), MedPy (#9748), Mini-XML (#10036), mkl_fft (#9887), Monocle3 (#9825), MoreRONN (#10255), motionSegmentation (#10228), NanoComp (#10212), NanoFilt (#10212), nanoget (#10212), nanomath (#10212), NanoPlot (#10212), ngspice (#9922), ntEdit (#9836), ntHits (#9833), occt (#9939), OCNet (#9955), OpenAI-Gym (#10347), OpenPIV (#9959), OpenPyXL (#10115), orca (#9518), Osi (#10361), PartitionFinder (#9983), pauvre (#10212), polymake (#9904), pretty-yaml (#10041)), PRSice (#9988), pycodestyle (#9626), pydot (#9899), pygraphviz (#9969), pylift (#10051), PyMC3 (#10279), pyparsing (#9983), PyRe (#10095), python-weka-wrapper3 (#9704), PyTorch-Geometric (#9995), qcat (#10244), RAxML-NG (#9990), Ray (#10302), rclone (#7934), Red (#9856), rstanarm (#9964), scikit-build (#9762), scVelo (#9805), SECAPR (#9721), segmentation-models (#10211), SentencePiece (#10192), SEPP (#10047), Shapely (#10309), Singular (#10030), SLATEC (#7529), spatialreg (#9767), split-seq (#9749), spoa (#9705), SSN (#9955), STEAK (#10337), stpipeline (#9736), SVG (#9905), Togl (#9868), torchtext (#10193), units (#9682), UQTk (#10279), WildMagic (#10044), Winnowmap (#10005), xtb (#9993), Zip (#9972)
- added additional easyconfigs for various supported software packages, including:
  - ABySS 2.1.5, Arrow 0.16.0, BCFtools 1.10.2, BEDTools 2.29.2, BUSCO 4.0.5, BerkeleyGW 2.1.0, binutils 2.34, CVXPY 1.0.28, CharLS 2.1.0, CheckM 1.1.2, Clang 10.0.0, CppUnit 1.15.1, canu 1.9, cutadapt 2.8, DIAMOND 0.9.30, davix 0.7.5, ELPA 2019.11.001, FastANI 1.3, FastQC 0.11.9, Ferret 7.5.0, GATK 4.1.4.1, GCCcore 9.3.0, GDB 9.1, GMAP-GSNAP-2019-09-12, GObject-Introspection 1.63.1, GPAW 20.1.0, GROMACS 2020, GTDB-Tk 1.0.2, GTK+ 3.24.13, Go 1.14.1, Gradle 6.1.1, GraphicsMagick 1.3.34, Graphviz 2.42.2, Gurobi 9.0.1, gSOAP 2.8.100, gnuplot 5.2.8, gtest 1.10.0, HDDM 0.7.5, HTSlib 1.10.2, HarfBuzz 2.6.4, Horovod 0.19.1, Hypre 2.18.2, IGV 2.8.0, IQ-TREE 1.6.12, IRkernel 1.1, iccifort 2020.0.166, igraph 0.8.0, impi 2019.6.166, ispc 1.12.0, Java 13(.0.2), Julia 1.4.0, Keras 2.3.1, Kraken2 2.0.8-beta, kim-api 2.1.3, LAST 1045, LASTZ 1.04.03, LLVM 9.0.1 + 10.0.0, LMfit 0.9.14, LS-PrePost

4.7.8, likwid 5.0.1, MAFFT 7.453, MATLAB 2019b, MMseqs2 10, Maven 3.6.3, Meson 0.53.1, MethylDackel 0.5.0, Mono 6.8.0.105, medaka 0.12.0, Nextflow 20.01.0, ncdf4 1.17, netcdf4-python 1.5.3, nodejs 12.16.1, numba 0.47.0, numexpr 2.7.1, Octave 5.1.0, OpenBLAS 0.3.8, OpenBabel 3.0.0, OpenCV 4.2.0, OpenFOAM-Extend 4.1-20191120, OrthoFinder 2.3.11, PETSc 3.12.4, PGI 19.10, PMIx 2.2.1, Pango 1.44.7, PyTables 3.6.1, PyTorch 1.4.0, parasail 2.4.1, pydicom 1.4.2, pyproj 2.4.2, Qhull 2019.1, QuantumESPRESSO 6.5, R-bundle-Bioconductor 3.10, RDKit 2019.09.3 Racon 1.4.10, ReFrame 2.21, Ruby 2.7.1, rjags 4-9, rpy2 3.2.6, SLEPc 3.12.2, SPAdes 3.14.0, SPAdes 3.14.0, STAR-Fusion 1.8.1, STAR 2.7.3a, Seaborn 0.10.0, SeqAn 1.4.2, Seurat 3.1.2, SimpleElastix 1.1.0, SimpleITK 1.2.4, Stacks 2.5, Stata 16, StringTie 2.1.0, scikit-optimize 0.7.4, statsmodels 0.11.0, TensorFlow 1.15.2 + 2.0.1, Tkinter 2.7.16, Trim_Galore 0.6.5, Trimmomatic 0.39, Trinity 2.10.0, tbb 2020.2, tqdm 4.41.1, XCrySDen 1.6.2, XGBoost 0.90, xarray 0.15.1, xmlf90 1.5.4

- minor enhancements, including:
  - add easyconfig for Java 11.0.6 on `ppc64le` and alter the Java 11 wrapper to support both x86_64 and ppc64le (#9371)
  - add additional extensions for R: `HiddenMarkov` (#9685), `lmerTest` (#9853), `VSURF` + `Rborist` (#10355)
  - change Mesa 19.1.7 + 19.2.1 easyconfigs to use custom easyblock for Mesa (#9764)
  - build shared libs and install header files for Ghostscript (#9785)
  - add MUMPS as dependency in PETSc 3.12.4 easyconfigs (#9880, #9891)
  - add Perl extensions: `Term::ReadLine::Gnu` (#9901), `URI::Escape` and `Set::IntervalTree` (#10049)
  - add dat directory to aNCI (#9929)
  - add patch to create a symlink from `libsvm.so.$(SHVER)` to `libsvm.so` in LIBSVM easyconfigs (#10045)
  - build SUNDIALS with '`pic`' (#10278)
  - add `BSgenome.Hsapiens.UCSC.hg38` + `MEDIPS` extensions to R-bundle-Bioconductor v3.10 (#10298)
  - fix checksums for `mkl-dnn` and `tbb` extensions (moved to oneAPI repo) in PyTorch easyconfigs (#10367)
  - update `Java/1.8` wrapper to `Java/1.8.0_241.eb` (#10305)
- various bug fixes, including:
  - use CMake for building double-conversion (#9659)
  - update recent libdrm easyconfigs to use custom easyblock & avoid hardcoded x86-specific sanity check (#9694)
  - add alternate checksum for OpenMolcas 18.09 (#9701)
  - use Github to download releases for MariaDB-connector-c (#9702)
  - add `-DOMPI_SKIP_MPICXX` in `configopts` for MathGL, to avoid using mpicxx during build (#9703)
  - make installing independent of build folder in pybind11 easyconfig (#9738)
  - add Lua as a dependency to gnuplot (#9773)
  - stick to `http://` source URLS for ISL in GCCcore easyconfigs, since `https://` doesn't work (#9784)
  - add alternative checksums for `farver`/`fracdiff`/`pkgmaker`/`rngtools`/`doRNG`/`cobs` extensions in R 3.6.2 easyconfigs (#9789)

  - add patch for OpenBLAS 0.3.4 w/ GCC/8.2.0-2.31.1 to fix broken tests (#9865)
  - revert removal of AVX512 vmovd with 64-bit operands in binutils 2.32 easyconfigs (#9866)
  - fix inline asm in dscal: mark x, x1 as clobbered, in OpenBLAS 0.3.8 (#9867)
  - add missing `sanity_check_commands` to cutadapt v1.18 and v2.7 easyconfigs (#9869)
  - don't overwrite `configopts` in BLAST+ easyconfigs, append to it (#9875)
  - add alternate checksum for `LaplacesDemon` in R 3.6.x easyconfigs (#9879, #10382)
  - fix redefining of preconfigopts in OpenCV easyconfigs (#9895)
  - use symlinks for terminfo files instead of hard links in ncurses 6.1 easyconfigs (#9912)
  - fix NCIPLOT build flags (#9915)
  - add missing patch to iccifort libxc easyconfigs (#9918)
  - use `checkout@v2` in GitHub Actions to fix broken re-triggered tests (#9925)
  - re-enable building utils in Siesta 4.1-MaX-1.0 release (#9936)
  - fix homepage and source URLs in SLEPc easyconfigs by using https (#9943)
  - fix source URLs for rgeos after source tarball was moved to CRAN archive (#9954)
  - add dependencies on Python 3 and SciPy-bundle in Trinity v2.9.1 easyconfig (#9957)
  - patch GCC `lisanitizer` for glibc 2.31 (#9966)
  - add Zip as build dependency for recent Bazel versions (#9972)
  - fix checksums in Jellyfish v2.3.0 easyconfigs (#9997)
  - fix source URLs for ParMGridGen easyconfigs (#10019)
  - disable unintended Octave support in all libsndfile easyconfigs (#10027)
  - fix sources for LS-PrePost 4.6 (#10236)
  - security update for vsc-mympirun 4.1.9 (#10185)
  - configure libwebp to also install `libwebpmux` (#10274)
  - ensure that CVS easyconfigs are included in source tarball produced by '`python setup.py sdist`' (#10326)
  - fix undefined reference error due to libxc 4.3.4 built with CMake (#10356)
  - fix source_urls for tbb: use (new) official '`oneapi-src`' GitHub repository (#10361)
  - update checksums and homepage in tbb easyconfigs (#10285)
- other changes:
  - use new custom easyblock in recent CMake easyconfigs (#9871, #9923)
  - add check for redefined easyconfig parameters in easyconfig tests (#9876)
  - use M4-1.4.18.eb for test installation in easyconfigs test suite (#9926)
  - use `https://` in `homepage`/`source_urls` of `zlib-1.2.11.eb` (#10018)
  - add `-GCCcore-9.2.0` versionsuffix for `intel/2020.00` components (#10083)
  - add checksum of new tbb 2019_U9 source tarball, next to original one + update homepage (#10237)
  - add comment informing about manually setting Gallium drivers in easyconfigs for Mesa v19.1.7 and v19.2.1 (#10276)

11.12.12 EasyBuild v4.1.2 (March 16th 2020)

bugfix release

Note: This release includes a bug fix for the leaking of your GitHub token in the EasyBuild (debug) log file.

We strongly encourage that you revoke the GitHub tokens you are using currently, via https://github.com/settings/tokens**, and to replace them using a new token (using** `eb --install-github-token --force` **).**

More information in https://github.com/easybuilders/easybuild-framework/pull/3248 .

---

framework

- fix gitdb dependency on Python 2.6 in test configuration (#3212)
- fix broken test for –review-pr by using different PR to test with (#3226)
- censor authorization part of headers before logging ReST API request (#3248)

easyblocks

(no changes)

easyconfigs

(no changes)

11.12.13 EasyBuild v4.1.1 (January 16th 2020)

bugfix/update release

framework

- various enhancements, including:
  - add `check_log_for_errors` function (in `easybuild.tools.run`) to detect and handle multiple errors (#3118)
  - implement support for `eb --show-ec` to show contents of specified easyconfig file (#3132)
  - also update `$XDG_DATA_DIR` (share/) and `$GI_TYPELIB_PATH` environment variables (lib*/girepository-*) in generated module files (#3133)
  - add support for `--copy-ec` to copy easyconfig file to specified location (#3142)
  - mention `--disable-*` option in `--help` output for boolean options enabled by default (#3151)
  - add `--cuda-compute-capabilities` configuration option (#3161)
- various bug fixes, including:
  - ignore imports from `vsc` namespace made from pkgutil.py (#3120)
  - only actually change permissions using `os.chmod` in `adjust_permissions` if the current permissions are not correct already (#3125)
  - use `shutil.copyfile` to just copy file contents if target path exists and is owned by someone else (#3127)
  - fix or avoid warnings that commonly arise in build log (#3129)
  - disable buffering in `asyncprocess.Popen` using `bufsize=0`, to fix `run_cmd_qa` missing output (#3130)

  - update pip & install wheel package in generated Singularity container recipes (#3136)
  - avoid crash in `modify_env` & unset `unset_env_vars` when using (older versions) of Python 3.5 & 3.6 by using `list(...)` (#3140)
  - take into account that lib64 could be a symlink to lib (or vice versa) in `get_software_libdir` function (#3141)
  - only parse docstring if it exists in `gen_easyblock_doc_section_rst` function (#3144)
  - only add useful entries for `$CPATH`, `$(LD_)LIBRARY_PATH` and `$PATH` (non-empty directories) (#3145, #3152)
  - fix `--list-software=detailed` when using Python 3 by leveraging `sort_looseversions` function from `py2vs3` module (#3146)
  - ensure subdirectories in software install directory have correct search (exec) permission (#3147)
  - take into account that a checksum value may be a tuple of valid checksum in `EasyBlock.check_checksums` (#3153)
- other changes:
  - bump to Lmod 8.2.9 in GitHub CI config (#3115)
  - update copyright statements for 2020 (#3149)
  - make Hound CI code style checker ignore "Black would make changes" produced by flake8-black (#3162)

easyblocks

- new software-specific easyblock for cryptography (to fix missing `-pthread` for all versions) (#1874)
- minor enhancements, including:
  - update WPS easyblock for v3.6 & newer (#1315)
  - update FSL easyblock to support FSL v6.0.2 & newer (#1860)
  - add `setup_cmake_env` function in CMakeMake easyblock which can be leveraged in other easyblocks, and use it for OpenFOAM (#1869)
  - remove obsolete configure options for Python + build with optimizations/LTO enabled for recent Python versions (#1876)
  - update WPS easyblock for recent versions: set `$WRF_DIR` to point to location of WRF installation (#1886)
  - make sure `$LIBLAPACK_MT` is set before using it in ESMF easyblock (#1887)
  - remove useless `PATH` entries + add `PKG_CONFIG_PATH` in imkl easyblock (#1900)
  - enhance tbb easyblock to support building on POWER (#1912)
  - enhance TensorFlow easyblock to pick up on `--cuda-compute-capabilities`, and issue a warning if no CUDA compute capabilities are specified (#1913)
  - add custom easyconfig parameter `build_type` to generic CMakeMake easyblock (#1915, #1922)
- various bug fixes, including:
  - fix for conda packages that rely on particular versions of Python (#1836)
  - fix path for intel and netCDF lib directories in NCL easyblock (#1862)
  - fix CUDA 10.1 installation on POWER (#1871)
  - change Bazel easyblock to prefer using Java dependency rather than included JDK (fix for POWER9) (#1875)

  - remove optarch warning in GROMACS for Cray toolchains (#1879)
  - also fix `$WM_COMPILE_OPTION` in OpenFOAM rc scripts to make debug builds work correctly (#1880)
  - limit MPI ranks used for running WRF test cases to max. 4 + include contents of rsl.error.0000 output file in case test failed (#1884)
  - update `$PYTHONPATH` + add `python -c 'import mrtrix3'` as sanity check command for recent MRtrix versions (#1889)
  - update sanity check in SAMtools easyblock for version 1.10 (#1890)
  - make sure `$PYTHONNOUSERSITE` it set when performing sanity check for (bundles of) Python package(s) (#1891)
  - fix install dir subdir for WPS v4.0+ that is considered for `$PATH` and `$LD_LIBRARY_PATH` (#1895)
  - impi: don't rebuild libfabric if the source code is not present (#1896)
  - also copy component patches to self.cfg in Bundle generic easyblock (#1897)
  - skip patch step in Bundle generic easyblock (per-component patches are still applied) (#1898)
  - derive easyblock for iccifort only from icc easyblock (not ifort), to avoid adding include subdir to `$CPATH` (#1899)
  - add `export LANG=C` to preinstallopts instead of install_script path in CUDA easyblock (#1902)
  - stop setting updating `$CPATH` and `$LIBRARY_PATH` for GCC and GCCcore, not required (#1903)
  - remove `$LIBRARY_PATH` entries in iccifort easyblock, already known to icc et al (#1904)
  - use major/minor version of Python command being used if `req_py_majver`/`req_py_minver` are not specified (#1907)
  - define `$EB_PYTHON` in module for EasyBuild installation, to make sure correct Python version is used at runtime (#1908)
  - fix Python easyblock to allow configuring build of Python v3.8 (`Setup.dist` script was renamed to `Setup`) (#1909)
  - fix netCDF easyblock for version 4.4.0 (#1911)
  - correct comment about when we set `RUNPARALLEL` in HDF5 easyblock (#1914)
  - do not sanity check on MATLAB compiler, since it requires a separate license (#1916)
  - build HDF5 without MPI C++ extension to avoid breaking linkage for C software that requires HDF5 (#1918, #1919)
  - override `set_pylibdirs` method in `VersionIndependentPythonPackage` to hard set `self.pylibdir` to 'lib' (#1924)
- other changes:
  - increase timeout for interactive installation command in CUDA easyblock to 1000 sec. (#1878)
  - disable running of `sudo apt-get update` in GitHub CI config, since it's failing (and we don't really need it) (#1882)
  - stop requiring Python dep for SWIG, just configure with `--without-python` if Python is not a dependency (#1894)
  - update copyright statements for 2020 (#1905)
  - make Hound CI code style checker ignore "Black would make changes" produced by flake8-black (#1923)

easyconfigs

- added example easyconfig files for 27 new software packages:
  - Autoconf-archive (#9658), breseq (#9603), CrossMap (#9483), CSBDeep (#9560), CNT-ILP (#9323), cytoolz (#9453), Faber (#9553), Fiji (#8748), GARLI (#9404), Globus-CLI (#9565), GtkSourceView (#9526), gradunwarp (#9648), gsettings-desktop-schemas (#9529), HyPhy (#9405), horton (#7449), IGM-Plot (#9438), LEMON (#9323), Meld (#9530), mhcflurry (#9554), NCIPLOT (#9419), ncl (#9632), OpenSlide (#9499), openslide-python (#9499), pythran (#9488, #9594), Qualimap (#9411), TinyDB (#9555), TreeShrink (#9381)
- added additional easyconfigs for various supported software packages, including:
  - Beast 1.10.4, Boost.Python 1.71.0, Clang 9.0.1, ESMF 8.0.0, FSL 6.0.3, fastp 0.20.0, freeglut 3.2.1, GDAL 3.0.2, GEOS 3.8.0, GROMACS 2019.4, GSL 2.6, hwloc 2.1.0, Jellyfish 2.3.0, Julia 1.3.1, LibTIFF 4.1.0, libxml2 2.9.10, lxml 4.4.2, Mothur 1.43.0, mayavi 4.7.1, molmod 1.4.5, netCDF-C++4 4.3.1, netCDF-Fortran 4.5.2, numactl 2.0.13, OpenFOAM 7, OpenFOAM v1912, OpenMM 7.4.1, OpenMPI 4.0.2, PLUMED 2.5.3, PROJ 6.2.1, plotly.py 4.4.1, pocl 1.4, QuickFF 2.2.4, R 3.6.2 w/ foss/2019b and fosscuda/2019b, ReFrame 2.20, SAMtools 1.10, SUNDIALS 5.1.0, SWIG 4.0.1, Salmon 1.0.0, SuiteSparse 5.6.0, snakemake 5.7.1, TensorFlow 2.1.0 w/ fosscuda/2019b, torchvision 0.4.2, WPS 4.1, WRF 4.1.3
- added easyconfigs for `intel/2020.00` toolchain (#9575)
- minor enhancements, including:
  - add POWER9 support to CUDA 10.1 easyconfigs (#9442)
  - build CMake in parallel (#9543)
  - use NCCL for GPU ops in Horovod 0.18.2 easyconfig (#9562)
  - update Java/1.8 wrapper to Java/1.8.0_231 (for x86_64) (#9585)
- various bug fixes, including:
  - fix remote launch of broker and workers for SCOOP (#9366)
  - fix failing RPATH sanity check for NCL 6.6.2 due to missing dependencies (+ add easyconfig using `foss/2018b`) (#9388)
  - add missing 'wheel' extensions to Spark 2.4.0 easyconfig using `intel/2018b` toolchain (#9424)
  - add missing OS dependencies in Java 1.8 easyconfig used on POWER systems (#9454)
  - fix build of recent Bazel versions on Power9 + stick to Java/1.8 as dependency (#9455)
  - fix CMake 3.15.3 build on Power (+ enable building in parallel) (#9469)
  - fix source URLs in xorg-macros easyconfigs (#9477, #9578)
  - add missing wcwidth extension to Python 2.7.15 + 2.7.16 easyconfigs & enable 'pip check' in sanity check (#9479)
  - remove (wrong) `GI_TYPELIB_PATH` and `XDG_DATA_DIRS` in various easyconfigs (#9528, #9577, #9615)
  - use xorg-macros as dependency in X11 easyconfigs (rather than installing it as a bundle component) (#9546)
  - fix lpsymphony extension for R-bundle-Bioconductor (#9548)
  - add correct 'old-versions' source URL to all Mesa easyconfigs (#9569)
  - add missing SHA256 checksums for Armadillo (#9572)
  - also define `$AUGUSTUS_BIN_PATH` and `$AUGUSTUS_SCRIPTS_PATH` in generated module file for AUGUSTUS (#9579)

  - add SSL OS dependencies for GDAL 3.0.0 (#9586)
  - add missing jupyter_contrib_core extension for IPython 7.7.0+ + consistently include jupyter_nbextensions_configurator extension (#9587)
  - patch libcxx (Clang 8.0.0) on pcc64le for incomplete IBM128 long double in GCC (#9590)
  - patch for GCCcore 8.2.0 to fix '__float128 is not supported on this target' on ppc64le (#9591)
  - fix broken easyconfigs for cyvcf2 v0.11.5 by adding missing 'monotonic' extension (#9601)
  - use absolute path for extraction to allow relocating the build dir for g2log-1.0 (#9604)
  - add alternate SHA256 checksum for kallisto-0.43.1 after re-release under same version without code changes (#9611)
  - add additional valid checksum for MASS 7.3-51.4 extension in R 3.6.0 easyconfigs (#9621)
  - update ctffind website (#9622)
  - make sure we use easybuild Clang in pocl easyconfigs (#9624)
  - make `postinstallcmds` independent of current working directory in OpenCV 3.1.0 easyconfigs (#9628)
  - update `source_urls` to include old releases folder in libsodium easyconfigs (#9632)
  - fix source URLs for ant v1.10.5 - v1.10.7 (#9633)
  - update URLs to new location of libxc (#9635)
  - add alternate SHA256 checksum for rda_1.0.2-2.1 extension in R 3.6.0 (#9644)
  - update source URLs in QCA 2.1.0 easyconfigs (#9647)
  - fix Python 3.5.1 easyconfig: bitstring 3.1.3 sources no longer available on PyPI (#9649)
  - fix tesseract 4.1.0 dependencies (#9650)
  - make ICU 64.2 depend on Python3 instead of Python 2, to avoid picking up system Python 3.x (#9652)
  - use `True` (boolean value) rather than `'True'` (string value) for boolean easyconfig parameters (#9657)
  - fix pyfits easyconfig by adding missing d2to1 extension (#9687)

- other changes:
  - disable running of `sudo apt-get update` in GitHub CI config, since it's failing (and we don't really need it) (#9492)
  - require that `sanity_pip_check` is enabled in new/changed easyconfigs (#9516, #9576)
  - update copyright statements for 2020 (#9598)
  - allow missing '-Python-*' versionsuffix for existing easyconfig files changed in PRs (#9634)

11.12.14 EasyBuild v4.1.0 (December 4th 2019)

bugfix/update release

framework

- various enhancements, including:
  - performance improvements:
    * skip validation when copying EasyConfig object for extension (#3071)

    - correctly specify that ActiveMNS & co are singleton classes when using Python 3 (#3073)
    - don't call out to prohibitively expensive `getRootLoggerName` in `getLogger`, just use 'root' instead (#3074)
    - fix inconsistent module path usage that leads to repeated reloading in HMNS (#3099)
  - add support for specifying different dependency version based on processor architecture (#3047)
  - support use of glob patterns for paths to files with external modules metadata (#3075)
  - take into account that external modules may not be visible directly (due to module hierarchy) (#3083)
  - add support for including 'extensions' statement in Lua modules with Lmod 8.2.8+ (#3085, #3107, #3110)
  - add support for `--sync-pr-with-develop` (#3087)
  - add support for `--new-branch-github`, `--new-pr-from-branch`, `--sync-branch-with-develop`, `--update-branch-github` (#3103)
  - fix typo in docstring for `new_branch_github` (#3106)
- various bug fixes, including:
  - correctly handle `iccifortcuda` toolchain with standalone `iccifort` in `det_toolchain_compilers` (#3055)
  - init git repo with `git.repo.clone()` instead of `copy_dir()` (#3062)
  - fix regular expression so depends-on statements are recognized correctly in Tcl module files (#3065)
  - update GitPython to latest version that supports Python 2.6 in requirements.txt to fix broken `test_new_update_pr` (#3066)
  - imply `--disable-pre-create-installdir` with `--inject-checksums` (#3069)
  - handle patches in extensions more like normal patches (#3067)
  - take into account that `platform.linux_distribution` and `platform.dist` was removed in Python 3.8 (#3078)
  - always include mandatory easyconfig parameters in dumped easyconfig (#3081)
  - hide backup module file when using Lmod 6.x (fixes #9302) (#3089)
  - add better error message when mandatory key is missing from a dictionary easyconfig parameter (#3092)
  - also create symlinks for default modules in class module folders (#3093)
  - fix semantics of `--set-default-module`: only set default for specified easyconfigs, not for the ones that are installed as dependencies via `--robot` (#3094)
  - fix various issues in extracting comments from original easyconfig file and including them again in dumped easyconfig (#3095)
  - don't use `%(version)s` template in `exts_default_options` in dumped easyconfig (#3096)
  - fix generated module statements in case only a single version is listed in `multi_deps` (#3097)
  - fix broken `test_show_system_info` on macOS due to 'Python' binary (#3105)
  - take into account that dependency version could be a dict rather than a string value in template_constant_dict (#3111)
- other changes:
  - deprecate running EasyBuild with Python 2.6 via new `check_python_version()` function (#3076)
  - deprecate support for using Lmod 6.x (#3077)

  - trim set of test configurations in Travis CI (#3086)
  - flesh out `env_vars_external_module` from `Toolchain._simulated_load_dependency_module` (#3088)

easyblocks

- new software-specific easyblock for cuDNN, to allow setting `cudnnarch` (#1855)
- refactored software-specific easyblock for Xmipp, based on Scons (#1837)
- minor enhancements, including:
  - add `prebuildopts` to Bazel build command (#1838)
  - add support to Toolchain generic easyblock for defining $EB* environment variables for toolchain components that use an external module (#1849)
  - add support for running 'pip check' during sanity check in generic PythonPackage easyblock (#1853)
- various bug fixes, including:
  - clean up `/tmp/cuda-installer.log` in CUDA easyblock, to avoid segfault in cuda-installer (#1835)
  - minor fix to name of Gctf binary (#1840)
  - move BLAS toolchain existence check earlier in SuperLU easyblock (#1842)
  - fixes for TensorFlow easyblocks w.r.t. Bazel build options & `__init__` in top-level google-protobuf package dir (#1843)
  - fix support for sequential version in MUMPS easyblock (#1845)
  - change default value of `files_to_copy` to None in MakeCp generic easyblock + code cleanup & use `change_dir`, `copy_dir`, `copy_file` and `mkdir` function from `filetools` (#1848)
  - prepend `-L$EBROOTZLIB/lib` to `LDFLAGS` in SCOTCH easyblock (#1850)
  - improve configuration in netCDF and netcdf4_python easyblocks (#1852)
  - fix CUDA header paths for TensorFlow versions < 1.14 (#1854)
  - handle incorrect regex better in generic CmdCp easyblock (#1861)
  - add missing docstrings in cmakeninja easyblock (#1867)
- other changes:
  - add GitHub Actions workflow to run easybuild-easyblocks test suite (#1844)

easyconfigs

- added example easyconfig files for 46 new software packages:
  - Amara (#9340), anvio (#9387), Arriba (#9226, #9244), attr (#7824), bibtexparser (#9284), bwa-meth (#9217), CITE-seq-Count (#9237), CoordgenLibs (#9374), dtcmp (#9052), fatslim (#9193), GromacsWrapper (#9177), GULP (#9243), hdf5storage (#9195), ITSTool (#7260), kim-api (#8786), kwant (#9238), libarchive (#9052), libcircle (#9052), libxml2-python (#7260), lifelines (#9215), lwgrp (#9052), maeparser (#9374), MaxQuant (#9281), MethylDackel (#9216), MoviePy (#9205), mpifileutils (#9052), mpiP (#9059), nanofilt (#8502), NOVOPlasty (#9326), openkim-models (#8786), parallel-fastq-dump (#9218), pasta (#9348), pyqstem (#9277), python-Levenshtein (#9237), RapidJSON (#9373), RDFlib (#9346), RQGIS3 (#9125), Short-Pair (#9376), SpliceMap (#9375), TRIQS-cthyb (#9230), TRIQS-dft_tools (#9230), TRIQS-tprf (#9230), UMI-tools (#9237), VarDict (#7283), Xmipp (#9257), XSD (#9347)
- added additional easyconfigs for various supported software packages, including:

  - awscli 1.16.290, BLIS 0.6.0, Bazel 1.1.0, Biopython 1.75, Blender 2.81, bokeh 1.4.0, CONCOCT 1.1.0, CUDA 10.2.89, Catch2 2.11.0, CellRanger 3.1.0, CheckM 1.0.18, dask 2.8.0, deepTools 3.3.1, FastANI 1.2, Flye 2.6, GDCM 3.0.4, GTDB-Tk 0.3.2, Glade 3.8.6, Hadoop 2.9.2, h5py 2.10.0, hypothesis 4.44.2, IPython 7.9.0, Kaiju 1.7.2, Kraken 1.1.1, libsodium 1.0.18, MEGAHIT 1.2.8, Mesa 19.2.1, MetaBAT 2.14, matplotlib 3.1.1, metaWRAP 1.2.2,cNCCL 2.4.8, NGS 2.10.0, NiBabel 2.5.1, netCDF 4.7.1, networkx 2.4, numba 0.46.0, OpenCV 3.4.7, OpenCoarrays 2.8.0, OpenEXR 2.4.0, OpenFOAM v1906, OpenImageIO 2.0.12, ParaView 5.6.2, Pillow 6.2.1, PyTorch 1.3.1, PyYAML 5.1.2, Pysam 0.15.3, picard 2.21.1, prokka 1.14.5, protobuf 3.10.0, R-keras 2.2.5.0, Racon 1.4.7, SCOTCH 6.0.9, SRPRISM 3.1.1, Salmon 0.14.2, SciPy-bundle 2019.10, Subread 2.0.0, scikit-image 0.16.2, scikit-learn 0.21.3, TRIQS 2.2.1, TensorFlow 1.15.0, TensorFlow 2.0.0 w/ fosscuda/2019b, Tkinter 3.7.4, ToFu 1.4.1, tbb 2019_U9, Xerces-C++ 3.2.2, Xmipp 3.19.04, yaff 1.6.0

- added easyconfigs for `intelcuda/2019a` toolchain (#9271)
- minor enhancements, including:
  - tweak Java 1.8 wrapper to use different Java version on POWER systems (#9081)
  - add jupyter_nbextensions_configurator extension to IPython 7.7.0 easyconfigs (#9133)
  - add additional extensions to R 3.6.0 easyconfigs (#9184, #9275)
  - add additional extensions to R-bundle-Bioconductor 3.9 easyconfig (#9185, #9349, #9410)
  - enhance sanity check in cutadapt 1.18 easyconfigs + consistently use PythonBundle & `use_pip` (#9219)
  - update cuDNN 7.6.4.38 easyconfigs to support both x86_64 and ppc64le (#9331)
  - tweak NCCL 2.4.8 easyconfig to support x86_64 and ppc64le (#9336)
  - define `$SPARK_HOME` in generated module file for Spark 2.4.0 (#9408)
  - add sanity check command for matplotlib 3.x with Python 3 to check import from mpl_toolkits (#9413, #9414)
- various bug fixes, including:
  - explicitly set `SYSCONFDIR` configure option in TurboVNC easyconfig (#9137)
  - patch pigz Makefile so zlib provided by EasyBuild is picked up (#9138)
  - add libjpeg-turbo as dependency to recent LibTIFF easyconfigs, to avoid picking up LibTIFF installed in system (#9146)
  - add freetype as dependency to OpenImageIO, to avoid picking up freetyype installed in system (#9147, #9152)
  - fix definition of `fosscuda/2019b` to make sure it works with hierarchical MNS (#9178)
  - add missing setuptools_scm extension required to build dateutil extension in Python 3.7.0 easyconfigs (#9209)
  - add Python as build dependency for recent Bazel versions (#9223, #9299, #9342)
  - fix homepage & description in Bioconductor easyconfigs (#9225)
  - fix checksum in Stacks 2.41 easyconfig after sneaky re-release (#9232)
  - apply fixes to ImageJ 1.51k easyconfig (#9245)
  - consistently use patch for OpenCV 3.4.7 (#9279)
  - use protobuf 3.10.0 as build dep for TensorFlow 2.0.0 w/ fosscuda/2019b + use nodocs variant of git as build dep (#9298)
  - add Jasper dependency to Qt5 v.5.13.1 (#9313)

  - fix Python 3.7.2 required OpenSSL version for old OS to the one provided on the same toolchain (#9324)
  - add missing extensions required by Sphinx & pytest to easyconfigs for Python 3.7.2 and 3.7.4 (#9329)
  - update TensorFlow v1.14.0 + v2.0.0 CUDA patch to handle compiler wrappers like ccache (#9333)
  - patch binutils 2.31.1 and 2.32 to fix compatibility with RHEL8 (#9335)
  - add missing extensions in TensorFlow 2.0.0 easyconfigs (+ update to tensorboard/tensorflow-estimator 2.0.1) (#9338)
  - fix logic to determine location of scripts dir + ensure right compiler flags are used in KAT easyconfigs (#9360)
  - add missing GCCcore-6.3.0_fix-sanitizer_linux.patch in GCCcore 6.4.0 easyconfig (#9362)
  - fix linker errors when linking with libhts.a for MetaBAT 2.12.1 (#9379)
  - add egg-info file via patch in VTK v8.2.0, for Pytho 2.7.15, 3.7.2, 3.7.4 (#9386)
  - promote binutils to a runtime dependency for Python in GCCcore based builds (#9402)
  - fix archive URL typo for ncdf4 (#9407)
  - fix problems with mpl_toolkits namespace for matplotlib easyconfigs using Python 2 (#9415, #9416, #9417)
- other changes:
  - ignore commented out lines in easyconfig files when checking for http:// URLs (#9224)
  - add GitHub Actions workflow to run easybuild-easyconfigs test suite (#9231, #9255)
  - archive old patches for Xmipp 3.1 (#9256)
  - speed up easyconfigs test suite by avoiding re-parsing and re-ordering of easyconfigs (#9236)
  - only run easyconfigs test suite with Python 2.7 & 3.6 + Lmod 7 in Travis CI (#9297)
  - archive ACML easyconfigs (#9367)
  - update CMake build in Eigen 3.3.7 to use more recent toolchain (#9398)

11.12.15 EasyBuild v4.0.1 (October 15th 2019)

bugfix/update release

framework

- various enhancements, including:
  - add '`parallel`' to list of config templates (#3036)
  - add GitHub Actions workflow to run easybuild-framework test suite (#3039)
  - add '`retest`' as a reason to `--close-pr`, to close/re-open PRs to trigger re-test in Travis (#3040)
  - define `$EB_SCRIPT_PATH` in '`eb`' wrapper script, and consider it before location of '`eb`' determined via `$PATH` in `get_paths_for` function (#3046)
  - add support for `--remove-ghost-install-dirs` configuration option, and warn about (potential) ghost install dirs by default when `--force`/`--rebuild` is used (#3050)
- various bug fixes, including:
  - update bootstrap script to support installing EasyBuild v4.0 (#3017)
  - fix broken `test_download_repo` due to archiving of easyconfigs (#3019, #3023)

  - avoid that `--inject-checksums` introduces list of patches for extensions as a single long line (#3025, #3034)
  - enhance regex in `fix_shebang` method to fix more Python/Perl shebangs + avoid patching binary files (#3029)
  - delete test gist that is created by `--check-github` (#3031)
  - disable templates when defining easyconfig parameters in `EasyConfig.set_keys()` (#3037)
  - avoid setting GC3Pie's max_in_flight to `None` if `--job-max-jobs` is not specified (#3038)
  - fix use of `obtain_file` method for extensions (#3042)
  - error out if some GC3Pie job failed (#3044)

easyblocks

- one new generic easyblock: `CMakeNinja` (#1829)
- new software-specific easyblock for Gctf (#1827), MotionCor2 (#1819)
- minor enhancements, including:
  - update OpenFOAM easyblock for changes in version 1906 w.r.t. wamke rules (#1772)
  - add `%(cudaarch)s` template variable so that it can be used in sources (#1797)
  - update Boost easyblock for versions >=1.71.0 (#1814)
  - update RepeatMasker easyblock for version 4.0.9 (#1815)
  - add `--verbose` flag to '`pip install`' when running EB in debug mode (#1822)
  - update TensorFlow easyblock to support TensorFlow 2.0 (#1823)
  - add support in TensorFlow easyblock to run a custom test script as smoke test (#1824)
  - add support for installing QScintilla on top of PyQt5 (#1825)
  - update VEP easyblock to make installation compatible with `Bio::EnsEMBL::XS` (#1828)
- various bug fixes, including:
  - enhance TensorFlow easyblock to support installing TF 1.14.0 with CUDA and MPI support (#1811, #1816)
  - avoid `UnicodeDecodeError` when reading '`configure`' script and checking for '`Generated by GNU Autoconf`' in `ConfigureMake` (#1817, #1821)
  - don't require '`python`' command to install libxml2 without Python bindings (#1818)
  - make sure "`generic=True`" actually turns on generic build in GCC easyblock (#1826)
  - fix compatibility of Trilinos easyblock with Python 3 (#1831)

easyconfigs

- added example easyconfig files for 58 new software packages:
  - ADOL-C (#9098), ALFA (#9106), ASTRID (#9088), Annif (#8536), bnpy (#8989), bpp-core (#9064), bpp-phyl (#9064), bpp-seq (#9064), Clang-Python-bindings (#9084), CPB (#5869), Centrifuge (#8714), Chromaprint (#9047), Con3F (#8755), DeepSurv (#8096), Essentia (#9054), FastRFS (#9088), GAT (#5871), Gaia (#9049), Gctf (#9097), GenomeMapper (#5872), Infomap (#9091), kpcalg (#8740), libglvnd (#9111, #9130), libsamplerate (#9046), libssh (#8865), libzip (#9073), MetaboAnalystR (#8773), Metaxa2 (#8939), MotionCor2 (#8942), NFFT (#9085), PhyML (#9103), PlaScope (#8714), PyCharm (#9100), pb-copper (#8928), pbmm2 (#8929), phylokit (#9088), phylonaut (#9088), phyx (#9090), pycma (#8834), Q6 (#9069), Qt5Webkit (#9120), ROME (#9050, #9062), rioxarray (#9007), SVDquest (#9088), savvy

(#9124), sciClone (#7806), shapAAR (#8983), shrinkwrap (#9124), Structure (#5866), trimAl (#9063), thurstonianIRT (#9080), TurboVNC (#9110, #9111, #9128), Tracer (#8970), TagLib (#9048), TRIQS (#8835), THetA (#8875), vcfnp (#5862), WebSocket++ (#8842)

- added additional easyconfigs for various supported software packages, including:
  - Armadillo 9.700.2, arpack-ng 3.7.0, BLASR 5.3.3, Bazel 0.26.1 + Bazel 0.29.1, Cufflinks 20190706, DL_POLY_Classic 1.10, FFmpeg 4.2.1, Go 1.13.1, Horovod 0.18.1, IOR 3.2.1, Julia 1.2.0, LLVM 9.0.0, Mesa 19.1.7, Molden 6.1, Mono 6.4.0.198, NCO 4.8.1, , Net-core 3.0.0, Nim 1.0.0, OpenFOAM 2.2.x, PGI 19.7, PLUMED 2.5.2, PMIx 3.1.4, PostgreSQL 11.3, psycopg2 2.8.3, QGIS 3.4.12, QScintilla 2.11.2, Qt5 5.13.1, ReFrame 2.19, Rust 1.37.0, Spack 0.12.1, TAMkin 1.2.6, TensorFlow 1.14.0 w/ fosscuda/2019a, TensorFlow 2.0.0 w/ foss/2019a, UCX 1.6.1, VEP 96.0, xarray 0.13.0
- added easyconfigs for `fosscuda/2019a` toolchain (#9066)
- minor enhancements, including:
  - add EBImage extension to easyconfig for R-bundle-Bioconductor 3.9 (#8982)
  - add check for `http://` URLs in easyconfig files added/changed in PRs (#9012)
  - add bbmle/emdbook/SOAR/rasterVis/tictoc extensions to R 3.6.0 easyconfigs (#9037)
  - updated PyQt5 5.12.1 easyconfig to also build sip files + minor readability changes (#9071)
  - enabled `SQLITE_ENABLE_COLUMN_METADATA`, which is needed for GDAL (and QGIS) (#9118)
  - also install include/GL/internal/ for recent Mesa installations (#9129)
- various bug fixes, including:
  - add ncurses as dependency to lftp (#8646)
  - add patch for gettext 0.19.8* to avoid picking up global git config that could break the installation (#8957)
  - fix source URLs in GlimmerHMM easyconfigs (#8980)
  - add patch for PyTorch 1.2.0 to use version of torchvision that is compatible with PyTorch 1.2.0 (#8986)
  - clarify the comment regarding the optarch setting in ITK-5.0.1 (#8991)
  - fix homepage & description in easyconfig file for YAPS (#8993)
  - add patch for PyTorch 1.2.0 to fix failing softmax test on Intel Sandy Bridge (#9010)
  - fix permissions for TRF (#9034)
  - ICU needs Python 2.7+ to build, so add that as builddependency (#9053)
  - fix urls for Anaconda and Miniconda (#9087)
  - use a cuDNN version that has support for the CUDA version in `fosscuda/2019a` (CUDA 10.1) in PyTorch, TensorFlow and Theano easyconfigs (#9112)
- other changes:
  - make sources in CUDA 10.1.105 use `%(cudaarch)s` template value (to use different source on POWER systems) (#8136)
  - update `Java/1.8` wrapper to `Java/1.8.0_221` (#9038)
  - allow divergent Java dep version as long as it's indicated by versionsuffix (#9041)

11.12.16 EasyBuild v4.0.0 (September 20th 2019)

feature release (incl. backwards-incompatible changes)

framework

- **fixed compatibility with Python 3.5+** (#2708, #2713, #2714, #2719, #2721, #2723, #2729, #2743, #2744, #2751, #2756, #2759, #2761, #2762, #2765, #2766, #2767, #2768, #2774, #2775, #2778, #2780, #2785, #2787, #2789, #2791, #2792, #2794, #2800, #2801, #2805, #2806, #2895, #2932, #2982, #2992, #3007, #3011)
  - supported Python versions: 2.6, 2.7, 3.5, 3.6, 3.7
  - some functionality from the Python standard library should be imported from the new `easybuild.tools.py2vs3` package
  - see py2_py3_compatibility & eb4_relocated_functions_classes_constants
- **ingested relevant code from** `vsc-base` & `vsc-install` (#2708, #2713, #2714, #2763, #2790, #2993)
  - `vsc-base` & `vsc-install` are no longer required as dependencies
  - the functionality that was provided by these packages is now (mostly) available from the `easybuild.base` package
  - any import statements from the '`vsc`' namespace will result in an error
  - see also eb4_changes_ingested_vsc_base and eb4_relocated_functions_classes_constants
- `setuptools` **is longer required** (neither for installing or using EasyBuild) (#2836, #2837, #2984, #2986, #2988)
  - see also eb4_no_required_deps
  - this required moving some classes and constants, see eb4_relocated_functions_classes_constants
- **the** '`dummy`' **toolchain is deprecated and replaced by the** '`system`' **toolchain** (#2877, #3001)
  - if '`dummy`' is still used as a toolchain in easyconfig files, a warning will be printed
  - use "`toolchain = SYSTEM`" instead
  - for more information, see system_toolchain
- **a warning is printed when local variables in easyconfig file don't follow the recommended naming scheme** (#2938, #2968)
  - see also *Local variables in easyconfig files*
- **names of software installation directories are independent of module naming scheme by enabling** `--fixed-installdir-naming-scheme` **by default** (#2999)
  - see eb4_changes_fixed_installdir_naming_scheme for more information
- various other small enhancements, including:
  - use requests instead of urllib2 for 403 errors (#2695)
  - use pip requirements file in Travis (#2874)
  - add support for `--fix-deprecated-easyconfigs` (#2881, 2898, #2938)
  - add `-L$EBROOTIMKL/lib/intel64` to `$LDFLAGS` for MKL (#2930)
  - handle dict-type checksums in check_checksums_for (#2974)
  - allow that `icc` & `ifort` are not actual dependencies in `iccifort` toolchain module (#2995)
  - define mapping for `iccifort`->''intel'' for `iccifort` compiler-only toolchain for HMNS (#2996)

  - also consider concatenation of compiler module names to determine details of toolchain compiler component (#2997)
  - update metadata for Cray-provided external modules (#3013)
- various bug fixes, including:
  - update the PyPI trove classifiers in setup.py (#2875)
  - reverse lists for `$LDFLAGS` and `$CPPFLAGS` (#2931)
  - enhance/fix to_template_str function to do a better job at using template values in dumped easyconfigs (#2948)
  - also take into account `--filter-deps` when finalizing parsed dependencies to fix problem with dependency filters using version ranges (#2983)
  - fix broken `--from-pr` tests due to archiving of easyconfigs (#2985)
  - fixes required to avoid breaking Cray support (#3008)
  - fixes for `--read-only-installdir`: avoid crash with `ModuleRC` easyblock + also make log file in installdir read-only (#3012)
- other changes:
  - add check to ensure that `--robot` argument specifies an existing directory (#2086)
  - remove old scripts that are no longer useful (+ minor fixes to others) (#2897)
  - use '`command -v`' to avoid requiring '`which`' in '`eb`' command (#2979)
  - add `modluafooter` & `modtclfooter` at the end of the generated module file (#3003)
  - print warning when '`eb`' command is not found in `$PATH` and for empty robot search path (#3006)

easyblocks

- fixes due to backwards-incompatible changes in easybuild-framework v4.0.0
  - use `is_system_toolchain()` rather than checking toolchain name against DUMMY_TOOLCHAIN_NAME constant (#1690, #1728)
- fix compatibility of various easyblocks with Python 3 (#1640, #1644, #1648, #1721, #1794, #1808, #1795, #1796, #1807, #1809)
- new software-specific easyblocks for `OpenMPI` (#1789, #1801), `iccifort` (#1799) and `numexpr` (#1803, #1804)
- various other enhancements, including:
  - removed checks for Boussinesq and sonic solvers for OpenFOAM 7, since those have been deprecated (#1733)
  - update Paraver easyblock to support recent versions (#1790)
- various bug fixes, including:
  - update the PyPI trove classifiers (#1723)
  - make the plugins build use the correct Tcl library version in VMD easyblock (#1786)
  - use `*_MT libs` for BLAS/LAPACK only if `openmp` is enabled in Trilinos easyblock (#1791)
  - filter out empty entries in `$CPATH` or `$C_INCLUDE_PATH` when building Perl, since that breaks the build (#1800)
  - set `$XDG_CACHE_HOME` in TensorFlow easyblock to avoid that `pip` (ab)uses `$HOME/.cache` (#1802)

  - don't load modules for dependencies in `CrayToolchain.prepare_step` (#1805)
- other changes:
  - drop requirement for setuptools as runtime dependency (#1689)
  - enable '`check_ldshared`' in generic PythonPackage easyblock by default for recent Python versions (#1788)
  - stop trying to use `setuptools.setup` in `setup.py`, always use `distutils.core.setup` instead (#1793)

easyconfigs

- fixes due to changes in easybuild-framework v4.0.0
  - use `SYSTEM` toolchain rather than deprecated `dummy` toolchain (#8369, #8711, #8822)
  - fix names of local variables (#8682-#8688, #8690, #8695-#8702, #8709, #8710, #8715, #8717, #8718, #8720-#8732, #8822)
  - enable `--local-var-naming-check=error` for easyconfigs tests (#8784)
  - stick to '`dummy`' toolchain for now in easyconfig for latest EasyBuild 3.x (#8829)
- fix compatibility with Python 3: also run easyconfigs tests with Python 3.5, 3.6 and 3.7 (#7778, #7836, #8293)
- added easyconfigs for new common toolchains: `foss/2019b` (#8567), `intel/2019b` (#8681)
  - `iccifort` is now installed as a single entity (no more separate `icc`/`ifort` installations from `intel/2019b` onwards) (see also #8879)
  - `versionsuffix` has been stripped down for toolchain components (GCC/binutils/OpenBLAS versions are no longer included)
  - see also https://easybuild.readthedocs.io/en/latest/Common-toolchains.html
- added example easyconfig files for 28 new software packages:
  - AGFusion (#8840), Bonmin (#8855), causalml (#8871), ClonalFrameML (#6082), Control-FREEC (#8794), corner (#8886), CVXPY (#8662), cytosim (#8368), dill (#8885), Dsuite (#8713), GDCHART (#8679), gifsicle (#8664), guenomu (#8677), JsonCpp (#8841), libxml++ (#8896), LOHHLA (#7227), Longshot (#8830), MDBM (#8850), nglview (#8860), ownCloud (#6804), ptemcee (#8884), pubtcrs (#7500), pyiron (#8860), qpth (#8665), QtKeychain (#6804), rgdal (#8826), smallgenomeutilities (#8507), umis (#8812)
- added additional easyconfigs for various supported software packages, including:
  - Blosc 1.17.0, bokeh 1.3.4, cURL 7.66.0, csvkit 1.0.4, dask 2.3.0, Extrae 3.7.1, FSL 6.0.1, GLibmm 2.49.7, git 2.23.0, IPython 7.7.0, numexpr 2.7.0, OSU-Micro-Benchmarks 5.6.2, OpenBLAS 0.3.7, OpenSSL 1.1.1d, ParaView 5.5.2, Paraver 4.8.1, Perl 5.30.0, PnetCDF 1.10.0, Porechop 0.2.4, PyTables 3.5.2, PyTorch 1.2.0, Python 2.7.16 + 3.7.4, parallel 20190622, phonopy 2.2.0, QIIME2 2019.7, Qiskit 0.12.0, REMORA 1.8.3, scikit-image 0.15.0, spglib-python 1.14.1.post0, torchvision 0.3.0, X11 20190717
- various additional minor enhancements, including:
  - add several extensions to R 3.6.0 easyconfigs (#8843, #8881)
  - add pRoloc to R-bundle-Bioconductor v3.9 (#8882)
  - clean up OpenMPI 3.1.* and 4.* easyconfigs to use custom OpenMPI easyblock (#8889, #8890)
  - update numexpr easyconfigs to use custom easyblock for numexpr (#8901)
  - switch to `PythonBundle` & enable use_pip in old dask easyconfig files (#8922)
  - update `CrayCCE`, `CrayGNU`, `CrayIntel` and `CrayPGI` toolchains to 19.06 (#8944)

- various bug fixes, including:
  - make TensorFlow 1.7.0 work for AMD CPUs (#6256)
  - make sure that right Python wrapper is used in VTK8 (#7296)
  - update the PyPI trove classifiers (#8298)
  - add missing checksum for matplotlib v3.0.3 (#8643)
  - add patch to `plugins/Make-arch` to use the correct Tcl library version in VMD (#8820)
  - fix issue where '`print_qiime_config.py -t`' sanity check command fails for QIIME 1.9.1 because of missing subdir in `$PYTHONPATH` (#8838)
  - update homepage info in likwid (#8846)
  - disable threading in `preprocessCore` extension included with Bioconductor 3.9 to work around conflict with OpenBLAS's threading (#8847)
  - add `-lrt` patch to PyTorch 1.1.0 easyconfig (#8852)
  - fix incorrect escaping in SIP configure options in PyQt5 easyconfigs (#8856)
  - add missing Autotools build dep for fastq-tools (#8858)
  - add missing deps for zlib, bzip2, and XZ for angsd (#8867)
  - apply patch to R package uroot in R 3.6.0 (#8872)
  - consider archive source URL for all extensions in R-tesseract easyconfig (#8897)
  - add pkg-config build dep for tesseract v4.0.0 (#8898)
  - fix `source_urls` in byacc easyconfig files (#8899, #8908)
  - add missing cairo dependency to PRINSEQ easyconfig file (#8902)
  - configure OpenMPI 1.10.x with `--without-ucx` to avoid problems when `ucx-devel` is installed in the OS (#8903)
  - add GDAL 3.0.0 for Python 2.7.15 and fix the Python 3.7.2 version (#8912)
  - fix homepage & description in scikit-image easyconfigs (#8916)
  - add faulthandler patches to Python 3.7.0 easyconfigs (#8832)
- other changes:
  - archive ancient CUDA 5.0.35 easyconfigs with creative way of determining sources (#7796)
  - remove ancient easyconfigs from archive (#8542)
  - archive easyconfigs using deprecated toolchains (#8557, #8558, #8585)
  - archive ancient versions of GC3Pie/GCC/OpenMPI/ORCA (#8586) & CPLEX (#8765)
  - rename SALMON to SALMON-TDDFT to fix name clash with Salmon (#8613)
  - bump AnnotationDb version in bundle for Bioconductor 3.9 (#8854)
  - stop trying to use `setuptools.setup` in `setup.py`, always use `distutils.core.setup` instead (#8866, #8892, #8894)
  - archive easyconfigs using ancient `Cray*` toolchains (#8945)

11.12.17 EasyBuild v3.9.4 (August 23rd 2019)

bugfix/update release

framework

- various enhancements, including:
  - add support for specifying checksum via dict providing a filename-to-checksum mapping (#2946)
  - enhance 'completed' message with how much time was needed for the installation (#2956)
  - add support for specifying tuple of alternative checksums (#2958)
  - support using '`system`' as alias for '`dummy`' toolchain + `SYSTEM` constant (#2960)
  - ensure '`docurls`' easyconfig parameter value is a list of string values, not a single string value (#2963)
  - automatically enable `--ignore-osdeps` when using `--check-contrib` or `--check-style` (#2965)
- various bug fixes, including:
  - escape '+' in search queries + handle invalid search queries better (#2967)
  - also consider extension patches when determining for which easyconfig a given patch file is intended in `--new-pr`/`--update-pr` (#2969)
  - unset `$CDPATH` early on if it is defined (#2970)
  - create specified temporary log directory if it doesn't exist yet in `init_logging` (#2972)
  - don't indicate first multi_deps version as default in module help text when `multi_deps_load_default=False` (#2973)
- other changes:
  - deprecate toolchains older than `gompi/2016a` and `foss/2016a` (#2951)

easyblocks

- new software-specific easyblock for MSM (#1770, #1775, #1776)
- minor enhancements, including:
  - allow specifying the license file directly in MATLAB easyblock (#1712)
  - enhance CPLEX easyblock to also build Python bindings (#1738)
  - workaround for XCrySDen for Tcl 8.6 (#1749)
  - update WIEN2k easyblock to support version 19.1 (#1758)
  - resolve custom `%(cudaarch)s` template value for CUDA sources (#1766)
  - enhance sanity check for Blender to make sure that Cycles render engine is available (#1779)
- various bug fixes, including:
  - only check for qtwebengine in custom easyblock for Qt(5) if glibc is sufficiently recent (#1771)
  - correctly define `comp.src` in Bundle easyblock, to fix compatibility with easyblocks that leverage `self.src` (#1777)
  - fix `Bundle.check_checksums` to checksums for extensions are also checked (#1778)

easyconfigs

- added example easyconfig files for 36 new software packages:

  - ArrayFire (#8461), BRAKER (#8437), bwidget (#8477), Catch2 (#8703), core-counter (#8749), CubeGUI (#6328), CubeLib (#6328), CubeWriter (#6328), dagitty (#8606), enaBrowserTool (#8795), GEMMA (#8270), GeneMark-ET (#8437), GenomeThreader (#8437), ieeg-cli (#8793, #8811), Judy (#8543), Julia (#8578), libaio (#8543), libtirpc (#8792), magick (#8545), MSM (#8556), MSPC (#8531), mygene (#8809), OpenMolcas (#7699), PhiPack (#8750), plc (#8796), plotly.py (#8756), pymemcache (#8663), PySCF (#8736), qcint (#8736), Qiskit (#7592), QuaZIP (#8672), re2c (#8543), SeqAn3 (#8651), snippy (#8635), spaln (#8437), V8 (#8676)

- added additional easyconfigs for various supported software packages, including:
  - ASE 3.18.0, BEDTools 2.28.0, Bowtie 1.2.3, bzip2 1.0.8, CPLEX 12.9, CUDA 10.1 update 2, cyvcf2 0.11.5, EIGENSOFT 7.2.1, GC3Pie 2.5.2, GCC(core) 9.2.0, GPAW 19.8.1, GlobalArrays 5.7, IMB 2019.3, imageio 2.5.0, jemalloc 5.2.0, nodejs 10.15.3, PyTorch 1.1.0, pybedtools 0.8.0, Quandl 3.4.8, R 3.6.0 w/ intel/2019a, R-bundle-Bioconductor 3.9, Salmon 0.14.1, Scalasca 2.5, Score-P 6.0, Stacks 2.41, TensorFlow 1.14.0, ToFu 1.4.0, WIEN2k 19.1, Wannier90 3.0.0, XCrySDen 1.5.60
- minor enhancements, including:
  - add patch and dependencies to easyconfig for Qt5 5.12.3 to fix Xlib support & enhance the installation (#8544)
  - update nodejs to version 10.15.3 and build libnode and libv8 shared libs (#8546)
  - add extensions to R 3.6.0 easyconfig: `MIIVsem` (#8565), `medflex` (#8680), `Rserve`/`spls` (#8758), `Boruta`/`CovSel`/`ctmle`/`BayesPen` (#8805)
  - include PyQtWebEngine bindings in easyconfig for PyQt5 5.12.1 using Python 3.7.2 (#8572)
  - switch GLX backend to Gallium in Mesa-19.0.1 (#8594)
- various bug fixes, including:
  - fix typo in description in GroopM easyconfig file (#8346)
  - add missing bugfix patch to easyconfig for OpenMPI 3.1.4 (#8566)
  - downgrade AtomPAW to last supported version in ABINIT 8.10.3 (#8571)
  - switch easyconfig for glew 2.1.0 to `ConfigureMake` easyblock (#8595)
  - fix checksum for source tarball in `foss/2018b` easyconfig of NAMD v2.13 (#8602)
  - update URL for bzip2 easyconfigs (#8614)
  - add patch for M4 1.4.17 to fix installation on top of glibc 2.28 (#8666)
  - add patch for Bison 3.0.4 to fix installation on top of glibc 2.28 (#8675)
  - avoid needless requirement for matplotlib < 3.0.0 in MultiQC easyconfigs (#8691)
  - fix checksum in OpenMPI 1.8.8 easyconfigs (#8692)
  - add alternative checksums for `nlme`/`mgcv`/`foreign`/`boot` extensions in R 3.5.1 and 3.6.0 easyconfigs (#8564, #8762)
  - add missing core-counter dependency for worker 1.6.8 (#8749)
  - add missing pkg-config build dependency in various easyconfigs for (#8763, #8775, #8777, #8776, #8764, #8787, #8816)
  - add patch to Python 3.7.2 easyconfig to fix faulthandler segfault (#8781)
  - set `$CPLUS_INCLUDE_PATH` in easyconfigs for older matplotlib versions (#8785)
  - patch out removed glibc 2.28 header from GCC libsanitizer (#8789)
  - include `sysmacros.h` directly to work around removal from glibc header in numactl easyconfig (#8790)

  - adapt gzip's bundled gnulib for glibc 2.28 (#8791)
  - add libtirpc and depend on it in easyconfig for libdap 3.20.3 (#8792)
  - add missing YAML extension to Perl 5.28.0 easyconfig (required by BioPerl scripts) (#8806)
- other changes:
  - remove broken easyconfigs for ciftify due to non-trivial missing dependencies (#8560)

11.12.18 EasyBuild v3.9.3 (July 8th 2019)

bugfix/update release

framework

- various enhancements, including:
  - add a URL whatis line to generated module files (#2933)
- various bug fixes, including:
  - stick to Ubuntu Trusty when testing with Python 2.6 in Travis (#2928)
  - honor `--tmp-logdir` when determining location of temporary log file (#2937)

easyblocks

- minor enhancements, including:
  - also install docs and demos in Rosetta easyblock (#1745)
  - update HEALPix easyblock to work with latest version of HEALPix' `configure.sh` (#1752)
  - make HPCG log pattern more generic, it changed in 3.1 (#1753)
  - allow bootstrapping GCC with profile-guided optimizations (#1756)
  - only require vsc-base for EasyBuild 2.x and 3.x in EasyBuildMeta easyblock (#1757)
  - extend question patterns with '`https`' URL entry in Modeller easyblock for recent versions (#1762)
  - `modextrapaths` implemented on easyblock level, `sanity_check_step`, `gcc_target` added in HEALPix easyblock (#1763)
  - update OpenBLAS easyblock to be aware of POWER9 support in OpenBLAS 0.3.6 (#1765)
- various bug fixes, including:
  - stick to Ubuntu Trusty when testing with Python 2.6 in Travis (#1751)
  - fix scripts installation path in SNPhylo easyblock (#1759)
  - be more patient when running interactive `Allwmake.firstInstall` command for recent OpenFOAM-Extend versions (#1761)
  - also add `--without-go` configure option for SWIG as we "disable everything by default" (#1754)
  - don't assume cmake arguments when configure_cmd is set in CMakeMake easyblock (#1750)

easyconfigs

- added example easyconfig files for 25 new software packages:
  - Arb (#8137), AtomPAW (#8506), ciftify (#8457), cysignals (#8459), deal.II (#8440), FastQTL (#8449), FLINT (#8137), gdbgui (#8488), gearshifft (#8482), jbigkit (#8442), lavaan (#8539), libRmath (#8449), OR-Tools (#8364, #8523), p4est (#8440), ppl (#8459), pplpy (#8459), PRISMS-PF (#8440), PyAPS3

(#8398), pyEGA3 (#8418), ReFrame (#8481), S4 (#8487), SALMON (#8478), TM-align (#8510), UnZip (#8474), XTandem (#8517)

- added additional easyconfigs for various supported software packages, including:
  - ABINIT 8.10.3, CFITSIO 3.47, GDB 8.3, GROMACS 2019.3, HEALPix 3.50, HPCG 3.1, Nilearn 0.5.2, OpenBLAS 0.3.6, Xerces-C++ 3.2.0
- minor enhancements, including:
  - update easyconfig for CFITSIO 3.45 with https and sanity check (#8472)
  - add AtomPAW and Wannier90 support to ABINIT 8.10.2 easyconfig (#8506)
  - add '`resample`' extension to R 3.5.1 + 3.6.0 easyconfigs (#8538)
- various bug fixes, including:
  - fix source URLs for Mesa 17.{2,3} with fosscuda toolchain (#8446)
  - add in Boost dependency to canu-1.8 easyconfigs using `foss` toolchain (#8470)
  - stick to Ubuntu Trusty when testing with Python 2.6 in Travis (#8483)
  - fix incorrect order of extensions for Python-2.7.14 easyconfigs (`pycparser` and `cffi` must come before `cryptography`) (#8495)
  - suppress installation of `libbfd` and `libopcode` for GDB (#8496)
  - fix KronaTools easyconfigs to make sure symlinks in bin are not broken (#8508)
  - make additional configopts in PETSc easyconfigs work after uncommenting (#8522)
  - add `pkg-config` build dep to easyconfig for pocl 1.2 (#8528)
  - download correct source tarball for Net-core 2.1.8 (#8530)
  - fix '`Permission denied`' error when running '`cp -a`' for ANTs 2.3.1 installation by first removing the `.git` subdirectories causing them (#8535)
  - fix checksum for boot 1.3-22 extension in R 3.6.0 easyconfig (#8537)

11.12.19 EasyBuild v3.9.2 (June 9th 2019)

bugfix/update release

framework

- various enhancements, including:
  - enhance (experimental) support for generating Singularity container recipes/images (#2884, #2900, #2902, #2903, #2907, #2909, #2910, #2913, #2915)
    - add support for specifying container configuration via `--container-config` (`--container-base` option has been removed)
    - add support for providing custom template for container recipe via `--container-template-recipe`
    - add support for '`sif`' Singularity 3.x container image format
    - add support for specifying non-image based Singularity bootstrap agents
    - install default set of OS packages in container recipes starting from scratch
    - configure Lmod and update Lmod spider cache in generated Singularity container recipes

    * configure EasyBuild via environment variables (to allow overriding configuration settings via options to '`eb`' command)
    * also consider `/tmp/easybuild/sources` in source path (to allow seeding in sources in container build environment)
    * updated documentation is available at https://easybuild.readthedocs.io/en/latest/Containers.html

- various bug fixes, including:
  - make sure that easyconfig file for EasyBuild version being bootstrapped is found in robot search path (#2899)
  - remove interpreter options when fixing script shebang (#2905, #2906)
  - fix crash on iterated installation when using Cray toolchains (#2914)
  - disable checking of read/exec permissions when checking for availability of '`sudo`' command (#2918)
- other changes:
  - stick to vsc-base < 2.9.0 in bootstrap script to avoid requiring '`future`' dependency (#2892)

easyblocks

- new software-specific easyblock for ELPA (#1621)
- minor enhancements, including:
  - allow Boost to build both Python 2 and Python 3 libraries (#1718)
  - update CPLEX sanity check: convert tool was removed in CPLEX 12.8 (#1737)
  - don't try to patch `CROSSTOOL` script for recent Bazel versions (since it's no longer there) (#1741)
- various bug fixes, including:
  - fix version check on using '`-log`' option to '`Allwmake`' in OpenFOAM easyblock (#1739)
  - fix GROMACS easyblock for list-type `configopts` (#1740)
  - fix configure step for GROMACS version < 4.6 (#1742)

easyconfigs

- added example easyconfig files for 17 new software packages:
  - adjustText (#8354), cowsay (#8380), fxtract (#8426), google-java-format (#8373), libtar (#8379), mkl-service (#8390), msprime (#8371), pygrib (#8395), pyhdf (#8394), pyproj (#8395), PyStan (#8410), Racon (#8358), rapidtide (#8256), SingleM (#8428), smafa (#8420), SVDetect (#8399), Unicycler (#8376)
- added additional easyconfigs for various supported software packages, including:
  - BLAST+ 2.9.0, Boost.Python 1.70.0, DIAMOND 0.9.24, EMAN2 2.3, ecCodes 2.12.5, GDAL 3.0.0, ImageMagick 7.0.8-46, Libint 2.5.0, matplotlib 2.2.4, NLopt 2.6.1, OrfM 0.7.1, PGI 19.4, PostgreSQL 11.3, R 3.6.0, R-bundle-Bioconductor 3.8, Rust 1.35.0, STAR 2.7.1a
- minor enhancements, including:
  - use CMake rather than configure script for libxc 4.3.4 (#8361, #8453)
  - add '`cobs`' extension to R 3.5.1 easyconfigs (#8389)
  - add sanity check command to PSI4 1.2.1 easyconfigs to ensure that '`import psi4`' works (#8393)
  - include the provided backports patch in QuantumESPRESSO-6.4.1 (#8405)
  - add `Logger::Simple`, `Scalar::Util::Numeric`, `YAML`, `Object::InsideOut` extensions to Perl 5.28.1 easyconfig (#8432)

  - update Java 1.8 to 1.8.0_212 (#8443)
- various bug fixes, including:
  - add missing build dependency on gettext to ATK/2.28.1 for fosscuda/2018b (#8402)
  - add patch to fix OpenBLAS v0.3.1 matrices multiplication issue (#8396)
  - make Eigen a build dependency for nanopolish (#8448)
- other changes:
  - clean up ELPA 2018.11.001 easyconfig to use custom easyblock for ELPA (#8360)
  - remove unused checksum for `ballgown` extension in Bioconductor 3.7 easyconfigs (#8363)

11.12.20 EasyBuild v3.9.1 (May 20th 2019)

bugfix/update release

framework

- various enhancements, including:
  - add function to determine whether specified easyblock is generic or not (#2838)
  - add support to `apply_regex_substitutions` function to skip creation of backup (#2844)
  - add support for `%(pymajver)s` (& co) (#2850)
  - add support for '`fix_python_shebang_for`' and '`fix_perl_shebang_for`' easyconfig parameters (#2878)
  - add `multi_deps` information to generated module, help and whatis sections (#2882)
  - add support for '`eb --missing`' ('`eb -M`') (#2883)
    - see also https://easybuild.readthedocs.io/en/latest/Using_the_EasyBuild_command_line.html#getting-an-overview-of-missing-installations-missing-m
- various bug fixes, including:
  - only call '`scontrol release`' when there's at least one job submitted (#2834)
  - fix small bug in `find_eb_script` + various minor issues with tests (related to environment in which tests are run) (#2835)
  - list build dependencies *before* runtime dependencies, so dependencies listed via `multi_deps` are loaded first in toolchain environment (#2839)
  - allow setting parallel to `0` or `False` to disable adding the `-j` argument (#2842)
  - be more careful when using single quotes for Tcl syntax (#2847)
  - append to existing `.modulerc` file rather than overwriting it (if `--force` is used) (#2848)
  - correctly resolve template values used for extensions (#2852)
  - update template values after updating iterative easyconfig parameters (#2854)
  - add top-level directories of hierarchical module naming scheme to `$MODULEPATH` before loading dependencies (#2857)
  - inject dependencies specified via `multi_deps` before normal build dependencies (#2861)
  - include Python version(s) in PR titles auto-generated by `--new-pr` (#2863, #2880)
  - update `HierarchicalMNS` for GCCcore toolchain (#2870)

  - bump iteration index counter at the correct time (#2886)
  - don't specify `--ntasks-per-node` when submitting Slurm jobs (#2887)
  - fix order of easyconfig parameters in output generated by '`eb --avail-easyconfig-params --output-format rst`' (#2888)
- other changes:
  - lower required version in Slurm job backend to 16.05 (#2833)
  - add Lmod 8 to test suite (#2849)
  - deprecate useless 'skip_lower' named argument in template_constant_dict, always define `*lower` templates (#2856)
  - fix Python classifiers in `setup.py`, should be (only) Python 2.6 & 2.7 (#2876)

easyblocks

- one new generic easyblock: `CMakeMakeCp` (#1725)
- new software-specific easyblock for Blender (#1595), Lua (#1396), OpenBLAS (#1693, #1697)
  - OpenBLAS easyblock also supports installation on POWER systems
- minor enhancements, including:
  - enhance VMD easyblock to also build Surf & Stride + pick up netCDF for VMD >= 1.9.3 (#1314)
  - update CGAL easyblock to support CGAL 4.12 & newer (#1679)
  - update QuantumESPRESSO easyblock for v6.4.1 (#1692)
  - enhance Advisor & VTune easyblocks, since no license is required starting with version 2019 update 3 (#1694, #1695)
  - handle new multi-deps Python setup with `$EBPYTHONPREFIXES` for Tensorflow (#1702)
  - make CHARMM easyblock pick up on `prebuildopts` and `runtest` (#1707)
  - change default value for '`use_pip`' to None, so we can discriminate from '`use_pip`' being set to False explicitly (#1709)
  - make PythonPackage aware of `(pre)testopts` (#1715)
  - enable GPU offloading in Clang if CUDA is included as a dep (#1716)
  - add support to Clang easyblock for also building libc++ and RTTI (#1717)
  - rebuild internal libfabric for impi 2019.x & more recent (#1730)
  - update Q&A patterns in SAS easyblock for version 9.4 + add support for custom license file path (#1731)
- various bug fixes, including:
  - only install mkl-dnn by default with TensorFlow on x86-64 systems (#1666)
  - make sure `ldconfig` can be found before using it in CUDA easyblock (#1684)
  - use temporary directory for pip cache (rather than `$HOME/.cache/pip`) in `PythonPackage` easyblock (#1687)
  - allow oversubscription when testing FFTW on top of OpenMPI >= 3.0 (#1688)
  - make sure `$HOME/.cache/pip` isn't used while installing Tkinter (#1691)
  - set default `exts_filter` in `PythonBundle`, so already installed extensions in a bundle can be skipped (#1701)

  - instruct Bazel to not use `$HOME/.cache/bazel` in TensorFlow easyblock (#1704)
  - also set `$XDG_CACHE_HOME` during installation of Python, to ensure that '`pip`' that comes along with it doesn't touch `$HOME/.cache/pip` (#1705)
  - use `-fp-model precise` if FMA instructions are missing in GROMACS easyblock (#1706)
  - make sure Perl install dir follows a standard format (#1708)
  - attach `source_urls` directly to sources for components in generic `Bundle` easyblock (#1711)
  - avoid that `--no-build-isolation` option is specified multiple times in `PythonPackage` easyblock (#1713)
  - fix prefix in impi compiler wrappers (#1727)
  - only embed zlib in binutils if it is listed a build dependency (#1732)
- other changes:
  - fix Python classifiers in `setup.py`, should be (only) Python 2.6 & 2.7 (#1724)

easyconfigs

- added easyconfigs for new toolchains:
  - `fosscuda/2019a` (#8063), `gimkl/2018b` (#8287), `gomkl/2018b` (#8216), `gomkl/2019a` (#8218), intelcuda/2019a (#8069)
- added example easyconfig files for 52 new software packages:
  - ADDA (#8207), AMD-LibM (#7164), AMD-RNG (#7165), AMD-SecureRNG (#7165), ARGoS (#8039, #8104), ARWEN (#8213), Canvas (#7716), cdsapi (#7970), charmm (#8202), emcee (#7989), FlexiDot (#8228, #8275), FreeImage (#8039), Hello (#7704), HLAminer (8094), hypothesis (#8307), imgaug (#8229), INTEGRATE (#8193, #8304), INTEGRATE-Neo (#8094), IRkernel (#8050, #8099), JiTCODE (#7148, #8327), libFLAME (#7163), libpsml (#5859), LibSoup (#8116), libutempter (#6426), LS-PrePost (#8070), LUSCUS (#7191, #8105, #8285), memory-profiler (#8255), metaWRAP (#7896), Net-core (#7716), netMHC (#8094), Nextflow (#8195), nvtop (#8024), openpyxl (#8121), py-cpuinfo (#8245), pyFFTW (#8198), PyQtGraph (#7525, #8253), R-tesseract (#7933), RBFOpt (#8178), rnaQUAST (#8040), RStan (#7996), scikit-multilearn (#8142), simpy (#8177, #8250), SMARTdenovo (#7630), socat (#8305), SymEngine (#7148, #8327), unixODBC (#8074), VAtools (#7938), VirtualGL (#8008), WebKitGTK+ (#8118, #8241), xmlf90 (#5858), YAPS (#7976), zingeR (#7264)
- added additional easyconfigs for various supported software packages, including:
  - Boost 1.70.0, cairo 1.16.0, CGAL 4.14, Clang 8.0, cutadapt 2.1, dask 1.1.4, ELPA 2018.11.001, FFmpeg 4.1.3, GCC 9.1.0, GEOS 3.7.2, GLib 2.60.1, gmsh 4.2.2, GROMACS 2019.2, GTK+ 3.24.8, h5py 2.9.0, hwloc 1.11.12, Hypre 2.15.1, Mathematica 12.0.0, matplotlib 3.0.3, Mesa 19.0.1, NBO 7.0, NCL 6.6.2, NCO 4.7.9, NiBabel 2.4.0, numba 0.43.1, OpenMPI 3.1.4, OrthoFinder 2.3.3, PCMSolver 1.2.3, PETSc 3.11.1, PROJ 6.0.0, PyQt5 5.12.1, PyTorch 1.0.1, PyYAML 5.1, Qt5 5.12.3, QuantumESPRESSO 6.4.1, R 3.5.1 (w/ intel/2018b), RNAIndel 1.0.0, Ruby 2.6.3, scikit-learn 0.20.3, SLEPc 3.11.0, sympy 1.4, Tkinter 3.7.2, Vim 8.1.1209, VTK 8.2.0, wrf-python 1.3.1, wxPython 4.0.4, wxWidgets 3.0.4, xarray 0.12.1, zstd 1.4.0
- minor enhancements, including:
  - enable auto-download of VMD 1.9.3 + add patches for Surf and Stride (#7305)
  - add mlegp extension in R 3.5.1 easyconfigs (#7814)
  - add pkg-config file to bzip2 easyconfigs (#8200)
  - allow use of '`use_pip = False`' in easyconfigs if pip doesn't work (#8220)
- various bug fixes, including:

  - fix checksums for nlme extensions in R easyconfigs (#7814, #8054)
  - add missing XZ dependency for Pysam > 0.12 (#7971)
  - define `$GRACE_HOME` in Grace easyconfigs, so that font dir can be located (#8048)
  - `$XDG_DATA_DIRS` must be set for GTK+ (#8089)
  - add missing FriBidi dependency for Pango 1.43.0 (#8103)
  - add (back) custom `sanity_check_paths` in recent Pango easyconfigs (#8106)
  - fix missing extensions in cutadapt 1.16 easyconfigs (#8130)
  - add missing cURL dependency for recent SAMtools versions (#8131)
  - add singledispatch extension to Python 2.7.15 easyconfig using `GCCcore/8.2.0` toolchain (#8164)
  - add missing X11 dependency for Gdk-Pixbuf 2.38.1 (#8222)
  - make sure `hdf5r` picks up HDF5 dependency in R 3.5.1 easyconfigs (#8223)
  - enable zstd compression in GRASS 7.6.0 easyconfig (#8224)
  - add missing ICU dependency on ICU for Harfbuzz 2.4.0 (#8226)
  - disable AVX512 DGEMM kernels in OpenBLAS 0.3.5 (#8227)
  - fix homepage/description in OrthoFinder easyconfig (#8234)
  - add `Parallel::ForkManager` extension to Perl 5.28.x easyconfigs (#8247)
  - replace LibUUID dependencies with util-linux (#8258)
  - add jemalloc & pkg-config as build deps for Salmon 0.12.0 (#8264)
  - fix MAJIQ easyconfig by fixing order of extensions + avoid numpy test hang (#8272)
  - fix shebang in GLib Python script + clarify runtime dependency on Python (#8277)
  - add pkg-config and expat as (build) dependency for DBus (#8283)
  - define `$GI_TYPELIB_PATH` in GTK+ and Pango easyconfigs (#8246, #8286)
  - add pkg-config build dep to PROJ 6.0.0 easyconfig (#8309)
  - fix source URLS in recent libcerf easyconfigs (#8332, #8243)
  - make zlib a real dependency rather than a build dep in recent binutils easyconfigs (>= 2.28) (#8340)
  - add fix-ib-query patch to OpenMPI 2.1.x and 3.0.x easyconfigs (#8341)
  - set `$XDG_CACHE_HOME` to $TMPDIR before '`pip install`' in Arrow 0.12.0 easyconfigs (#8347)
- other changes:
  - remove xbitmaps dependency from motif (#7530)
  - require custom `sanity_check_paths` in easyconfigs touched in PRs when generic easyblock is used (#8101, #8123)
  - use CMake built with GCCcore toolchain when installing Eigen 3.3.4+ (#8261)
  - fix Python classifiers in `setup.py`, should be (only) Python 2.6 & 2.7 (#8299)
  - use custom easyblock for OpenBLAS in OpenBLAS 0.3.x easyconfigs (#8345, #8339)

11.12.21 EasyBuild v3.9.0 (April 12th 2019)

feature release

framework

- various enhancements, including:
  - add `(pre)testopts` easyconfig parameters (#2793)
  - add support for `multi_deps` easyconfig parameter (#2741, #2810, #2811, #2812, #2813, #2825, #2826, #2827, #2828)
- various bug fixes, including:
  - fix argument name in `close_pr` (#2752)
  - fix undefined variable 'prefix' in `Compiler._set_compiler_vars` (#2757)
  - fix test account for GitHub-related tests (#2760)
  - make sure read permissions are always set in `permissions_step` (#2770)
  - make test for `dep_graph` function robust against changing order of lines in resulting dot file (#2772)
  - fix problem with `import_available_modules` when running from easybuild-framework directory (#2786)
  - prepend location of test easyblocks to `$PYTHONPATH` to `test_generate_software_list`, rather than append (#2788)
  - use pid in backup name candidate to reduce risk of two processes colliding (#2796)
  - automatically enable `--ignore-osdeps` when using `--preview-pr` (#2799)
  - use temporary checkout of repository as robot path for `--new-pr` and `--update-pr` to determine locations for patch files (#2803)
  - avoid relying on order in which keys are processed in `to_dependency` (#2804)
  - fix yeb format to work with PyYAML 5 (#2809)
  - fix typo in error message for use of unknown easyconfig parameter (#2817)
  - enhance `log_file_format` to fix problem when `%(name)s` template value is used for log directory (#2818)
  - iterate over subdirectories in order in `find_egg_dir_for` of bootstrap script, to ensure oldest vsc-install is picked (#2819)
  - replace `-Wl,--enable-new-dtags` compiler option with `-Wl,--disable-new-dtags` in RPATH wrapper script (#2821)
  - updated `COMPILER_F77` for PGI >= 19.1 (#2823)
  - added `FFTW_STATIC_LIBS_MT` to `FFTW_MAP_CLASS`, otherwise `FFTW_STATIC_LIBS_MT` are incorrectly defined (#2822)
  - sort available version list when selecting/generating easyconfig (#2732)
- other changes:
  - use `assertEqual` rather than `assertEquals` (which is deprecated) (#2746)
  - test with Tmod 4.1.4 (packaged in RHEL8) in Travis CI (#2747)
  - replace deps by hidden deps instead of removing them from the lists (#2748)

  - ensure non-zero exit code for all test subsuites (+ fix trivial style issues) (#2773)
  - silence printed deprecation warnings for easyconfigs/toolchains while running (easyconfigs) test suite (#2781)
  - avoid printing of messages/warnings in framework test suite (#2782)
  - enable silent mode for `--new-pr` and `--update-pr` (#2802)
  - update Travis config to reflect that PyYAML 5.x is no longer compatible with Python 2.6 (#2807)

easyblocks

- new software-specific easyblock for TensorRT (#1627)
- minor enhancements, including:
  - make sure CMake doesn't pick up on system Boost in CMakeMake generic easyblock (#1618)
  - modified `ConfigureMake` and `CMakeMake` generic easyblocks to allow changing `configure_cmd`, `build_cmd` and `install_cmd` to fit various needs (#1628, #1658)
  - update ANSYS easyblock to deal with changed versioning scheme (#1631)
  - add extra path to builddir for SNPhylo >= 20160204 (#1632)
  - add better support for TensorRT to TensorFlow (#1634)
  - add support for newer versions of wxPython to wxPython easyblock (#1635)
  - update OCaml easyblock to support OPAM 2.x (#1638)
  - update sanity check in TensorFlow easyblock for TensorFlow 1.13.x (#1642)
  - make `ConfigureMake`, `MesonNinja` and `SCons` generic easyblocks aware of pretestopts (#1645)
  - add support for installing Python packages with '`pip`' without using `--ignore-installed` (#1647)
  - add support to `CMakeMake` easyblock to specify compilers using absolute file path (#1652, #1655)
  - stop checking for deprecated `pgf77` in sanity check of PGI easyblock (#1653)
  - accommodate version 10.1 in CUDA easyblock (#1654)
  - install `sitecustomize.py` with Python to support Python package installations for multiple Python versions in a single directory (#1657)
  - enhance SAMtools easyblock to also install `libhts.a` and include/htslib (when applicable) (#1661)
  - tweak `PythonPackage` & `PythonBundle` to update `$EBPYTHONPREFIXES` rather than `$PYTHONPATH` for multi-Python installations + make `sanity_check_step` aware of multi_deps (#1664, #1678)
  - replace `-openmp` with `-fopenmp` when building TINKER with Intel compilers (+ modernize easyblock) (#1668)
  - make sure Meson is configured with `-Dlibdir=lib` so it doesn't install libraries in `lib/x86_64-linux-gnu` on Debian/Ubuntu multiarch systems (#1673)
  - allow for external libfabric via `ofi_libfabric` parameter in impi easyblock (#1676)
  - pass down compiler names and options to Qt5 5.8 & more recent via corresponding `QMAKE_*` configuration options (#1677)
- various bug fixes, including:
  - create symlinks to `.modulerc` in symlinked module directories (#1636)
  - add missing import in FFTW easyblock (#1641)

  - fix finding of Python 3 include subdir + `libpython*.so` in OpenBabel easyblock (#1633)
  - add `no_qa` pattern for interactive installation command for ABAQUS (#1637)
  - make serial, smpar and dm+sm builds of WRF work and pass tests (#1646)
  - configure CP2K with correct `DATA_DIR` value rather than relying on `$CP2K_DATA_DIR` environment variable (#1650)
  - CP2K fixes: only call configure_LAPACK/ScaLAPACK when imkl is not used, prefer using 2^2 MPI processes for tests (#1651)
  - also take into account custom easyconfig parameters from `CMakeMake` in BamTools easyblock (#1656)
  - make sure Perl scripts are installed in the right location (#1660)
  - make sure setenv is used for CUDA_PATH etc. (#1663)
  - cleanup `sanity_check_paths` in numpy & scipy easyblocks (#1665, #1667)
  - always set `self.pylibdir` in `PythonPackage.set_pylibdirs()`, which is important when installing Python packages for multiple Python versions via `multi_deps` (#1671)
  - fix custom definition of `load_module` method in LAPACK module (#1675)
- other changes:
  - replace '`except OSError, err:`' pattern with '`except OSError as err:`' (for compatibility with Python 3.x) (#1639)
  - clean up Java easyblock to use `filetools` functionality (#1649)

easyconfigs

- added easyconfigs for new toolchains: `intel/2019.02` (#7598), `intel/2019.03` (#7846)
- added example easyconfig files for 68 new software packages:
  - ACT (#7928), aiohttp (#7728), at-spi2-atk and at-spi2-core (#7658), Bader (#7804), barrnap (#7738), BCEL (#7937), biscuit (#5868), bitarray (#7772), BlobTools (#7565, #7583), bmtagger (#7890), bsddb3 (#7642), CheckM (#7712), Cheetah (#7952), CONCOCT (#7891), cyvcf2 (#8031), DAS_Tool (#7741), ExaBayes (#7801), FastANI (#7992), fastp (#7693), Flask (#7734), giflib (#7663), Giza (#7843), glew (#7685), gpustat (#8025), GRASS (#7489), GTDB-Tk (#7995), HPCX (#7725), IntelPython (#7920), KNIME (#7554), KronaTools (#7721), KyotoCabinet (#7955), Leptonica (#7932), libepoxy (#7655), libpsl (#7666), MAGMA (#7829), MATLAB-Engine (#7758), MaxBin (#7767), MetaBAT (#7746, #7931), MinPath (#7763), ncdu (#7505), NGSadmix (#7524), NIMBLE (#7564), PCAngsd (#7727), pizzly (#7724), Ploticus (#7545), pocl (#7681), POT (#8011), ProjectQ (#7576), pullseq (#7740), pyBigWig (#7600), Pyke3 (#8034), PyRETIS (#8041), RDKit (#7973), RNAIndel (#8009), scikit-optimize (#7613), SciPy-bundle (#7922), sep (#8032), slidingwindow (#7909), SPLASH (#7843), SqueezeMeta (#7771), SRPRISM (#7890), taxator-tk (#7894), TensorRT (#7584), tesseract (#7932), Transrate (#5108), VCF-kit (#7786, #7882), VV (#7297)
- added additional easyconfigs for various supported software packages, including:
  - AFNI 19.0.01, Arrow 0.7.1, BLAST+ 2.8.1, CUDA 10.1.105, GCC(core) 8.3.0, GTK+ 3.22.30, Java (OpenJDK) 11(.0.2), Meson 0.50.0, MultiQC 1.7, Nim 0.19.2, Ninja 1.9.0, netCDF 4.6.2, netCDF-Fortran 4.4.5, PCRE 8.43, Perl 5.28.1, PGI 19.1, Python 3.7.2, RELION 3.0.4, Ruby 2.6.1, SCons 3.0.4, SQLite 3.27.2, SuiteSparse 5.4.0, TINKER 8.6.1, TensorFlow 1.13.1, X11 20190311
- minor enhancements, including:
  - add various extensions to R 3.5.1 easyconfigs: `asnipe` (#7572), `liquidSVM` (#7597), `oddsratio`/`mltools`/`h2o` (#7744), `mlegp` (#7814), `bartMachine`/`lqa` (#7865), `PresenceAbsence`/`GUTS`/`GenSA` (#7905), `parsedate` (#7935), `circular` (#7975)

  - add `ujson` extension in recent Python easyconfigs (#7517)
  - run various checks on easyconfigs that are touched in pull requests and involve Python packages (#7754)
  - add `cpanminus` extension to recent Perl easyconfigs (#7866)
  - also install ANTs scripts (and set `$ANTSPATH` as required by those scripts) (#7940)
  - add missing `configopts` in GATE 8.1 easyconfig to enable Davis feature (#8000)
- various bug fixes, including:
  - add patch for Mesa 18.1.1 to detect MIT-SHM (#7536)
  - add proper description to MINC-2.4.03 (#7551)
  - add libunwind dependency to recent Mesa easyconfig when building with foss/GCC (#7629)
  - add/reorder missing/misplaced extensions in Python 2.7.15 and 3.6.6 easyconfigs (#7696)
  - fix Jellyfish dependency in easyconfig for Kraken 1.0 (Jellyfish 1.x is required) (#7743)
  - use `https://` in most recent XZ easyconfigs (#7782)
  - add patch for OpenMPI 3.1.x to fix ib-query '`Invalid argument`' error (#7789)
  - build OpenBLAS with `-fno-tree-vectorize` (asm constraint bugs for <0.3.6) + cleanup & SHA256 checksums (#7790, #7793)
  - extra patch for TensorFlow 1.12.0 to remove `-B/usr/bin` from `linker_bin_path_flag` in `cuda_configure.bzl` (#7800)
  - fix easyconfig for STAR-Fusion 1.5.0 (#7802)
  - fix checksums for boot/nlme extensions in R easyconfigs (#7814, #8054)
  - add patch for OpenMPI 3.x to fix UCX memory leak (#7535, #7824)
  - replace `ncurses-devel` OS dependency in CMake easyconfigs using `dummy` toolchain with proper ncurses dependency (#7834)
  - use `PythonBundle` for `snakemake-5.2.4-foss-2018b-Python-3.6.6.eb` (+ fix module-class) (#7842)
  - use correct `buildopts` + add missing zlib dependency in StringTie 1.3.5 easyconfig (#7845)
  - update GStreamer to not enable dw and fix some missing dependencies (#7889)
  - add missing XZ dependency to most recent Pysam easyconfigs (#7897)
  - expat: add configure option `--without-docbook` to avoid docbook2X dependency (#7930)
  - fix source URLs for mawk (#7960)
  - fix LWM2, OTF2, OPARI2, and Score-P download URLs (#7994)
  - use `https://` in homepage & source_urls for OpenMPI and hwloc easyconfigs (#8013, #8014, #8015 and #8016)
  - add missing bokeh dependency for dask 1.0.0 (+ add dask-jobqueue) (#8029)
  - fix checking of binutils build dep in easyconfig tests (#8038)
- other changes:
  - avoid use of `.items()` in R (bundle) easyconfigs, to fix compatibility with EasyBuild running on top of Python 3 (#7791)
  - trim down test configuration: only test with Lmod 6.x with Tcl/Lua on Python 2.6/2.7 (#7795, #7798)

- use `%(pyshortver)s` template in (old) SIP easyconfigs (#7797)
- add PyTorch to whitelist for not having '`use_pip`' enabled (#7844)
- don't use local variable '`pylibdir`' in list comprehension in PyQt easyconfig, since that doesn't work in Python 3 (#7848)
- use pip instead of setup.py with h5py/2.7.1 and 2017b toolchains (#7864)
- prefer `https://` over `ftp://` for source_urls in recent GROMACS easyconfigs (#7948)
- rename arrow to Arrow for old easyconfig (#8007)

11.12.22 EasyBuild v3.8.1 (January 29th 2019)

bugfix/update release

framework

- various minor enhancements, including:
  - speed up checking of OS dependencies (#2703)
  - add support for '`eb --show-system-info`' (#2722)
  - add support for '`%(arch)s`' template value in easyconfig files (x86-64, aarch64, ppc64le, ...) (#2728)
- various bug fixes, including:
  - add timestamp to reprod dir while in tmp space (#2705)
  - avoid '+' in directory name for `g++` rpath wrapper (#2710)
  - pre-install vsc-install < 0.11.4 in bootstrap script to avoid requiring '`mock`' Python package (#2717)
  - fix check in `--merge-pr` whether PR is eligible for merging in to only consider the status of the last test report (#2720)
  - don't recreate build directory when '`buildininstalldir`' is enabled for iterative installations (#2724)
  - filter (CUDA) `lib*/stubs` paths in RPATH wrapper script (#2725)
- other changes:
  - replace '`except IOError, err:`' pattern with '`except IOError as err:`' for compatibility with Python 3.x (#2711)
  - make all `print` statements compatible with Python 3 (#2715)
  - avoid use of `sys.maxint` in dependencies_for (#2716)

easyblocks

- minor enhancements, including:
  - update Ferret easyblock to handle Ferret 7.3 (#1349)
  - add support for defining `$LDSHARED` when installing Python packages if Python's value doesn't use toolchain compiler ($CC) (#1455)
  - update CP2K easyblock to add support for CP2K 6.1, and fix incorrect LibInt references (#1545)
  - make OpenFOAM easyblock handle debug build (#1609)
  - handle multiple installation keys for MATLAB (#1610)
  - run '`make check`' in parallel for GROMACS since it involves more compilation (#1611)

  - update OpenCV easyblock for recent versions (3.4.x >= 3.4.4 + 4.0.x) (#1616)
- various bug fixes, including:
  - check current `start_dir` value before appending '`src`' subdirectory in MrBayes easyblock (#1582)
  - avoid that (system) Intel compilers are always considered when building SuiteSparse (#1612)
  - fix missing import statement in ROOT easyblock (#1614)
  - answer SELinux question with '`no`' in Mathematica easyblock (#1617)
  - disable "build isolation" feature in pip > 10.x in PythonPackage generic easyblock (#1623)
- other changes
  - cleanup in Perl and PerlModule easyblocks (#1603)

easyconfigs

- added easyconfigs for new toolchains `foss/2019a` (#7371), `intel/2019a` (#7372) and `iomkl/2019.01` (#7375)
  - see also https://easybuild.readthedocs.io/en/latest/Common-toolchains.html
- added example easyconfig files for 43 new software packages:
  - Assimulo (#6740), C3D (#7059), CellMix (#7422), CESM-deps (#6654, #6675, #6823), DSA (#7422), DeMixT (#7422), deconf (#7422), DeconICA (#7422), double-conversion (#7307), dxpy (#7079), EPIC (#7422), expect (#7387), Flye (#7430), FMILibrary (#6740), FMRIprep (#7059), FUSE (#7078), GDCM (#7310), HDDM (#7396), ICA-AROMA (#7059), IntaRNA (#7334), imageio (#6738), KWIML (#7308), kWIP (#7444), MagresPython (#7395), medaka (#7426), MINC (#7311), MuSiC (#7422), netMHCIIpan (#7377), NIfTI (#7311), OPERA (#7408), psrecord (#7331), pyfits (#7273), PyFMI (#6740), PyGWAS (#5852), Pylint (#6675), QDD (#7284), RERconverge (#7289), samclip (#7226), SCIPhI (#7419), swarm (#7453), VXL (#7309), wrf-python (#6736), xCell (#7422)
- added additional easyconfigs for various supported software packages, including:
  - Amber 18, Arrow 0.12.0, Biopython 1.73, Boost 1.69.0, CP2K 6.1, canu 1.8, Ferret 7.3, GATE 8.1.p01, GROMACS 2019, Geant4 10.5, HPL 2.3, libxsmm 1.10, matplotlib 3.0.2, NEURON 7.6.5, Octave 4.4.1, OpenBLAS 0.3.5, OpenCV 3.4.5 + 4.0.1, OpenFOAM v1812, PGI 18.10, PLUMED 2.5.0, ROOT 6.14.06, Theano 1.0.3, TopHat 2.1.2, Yade 2018.02b
- minor enhancements, including:
  - add additional extensions to R 3.5.1 easyconfigs:
    - statnet (#7370), NMF, ComICS, dtangle, MCMCpack, shinythemes (#7420), csSAM (#7423), bridgedist (#7477)
  - add DeconRNASeq and GSVA extensions to R-bundle-Bioconductor 3.7 easyconfigs (#7421)
  - also install header files and CMake module for SeqAn 2.4.0 built with `foss/2018b` (#7434)
  - add download URL for `intel/2018a` components (#7436)
  - add missing checksums for protobuf easyconfigs using dummy toolchain (#7492)
- various bug fixes, including:
  - fix source URL in Doxygen easyconfigs (#7324)
  - add fallback source URL for recent pigz versions (#7346)
  - promote FriBiDi to runtime dep of Pango rather than only build dep (#7369)
  - add `--without-systemdsystemunitdir` configure option to recent DBus easyconfigs (#7373)

  - fix issue of building M4 1.4.18 with glibc 2.28 (#7384)
  - fix broken source URLs in MariaDB easyconfigs (#7413)
  - bump up build dependency of flex 2.6.4 from Bison 3.0.4 to 3.0.5 (#7414)
  - disable building of OpenColorIO Python bindings since Python is not included as a dependency (#7416)
  - add missing build dependencies in old Harfbuzz/Pango easyconfigs (2016a generation) (#7433)
  - fix XML-LibXML linking with Intel compiler and GCCcore Perl (#7440)
  - prevent non-critical error from stopping Guile v1.8.8 build (#7446)
  - use `%(arch)s` template rather than hardcoding '`x86_64`' in XML-Parser easyconfigs (#7450)
  - use uniform configopts for Guile 1.8.8 (#7452)
  - fix configopts in PyQt5 easyconfig files to avoid installation of files in Python/Qt5 installation directories (#7470)
- other changes:
  - fix homepage in easyconfigs for `foss` toolchains (#7482) and `intel` toolchains (#7483)

11.12.23 EasyBuild v3.8.0 (December 18th 2018)

feature release

framework

- various enhancements, including:
  - support use of version ranges in `--filter-deps` (#2357)
  - add support for `--list-prs` (#2400, #2668)
  - add support for `--close-pr` (#2401)
  - allow setting `optarch` compiler flags in the easyconfig via `toolchainopts` easyconfig parameter (#2595)
  - add option to skip pre-creation of install directory: `--disable-pre-create-installdir` (#2629, #2637)
  - flesh out setting up of configuration into dedicated '`set_up_configuration`' function (#2638)
  - clean error when '`eb`' is cancelled by user (#2641)
  - add support for using Slurm as backend for `--job`, via `--job-backend=Slurm` (#2642, #2666)
  - save easyblocks along with easyconfig in '`reprod`' subdirectory of install directory (#2653)
  - add support for deprecating easyconfig files & toolchains (#2656)
  - changed `EasyConfig.update` to be able to specify not to allow duplicate values (#2657)
  - move adding of dependencies to prepare step instead of check readiness step (#2674)
  - add support for disabling mapping of (sub)toolchains when `--try-toolchain` is used, via `--disable-map-toolchains` (#2682)
  - add '`astro`' and '`quantum`' module classes (#2693)
- various bug fixes, including:
  - make `--from-pr` always try to apply PR patch on top of PR target branch (#2631)

  - fix bug that could cause to silently overwrite an existing easyconfig when using `--try-*` (#2635)
  - don't pass down specific environment variables into submitted jobs (#2643)
  - update template constants for source URLs to use https (#2648)
  - fix lib64 fallback for '`lib`'/'`lib64`' dirs entry in `sanity_check_paths` (#2649)
  - change PGI F90 Fortran compiler to pgf90 (#2650)
  - use `--set-upstream-to` in `install-EasyBuild-develop.sh` script to deal with deprecated/no longer supported `--set-upstream` (#2651)
  - fix order of keys in '`toolchain`' value for dumped easyconfig file (name, version) + run style check on dumped easyconfigs in dump tests (#2660)
  - ensure checksums beside sources in dumped easyconfigs (#2661)
  - fix problems with easyconfig file saved in '`reprod`' directory by copying it before running any installation steps (#2664)
  - make print functions more robust w.r.t. arguments being passed to format the message to be printed (#2670)
  - use '`git am`' to apply patch for PRs in `fetch_easyconfigs_from_pr` (#2680)
  - correctly define `$LIBFFT_MT` for Intel MKL (#2688)
  - don't always require easyconfig files to resolve dependencies (#2690, #2692, #2697, #2698, #2699)
- fixes for Travis CI config:
  - stop testing with Lmod 6.6.3, testing with Lmod 6.5.1 is sufficient (#2627)
  - Python 2.6 requires python-daemon 1.x as dep for GC3Pie (#2673)
  - stick to idna<2.8 with Python 2.6 in Travis + distable broken test for HgRepository (#2678)
- other changes:
  - always skip symlinks in `adjust_permissions` (#2644)
  - flesh out '`avail_easyblocks`' function from support for `--list-easyblocks` (#2663)
  - deprecate the `ictce` toolchain (#2667)
  - deprecate `goolf` and `goolfc` toolchains (#2676)
  - deprecate `intel` toolchains older than `intel/2016a` (+ `iccifort` & `iimpi` subtoolchains) (#2677)
  - deprecate ancient `gompi` toolchain versions (#2684)
  - use '`tar xzf`' (gzip) rather than '`tar xZf`' (ancient `compress`) to unpack `*.tar.Z` source files (#2686)
  - flesh out '`set_parallel`' method to it can be called separately (#2687)

easyblocks

- new generic easyblocks:
  - `PythonBundle` for installing a bundle of Python packages (#1553)
  - `MesonNinja` for installing using Meson & Ninja (#1561)
- new software specific easyblock for `RepeatMasker` (#1600)
- minor enhancements, including:
  - add `-fno-delete-null-pointer-checks` compiler flag for OpenFOAM versions older than v3.0 (#1311)

  - add support for building Tau with OTF included as dependency (#1313)
  - add support for Intel MPI version 2019 (#1546)
  - also populate the include dir for CP2K (#1554)
  - add custom easyconfig parameter in GCC easyblock to control use of gold linker: `use_gold_linker` (#1555)
  - replace '`-`' with '`_`' in default '`import`' check for Python packages (#1560)
  - update WRF and WPS to support version 4 (#1563)
  - adapt PSI easyblock for PSI4 > 1.2 (#1568)
  - update Siesta easyblock to 4.1-b4 and add custom test_step (#1573)
  - enhance Mothur easyblock to support use of Boost and HDF5 as dependencies (#1576)
  - enhance sanity check for icc & ifort: also check for `compilers_and_libraries_*/linux` subdirectory (#1577)
  - update Trinity easyblock for latest version 2.8 (#1579)
  - add version check in sanity check step of SCOTCH easyblock (+ code cleanup) (#1580)
  - add support in Clang easyblock to skip running of all tests (#1584)
  - update WRF easyblock to allow serial HDF5 + pick up on parallel netCDF (#1592)
  - avoid hardcoding '`PREFIX=<installdir>`' in build/install options when using `SCons` easyblock (#1594)
  - enable TensorFlow to detect any MPI runtime (#1597)
  - enhance `Bundle` easyblock to allow installation of bundle components with additional easyblocks + build components in parallel (#1598)
  - enhance Trilinos easyblock to support building against MKL (#1601)
- various bug fixes, including:
  - make FDTD_Solutions easyblock do the install by copying files instead of '`rpm rebuild`' (#1307)
  - re-add missing `VT_(S)LIB_DIR` env variable to itac module (#1309)
  - fix in QuantumESPRESSO easyblock: ifort compiler needs `-assume byterecl` (#1556)
  - extract targets from buildopts in Quantumespresso easyblock (#1558)
  - fixes for TensorFlow easyblock (#1559)
    - pass `$PYTHONPATH` while building TensorFlow
    - disable cross-compilation mode if optarch is set
    - fix sanity check for installing TensorFlow as extension
  - added a test to disable compiling FFTW with MPI if the toolchain does not support MPI (#1562)
  - fix permissions problem with CUDA nvvp tar files + correctly handle numactl symlink in LLVM subdir in PGI easyblock (#1569)
  - fix TensorFlow test tempdir problem (#1572)
  - tweak VMD easyblock so that `configopts` don't contain duplicate values (otherwise it fails to rebuild) (#1575)
  - fix running netcdf4-python tests to also support installation as extension (#1578)

  - added the openmp flags in the linker flags in MUMPS easyblock (#1585)
  - avoid hardcoding `--with-rdma` configure option in MVAPICH2 easyblock (#1586)
  - fix location of path-to-source argument in configure command of `CMakeMake` (#1591)
  - fix GROMACS use of MKL for non-Intel compilers (e.g. gomkl toolchain) (#1596)
- other changes
  - inform Hound CI about Python 2 builtins (#1604)

easyconfigs

- added easyconfigs for new toolchains: `intel/2018.04` (#7171), `intel/2019.00` (#7218), `intel/2019.01` (#7219)
- added example easyconfig files for 36 new software packages:
  - 4ti2 (#7040), ARPACK++ (#6750), CellRanger (#7242), CharLS (#6762) cscope (#7057), ctags (#7057), datamash (#6693), dcm2niix (#6762), DCMTK (#6761), deepdiff (#7109), Drake (#7182), fast5 (#7250), gap (#7040), Gerris (#7211), GPAW-setups (#6984), iCount (#7080), ipyparallel (#6797), Kratos (#7149), LCov (#7160), libvdwxc (#6984), libwebp (#7065), lrslib (#7040), MuPeXI (#6991), netMHCpan (#6991), Normaliz (#7040), OpenColorIO (#7146), OpenJPEG (#7216), PHLAT (#7036), poppler (#7069), RECON (#7042), RepeatMasker (#7281), TreeMix (#7133), unrar (#7117), utf8proc (#7083), VSEARCH (#7153), XMDS2 (#7121)
- added additional easyconfigs for various supported software packages, including:
  - CheMPS2 1.8.8, Clang 6.0.1 + 7.0.0, dask 1.0.0, FFmpeg 4.1, GCC(core) 6.5.0 + 7.4.0, GPAW 1.4.0, HMMER 3.2.1, IPython 7.2.0, ITK 4.13.1, Keras 2.2.4, LLVM 7.0.0, Mothur 1.41.0, MultiQC 1.6, NAMD 2.13, OpenBLAS 0.3.4, OpenMPI 3.1.3 + 4.0.0, PSI4 1.2.1, Qt5 5.11.2, QuantumESPRESSO 6.3, Rust 1.30.1, Spark 2.4.0, Spyder 3.3.1, TensorFlow 1.11.0 + 1.12.0
- minor enhancements, including:
  - install misc tools for angsd (#6995)
  - add libXp component to X11 bundle for GCCcore 6.4.0 (#7062)
  - also install shared libraries for LZO (#7073)
  - add extensions to R 3.5.1 easyconfigs: `gllvm` (#7123), `grpreg` (#7140), `gamlss` + `gamlss.tr` (#7263)
  - install Tk private headers (#7155)
  - add `File::Next` extension to Perl 5.28.0 easyconfig (#7276)
- various bug fixes, including:
  - fix installation of scikit-image 0.13.x by including missing required `PyWavelets` extension (#7061)
  - add missing dependencies to easyconfig for GC3Pie 2.5.0 (#7066)
  - stop setting ignored '`full_sanity_check`' easyconfig parameter in easyconfigs (#7094)
  - add missing checksums in X11/2016* easyconfigs (#7095)
  - add missing checksums in X11/2017* easyconfigs (#7096)
  - add missing `cffi` & `pycparser` extensions to recent Python easyconfigs (#7105, #7118)
  - fix homepage in QuantumESPRESSO easyconfigs (#7114)
  - use empty toolchain version in picard easyconfigs to ensure Java dependency is loaded during installation (#7116)

  - fix checksum for `mgcv` extension in R 3.5.* easyconfigs (#7122)
  - add missing 'patsy' extension in ARCH easyconfig + switch to using `PythonBundle` easyblock (#7124)
  - add missing (dummy) sklearn extension in NeuroKit easyconfig (#7126)
  - fix order of extensions in recent IPython easyconfigs to make '`pip check`' pass (#7131)
  - fix missing required Python packages in ASE 3.16.2 easyconfigs (#7139)
  - fix `source_urls` in CP2K easyconfigs (for official releases) (#7240)
  - update SCOTCH 6.0.5 easyconfigs in-place to 6.0.6 to fix wrong download URL (#7159)
  - add Autotools build dep in patchelf easyconfig (#7175)
  - STREAM fixes: use https + versioned source (#7193)
  - fix checksum in Molden easyconfig (#7251)
  - fix source URL for ADMIXTURE (#7258)
  - fix source URLs & versions + build procedure for SKESA (#7275)
  - fix checksum for TRF 4.09 (#7278)
  - add missing zlib dependency for NSS (#7293)
  - add required build deps for QtWebEngine in recent Qt5 easyconfigs (#7300)
  - fix ImageMagick source URLs: use `launchpad.net` for old versions, switch to GitHub for 7.0.8-* onwards (#7301)
- other changes:
  - update `Java/1.8` wrapper to Java 1.8.0_192 (#7097)
  - don't enable `optarch` toolchain option in VCFtools easyconfigs, since it's enabled by default (#7106)
  - stop using lowopt in libxc 4.2.3 easyconfigs (#7115)
  - deprecate easyconfigs for `ictce` toolchains + `intel` toolchains older than `intel/2016a` (#7231)
  - deprecate easyconfigs for `goolf(c)` + corresponding `gompi(c)` subtoolchains (#7243)

11.12.24 EasyBuild v3.7.1 (October 18th 2018)

bugfix/update release

framework

- various enhancements, including:
  - generate `.modulerc.lua` when Lua syntax and Lmod >= 7.8 is used (#2597)
  - allow `--force` to use regex if `--try-toolchain` can not map intelligently (#2605)
  - add support for disabling modules tool version check (#2610)
  - add support to `ModuleGenerator.modulerc` method to also write `.modulerc` file (#2611)
  - check whether module file being wrapped exists in same directory as module wrapper when using Lmod 6.x (#2611)
- various bug fixes, including:
  - stop relying on '`easy_install`' in bootstrap script, use '`python -m easy_install`' instead (#2590)

  - fix templating of values in `list_software` function (#2591)
  - fix composing of `lib64` fallback paths in sanity check (#2602)
  - determine `file_info` for all easyconfigs before any actual copying in `copy_easyconfigs` function (#2604)
  - also check for module wrappers in `ModulesTool.exist` method (#2606)
  - add trailing newline to module load message if it's not there yet (#2613)
  - retain all dependencies when determining dependency tree of a toolchain (#2617)
  - protect `exts_lists` from templating in `dump` method (#2619)
  - making CUDA capability detection more robust (#2621)
- other changes:
  - lower required Lmod version to 6.5.1 (#2593)

easyblocks

- new software specific easyblocks for fastStructure (#1529)
- minor enhancements, including:
  - support bypassing use of system type obtained with recent `config.guess` in `ConfigureMake` (#1531)
  - enhance generated module file for FreeSurfer (#1543)
  - add option in Qt easyblock to check for QtWebEngine component in sanity check (#1544)
  - also install CP2K as a library + code cleanup in CP2K easyblock (#1547)
- various bug fixes, including:
  - fix checking for downloaded dependencies for stand-alone installations in `PythonPackage` generic easyblock (#1530)
  - also specify `--host` option to configure script based on config.guess result in `ConfigureMake` easyblock (#1532)
  - use short module name when creating module wrapper in `ModuleRC` generic easyblock (#1535)
  - use `DOT_MODULERC` constant in `ModuleRC` easyblock rather than hardcoding '`.modulerc`' (#1533)
  - use `--no-deps` when installing .whl in TensorFlow easyblock if extension are being installed, move test run to sanity check (#1537)
  - use `os.getcwd()` rather than `self.startdir` in TensorFlow easyblock to fix installation of TensorFlow as extension in a bundle (#1540)
  - add symlink to wrapped module file when creating `.modulerc` in temporary location (#1539)
  - properly handle Python dependency in Qscintilla easyblock (#1499)
- other changes:
  - cleanup Tarball easyblock by using `copy_dir` function (#1541)

easyconfigs

- added easyconfigs for new toolchain `foss/2018.08` with GCC/8.2.0 (#6992)
- added example easyconfig files for 26 new software packages:

  - ARCH (#6939), fbm (#6948), GenomeTester4 (#6970), GlobusConnectPersonal (#6974), MMseqs2 (#6964), NAG (#5772), NAGfor (#5772), NSPR (#7005), NSS (#7005), NeuroKit (#6947), novoalign (#6944), OptiType (#6924), OrthoFinder (#6964), pandas-datareader (#6938), pFUnit (#6949), PMIx (#6930), PSolver (#6888), PyDatastream (#6951), PyFR (#6846), Pyomo (#6910), Quandl (#6950), RTG-Tools (#6862), seq2HLA (#6969), suds (#6951), UCX (#6931), XMLSec (#6929)

- added additional easyconfigs for various supported software packages, including:
  - binutils 2.31.1, dask 0.19.4, GCC(core) 8.2.0, OpenBLAS 0.3.3, OpenMPI 3.1.2, Pillow 5.3.0, PyCUDA 2018.1
- minor enhancements, including:
  - add clustree + plotly extensions (+ deps) for R 3.5.1 (#6901)
  - add tclsh symlink to recent Tcl easyconfigs (#6915)
  - update GROMACS 2016.3 with NVML patches (#6936)
  - use `make=make` in build options for ParMGridGen easyconfigs (#6952)
  - add `Data::Dump` as extension to Perl (dependency for GIMIC) (#7004)
- various bug fixes, including:
  - inhibit `-Werror` in binutils 2.26 as new system GCC has case fallthrough warnings (#5793)
  - change fastStructure easyconfig to use custom easyblock (#6893)
  - fix toolchain for tbb dependency in `Bowtie2-2.3.4.2-foss-2018b.eb` (#6927)
  - fix uroot installation in R-3.5.1-foss-2018b (#6934)
  - add missing Python packages in TensorFlow 1.10.x easyconfigs (#6940)
  - add missing dep for Szip in 2017b builds of netCDF (#6942)
  - fix missing comment from `OpenBabel-2.4.1-fix-link-path-tests.patch` (#6943)
  - add patch to Python 3.6.1 - 3.6.3 that removes comment in comment (#6946)
  - add missing `ulimit_unlimited=True` (see issue #6484) in the newest Python builds (#6959)
  - solve '`version UUID_1.0 not found`' problem in LibUUID easyconfigs (#6962)
  - eliminate dependency on ancient problematic LibUUID library, replace with util-linux (#6963)
  - force building of `ccmake` for CMake 3.12.1 + fix deps (#6967)
  - fix broken installation for Python 3.6.2 & 3.6.3 with PyNaCl as dep for paramiko extension by explicitly including previous PyNaCl version as extension (#6971)
  - fix source URL for ADMIXTURE (no https) + add SHA256 checksum (#6982)
  - add missing NSS/DBus dependencies to Qt 5.10.1 easyconfigs built with foss toolchain to ensure that QtWebEngine component gets installed (#7005)
  - add '`openssl`' OS deps in Perl 5.28.0 easyconfig for `Net::ssleay` (#7008)
  - add missing checksum for ipaddress extension in Python easyconfigs (#7021 and #7033)
  - add missing extensions to Python 2.7.14 (#7022 and #7023), 3.6.2 (#7025), 3.6.3 (#7027) and 3.6.4 (#7029 and #7030)
  - add missing OS dependencies for git (#7028)
  - create default configuration for RTG-Tools (#7032)
  - also run checks on changed files when target branch for PR is something different than 'develop' (#7034)

- other changes:
  - removed dead 'bzip.org' source URL for bzip2 source tarball (#6983)

11.12.25 EasyBuild v3.7.0 (September 25th 2018)

feature release

framework

- minimal Lmod version requirement bumped to 6.6.3 (#2575)
- various enhancements, including:
  - add support to bootstrap script to force install specific EasyBuild version (#2382, #2580)
  - consider potential of multiple subtoolchains when resolving dependencies (#2464, #2465, #2466, #2585)
  - fall back to downloading using the `requests` Python package (if installed) when `urllib2` fails due to SSL error (#2538)
  - make `--try-toolchain` more aware of subtoolchains (#2539)
    - subtoolchain of original toolchains are now mapped to subtoolchains of target toolchain
  - add support for BLIS and goblf toolchain that uses BLIS for BLAS (#2540)
  - allow skipping of sanity check step via '`skipsteps`' easyconfig parameter (#2549)
  - add support for `--check-contrib` (#2551)
    - equivalent with `--check-style`, but also verifies presence of SHA256 checksums (+ more checks in the future)
  - added support to 'download' sources from git (#2555)
    - see https://easybuild.readthedocs.io/en/latest/Writing_easyconfig_files.html#downloading-from-a-git-repository
  - add '`parse`' hook to add support for applying site-specific customisations to the 'raw' easyconfig (#2562, #2566)
    - see https://easybuild.readthedocs.io/en/develop/Hooks.html
  - lift invalidating of module caches into helper method that can be used by easyblocks (#2571)
  - always dump a fully parsed easyconfig to the '`reprod`' subdir of the installation directory (#2574)
  - add '`modulerc`' method to `ModuleGenerator` class (#2575)
- various bug fixes, including:
  - make GC3Pie stop build process if a dependency failed (#2474)
  - filter out patched files in test/ in `fetch_easyconfigs_from_pr` (#2547)
  - check GC3pie version using the `pkg_resources` API rather than using `__version__` (which was removed in GC3Pie 2.5.0) (#2554)
  - fix enforcing of checksums for extensions (#2561, #2570, #2579)
  - skip running of configuration checks while only a single configuration level is taken into account during `--show-config` (#2567)
  - fix error statements in modules tool version checks (#2576)
  - fix finding of software subdirectory for specified patch file in `--new-pr`/`--update-pr` (#2577)

  - take into account dependency 'wrappers' in check_conflicts (#2583)
  - stick to pycparser < 2.19 with Python 2.6 in Travis config (#2584)
- other changes:
  - use `namelower` as default for '`github_account`' easyconfig parameter (#2528)
  - use `.counts()` rather than deprecated `.stats()` for GC3Pie (#2573)

easyblocks

- new generic easyblock: `ModuleRC` (#1503, #1518)
- new software specific easyblocks for BWISE (#1497) and VEP (#1512)
- minor enhancements, including:
  - update QuantumESPRESSO easyblock: stop building in installation dir, do not use external FoX dependency, support for recent versions (#1312)
  - updates to TensorFlow easyblock:
    - require cuDNN if CUDA support is enabled, enable mkl-dnn by default, fix problem with internal protobuf problem, add awareness for TensorRT & NCCL (#1453)
    - add support for IntelMPI (#1507)
  - update WIEN2k easyblock for version 18.1 (#1460)
  - add CUDA '`stubs`' subdirectory to $LIBRARY_PATH (#1464)
  - add support for building ScaLAPACK on top of BLIS (#1467)
  - handle X11 better and make '`static`' a build flag in Amber easyblock (#1468)
  - update Boost easyblock for Boost 1.67.0 (name change in Python 3.x library files) (#1472)
  - add CEI/bin to $PATH for ANSYS 19 & newer (#1476)
  - enhance SCons easyblock to enable building in parallel (#1477)
  - add awareness for CCOLAMD and CAMD in Trilinos easyblock (#1480)
  - add support to apply (binary) patches after main CUDA install (#1481, #1483)
  - add support for only building Python bindings (+ code cleanup) in Boost easyblock (#1484, #1495)
  - provide control over subdirectory in which R packages are installed (#1485)
  - added regex to fix `$WM_PROJECT_VERSION` correctly in OpenFOAM easyblock (#1489)
  - improve PGI siterc so it allows `-pthread` switch (#1494)
  - customise `check_checksums method` in Bundle easyblock to fix checking of checksums for components (#1496)
  - include a pkgconfig file '`hdf5.pc`' to HDF5 installations (#1504)
  - (download &) use an updated `config.guess` script in generic ConfigureMake easyblock (#1506, #1522, #1523, #1524)
  - make `IntelBase` generic easyblock aware of `(pre)installopts` (#1509)
  - update Siesta easyblock for v3.2 to 4.1-b3 (#1510)
  - take `(pre)installopts` into account in RPackage generic easyblock (#1513)
  - update DOLFIN easyblock for latest version (#2018.1) (#1521)

- various bug fixes, including:
  - drop useless definition of `$NLSPATH` in `IntelBase` + fix ipp library paths (#1442)
  - fix order of arguments in log message in `PythonPackage` easyblock (#1459)
  - run `ldconfig` in post-install step of CUDA easyblock to create missing symlinks in 'stubs' subdirectory (#1473)
  - take into account that only name/version may be specified for some components in `Bundle` easyblock (#1474)
  - make SuperLU easyblock consider both `lib` and `lib64` subdirectories (#1479)
  - use short build dir for Trilinos to dance around "`Argument list too long`" problem + link with `libmetis.a` (#1486)
  - correct check for Red Hat 6 based OS in TensorFlow easyblock (#1487)
  - improve configuration choice in FSL easyblock (#1498)
  - don't check for `mcc` in MATLAB sanity check as it requires a specific toolbox license (#1514)
  - make sure Bazel doesn't write files in $HOME/.cache when building TensorFlow (#1519)
  - enable VSX on POWER for FFTW >= 3.3.7 (#1520)
  - add `librt` as dependency when linking Trilinos with SuiteSparse (#1525)
- other changes:
  - switch to using CMake install procedure for Eigen 3.3.4 & newer (#1482)
  - bump Lmod version used in Travis config to 6.6.3 (now required by framework) (#1505)

easyconfigs

- added easyconfigs for new toolchains: `fosscuda/2017b` (#6706), `intelcuda/#2017b` (#6709), `iomkl/2018b` (#6661)
- added example easyconfig files for 49 new software packages:
  - alleleCount (#6676), BCALM (#6796), BDBag (#6672), BFC (#6647), Bio-DB-HTS (#6854), bioawk (#6865), biomart-perl (#6745), BLIS (#6614), Boost.Python (#6763), BWISE (#6802), CapnProto (#6542), CaVEMan (#6708), CCL (#5802), cDNA_Cupcake (#6787), cget (#6780), coevol (#6589, #6642), Delly (#6735), earthengine-api (#6556), fineRADstructure (#6586), GIMIC (#6575), GitPython (#6850), HiC-Pro (#5873), Inelastica (#6831), JAXFrontCE (#6837), jq (#6632), KAT (#6808), KMC (#6553), Kraken2 (#6569), Lighter (#6553), Mash (#6542), MetaPhlAn2 (#6600), Minimac4 (#6781), mordecai (#6670), NCCL (#5802), NxTrim (#6646), parasail (#6601), PheWAS (#6030), Pilon (#6553), python-parasail (#6601), SearchGUI (#6637), shovill (#6553), SKESA (#6553), snakemake (#6851), strelka (#6742), SWIPE (#6795), ToFu (#6322), tqdm (#6721), TRUST (#6601), VEP (#6854)
- added additional easyconfigs for various supported software packages, including:
  - Bazel 0.16.0, Boost 1.67.0 + 1.68.0, CUDA 10.0.1, DOLFIN (FEniCS) 2018.1, FSL 5.0.11, GC3Pie 2.5.0, GROMACS 2018.3, HTSlib + SAMtools 1.9, IPython 6.4.0, matplotlib 3.0.0, OpenCoarrays 2.2.0, OpenFOAM 6 & v1806, PyTorch 0.4.1, Python 3.6.6, R 3.5.1, TensorFlow 1.10.1, Trilinos 12.12.1, WIEN2k 18.1
- minor enhancements, including:
  - add test to enforce SHA256 checksums in touched files in pull requests to develop (#5005)
  - add various extensions to recent R easyconfigs (#6590, #6686, #6858)
  - add `mpmath` extension to Python 3.6.4 + 3.7.0 easyconfigs (#6607, #6713)

  - add Java 1.8 easyconfig that can be updated in-place to more recent JDK 1.8.x (#6712)
    * see also https://easybuild.readthedocs.io/en/latest/Wrapping_dependencies.html
  - add `iso_c_binding` support to arpack-ng 3.6.2 (#6718)
  - add `XML::Parser`, `XML::RegExp` & `XML::DOM` extensions to Perl(#6744)
  - add `tabulate` extension to all Python > 3.6 easyconfigs (#6809)
  - enable MPI version of arpack-ng 3.5.0 (#6840)
  - add `wish` symlink to `wish8.6` in recent Tk easyconfigs (#6870)
  - add `libXp` and `printproto` to X11 bundle (#6873)
- various bug fixes, including:
  - define '`_GNU_SOURCE`' to ensure that '`reallocarray`' is defined in flex 2.6.4 easyconfigs, fixes bootstrap crash (#5792, #6766)
  - added tabix dependency in VCFtools (#6584)
  - added zlib dependency to BWA (#6591), BEDTools (#6592), VCFtools (#6653)
  - add checksums to googletest easyconfigs (#6611)
  - fix source URL + add checksum to wkhtmltopdf-0.12.3-Linux-x86_64.eb (#6628, #6641)
  - don't use external FoX dependency in QuantumESPRESSO 6.2 (#6638)
  - update hwloc easyconfigs with libxml2 and libpciaccess dependencies (#6639)
  - fix gettext build dep for Mono-5.10.0.160 (#6640)
  - remove no longer existing config option `--enable-mpi-thread-multiple` in OpenMPI 3.x easyconfigs (#6645)
  - fix homepage in recent HDF5 easyconfigs (#6687)
  - add expat dependency to Mesa v18.1.1 (#6706)
  - add missing `--with-trio-flavor=netcdf` flag to ABINIT 8.x easyconfigs (#6711)
  - remove useless definition of $TORCH_CUDA_ARCH_LIST in PyTorch easyconfig that doesn't use CUDA (#6719)
  - fix builing GCCcore 6.3.0 on recent OSs by backporting patches from 6.4.0 (#6727)
  - fix `source_urls` in bzip2 easyconfigs (#6728)
  - specify location of dependencies in configure options for libgd (to avoid system libraries being used instead) (#6731)
  - also install docs/man pages in recent git easyconfigs (versions 2.1x.y) (#6751)
  - add CMake build dep for Eigen 3.3.4 (#6756, #6784)
  - add expat dependency to Perl 5.26.0 (#6758)
  - add ncurses OS dependency to CMake for Debian distros (#6783)
  - fix checksums for pkgmaker, rngtools & RcppProgress extensions in R 3.4.3 easyconfigs (#6815)
  - add pkg-config build dependency to GTS (#6819)
  - do not check if hwloc-dump-hwdata utility was installed, since it's only built on x86 systems (#6836)
  - patch libxc 3.x and 4.0.x to compile on POWER (#6868)

  - consistently add patch for paycheck extension in Python 3.6/3.7 to fix UTF8 issue with README (#6892)
- other changes:
  - bump Lmod version used in Travis config to 6.6.3 (now required by framework) (#6834)

11.12.26 EasyBuild v3.6.2 (July 11th 2018)

bugfix/update release

framework

- various enhancements, including:
  - add support for including environment variable that is resolved at "module load time" in user module path (#2395)
    * `{RUNTIME_ENV::EXAMPLE}` is replaced by value of `$EXAMPLE` when module is loaded
  - also support generating Docker container recipes & image via `--containerize` (still experimental) (#2479)
  - add support for specifying source URLs directly in '`sources`' (#2520)
  - perform early 'raw' parse of provided easyconfig file to check for syntax error or faulty inputs (#2523)
  - add '`bitbucket_account`' easyconfig parameter and template, and let `BITBUCKET*` templates use it (#2525)
- various bug fixes, including:
  - take into account `--filter-deps` when re-loading build dependencies in `extensions_step` (#2516)
  - fix for offline use of bootstrap script: ignore errors when determining source URLs if source tarballs are provided (#2517)
  - fix error message that is raised for incorrect type of value in `sanity_check_paths` (#2524)
- other changes:
  - move flake8 config into `setup.cfg` + fix style issues in `easybuild/tools/options.py` (#2511)
  - make test that verifies that `BuildOptions` does not support updating a bit more flexible (#2518)

easyblocks

- new software-specific easyblock for OpenCV (#1444)
- minor enhancements, including:
  - use `$CPATH`/`$LD_LIBRARY_PATH` for CMake's `find_path`/`find_library` functions in `CMakeMake` easyblock (#1165)
  - make cdft lib compilation optional for Intel MKL, by detecting MPI availability (#1393)
  - add `use_glibcxx11_abi` easyconfig parameter in Boost easyblock (#1434)
  - enable filtering of paths in `$CPATH` and `$LIBRARY_PATH` in TensorFlow easyblock (#1436)
  - add support for building GROMACS with `--optarch=GENERIC` (#1440)
  - check current stack limit and set it to '`unlimited`' if possible in Python easyblock (#1441)
  - trivial update for Q&A in SAS easyblock (#1448)
  - allow skipping tests when installing Perl extensions, by setting '`runtest`' to `False` (#1451)

  - add support for installing Intel products using serial numbers (#1452)
  - update version check to FFTW 3.3.8 for tests to pass on POWER (ppc64le) (#1454)
- various bug fixes, including:
  - build MPFR in GCC stage 1 without LTO if (system) GCC used is too old (#1435)
  - make sure xmlpatterns always gets built with Qt (#1437)
  - fix symlink check in NWChem easyblock + use `change_dir`/`remove_file`/`symlink` functions (#1438)
- other changes:
  - move flake8 configuration to `setup.cfg` and make HoundCI aware of it (#1430)

easyconfigs

- added easyconfigs for new toolchains: `foss/2018b` (#6424), `fosscuda/2018b` (#6555) and `intel/2018b` (#6409)
- added example easyconfig files for 28 new software packages:
  - CUnit (#6469), eggnog-mapper (#6513), FANN (#6468), FTGL (#6421), FreeFem++ (#5918), fastStructure (#6448), fastq-tools (#5396), fullrmc (#6422), GDGraph (#6529), heaptrack (#6450), libgpuarray (#5429), lz4 (#6449), MAJIQ (#5983), makedepf90 (#6504), nanopolish (#6464), opencv_contrib (#6441), PRC (#6477), Pillow-SIMD (#6152), Pytorch (#6152), poretools (#6467), pystran (#6395), R-keras (#6530), Scoary (#6521), Scrappie (#6469), torchvision (#6152), WISExome (#6524), WannierTools (#6539), zstd (#6449, #6452)
- added additional easyconfigs for various supported software packages, including:
  - GROMACS 2018.2, HDF5 1.10.2, IPython 6.3.1, Kraken 1.0, Mesa 18.1.1, netCDF 4.6.1, NWChem 6.8, OpenBLAS 0.3.1, OpenMPI 3.1.1, Perl 5.28.0, Python 2.7.15, R 3.5.0, X11 20180604
- minor enhancements, including:
  - add hint on file name differences between downloaded and expected in cuDNN easyconfig (#6042)
  - add '`gee`' extension to R 3.4.4 easyconfigs (#6376)
  - enable building of MPI libraries in VTK 8.1.0 easyconfigs (#6460, #6429)
- minor changes, including:
  - rename '`Canu`' to '`canu`' for v1.7 (#6389)
  - update existing easyconfigs for OpenCV 3.4.1 to use new custom easyblock for OpenCV (#6509)
  - fix software name in `Bsoft` easyconfig (#6557)
- various bug fixes, including:
  - fix SAMtools dependency for ChimPipe, required SAMtools < 1.0 (#5930)
  - skip installing of documentation in easyconfig for jemalloc 5.0.1 (#6428)
  - stop including GCCcore 6.4.0 as build dep for GCCcore 8.1.0 (#6431)
    - no longer needed with updated GCC easyblock
  - add patch for GCCcore 6.4.0 to fix compilation problems on glibc 2.26 systems (#6432, #6495)
  - fix checksums for `pkgmaker`/`rngtools`/`RWeka`/`RcppProgress`/`mgcv` extensions in R 3.4.4 easyconfigs (#6446, #6502)
  - added necessary compiler flag for Guile 1.8.8 (#6457)

  - build hwloc 1.11.10 with `-fno-tree-vectorize` to avoid segfaulting lstopo on Intel Skylake (#6461)
  - add patch for binutils 2.30 to fix assertion failure (#6463)
  - sync extensions in `Python-3.6.4-iomkl-2018a.eb` easyconfig with other Python 3.6.4 easyconfigs using 2018a toolchain (#6471)
  - enable checking/setting of unlimited stack limit in Python 3.6.x easyconfigs using an '`intel`' toolchain (#6485, #6492)
  - add missing `libxml2` dependency for HDF5 1.10.1 (#6498)
  - also copy eggnog-mapper scripts (#6522)
  - fix missing dependencies for Perl modules included as extensions for Perl 5.26.1 (#6532) and 5.28.0 (#6533)
  - fix location of include directory in libffi 3.2.1 easyconfigs (#6565)
- other changes
  - also check for multiple dependency variants in easyconfigs using GCCcore 7.3.0 & newer (#6444)

11.12.27 EasyBuild v3.6.1 (May 26th 2018)

bugfix/update release

framework

- various enhancements, including:
  - add support for enabling fallback in sanity check to consider `lib64` equivalent for seemingly missing libraries (#2477)
  - add `GITHUB_LOWER_SOURCE` constant (#2491)
  - add '`exts_download_dep_fail`' as known easyconfig parameter (#2493)
  - add support for passing custom messages on failing sanity check for extensions (#2494)
  - add definition for `fosscuda` toolchain (#2507)
- various bug fixes, including:
  - make `--inject-checksums` always re-order `source_urls`/`sources`/`patches`/`checksums` (#2487)
  - fix git remote url in `CONTRIBUTING.md` (#2490)
  - make `flake8` happy in `easyblock.py` (#2492)
  - handle missing permissions for adding labels in `--new-pr` (#2497)
  - restore tweaked `$TMPDIR` value after loading module (for sanity check) (#2498)
  - enhance `get_module_path` function to auto-detect generic vs software-specific easyblock class names (#2502)
  - don't blindly overwrite an existing easyconfig in `tweak_one` (#2504)
  - take account that `PlaintextKeyring` may be provided via `keyrings.alt` (#2505)
  - prepend location for temporary module file to `$MODULEPATH` with high priority + mark it as default in `load_fake_module` method (#2506)

easyblocks

- minor enhancements, including:
  - add support for detecting auto-downloaded dependencies in `PythonPackage` easyblock (#1377)
    - disabled by default, can be enabled using "`download_dep_fail = True`" in easyconfig file
  - add support to enable integration of pscom in psmpi easyblock (#1397)
  - set `$CMAKE_*_PATH` when `CMake` is loaded in `PythonPackage` easyblock (#1398)
  - update WIEN2k easyblock for v17 (#1405)
  - disable jemalloc support in TensorFlow on CentOS 6 & co (+ minor cleanups) (#1412)
  - update Maple easyblock to support recent versions (#1414)
  - enable `nc-config` usage for netCDF in ESMF >=7.1 (#1419)
  - enhance PETSc easyblock for version 3.9 (#1421)
  - check output of MATLAB installation command for invalid installation key error (#1423)
  - fix suffix for Boost Python library in Boost 1.67.0 & newer (#1424)
  - support adding specific paths to `$PATH` for generic Binary easyblock via '`prepend_to_path`' custom easyconfig parameter (#1426)
- minor changes, including:
  - assume PGI Community edition is used when no license file is specified (#1427)
- various bug fixes, including:
  - fix imkl sanity check overwriting base libs with interface libs (#1392)
  - install Chimera in a subdirectory to avoid its dependencies being added to the environment (#1413)
  - add conditional so "`--with-x`" is only added to configopts if left unspecified in R easyblock (#1415)
  - make `configure` and `make` look for FoX in `$EBROOTFOX` in QuantumESPRESSO easyblock (#1420)
  - fix path for `$ICEM_ACN` in ANSYS easyblock (#1422)
  - avoid hardcoding defaults in question patterns in Doris easyblock (#1428)

easyconfigs

- added easyconfigs for new toolchains `fosscuda/2018a` (#6363) and `giolfc/2017b` (#5799)
- added example easyconfig files for 24 new software packages:
  - BAGEL (#6332), Bottleneck (#6334), cisTEM (#6370), cftime (#6337), dotNET-Core-Runtime (#6345), ecCodes (#6235), feh (#6300), Graphene (#5043), gffread (#6306), HOME (#6227), HiCExplorer (#6342), ICU (#6371), Imlib2 (#6300), KmerGenie (#5929), libgeotiff (#6262), NetPIPE (#6062), Pandoc (#6247), Pisces (#6347), Proteinortho (#6333), pyshp (#6364), SIMPLE (#6019), STIR (#6349), SimpleElastix (#6114), wxWidgets (#6370)
- added additional easyconfigs for various supported software packages, including:
  - Boost 1.67.0, ESMF 7.1.0r, GATK 4.0.4.0, GCC 8.1.0, GROMACS 2018, OpenBLAS 0.3.0, PETSc 3.9.1, PGI 18.4, TensorFlow 1.8.0, WIEN2k 17.1
- minor enhancements, including:
  - add `py_expression_eval` extension to (recent) Python 2.7.14 & 3.6.4 easyconfigs (#6285)
  - add `README` for Java with information on downloading source tarball (#6294)
  - add several extensions to Perl 5.26.1 easyconfigs, incl `Dist::Zilla` & dependencies (#6297)

  - also include archive URLs for Bioconductor 3.6 (#6311)
  - add `README.md` file for installing icc/ifort (#6317)
- various bug fixes, including:
  - consistently specify '`intel-mkl`' component in recent Intel MKL easyconfigs (#6234)
  - add `pkg-config` as build dependency to libdrm (#6243)
  - add `pkg-config` build dep to most recent libdrm easyconfigs (#6244)
  - fix checksum for foreign extension in R 3.4.3 and R 3.4.4 easyconfigs (#6245)
  - fix installation of Libint 2.4.2 by building with `-std=c++11` (#6251)
  - fix source spec for networkx 2.1 extension in scikit-image easyconfig (#6254)
  - avoid that Nipype downloads dependencies for included extensions (#6261, #6263)
  - consistently include patch for FLTK 1.3.4 (#6265)
  - fix Perl shebang in MCL v14.137 scripts (#6269)
  - add patch to build particular source file of matrixStats extension in R 3.4.4 easyconfig with `-O1` to work around ICE in Skylake systems (#6278)
  - add `pkg-config` build dependency to FFmpeg >= 3.3.1 (#6291)
  - change back checksum for libdap 3.19.1 (#6305)
  - add patch for Automake 1.15 to fix issue with recent Perl versions (#6358)
  - fix `glog` causing intel error in Intel compilers on Intel Skylake (#6360)
  - include ICU as dependency in recent R easyconfigs (v3.4.3 & v3.4.4) (#6371)
  - fix checksum for RSEM 1.3.0 after sneaky re-release (#6379)

11.12.28 EasyBuild v3.6.0 (April 26th 2018)

feature release

framework

- (experimental) support for generating Singularity container recipes & (optionally) images via '`sudo singularity`' (#2332, #2480, #2481, #2482, #2483)
  - see documentation at http://easybuild.readthedocs.io/en/latest/Containers.html
- include `-ftree-vectorize` and `-fno-math-errno` in default compiler optimisation flags for GCC (#2388)
  - this significantly improves performance of generated binaries when building with a GCC-based toolchain
  - can be disabled if needed via the `vectorize` toolchain option
- several enhancements/fixes to GitHub integration support:
  - loosen commit message requirements for `--new-pr` w.r.t. to patches as long as all easyconfigs are new (#2438)
  - automatically add '`new`' and/or '`update`' labels in `--new-pr` (#2384)
  - add `force_in_dry_run=True` to `copy_file` in `copy_patch_files` (#2442)
  - test for custom commit message when deleting a file instead of when adding a patch (#2443)

  - make `diff_stat` pattern also match output of older git versions in tests for `--new/update-pr` (#2444)
  - remove duplicates from `--new-pr` title (#2478)
- add support for "`eb --fetch`" to only download sources (even without having a modules tool installed) (#2457)
- add definitions for a bunch of new toolchains:
  - `golf` (#2458), `gmkl` (#2460), `gomkl` (#2455), `pmkl` (#2460)
  - toolchains including CUDA: `gmklc`, `gomklc`, `iimklc`, `iompic`, `iomklc` (#2461)
- various minor enhancements, including:
  - avoid that '`--inject-checksums`' adds lines longer than 120 characters (#2434)
  - enable caching of `$HOME/.cache/pip` in Travis config (#2435)
  - replace raw strings with bytes literal as iterator sentinels in checksum calculation (#2446)
  - new command-line option '`--job-max-jobs`' to cap nr of submitted build jobs with GC3Pie (#2378)
  - add configuration for houndci + flake8 (#2451)
  - add `Accept` header when downloading file (#2437)
  - include running of '`eb --check-github`' in Travis config (#2449, #2454)
  - flesh out common code blocks in `test/framework/options.py` (#2452)
  - support '`depends_on`' load statements in generated modules via `--module-depends-on` and `module_depends_on` easyconfig parameter (#2391)
  - fix compatibility with Modules v4.1.x (#2470)
  - add support to `run_cmd` to enable streaming output (#2476)
- various bug fixes, including:
  - determine whether included easyblocks are generic or not based on class they define (#2432)
  - make sure GitHub token is used in test for `--preview-pr` (#2436)
  - take into account that toolchain components may be hidden when determining toolchain composition (#2440)
  - stick to autopep8 1.3.4 when testing with Python 2.6 (#2462)
  - strip off `.lua` extension when backing up modules to ensure Lmod 6.x doesn't pick up on them (#2463)
  - check for `modulecmd.tcl` before modulecmd in bootstrap script to discriminate between Modules 4.1.x vs 3.2.10 (#2468)
  - fix `derive_alt_pypi_url` after PyPI switching to sha256 in package URLs + fix broken test for `pypi_source_urls` + fix bootstrap script (#2471)
  - make sure that both '`get_git_revision`' and '`this_is_easybuild`' return regular strings rather than Unicode strings (#2472)

easyblocks

- new software-specific easyblock for Nim (#1402)
- minor enhancements, including:
  - enhance `RPackage` easyblock to support installing from unpacked sources (#1383)

  - add support to `PythonPackage` easyblock to install with '`pip install --editable`' (#1384)
  - add `$EBROOTIFORT/include` in `$CPATH` for ifort (#1385)
  - add houndci + flake8 configuration (#1388)
  - add additional location to `$PATH` for FLUENT installations (#1389)
  - make `PythonPackage` generic easyblock aware of '`unpack_options`' easyconfig parameter (#1391)
  - minor updates to ABAQUS easyblock to support latest version (#1394)
  - add support for extracting sources in `Binary` easyblock (#1401)
- various bug fixes, including:
  - fix linking to FFTW for Doris: should be `-lfftw3f` (#1387)
  - fix for installing TensorFlow 1.6.0: use the absolute path for the C compiler when compiling with GPU support (#1386)
  - also take `lib64` into account for binutils libraries (#1399)
  - make sanity check in MPICH easyblock aware of libraries in `lib64` subdir (#1403)
  - take into account that `self.debuggerpath` may not be set in icc easyblock (#1408)
  - extend noqa in configure step of Qt easyblock (#1409)

easyconfig

- added easyconfigs for new toolchain golf/2018a (#6080)
- added example easyconfig files for 39 new software packages:
  - ARAGORN (#6138, #6219), amask (#6105), BamBam (#5490), Bandage (#6053), bcgTree (#6057), biobambam2 (#6040), CODEX2 (#5972), coverage (#5964), DANPOS2 (#5870), EasyQC (#6175), elastix (#6074), FRANz (#6035), Filtlong (#6050), FriBidi (#6181), Gblocks (#6057), Grace (#6036), LMfit (#6119), libmaus2 (#6040), MathGL (#6033), MiGEC (#6047), MiXCR (#6045), Miniconda3 (#6075), Multiwfn (#5403), mosdepth (#6169), Nim (#6167), Porechop (#6051), propy (#6091), Roary (#6056), Rtree (#6222), SALib (#6145), SMRT-Link (#6218), SimpleITK (#6172), spectral.methods (#6104), supernova (#6193), TEToolkit (#5912), tbl2asn (#6139), udocker (#5770), vartools (#6102), XCfun (#5975)
- added new easyconfigs for existing toolchains:
  - `gmpich/2017.08` (#6143), `goolfc/2018a` (#6129), `intel/2018.02` (#6077), `iomkl/2018.02` (#6088)
- added additional easyconfigs for various supported software packages, including:
  - ABINIT 8.6.3, Anaconda2 5.1.0, Anaconda3 5.1.0, binutils 2.30, dask 0.17.2, FFmpeg 4.0, GCC(core) 7.3.0, GROMACS 2016.5, HTSlib 1.8, LLVM 6.0.0, Mesa 17.3.6, netCDF 4.6.0, Octave 4.2.2, OpenCV 3.4.1, PLUMED 2.4.1, PROJ 5.0.0, PostgreSQL 10.3, Qt5 5.10.1, R 3.4.4, SAMtools 1.8, Spack 0.11.2, TensorFlow 1.7.0, VTK 8.1.0
- minor enhancements, including:
  - add test to ensure there's only one variant for each dependency in dep graph of easyconfigs using particular toolchains (#5970, #6023)
  - add extensions required by CODEX2 v20180227 to Bioconductor w/ R 3.4.3 (#5972)
  - add config file for https://pre-commit.com/ (#5785)
  - add custom sanity check paths for GraphicsMagick consistently (#5997)

  - add custom sanity check paths to recent Ghostscript easyconfigs (#5998)
  - add svd, Rssa, JBTools, RUnit, DistributionUtils and gapfill extensions for R 3.4.3 (#6099)
  - add additional extensions to Bioconductor 3.6 bundle (#6136)
  - add xlrd extension to recent Python 2.x and 3.x easyconfigs (#6162)
- minor changes, including:
  - use gettext 0.19.8.1 on top of libxml2 2.9.7 as dep for git built with `foss/2018a` (#5993)
  - bump hwloc dep for OpenMPI 2.1.2 that is part of `iomkl/2018a` to v1.11.8 (#5994)
  - use non-interactive plotting backend by default for matplotlib 2.1.2 (#6024)
  - don't use bare Perl as dependency for git with `foss/2018a`, use variant with extensions (#6058)
- various bug fixes, including:
  - fix `moduleclass` & add custom `sanity_check_paths` in gettext easyconfigs (#5991)
  - drop use of `--disable-dlopen` in (recent) OpenMPI easyconfigs due to negative performance impact (#6060)
  - add missing XZ dep in Python 3.6.4 easyconfigs built with `*/2018a` toolchain (#6065)
  - add 10 packages that were previously downloaded in Python 3.6.4 easyconfigs (#6081)
  - add patch for Tensorflow 1.6 & 1.7 to include missing -lrt link flag (needed in CentOS6) (#6089)
  - fix checksums for R extensions that were updated in place in easyconfigs for R versions 3.4.3 & 3.4.4 (#6118)
  - include pkg-config as build dep in recent R easyconfigs (required for at least nloptr) (#6122)
  - remove Intel-specific workaround for '`undefined symbol: __stack_chk_guard`' issue from Python 3.6.4 foss/2018a easyconfig (#6130)
  - add configopt `--without-matlab/octave` to all NLopt easyconfigs (#6132)
  - also consider `lib64` in `sanity_check_paths` for Bison 3.0.4 (#6170)
  - don't use libyaml built with dummy as dep for PyYAML (#6208)
  - add missing pkg-config build dep in recent GObject-Introspection, GLib & cairo easyconfigs (#6216)
  - don't include (ancient version of) `Time::HiRes` as Perl extension, since it's a core Perl module (#6225)
    - this fixes problems with the installation of BioPerl and proovread
  - add missing XML-LibXML dependency in recent BioPerl easyconfigs (#6226)
  - add `--without-ada` configure option in recent ncurses easyconfigs (#6228)
  - add patch for snaphu to fix segmentation fault due to use of short integer (#6230)

11.12.29 EasyBuild v3.5.3 (March 7th 2018)

bugfix/update release

framework

- various enhancements, including:
  - re-enable testing against environment modules, bump Lmod to 7.7.16 (#2425)
  - print which hook is being executed in the command line output (#2427)

- various bug fixes, including:
  - fix order in result of gen_list_easyblocks and gen_easyblocks_overview_rst (#2421)
  - fix target account for branch pushed when using `--new-pr` (#2426)

easyblocks

- minor enhancements, including:
  - make GROMACS easyblock select build type based on value for '`debug`' in '`toolchainopts`' (#1374)
  - re-enable testing against environment modules, bump Lmod to 7.7.16 (#1376)
  - enhance Gurobi easyblock to support installing Python bindings (#1378)

easyconfigs

- added example easyconfig files for 2 new software packages:
  - CNVkit and hmmlearn (#5954)
- added additional easyconfigs for various supported software packages, including:
  - matplotlib 2.1.2, TensorFlow 1.6.0
- minor enhancements, including:
  - re-enable testing against environment modules, bump Lmod to 7.7.16 (#5944)
  - add cghFLasso extension to R 3.4.3 easyconfigs (#5953)
  - add '`Math::CDF`' extension to recent Perl modules (#5957)
- various bug fixes, including:
  - add missing `--enable-ld-version-script` configure option for LibTIFF 4.0.9 built with `GCCcore/6.4.0` (#5945)
  - hard disable UCX support in recent OpenMPI versions, to dance around bug in OpenMPI configure script (#5949)

11.12.30 EasyBuild v3.5.2 (March 2nd 2018)

bugfix/update release

framework

- various enhancements, including:
  - add functionality to skip devel module with naming scheme (#2374)
  - add pagination support in `clean_gists.py` (#2379)
  - allow basic compiler modulenames to be specified as keys in `--optarch` (#2387)
  - initial set of OHPC module meta data for EasyBuild (#2392)
  - allow different target account in `post_comment_in_issue` (#2399)
  - declare support for RPATH linking stable (#2409)
- various bug fixes, including:
  - update bootstrap script to be compatible with Modules v4 (#2390)
  - avoid fatal error when determining glibc version on non-glibc Linux system (e.g. Alphine Linux) (#2398)
  - exclude location of RPATH wrappers from `$PATH` to avoid fork bomb (#2410)

  - fix target account for `--update-pr` in case it's different from GitHub account being used to push branch (#2419)

easyblocks

- new software-specific easyblocks for COMSOL (#1317), Stata (#1241) and TensorFlow (#1287, #1361)
- enhance GCC easyblock to support building generically (via '`generic`' easyconfig parameter or `--optarch=GENERIC`) (#1336)
- minor enhancements, including:
  - make GROMACS easyblock aware of building for KNL via `--optarch=MIC-AVX512` (#1360)
  - unset `$PERL_MM_OPT` and `$PERL_MB_OPT` when installing Perl modules to avoid problems (#1362)
  - add custom '`use_pip_for_deps`' easyconfig parameter to `PythonPackage` easyblock (#1366)
  - add support for '`default_component_specs`' easyconfig parameter in `Bundle` easyblock (#1369)
- various bug fixes, including:
  - fix logic in icc easyblock w.r.t. location of debugger libraries (libipt library for gdb) (#1224)
  - fix Tkinter easyblock to install Tkinter 3.x (#1347)
  - let impi modules also update `$MANPATH` (#1354)
  - enhance Octave extension filter to avoid false positives (#1355)
  - make CUDA easyblock aware of '`preinstallopts`' easyconfig parameter (#1367)
  - fix handling of per-component (checksums for) patches in `Bundle` easyblock (#1369)

easyconfigs

- added example easyconfig files for 24 new software packages:
  - AMPL-MP (#5800), AmberTools (#5632), bcolz (#5864), detonate (#5709), dropEst (#5734), EvidentialGene (#5627), faceswap (#5825), fineSTRUCTURE (#5663), Gradle (#5828), gbs2ploidy (#5877), HIPS (#5725), Ipopt (#5800), libMemcached (#5804), MEGAHIT (#5748), Mmg (#5807), methylpy (#5874), pstoedit (#5884), python-igraph (#5905), RNAcode (#5854), Seq-Gen (#5695), Stata (#5102), scikit-allel (#5864), TetGen (#5681), zarr (#5864)
- added new easyconfigs for existing toolchains: giolf/2018a (#5777), goolfc/2017b (#5768), iomkl/2018a (#5878)
- added additional easyconfigs for various supported software packages, including:
  - CP2K 5.1, IPython 6.2.1, OpenFOAM v1712, Perl 5.26.1, Python 3.6.4, TensorFlow 1.5.0, X11 20180131
- minor enhancements, including:
  - add `feather` and `tidyverse` as extensions for R 3.4.3 (#5693)
  - build recent PLUMED versions with all modules enabled (#5696)
  - add MAST/splatter/scDD to Bioconductor 3.6 bundle (#5704)
  - add `dummies` as extension for R 3.4.3 + `monocle`/`scde`/`ROTS` (+ deps) to Bioconductor 3.6 bundle (#5724)
  - include `io` and `statistics` extensions to Octave 4.2.1 easyconfigs (#5798)
  - switch to using TensorFlow as backend for recent versions of Keras (#5821)
- various bug fixes, including:
  - using the correct binutils in jemalloc 5.0.1 easyconfig using `GCCcore/6.4.0` (#5659)

- fix source URLs for MPC (#5662)
- add Tkinter as dependency for ASE & matplotlib using Python 3.6.3 (#5658)
- fix versions for updated extensions in Bioconductor 3.6 bundle (#5704, #5724, #5880)
- fix missing M4 build dependency in nettle easyconfigs (#5722)
- fix homepage for OpenFOAM 4.x & 5.x, should be `openfoam.org` (#5422, #5780)
- add matplotlib, cairo & PyCairo dependencies for graph-tool 2.26 + enhance sanity check (#5787)
- fix hardcoded version in scikit-image easyconfigs (#5822)
- fix pkgconfig moduleclass, '`data`' doesn't make much sense (#5817)
- add sanity check in recent matplotlib easyconfigs to ensure that Tkinter is available (#5689, #5896)
- correct MPI path when building ABINIT with '`foss`' toolchain (#5760)
- patch Hyperopt to make it work with networkx 2.0 (#5642)
- add Tkinter sanity check to ASE easyconfigs (#5691, #5909)
- include autotools as build dep in git easyconfigs (#5718)
- fix sanity check paths for Debian OS in JasPer easyconfigs (#5897)
- fix source_urls for recent versions of ABINIT (#5908)
- add patch for recent GDAL easyconfigs using `intel` toolchain to fix icc get stuck on compiling `ceos.c` (#5915)
- add explicit zlib dependency in Tkinter 2.x and 3.x (#5797, #5926)
- fix source URLs & homepage in Singularity easyconfigs (#5927)
- fix installation of TensorFlow 1.3 via binary wheel after introducing TensorFlow easyblock (#5938)

11.12.31 EasyBuild v3.5.1 (January 16th 2018)

bugfix/update release

framework

- various enhancements, including:
  - add definition of `giolfc` toolchain (#2359)
  - add support for Environment Modules 4 (#2365)
- various bug fixes, including:
  - install SQLAlchemy < 1.2.0 with Python 2.6 in Travis config (#2367)
  - make code in `easybuild/tools/job/gc3pie.py` forward-compatible with GC3Pie 2.5 (#2373)

easyblocks

- minor enhancements, including:
  - auto-detect default build target for Clang (#1115)
  - build GROMACS for target architecture based on `--optarch` (#1163)
  - ensure correct $PYTHONPATH for recent OpenBabel versions (#1219)
  - enhance Amber easyblock with support for OpenBLAS and better Intel MPI support (#1305)

  - also support only installing AmberTools through Amber easyblock (#1305)
  - also pick locations for CUPTI headers & libraries in CUDA easyblock (#1306)
  - update patching out of sanitizer tests for recent Clang versions (>=5.0) (#1327)
  - update known questions for Qt5 to support installing recent versions (#1328)
  - update BamTools easyblock for v2.5.0 (#1332, #1337)
  - disable `libfox` target if external module found in QuantumESPRESSO easyblock (#1333)
  - add support for linking Octave with multi-threaded BLAS/LAPACK library (#1340)
  - support `install_target` in PythonPackage + deprecate `use_easy_install` & `use_setup_py_develop` (#1341, #1342)
- various bug fixes, including:
  - make RubyGem easyblock use `$GEM_*` environment variables except if as extension of Ruby itself (#1247)
  - move initialisation in `SystemCompiler` & `SystemMPI` easyblocks to the prepare step (#1282)
  - enable skipping sanitizer tests by default in Clang easyblock, they can't be relied on (#1329)
  - fix quotes when using `$ORIGIN` in `RPATH` locations for DOLFIN (#1338)
  - fix sanity check for shared libraries in Trilinos easyblock (#1339)

easyconfigs

- added easyconfigs for `foss/2018a` and `intel/2018a` common toolchains (#5577), (#5578)
- added example easyconfig files for 26 new software packages:
  - BeautifulSoup (#5601), Calendrical (#5588), ChimPipe (#5560), crb-blast (#5124)), dammit (#5125), deepTools (#5536), FastQ_Screen (#5404), FoX (#5496), GffCompare (#5581), GlimmerHMM (#5559), LocARNA (#5548), MapSplice (#5566), MariaDB-connector-c (#5557), NextGenMap (#5430), nd2reader (#5545), PIMS (#5545), Pysolar (#5585), phono3py (#5551), preseq (#5569), proovread (#5513), QUAST (#5610), RNA-SeQC (#5589), RNAclust (#5607), Ragout (#5608), SOAPfuse (#5417), TransDecoder (#5125)
- added additional easyconfigs for various supported software packages, including:
  - BLAST+ 2.7.1, BamTools 2.5.0, Boost 1.66.0, Clang 5.0.0, dask 0.16.0, FFmpeg 3.4.1, GROMACS 2016.4, HDF5 1.8.20, matplotlib 2.1.1, PLUMED 2.4.0, Pillow 5.0.0, Qt5 5.9.3, QuantumESPRESSO 6.2, Ruby 2.5.0, Rust 1.22.1
- minor enhancements, including:
  - include `gomms` in list of extensions for R 3.4.3 (#5547)
  - clean up BamTools easyconfigs to rely on updated easyblock + add SHA256 checksums (#5575)
  - add `Time::HiRes` to recent Perl versions (#5616)
  - add `DNAcopy` & `dupRadar` extensions to bundle for Bioconductor 3.6 (#5587, #5618)
  - switch to using `install_target` rather than now deprecated `use_easy_install` and `use_setup_py_develop` (#5625)
- various bug fixes, including:
  - avoid auto-downloading of parcel in gdc-client 1.3.0 easyconfig (#5523)
  - fix permissions on `make_raw_alos.pl` script in ROI_PAC installation (#5546)

- remove erroneous patch for Intel compiler support in Perl 5.26.0 easyconfig built with `GCCcore/6.4.0` (#5561)
- include `HWxtest` as extension to fix issue with `diveRsity` in R 3.4.3 easyconfig file (#5570)
- add pkg-config as build dependency for fontconfig, harfbuzz, gnuplot, pango (#5580)
- fix versions of updated extensions in BioConductor bundle (#5587, #5618)
- add missing libpng dependency in ROOT 6.10.08 easyconfigs (#5595)
- fix option passed to configure in M4 (#5606)
- rename SIBELia to Sibelia (#5603)
- add patch for binutils 2.26 to fix compatibility with GCC 6.x (#5611)
- fix for dependencies was set twice in OpenMPI 3.0.0 easyconfig (#5619)
- fix download URL in comment of Kent tools easyconfigs (#5633)
- add SHA256 checksums to various easyconfigs (#5635, #5636, #5639)
- add `rdma-core-devel` to OS dependencies for OpenMPI 3.0.0 (#5648)

11.12.32 EasyBuild v3.5.0 (December 15th 2017)

feature release

framework

- add support for implementing pre- and post-step hooks (#2343)
  - documentation available at http://easybuild.readthedocs.io/en/latest/Hooks.html
- various enhancements, including:
  - add support for foss-like toolchain with Spectrum MPI: `gsolf` (#2329)
  - add support for `--preview-pr` (#2331, #2337, #2348)
    - see also http://easybuild.readthedocs.io/en/latest/Integration_with_GitHub.html#previewing-easyconfig-pull-requests-preview-pr
  - flesh out `find_extension` function, hoist dict with extract commands into a constant (#2336)
  - add support for using `self.start_dir` rather than using `self.cfg['start_dir']` (#2339)
  - add support for '`exts_default_options`' easyconfig parameter (#2345, #2346)
  - allow use of '`start_dir`' easyconfig parameter in extensions (#2353)
- various bug fixes, including:
  - fix typo in '`giolf`' toolchain definition (#2327)
  - fix minor issues with `--inject-checksums` (#2333)
  - fix error message when '`gv`' Python package is not available (#2340)
  - install paramiko version < 2.4.0 for Python 2.6 in Travis config (#2344)
  - disable broken log rotation, avoid duplicate logging of output of executed commands under '`--debug`' (#2347)
  - also include $ORIGIN and absolute paths to '`lib`' and '`lib64`' subdirectories in RPATH locations (#2358)

  - fix repo references in `install-EasyBuild-develop.sh` script (#2360)

easyblocks

- new software-specific easyblock for Bazel (#1286) and Octave (#1304)
- new generic easyblock for installing Octave packages as extensions (#1304, #1318)
- minor enhancements, including:
  - remove `foamExec` & `wdot` from sanity checks, add `blockMesh` & `checkMesh` + enable logging for recent OpenFOAM versions (#1205, #1272)
  - add exceptions for FFTW/3.3.6 on POWER with GCC 5/6/7 (#1274)
  - add support for Spectrum MPI to the `SystemMPI` easyblock (#1275)
  - allow skipping of steps for `Bundle` components + fix issues with templates & formatting of error message (#1278)
  - update HPCG easyblock for v3.0 by changing configure syntax (#1284)
  - correctly configure for BLAS/LAPACK in R easyblock & check configure output (#1292, #1300)
  - make R easyblock set configure options for dependencies (#1297, #1303)
  - allow tuning of build command in `PythonPackage` via custom '`buildcmd`' easyconfig parameter (#1299)
  - set default Java encoding to `UTF8` when installing Trinity (#1302)
  - also define `$CUDA_ROOT` in generated module for CUDA (#1234)
  - make the ScaLAPACK easyblock capable of building in parallel (#1288, #1321, #1324)
- various bug fixes, including:
  - avoid changing `$CPATH`, `$LD_LIBRARY_PATH` an `$LIBRARY_PATH` in generated modules for Intel Advisor, Inspector, and VTune (#1229)
  - fix check for Intel MKL in PSI easyblock (#1273)
  - fix missing space in fftw easyblock (#1277)
  - fix use of FFTW on top of Intel MKL in CP2K easyblock (#1281)
  - fix wrong sanity check for Boost when using Python 3.x (#1283)
  - pick up per-component checksums in `Bundle` generic easyblock (#1285)
  - correctly pass down optimization flags in CP2K easyblock (#1293)

easyconfigs

- added example easyconfig files for 39 new software packages:
  - AmberMini (#5476), arrow (#5416), bat (#5416), CIRCexplorer (#5356), CIRCexplorer2 (#5470), CIRI (#5358), FALCON (#5265), FastaIndex (#5465), find_circ (#5348), future (#5236), GapCloser (#5465), glibc (#5428), GRNBoost (#5373), HDFView (#5391), Horovod (#5239), HPDBSCAN (#5371), Hyperopt (#5455), IntelClusterChecker (#4970), ITK (#5434), LAST (#5465), MDTraj (#5317), Meson (#5228), mkl-dnn (#5319, #5362), Ninja (#5228), OpenKIM-API (#5479), ParmEd (#5476), PCRaster (#5386), piSvM (#5308), piSvM-JSC (#5316), PTESFinder (#5359), pybedtools (#5347), pymbar (#5476), pyScaf (#5465), QIIME2 (#5355), QTLtools (#5361), Redundans (#5465), Rmath (#5361), sbt (#5373), SNAP (#5465)
- added new easyconfigs for existing toolchains: `intel/2017.09` (#5303), `intel/2018.00` (#5129), `intel/2018.01` (#5345)

- added additional easyconfigs for various supported software packages, including:
  - CGAL 4.11, CMake 3.10.0, Caffe 1.0, FFTW 3.3.7, GATE 8.0, gnuplot 5.2.2, HPCG 3.0, HTSlib 1.6, Keras 2.1.2, LLVM 5.0.0, Mesa 17.2.5, mpi4py 3.0.0, netCDF 4.5.0, OpenFOAM 5.0, ParaView 5.4.1, R 3.4.3, Ruby 2.4.2, Rust 1.21.0, SCons 3.0.1, Szip 2.1.1, Theano 1.0.0, VTK 8.0.1, X11 20171023
- minor enhancements, including:
  - add SHA256 checksums to Autoconf (#5304), Automake (#5305), libtool (#5306) and M4 (#5307) easyconfigs
  - avoid hardcoding extension versions in worker easyconfig, include '`perl`' as OS dep (#5324)
  - enable building ScaLAPACK in parallel with enhanced ScaLAPACK easyblock (#5331)
  - include `RInside` extension in R 3.4.0 easyconfig (#5354)
  - clean up configure options that are now handled by R easyblock (#5478, #5485)
  - update Octave easyconfigs to use Octave easyblock, add extensions for latest Octave versions (#5484, #5503, #5507)
  - add `Log4perl` extension to Perl 5.26.0 easyconfigs (#5511)
- various bug fixes, including:
  - add missing extensions required by MultiQC & stick to networkx 1.11 (#5240)
  - disable `optarch` when using Intel compilers and enable tests in all libxc easyconfigs (#5256, #5257)
  - avoid downloads when installing matplotlib 2.1.0 w/ Python 2.7.14 (#5267)
  - fix `$CPATH` in recent libffi easyconfigs + enhance sanity check & add checksum (#5271)
  - use Github source URL, run `autogen.sh` + include Autotools build dep for numactl (#5286, #5296, #5297, #5299, #5302)
  - add hwloc to GROMACS dependencies (#5295)
  - fix `source_urls` (& `sources` spec) for GATE (#5367)
  - consistently add Autotools build dep in recent netCDF easyconfigs (#5394)
  - fix check in test suite for binutils build dep when `GCCcore` used as toolchain (#5436)
  - libdap 3.19.1 checksum changed (#5473)
  - disable new rfkill feature in easyconfig for util-linux 2.31 since it requires a recent kernel (#5480)
  - add missing PROJ dependency in recent GDAL easyconfigs (#5481)
  - fix name in TensorFlow easyconfigs (was 'TensorFlow') (#5495)
  - fix checksum for Szip 2.1.1 (#5517)
  - fix order of extensions for FSLeyes, add missing MarkupSafe (dep for Jinja2) (#5520)

11.12.33 EasyBuild v3.4.1 (October 17th 2017)

bugfix/update release

framework

- various enhancements, including:
  - improve trace output for executed commands + drop requirement for `--experimental` for `--trace` (#2306)

  - add `giolf` toolchain definition: GCC, IntelMPI, OpenBLAS, (Sca)LAPACK and FFTW (#2310)
  - add support for `--force-download` and `--ignore-checksums` (#2313, #2314)
  - flesh out `pypi_source_urls` from `derive_alt_pypi_url` (#2319)
- various bug fixes, including:
  - also check for use of `--rebuild` next to `--force` to skip sanity check with `--module-only` (#2307)
  - ensure `$TMPDIR` is set to a short path for OpenMPI v2.x (#2311)
  - guard '`module load`' commands in generated modules under `--recursive-unload` to avoid load storms (#2316)
  - correctly deal with use of special characters in description & co (#2320)
  - fix incorrect `FFT_INC_DIR` for Intel MKL (#2323)

easyblocks

- add generic '`SystemMPI`' easyblock (#1106, #1261, #1262)
- add software-specific easyblock for SAS (#1263)
- minor enhancements, including:
  - run '`wcleanAll`' or '`wcleanPlatform -all`' before building OpenFOAM (#780, #1258)
  - enhance generic '`SystemCompiler`' easyblock (#1106)
  - clean up `--trace` output for Python & Python packages (#1248)
  - update Intel MPI easyblock to support 2018.* versions (#1253)
  - add support for Intel MPI and Intel MKL to ScaLAPACK easyblock (#1255)
  - enhance GCC easyblock to also put symlinks in place for `cc`/`c++`/`f77`/`f95` commands (#1256)
- various bug fixes, including:
  - allow that '`gcc -print-multiarch`' fails in icc easyblock (#1249)
  - fix prefix subdirectory for older versions of icc (in particular 2011.3.174) (#1250)
  - use `remove_file` rather than `os.remove` in generic `IntelBase` easyblock to correctly deal with broken symlinks (#1251)
  - fix sanity check for MXNet easyblock + correctly detect unpacked source directory (#1257)
  - avoid building CP2K twice due to incorrect attempt at running '`make clean`' first (#1266)

easyconfigs

- added easyconfigs for new toolchain `giolf/2017b` (#5140)
- added example easyconfig files for 13 new software packages:
  - ASAP3 (#5200), Albacore (#5153), CatMAP (#5225), DLCpar (#5209), FSLeyes (#5192), IQ-TREE (#3695), NEST (#5515), nanonet (#5149), OMA (#5211), , oxford_asl (#5193), QEMU (#5220), REMORA (#5159), SAS (#5208), supermagic (#5187)
- added additional easyconfigs for various supported software packages, including:
  - Anaconda2/3 4.4.0, Blender 2.79, Boost 1.65.1, CMake 3.9.4, FFmpeg 3.3.4, GCC 5.5.0, h5py 2.7.1, Keras 2.0.8, matplotlib 2.1.0, mympingpong 0.8.0, OpenCV 3.3.0, OpenFOAM-Extend 4.0, OpenMPI 2.1.2 + 3.0.0, Pillow 4.3.0, Python 2.7.14 + 3.6.3, SAMtools 1.6, scikit-image 0.13.0, scikit-learn 0.19.0, Tensorflow 1.3.0, vsc-mympirun 4.0.2

- minor enhancements, including:
  - add `xkeyboard-config` component in X11 bundle (#5066)
  - clean up use of `wcleanAll` in OpenFOAM-Extend easyconfigs, now handled by OpenFOAM easyblock (#5131)
  - also install `run_rcorrector.pl` with Rcorrector (#5135)
  - add SHA256 checksum to PyCUDA easyconfig (#5154)
  - fix/improve description in HDF5 easyconfigs (#5182)
  - include heatmap3 extension for R 3.4.0 (#5185)
  - add ComplexHeatmap to Bioconductor 3.5 bundle + dep pkgs in R 3.4.0 easyconfig (#5195)
- various bug fixes, including:
  - fix source URLs for AUGUSTUS 3.2.3 (#5119)
  - fix building Bison 2.5 on systems with recent glibc (#5130)
  - also consider `rdma-core-devel` as alternative to `libibverbs-devel` OS dependency (#5132)
  - consistently use empty toolchain version in icc & ifort easyconfigs to ensure that GCC(core) dep is loaded during installation (#5133, #5134)
  - add patches for Boost 1.64.0 to fix known issues (#5148)
  - remove PyCUDA easyconfig for `intel/2017a`, doesn't work due to incompatibility between GCC 6.3.0 & CUDA 8 (#5156)
  - add missing Perl extensions in easyconfig for Worker 1.6.7 (#5157)
  - fix checksums for packages that download from `github.com/x/y/archive` (#5162)
  - add missing libpng dependency to g2lib-1.4.0 easyconfig using intel-2017a (#5196)
  - fix source_urls for Szip 2.1 & include SHA256 checksum (#5206)
  - remove unneeded `--with-fft-incs` configure option for ABINIT (#5207)
  - disable optarch for libjpeg-turbo 1.5.1 built with `intel/2017a` (#5214)
  - update `$R_LIBS` in plotly easyconfig (#5215)
  - include NLopt as a dependency in R easyconfigs that include `nloptr` as extension (#5217)

11.12.34 EasyBuild v3.4.0 (September 10th 2017)

feature release

framework

- various enhancements, including:
  - add support for backing up modules via `--backup-modules` (#2134)
    - enabled automatically with `--module-only` and `--skip`
    - see also http://easybuild.readthedocs.io/en/latest/Backup_modules.html
  - add support for `--search-paths` to extend list of locations considered by `--search`/`-S` (#2255)
  - include `userInGroup` check in Lua modules when installation is group-restricted (#2274)
  - add experimental support for '`eb --trace`' (#2285)

    - see also http://easybuild.readthedocs.io/en/latest/Tracing_progress.html
  - add support for '`eb --inject-checksums`' (#2286, #2292, #2293)
    - see also http://easybuild.readthedocs.io/en/latest/Writing_easyconfig_files.html#adding-or-replacing-checksums-using-inject-checksums
  - add support for `append_paths` in module generator (#2294)
- various bug fixes, including:
  - strip provided GitHub token of spaces in `--install-github-token` (#2270)
  - remove '`provides`' line from `setup.py` (#2275)
  - pass down stdin in '`import`' check for extensions during sanity check (#2276)
  - make sure location to 'eb' installed during stage 1 is included in `$PATH` during stage 2 of bootstrap procedure (#2281)
  - make resolve_path robust against `None` path being provided (#2282)
  - ensure clean error message on easyconfig file parse failure (#2290)
  - fix regex to avoid sucking up additional lines prior to module file path in `modulefile_path` (#2291)
  - fix error message when `--use-ccache` is used but `ccache` is not available in `$PATH` (#2295)

easyblocks

- minor enhancements, including:
  - update Siesta easyblock for versions 4.0.1 and 4.1-b3 (#1218)
  - updates GAMESS-US easyblock for version 20170420R1 + move `ddikick.x` when `ddi_comm` is set to '`sockets`' (#1221)
  - update MRtrix easyblock for 3.0 & beyond + use `copy` function (#1230)
  - update ROOT easyblock to support recent versions that require using CMake, add sanity check, clean up/enhance `make_module*` (#1236)
  - enhance icc easyblock to include multipath include dir in `$CPATH` (#1237, #1242)
- various bug fixes, including:
  - use `plumed-patch` command rather than '`plumed patch`' in GROMACS easyblock (#1212)
  - remove '`provides`' line from `setup.py` (#1217)
  - fixed wrong use of `build_type` in `self.cfg` in WRF easyblock that resulted in an raised exception (#1220)
  - added a call to super `post_install_step` in CUDA easyblock (#1226)
  - fix `$MCRROOT` definition in generated module file under `--module-only` in MCR easyblock (#1228)
  - fix permissions for directories in SuiteSparse (#1238)
  - fix function signature for `fetch_extension_sources` in OCaml easyblock (#1240)

easyconfigs

- added easyconfigs for `foss/2017b` and `intel/2017b` common toolchains (#4768), (#4618)
- added new easyconfigs for existing toolchains:
  - `iomkl/2017b` (#5097)
- added example easyconfig files for 31 new software packages:

  - BAMM (#4650), BamM (#4650), bcl2fastq2 (#4902), CGNS (#5078), CLAPACK (#5096), CLISP (#4926, #4986), cadaver (#4873), destiny (#5009), GroopM (#4650), Lucene-Geo-Gazetteer (#5064), libffcall (#4850), libsigsegv (#4840), MERCKX (#5056), minimap2 (#4991), ncompress (#4852), OpenNLP (#5059, #5061), OpenRefine (#5058), PHAST (#5096), PYTHIA (#5083), ParallelIO (#5071), PnetCDF (#5071), plotly (#5010), QML (#5101), Quorum (#5095), Rcorrector (#5095), SCnorm (#5008), SOAPdenovo-Trans (#5095), Shannon (#5095), Tika (#5063), UNAFold (#4997), VERSE (#4843)

- added additional easyconfigs for various supported software packages, including:
  - Boost 1.65.0, binutils 2.29, GAMESS-US 20170420-R1, GCC(core) 7.2.0, gzip 1.8, HDF5 1.8.19, LLVM 4.0.1, MRtrix 3.0_RC2, Perl 5.26.0, ROOT 6.10.04, Spark 2.2.0
- minor enhancements, including:
  - add checksums to Perl 5.24.1 easyconfigs (#4973, #4992)
  - add additional extensions for R 3.4.0 and Bioconductor 3.5 bundle (#5007, #5028, #5029, #5062, #5079)
  - also define `$INCLUDEPATH` and `$LIBRARY_PATH` in Tesla-Deployment-Kit easyconfig (#5018)
  - add check to see whether binutils is included as build dep when GCCcore toolchain is used (#5084)
- various bug fixes, including:
  - added Rmpi patch file for R built with intel toolchains incl. impi 5.x (#4623)
  - use single-line description in `setup.py` (#4881)
  - fix version and source for Seurat + add extensions required by Seurat in R 3.4.0 easyconfig (#4889)
  - add zlib as dependency to util-linux easyconfigs (#4900, #4998)
  - use `modextrapaths` instead of `modextravars` in OpenMM easyconfig (#4903)
  - update deprecated PLINK urls (#4920, #5006)
  - fix `moduleclass` for Cookiecutter (#4947)
  - fix order of OpenMPI dependency in `iomkl/2016.09*` easyconfigs, must come after icc/ifort (#5024)
  - fix typo in comment in util-linux easyconfigs & add SHA256 checksums (#5052)
  - remove superfluous `$CPATH` update in GLib 2.44.0 easyconfig (#5053)
  - update ncurses to not build a separate libtinfo but provide a soft link instead + force linking to ncurses in libreadline (#5067, #5074)
  - include Autotools as build dep for netCDF (#5077)
  - add missing binutils build dep for texinfo (#5099)

11.12.35 EasyBuild v3.3.1 (July 12th 2017)

bugfix/update release

framework

- various enhancements, including:
  - add support for '`allow_prepend_abs_path`' easyconfig parameter (#2254)
  - support for `--merge-pr` (#2266)
- various bug fixes, including:
  - resolve symlinks to location of '`eb`' in `get_paths_for` (#2248)

  - fall back to checking location relative to 'eb' location in `find_eb_script` (#2249)
  - respect `--suffix-modules-path` value for user-specific module path extensions (#2250)
  - update EasyBuild bootstrap script to download `distribute` tarball from http://easybuilders.github.io/easybuild/files (#2256)
  - fix default target GitHub account/organisation for `--new-pr` & co + fix tests that got broken by migration to github.com/easybuilders (#2258)
  - fix checking for new easyconfigs in `copy_easyconfigs`, pick up specified commit message as PR title if none was provided (#2259)
  - get rid of references to `hpcugent` after move to github.com/easybuilders (#2261)
  - automatically enable `--ignore-osdeps` under `--new-pr` and `--update-pr` (#2262)

easyblocks

- minor enhancements, including:
  - enhance HDF5 easyblock: define `$HDF5_DIR` & include `-DMPICH_IGNORE_CXX_SEEK` in `$CXXFLAGS` (#1200)
  - consistently pass down (named) arguments in `prepare_step`, and check for it in the tests (#1202)
  - remove no longer supported `VersionIndependentPythonPackage` generic easyblock (#1202)
  - update ABAQUS easyblock for recent versions, incl. support for installing hotfixes (#1203)
  - get rid of references to 'hpcugent' organisation after move to github.com/easybuilders (#1206)
  - make Boost easyblock fully aware of `(pre)configopts`, `(pre)buildopts` and `(pre)installopts` (#1207)
  - drop check for 'ipython' in sanity check of Anaconda easyblock, to also support Miniconda (#1210)

easyconfigs

- added example easyconfig files for 6 new software packages:
  - FastME (#4811), geopy (#4821), Miniconda2 (#4841), ngmlr (#4818), OpenCoarrays (#4799), Seurat (#4832)
- added additional easyconfigs for various supported software packages, including:
  - ABAQUS 2017, GCC(core) 6.4.0, Keras 2.0.5, NCBI-Toolkit 18.0.0, numpy 1.13.0, Tensorflow 1.2.0
- minor enhancements, including:
  - add `SVG` and `Statistics::Basic` to recent Perl versions (#4796)
  - remove `buildopts` from HDF5 easyconfigs, taken care of by updated HDF5 easyblock now (#4779)
  - include `joblib` as extension in recent Python easyconfigs (#4805)
  - changed moduleclass in mpi4py to better reflect what it is and to not confuse HMNS (#4807)
  - get rid of references to 'hpcugent' organisation after move to github.com/easybuilders (#4815, #4837)
  - add `Rtsne` as extension to R 3.4.0 (#4831)
- various bug fixes, including:
  - use `PYPI_SOURCE` as source URL in Tensorflow easyconfigs (#4786)
  - fix `homepage` for skewer (#4791)
  - sync/fix `source_urls` & `homepage` in HDF5 easyconfigs (#4800)

  - fix `ubsan` error blocking build of GCCcore 6.1.0, 6.2.0, 6.3.0 with system GCC 7.1 (#4813)

11.12.36 EasyBuild v3.3.0 (June 26th 2017)

feature release

framework

- various enhancements, including:
  - clean up easyconfigs that are copied for inclusion in pull request (#2197, #2227)
  - use devel logging where relevant in `easybuild.tools.toolchain` (#2198)
  - check exit code for executed `module` commands (#2200)
  - also copy patches to installation directory & easyconfigs archive along with easyconfig file (#2202, #2241)
  - add support for SHA256 checksums (#2215)
    - also auto-detect whether provided checksum is MD5 or SHA256 based on length (if not checksum type is specified)
    - add support for `--enforce-checksums`, to require availability of checksums for sources/patches
    - see http://easybuild.readthedocs.io/en/latest/Writing_easyconfig_files.html#source-files-patches-and-checksums
  - add support for renaming sources on download (#2223)
    - also involves deprecating use of 2-tuple elements in list of sources, see http://easybuild.readthedocs.io/en/latest/Deprecated-functionality.html#depr-sources-2-element-tuple
  - add support for `--detect-loaded-modules` (#2228)
  - give extensions access to `module_generator` of parent (#2229)
  - pass down additional arguments to `copy_dir` down to `shutil.copytree` (#2230)
  - avoid reloading already loaded modules that extend `$MODULEPATH` (#2232)
- various bug fixes, including:
  - make sure test account & accompanying token is used in tests for `github.py` (#2220, #2224)
  - only use MPD for old versions of Intel MPI (<4.1) in `mpi_cmd_for` (#2221)
  - escape dots in package filename to ensure correct match in `derive_alt_pypi_url` (#2225)
  - fix `GNU_SOURCE` template by adding missing `/gnu/` (#2235)
  - catch exception `shutil.Error` in `copy_file` (#2239)
  - report full error and traceback on unhandled exception in test report (#2240)
  - fix `--set-default-module` flag (#2243)

easyblocks

- added easyblocks for MXNet (#1135), Tkinter(#1184)
- minor enhancements, including:
  - enhance sanity check for NCL (#1169, #1179)
  - enable building of shared FFTW libraries (#1180)

  - include update statements for `$CPATH` and `$*LIBRARY_PATH` in generated module in numpy easyblock (#1183)
  - stop using deprecated '`copytree`' function from `easybuild.tools.filetools` (#1185)
  - update SAMtools easyblock for v0.1.17 (#1189)
  - update MATLAB easyblock for 2016b & 2017a versions (adjust permissions and change dir) (#1182, #1197)
  - consider `$EB_*_LICENSE_SERVER(_PORT)` in MATLAB and ANSYS easyblocks (#1195)
  - add `omp_num_threads` custom parameter in CP2K easyblock to allow defining `$OMP_NUM_THREADS` during testing (#1196)
- various bug fixes, including:
  - fix Siesta easyblock to enable and verify parallel build (#1186)
  - fix bug in alias definition in impi easyblock for `mpigxx`, `mpiicpc` and `mpiifort` (#1192)

easyconfigs

- enable automatic style checks in easyconfig tests (#2506)
- added example easyconfig files for 28 new software packages:
  - ada (#4594), Aspera-CLI (#4635), AUGUSTUS (#4624), ada (#4594), Bio-SamTools (#4637), Bpipe (#4590), BUSCO (#4624), CNVnator (#4649), davix (#4755), EricScript (#4594), FUNWAVE-TVD (#4743), gmpy2 (#4609), gSOAP (#4755), libsndfile (#4628), lpsolve (#4264), LUMPY (#4682), MaSuRCA (#4706), modred (#4729), MXNet (#4765), NRGLjubljana (#4651), OrfM (#4703), Perl4-CoreLibs (#4670), prodigal (#4468), pydlpoly (#4756), SeqAn (#4603), sharutils (#4745), Spyder (#4627), Tkinter (#4620), VariantMetaCaller (#4632)
- added new easyconfigs for existing toolchains:
  - `gimkl/2017a` (#4646)
- added additional easyconfigs for various supported software packages, including:
  - ABINIT 8.2.2, BLAST+ 2.6.0, Bowtie2 2.3.2, FFmpeg 3.3.1, NCL 6.4.0, Rust 1.18.0, SAMtools 1.5, VTK 7.1.1
- minor enhancements, including:
  - use '`git diff --name-only`' and `$TRAVIS_COMMIT_RANGE` in Travis config to get list of changed files (#4606, #4619)
  - add tuneR, seewave, soundecology, vcfR extensions for R 3.4.0 (+ libsndfile as dep) (#4628, #4680, #4747)
  - also copy `README` for GapFiller (#4631)
  - enable inclusion of version symbol by using `--enable-ld-version-script` configure option for LibTIFF (#4639)
  - add SHA256 checksums for libpciaccess to discriminate from old tarballs that required running `autogen.sh` (#4688)
  - fix `NE_GLOBAL_DIR` path for ne by also specifying `PREFIX` in `buildopts` (#4698)
  - more (trivial) style fixes (#4761)
  - avoid use of `import` in ANSYS & MATLAB easyconfigs (#4762)
  - set `$OMP_NUM_THREADS` during CP2K test step via dedicated easyconfig parameter (#4763)
  - avoid use of '`import`' in BFAST easyconfigs, just strip of '`a`' from version (#4764)

  - add the docopt module to all Python 2017a easyconfigs (#4770)
  - stop using deprecated 2-element tuple format in sources, use equivalent dict format instead (#4774)
- various bug fixes, including:
  - fix typo in statsmodels source url (`/sources/` -> `/source/`) (#4612)
  - add location to `DotLib.pm` to `$PERL5LIB` for SSPACE_Basic (#4638)
  - add missing `/gnu/` in ftpmirror.gnu.org `source_urls`, or use `GNU_SOURCE` where possible (#4653)
  - consistently use `--with-harfbuzz=no` configure option in freetype easyconfigs (#4668)
  - use `--with-x=yes` in R easyconfigs that include X11 as a dependency (#4701)
  - remove '`4.0`' in tbb description (#4707)
  - add lib sanity check paths for Debian compatibility in nettle (#4722)
  - fix source URLs for HDF5 (#4749)

11.12.37 EasyBuild v3.2.1 (May 12th 2017)

bugfix/update release

framework

- various enhancements, including:
  - make hardcoded max ratio for failures in `adjust_permissions` configurable (#2213)
  - allow `https://` on direct download in sources (#2214)
- various bug fixes, including:
  - bump version bootstrap script to sync with latest update (#2208)
  - fix crash during module generation when '`%`' character is used in description (#2209)

easyblocks

- added easyblock for Siesta (#1105)
- minor enhancements, including:
  - enhance GROMACS easyblock to build with PLUMED support (#1121)
  - enhance NAMD easyblock: add OpenMP support, update for recent NAMD versions (2.12), fix compatibility with Tcl versions other than 8.5 (#1173)

easyconfigs

- added example easyconfig files for 12 new software packages:
  - AdapterRemoval (#4509), blasr_libcpp (#4566), canu (#4473), enchant (#4567), hunspell (#4567), memkind (#4544), NLTK (#4565), pbbam (#4566), pbdagcon (#4566), pyenchant (#4567), Siesta (#4562), xarray (#4556)
- added new easyconfigs for existing toolchains:
  - goolfc 2017.01 (#4577)
- added additional easyconfigs for various supported software packages, including:
  - HDF5 1.10.1, NAMD 2.12, OpenFOAM 4.1, pandas 0.20.1, ParaView 5.2.0, R 3.4.0, R-bundle-Bioconductor 3.5, Tensorflow 1.1.0

- minor enhancements, including:
  - update source URLs in libpciaccess easyconfigs (#3960)
  - enable use of double precision floating point in METIS 5.1.0 `foss/2016a` easyconfig (#4555)

11.12.38 EasyBuild v3.2.0 (May 5th 2017)

feature release

framework

- various enhancements, including:
  - add support for marking installed module file as new default version using `--set-default-module` (#2110)
  - additional easyconfig parameters for documentation: `docpaths`, `examples`, `site_contacts`, `upstream_contacts`, `usage` (#2113)
  - add support for `--allow-use-as-root-and-accept-consequences` (#2123)
  - enable extraction of patches from compressed files before applying them (#2128)
  - alphabetically sort functions and methods in `module_generator` module (#2132)
  - introduce function `ensure_iterable_license_specs` (#2157)
  - bump Travis config to use Lmod 7.4 (#2176)
  - implement `copy_dir` function in `filetools` (#2177)
  - clarify error message when no software-specific easyblock was found (#2178)
  - make `log.deprecated` more verbose by also printing deprecation warnings to stderr (#2179)
  - add `copy` function to `filetools` for easy copying of lists of files/directories (#2180)
  - add support for `--verify-easyconfig-filenames` (#2185)
  - add support for `--package-tool-options` (#2187)
  - take into account inline trailing comments in `fetch_parameters_from_easyconfig` (#2192)
  - add support for customising easyconfig parameters on a per-extension basis (#2194)
  - perform sanity check after stage 2 of EasyBuild bootstrap script, module file should be in place (#2199)
  - change order in which module commands are checked, consider Lmod first (#2201)
- various bug fixes, including:
  - fix use of compiler-specific `--optarch` value in combination with `--job` (#2183)
  - call `run_all_steps` in regtest mode rather than running steps individually (#2203)

easyblocks

- added easyblocks for Doris (#1154, #1161), VMD (#1148) and WRF-Fire (#1153, #1159)
- minor enhancements, including:
  - update `IntelBase`, PGI and TotalView easyblocks to allow list of license files/servers via '`license_file`' easyconfig parameter (#1129)
  - update Bowtie2 easyblock to set correct build options, copy more files, extend sanity check (#1146)
  - added the option to build Boost with multi-threading support (#1147)

  - allow libpng as OS dependency for WPS (#1150)
  - extend Boost `TIME_UTC` patch to Boost versions <= 1.49.0 (#1152)
  - enhance Python sanity check to check for Tkinter support if Tk is included as a dependency (#1156, #1158)
  - add support to install Python extensions without unpacking (#1166)
  - enhance TBB easyblock to also support building open source versions (#1168)
- various bug fixes, including:
  - update FFTW easyblock: `--enable-avx-128-fma` depends on the `fma4` CPU feature (AMD), not `fma` (#1142)
  - fix problems when `usempi` not defined in toolchain in NAMD easyblock (#1162)

easyconfigs

- added example easyconfig files for 34 new software packages:
  - ACTC (#4386), atomate (#4484), BreakDancer (#4455), bx-python (#4486), ClusterShell (#4432), custodian (#4484), DFTB+ (#4398), Doris (#4404), ED2 (#4402), FireWorks (#4484), GETORB (#4414), GapFiller (#4462), IPy (#4450), imbalanced-learn (#4373), ipyrad (#4507), libiconv (#4499), MultiQC (#3564), NLopt (#1750), Node-RED (#4542), PyCUDA (#4523), pymatgen-db (#4484), QuTiP (#4371), ROI_PAC (#4414), Rascaf (#4459), RepastHPC (#4395), rootpy (#4242), SSPACE_Basic (#4461), Sambamba (#4442), Spack (#4431), SpiceyPy (#4406), StaMPS (#4454), samblaster (#4435), VMD (#4391), WRF-Fire (#4403)
- added additional easyconfigs for various supported software packages, including:
  - BamTools 2.4.1, Boost 1.64.0, GCC 7.1.0, IPython 5.3.0, LLVM 4.0.0, Mesa 17.0.2, Octave 4.2.1, OpenMPI 2.1.0, PETSc 3.7.5, PGI 17.3, Perl 5.24.1, Python 2.7.13 + 3.6.1 (incl. numpy 1.12.1, scipy 0.19.0), R 3.3.3, SuiteSparse 4.5.5
- various enhancements, including:
  - sync Bowtie2 easyconfigs, consistently use Bowtie2 easyblock (#4380)
  - use `pl2` source tarball for FFTW 3.3.6 which already includes patch for `F03` interface (#4529)
- various bug fixes, including:
  - add missing XZ dep in easyconfig for libunwind 1.1 w/ `GCC/4.9.2`, sync `sanity_check_paths` across libunwind easyconfigs (#4369)
  - use '`use_fma4`' rather than deprecated '`use_fma`' easyconfig parameter in FFTW easyconfigs using `intel` toolchain (#4384)
  - fix `pkgconfig` patch for Qhull (#4451)
  - also use patch for METIS 5.1.0 to enable use of doubles in easyconfig for `foss/2016b` (#4467)
  - add dependency NLopt for R extension `nloptr` (#4481)
  - fix issue with `configparser` and `backports` namespace blocking inclusion of `nbconvert` with IPython (#4504)
  - change source_urls of pycrypto to encrypted `https://pypi.python.org/...` (#4505)
  - fix sources spec for HMMER 3.1b2 + minor style fixes + better sanity check (#4531)
  - also build IMB-IO in IMB 4.1 easyconfig using `foss/2016a` (#4539)

11.12.39 EasyBuild v3.1.2 (March 20th 2017)

bugfix/update release

framework

- fix broken packaging support by fixing `run_cmd` bug with `shell=False` (#2153)
- minor enhancements, including:
  - implement `change_dir` function in `filetools` module (#2155)
  - use `checker_state` in trailing whitespace style check + add dedicated test (#2160, #2169)
  - consider both `pycodestyle` and `pep8` Python modules in style checks (#2161)
  - make bootstrap script aware various modules-related `$EASYBUILD_*` environment variables (#2170)
- various bug fixes, including:
  - interpret statements that extend `$MODULEPATH` in `modpath_extensions_for` method (#2104)
    - also fixes inclusion of superfluous load statements in modules installed in user HMNS module tree (cfr. #2172)
  - take into account that `$PATH` or `$PYTHONPATH` may be empty when running tests (#2149)
  - handle duplicates in `--include-*` (#2151)
  - exclude dependencies of dependencies that extend `$MODULEPATH` (#2152)
  - add `ld.bfd` to RPATH wrappers (#2156)
  - fix `test_vsc_location`, `vsc.__file__` may not be available when `vsc` is installed as a namespace package (#2159)
  - fix reported problems with scripts & docs tests (#2164)
  - fix `--try-software-version` using non-greedy matching + lookahead assertion, add test for `tweak_one` (#2166)
  - avoid creating empty modulefile directories when using `modaltsoftname` (#2168)
  - fix `fftw_libs` for MKL without interface libraries (#2171)

easyblocks

- add easyblock for QScintilla (#1127)
- minor enhancements, including:
  - auto-disable quad precision on POWER and ARM in FFTW (#1102, #1138)
  - allow "download install" for PETSc (#1114)
  - modify Trinity sanity check for versions 2.3 and above (#1133)
  - make unpacking Python extensions optional (#1134)
  - update SAMtools easyblock for version 1.4 (#1139)
- various bug fixes, including:
  - reduce number of environment variables in `icc` and `ifort` modules (#1126, #1143)
  - also run tests with Tcl module syntax, Lua is the default in EasyBuild v3.x (#1128)
  - rename member variable to avoid conflict with member of base class in PGI (#1137)
  - don't set `$FPATH` environment variable in multiple easyblocks (#1140)

easyconfigs

- add patch to FFTW 3.3.6 easyconfigs to fix MPI F03 interface (#4334)
  - note that this warrants rebuilding FFTW that is a part of `foss/2017a`
- added example easyconfig files for 14 new software packages:
  - DBG2OLC (#4281), disambiguate (#4125), fqtrim (#4280), GFOLD (#4293), Kaiju (#4349), LSMS (#4335), L_RNA_scaffolder (#4282), PileOMeth (#4289), psycopg2 (#4319), QGIS (#4307, #4332), QJson (#4305), QScintilla (#4306, #4313), sketchmap (#4360), snaphu (#4362)
- added additional easyconfigs for various supported software packages, including GROMACS 2016.3, PGI 17.1, SAMtools 1.4
- various enhancements, including:
  - fix style in several easyconfigs (#4267-#4271, #4274, #4275, #4277, #4279, #4286-#4288, #4318)
- various bug fixes, including:
  - correctly set `$PYTHONPATH` in ROOT easyconfigs (#4239, #4331)
  - correct libjpeg turbo references in GDAL (#4276)
  - make sure that HDF5 provided via EasyBuild is used in BLASR easyconfigs (#4278)
  - fix issues with miRDeep2 installation (#4291, #4301, #4316)
  - also run tests with Tcl module syntax, Lua is the default in EasyBuild v3.x (#4315)
  - fix PostgreSQL homepage + minor style fixes (#4318)
  - detect use of '`$root`', which is not compatible with module files in Lua syntax (#4330)
  - fix `homepage`/`source_urls` for ViennaRNA (#4338)
  - pass down `$FFLAGS` via `FLAGS_OPT` in SWASH easyconfigs (#4341)
  - remove `include/GL/wglext.h` from Mesa sanity check (#4354)
  - rename `S.A.G.E.` to `SAGE`, can't have directories with trailing dot in Windows (#4368)

11.12.40 EasyBuild v3.1.1 (March 7th 2017)

bugfix/update release

framework

- minor enhancements, including:
  - print more useful error message when no compiler-specific `optarch` flag is defined (#1950)
  - add `ec` parameter to `expand_toolchain_load()` (#2103)
  - clarify unstable/closed PR warning message (#2129)
- various bug fixes, including:
  - ensure that `$EBEXTSLIST*` is also included in generated module under `--module-only` (#2112)
  - fix formatting issues in generated documentation for `--list-software` and `--avail-easyconfig-licenses` (#2121)
  - fix problem with backticks in description breaking '`fpm`' packaging command (#2124)
  - replace `--enable-new-dtags` with `--disable-new-dtags` instead of removing it in RPATH wrapper script (#2131)

  - only perform `is_short_modname_for` sanity check in `det_short_module_name` if `modaltsoftname` is available (#2138)
  - fix logic in `make_module_dep` w.r.t. excluding loads for toolchain & toolchain components (#2140)
  - skip `test_check_style` if `pep8` is not available (#2142)

easyblocks

- minor enhancements, including:
  - change the sanity check for MCR 2016b since the directory structure has changed (#1096)
  - update NWChem easyblock for version 6.6.x and to handle different versions of OpenMPI for older versions (#1104)
  - allow per-component `source_urls` with templating in `Bundle` easyblock (#1108)
  - add slib to `$LD_LIBRARY_PATH` for itac (#1112)
  - consider both `lib` and `lib64` in CGAL sanity check (#1113)
  - add support for installing Intel tools that do not require license at installation time (#1117)
    * required for Intel MPI and Intel MKL version 2017.2.174
  - remove `prefix_opt` as custom easyconfig parameter for Qt easyblock (#1120)
- various bug fixes, including:
  - use '`-prefix <path>`' rather than '`--prefix=<path>`' for `configure` in Qt (#1109)
  - fix indentation problem in PETSc easyblock (#1111)

easyconfigs

- added example easyconfig files for 16 new software packages:
  - Caffe (#3667), DIAMOND (#4107), fmt (#4131), googletest (#4132), igraph (#4172), MEGA (#4202), meRanTK (#4175), meshio (#4178), miRDeep2 (#4229, #4255), OOMPA (#4211), PBSuite (#4224, #4230), randfold (#4217), skewer (#4246), Smoldyn (#4110), SpiecEasi (#4215), stress (#4180)
- added additional easyconfigs for various supported software packages, including:
  - binutils 2.28, Cantera 2.3.0, CGAL 4.9, GMP 6.1.2, IPython 5.2.2, JasPer 2.0.10, NWChem 6.6, matplotlib 2.0.0, PCRE 8.40, Qt5 5.8.0, Vim 8.0, X11 bundle v20170129, VTK 7.1.0, Yade 2017.01a
- added new easyconfigs for existing toolchains:
  - `iomkl/2017a` (#4216), `intel/2017.02` (#4248)
- various enhancements, including:
  - fix style in several easyconfigs (#4174, #4176, #4190, #4233)
  - add sanity check command to Yade easyconfig to make sure that '`import yade`' works, include bzip2 as dep (#4193)
  - add `PDF::API2` extension to Perl 5.24.0 easyconfigs + sync `exts_list` (#4221)
- various bug fixes, including:
  - add Bison and gettext as build deps for X11 (#4111)
  - clean up dependencies in libdrm (#4113)
  - make sure Ghostscript picks up external libraries (#4118)
  - fix ippicv source download and library install for OpenCV v3.1.0 (#4126)

  - fix software name for OrthoMCL + modernise OrthoMCL easyconfigs (#4134, #4135)
  - get rid of backticks in gettext descriptions, causes problems when packaging with FPM (#4146)
  - remove duplicate sources specification in OpenMPI (#4150)
  - fix definition of `buildopts`/`installopts` in Cantera easyconfig (#4133, #4164, #4177)
  - use `http://` rather than `ftp://` source URLs in CFITSIO easyconfigs (#4167)
  - add patch for XZ 5.2.2 to include `5.1.2alpha` symbols required by '`rpm`' command on CentOS 7.x (#4179)
  - add patch for Boost v1.61-1.63 to fix problem with `make_array`/`array_wrapper` in Boost serialization library (#4192)
  - set `CMAKE_PREFIX_PATH` to ncurses install directory in CMake easyconfigs (#4196)
  - switch to `lowopt=True` for libxc v2.2.* and v3.* (#4199)
  - remove custom `sanity_check_paths`, since it's identical to that used by the R easyblock (#4200)
  - fix `version` (& `homepage`) in ea-utils easyconfigs (#4205)
  - remove `--with-threads` configure option in OpenMPI-2.* (#4213)
  - fix check for Szip library in configure script for netCDF 4.1.3 (#4226)
- fix `source_urls` in several easyconfigs, including:
  - bsoft, cutadapt, EMBOSS, GnuTLS, ImageMagick, LibTIFF, Mercurial, netCDF, netCDF-Fortran, pigz, ROOT and Subversion (#4227)

11.12.41 EasyBuild v3.1.0 (February 3rd 2017)

feature release

framework

- various enhancements, including:
  - ARM: GCC optimal/generic architecture compiler flags (#1974)
  - add support for `--check-style` to check style in easyconfig files (#1618, #2038)
  - add `HOME` and `USER` from env to available cfg file constants (#2063)
  - `--optarch` can now be specified on a toolchain basis (#2071)
  - implement `get_cpu_features` function in systemtools (#2074, #2078)
  - support use of `linalg` without MPI, add `iimkl` toolchain definition (#2082)
  - spoof HTTP request header with empty agent (#2083)
  - exclude dependencies of dependencies that extend $MODULEPATH in make_module_dep (#2091)
- various bug fixes, including:
  - make `fetch_github_token` more robust against `RuntimeError` from `keyring` (#2070)
  - POWER: Fix `--optarch=GENERIC` for GCC (#2073)
  - fix docstring in toolchain class (#2075)
  - skip test cases involving `.yeb` if `PyYAML` is not installed, silence test in options subsuite (#2081)
  - fix traceback with '`eb --check-github`' if `GitPython` is not installed (#2085)

  - fix regex for determining list of patched files in GitHub diff (#2088)
  - modify robot so that it only appends dependencies of tweaked easyconfigs (#2090)
  - escape metacharacters in paths passed to `re.compile` in `dry_run_set_dirs` (#2098)
  - fix broken error message in `get_toolchain_hierarchy` + dedicated test case (#2099)

easyblocks

- new easyblock for FFTW (#1083)
- various enhancements, including:
  - update sanity check for flex 2.6.3, no more `libfl_pic.a` library (#1077)
  - cleanup build before proceeding with full Boost (#1080)
  - update CP2K easyblock: copy data dir, support version 4.1, support ELPA, fix psmp build with foss toolchain (#996, #1020, #1043, #1084)
  - add sanity check support for OpenSSL 1.1 (#1087)
  - support the latest changes in Inspector 2017 (#1047)
  - update NEURON easyblock to support the lack of `hoc_ed` in 7.4 (#987)
  - add support for WPS 3.8 (#1079)
  - also consider `setuptools` in `EasyBuildMeta` easyblock (#1093)
- various bug fixes, including:
  - (correctly) define `$ROSETTA3_DB` in Rosetta easyblock (#1092)

easyconfigs

- added easyconfigs for `foss/2017a` and `intel/2017a` common toolchains (#3968, #3969)
- added example easyconfig files for 16 new software packages:
  - ack (#3983), cclib (#4065), ConnectomeWorkbench (#3411), GroIMP (#3994), hyperspy (#3991), I-TASSER (#1216), ImageJ (#4023, #4062), libconfig (#4051), libspatialindex (#4002), mahotas (#3990), Minia (#3949), muParser (#4007), NetLogo (#3941), QIIME (#3868), QwtPolar (#4019), Tensorflow (#4084, #4095)
- added additional easyconfigs for various supported software packages, including:
  - Boost 1.62.0 + 1.63.0, CP2K 4.1, GSL 2.3, PLUMED 2.3.0, Qt5 5.7.1, WRF 3.8, WPS 3.8, Yade 2016.06a, zlib 1.2.11
- various enhancements, including:
  - update FFTW 3.3.5 easyconfigs to use FFTW easyblock & enable running of tests (#3985)
  - add FME extensions (+ deps) in R 3.3.1 easyconfigs (#4063)
- various bug fixes, including:
  - add libxml2 dependency on HDF5 (#3759)
  - remove unnecessary dependency in libmatheval (#3988)
  - fix permissions on SWASH binaries (#4003)
  - add conda-forge channel to perl-app-cpanminus (#4012)
  - add missing deps (libpthread-stubs, libpciaccess) to libdrm 2.4.70 (#4032)
  - modloadmsg style fixes in multiple easyconfigs (#4035)

  - include X11 as dep for Molden (#4082)
  - remove incorrect definition for $ROSETTA3_DB, now (correctly) defined via Rosetta easyblock (#4083)
- fix source URLs for several easyconfigs, including:
  - arpack-ng 3.1.3 + 3.1.5 (#4050), ChIP-Seq 1.5-1 (#4050), Ghostscript 9.10, 9.14 + 9.16 (#4050), Git 1.7.12, 1.8.2 + 1.8.3.1 (#4050), HBase 1.0.2 (#4043), libevent 2.0.22 (#4037), libsodium 1.0.3 (#4046), lynx 2.8.7 (#4050), Maven 3.2.2 and 3.3.3 (#4039), MEME 4.8.0 (#4050), PCC 20131024 (#4044), S-Lang 2.3.0 (#4045), Spark 1.3.0 (#4041), splitRef 0.0.2 (#4040)

11.12.42 EasyBuild v3.0.2 (December 22nd 2016)

bugfix release

framework

- various bug fixes, including:
  - also skip dependencies of dependencies marked as external module in get_toolchain_hierarchy (#2042)
  - disable verbose setvar in modules.py (#2044)
  - force copying of easyconfigs in –new-pr/–update-pr, even when combined with -x (#2045)
  - fix verification of filename for easyconfigs used to resolve deps (#2051)
  - skip RPATH sanity check when toolchain did not use RPATH wrappers (#2052)
  - check whether file-like paths are readable before reading them in systemtools module (#2065)
- various small enhancements, including:
  - add ‘rpath’ toolchain option to selectively disable use of RPATH wrappers (#2047)

easyblocks

- various enhancements, including:
  - enhance DL_POLY_Classic easyblock to support building with Plumed support (REVIEW) (#829)
  - make the Allinea easyblock search for the templates in the easyconfig paths (#1025)
  - make FortranPythonPackage aware of (pre)buildopts (#1065)
  - update sanity check for Mono to support recent versions (#1069)
  - fix Eigen sanity check for latest version 3.3.1 (#1074)
- various bug fixes, including:
  - skip RPATH sanity check for binary installations (#1056)
  - pass CXXFLAGS and LDFLAGS to Boost bjam (#1064)
  - make pip ignore already installed versions of the package being installed (#1066)
  - don’t pass empty string as custom installopts for numpy in test_step (#1067)
  - make the Rosetta EasyBlock work in –module-only mode (#1073)

easyconfigs

- added example easyconfig files for 13 new software packages:
  - CryptoMiniSat (#3952), MATSim (#3902), Molcas (#2084), ne (#3376), psmc (#3910), PyCogent (#3897), PyNAST (#3897), RASPA2 (#3903, #3946), SimPEG (#3876), SolexaQA++ (#3892), taco (#3882), UCLUST (#3896), USPEX (#3767)

- added additional easyconfigs for various supported software packages, including:
  - Mono 4.6.2.7, PGI 16.10, ROOT 6.08.02
- various enhancements, including:
  - trivial style fixes (#3878, #3893, #3895)
- various bug fixes, including:
  - add X11 develop libs to ncview easyconfig (#3881)
  - fix source_urls in pkg-config easyconfigs (#3907)
  - install numpy/scipy as .egg to ensure shadowing of numpy/scipy in parent Python installation (#3921)
  - fix broken source URL + homepage for Infernal (#3928)
  - fix test that verifies dumped easyconfig, take into account that dumped dependencies may include hard-coded dependency (#3932)
  - include libGLU as dependency in freeglut easyconfigs with recent Mesa (#3936)
  - add patch for FreeSurfer to fix issue with MATLAB 2013 (#3954)

11.12.43 EasyBuild v3.0.1 (November 30th 2016)

bugfix release

framework

- important changes
  - always use Intel-specific MPI compiler wrappers (`mpiicc`, `mpiicpc`, `mpiifort`) for toolchains using both Intel compilers and Intel MPI (#2005)
- various small enhancements, including:
  - use `setvar` in `modules.py` to define environment variables (#2011)
  - include output of `sanity_check_commands` in the build log (#2020)
- various bug fixes, including:
  - fix testing of bootstrap script in Travis config (#2003)
  - use correct module syntax in bootstrap script if Lmod is not used (i.e. Tcl) (#2007)
  - fix packaging issue with non-Python scripts in easybuild/scripts (#2015)
    - fixes issue where RPATH wrapper template script (`rpath_wrapper_template.sh.in`) was not included in the v3.0.0 release
  - make tests more robust against running headless (#2016)
  - avoid rewrapping already wrapped compiler/linker command with RPATH wrapper script (#2022)
  - fix `log.error` traceback due to '`raise EasyBuildError`' involving a '`%s`' in error message (#2024)
  - make sure '`modules_tool`' attribute is also defined for extensions (#2026)
  - only dump easyconfig with modified deps due to `--minimal-toolchains` to 'reprod' subdir of install dir (#2028)
  - avoid appending '`-h'` to sanity check commands specified as a string (#2030)
  - fix bug in `list_software_rst`: always include version suffix regardless of value (#2032)

easyblocks

- various enhancements, including:
  - update SAMtools easyblock for recent versions (#1048)
- various bugfixes, including:
  - fix QuantumESPRESSO easyblock to handle gipaw correctly (#1041)
  - always specify name of serial Fortran compiler to ALADIN, it already knows to use MPI wrapper commands (#1050)

easyconfigs

- added example easyconfig files for 7 new software packages:
  - Cookiecutter (#3827), ETE (#3857), findhap (#3860), graphviz (Python bindings, #3826), LoFreq (#3856), PhyloBayes-MPI (#3859), XGBoost (#3849)
- added additional easyconfigs for various supported software packages
- various enhancements, including:
  - add `ipywidgets` and `widgetsnbextension` extensions to IPython 5.1.0 easyconfigs (#3818, #3823)
  - run dadi test suite as a sanity check command (#3858)
- various bug fixes, including:
  - fix incorrect descriptions for ifort (#3817)
  - fix `modulename` for Jinja2 and Pygments (#3823)
  - fix download URL in BLAST 2.2.26 easyconfig (#3861)

11.12.44 EasyBuild v3.0.0 (November 16th 2016)

feature release

framework

- backward-incompatible changes:
  - make robot always consider subtoolchains, even without `--minimal-toolchains` (but in reverse order) (#1973)
  - clean up behaviour that was deprecated for EasyBuild v3.0 (#1978)
  - change *default* config to use `Lmod`/`Lua` for modules tool/syntax, `GC3Pie` as job backend (#1985)
  - the minimal required version of Lmod was bumped to 5.8 (#1985)
- major new features:
  - (experimental) support for RPATH linking via `--rpath` (#1942)
    - see http://easybuild.readthedocs.org/en/latest/RPATH-support.html
  - add support for `--consider-archived-easyconfigs` (#1972)
    - see http://easybuild.readthedocs.org/en/latest/Archived-easyconfigs.html
  - stabilize `--new-pr` and `--update-pr` (#1979)
    - see http://easybuild.readthedocs.org/en/latest/Integration_with_GitHub.html

- various other small enhancements, including:
  - add support for 'devel' log level (#1815)
  - make remove_file aware of --extended-dry-run + add dedicated unit test (#1932)
  - add support for filtering out setting/updating particular environment variables from generated modules (#1943)
    - see --filter-env-vars
  - clean up output of EasyBuild bootstrap script & add version (#1944)
  - improved ARM platform/CPU detection (#1953)
  - use '0' as letter dir for funky software names that don't start with a letter, e.g., 3to2 (#1954)
  - make bootstrap script aware of vsc-install for offline installation (#1955)
  - add support for blas_family() and lapack_family() methods in Toolchain instances (#1961)
  - make copy_file dry-run aware (#1963)
  - reorganise test easyconfigs to match structure in easyconfigs repo (#1970)
  - add a toolchain compiler option for enforcing IEEE-754 conformance (#1975)
  - support for intelcuda compiler toolchain (#1976)
  - check that each glob pattern matches at least one file expand_glob_paths (important for --include-*) (#1980)
  - enhance bootstrap to auto-skip stage 0 in case a suitable setuptools is already available (#1946, #1984)
  - simplify GC3Pie version check (#1987)
  - include suggestion on how to change configuration w.r.t. modules tool/syntax (#1989)
- various bug fixes, including:
  - fix test for find_easybuild_easyconfig (#1931)
  - remove existing module file under --force/--rebuild (#1933)
  - fix combining --search and --try-* (#1937)
  - fix appending to existing buildstats in FileRepository.add_easyconfig (#1948)
  - fix handling of iterate easyconfig parameters, restore them during cleanup (#1949)
  - fix filtering loads for (hidden) build deps from generated module (#1959)
  - handle multi-flag compiler options on all types of options (#1966)
  - fix --list-software by making letter_dir_for function aware of '*' wildcard name (#1969)
  - skip dependencies of toolchain marked as external modules when determining module hierarchy (#1977)
  - bump page limit in fetch_latest_commit_sha, spit out more meaningful error if too many branches were encountered (#1981)
  - fix CUDA-related issues in HierarchicalMNS (#1986)

easyblocks

- backwards incompatible changes:
  - remove deprecated GenomeAnalysisTK/GATK easyblock (#1001)
  - remove deprecated 'get_netcdf_module_set_cmds' function from netCDF easyblock (#1015)

  - remove deprecated 'get_blas_lib' function from LAPACK easyblock (#1016)
  - remove QLogicMPI easyblock (#1023)
- new easyblock for installing Anaconda (#950)
- add generic easyblock for Conda installations (#950)
- various enhancements, including:
  - enable use of GCCcore as toolchain for Clang, fail if no GCC prefix is found (#1002)
  - also build Boost MPI library in parallel (#1005, #1038)
  - enhance g2clib easyblock to allow to install 1.6.0 or higher (#1006)
  - update QuantumESPRESSO easyblock to support packaging changes in 6.0 (#1007)
  - add support to Scons generic easyblock to provide argument to specify installation prefix (#1008)
  - update IntelBase and imkl easyblocks to handle the 2017 versions of compilers/imkl (#1012)
  - leverage toolchain.linalg functionality in ScaLAPACK easyblock, use copy_file (#1014)
  - allow netCDF-C++4 to be used with ESMF (#1019)
  - update Advisor easyblock to support latest versions (#1021)
  - update CBLAS easyblock to build with foss toolchain (#1024)
  - update Gurobi easyblock to use copy_file (#1028)
  - add support for giving /lib preference over /lib64 & co in GCC installation (#1030, #1035)
  - enable installation of libiberty by default for binutils (#1030)
  - avoid CMake fiddling with the RPATHs injected by EasyBuild via --rpath in CMakeMake and METIS easyblocks (#1031, #1034)
  - simplify scipy sanity check to make it more robust w.r.t. version updates (#1037)
- various bug fixes, including:
  - make sure 'None' doesn't appear in modules generated with --module-only (#998)
  - fix ATLAS easyblock for non-x86 systems (#1003)
  - fix 'usempi' and 'with_mpi' usage to allow for a serial build of Amber 16 (#1013)
  - add both lib/python2.7/site-packages/{,wx-3.0-gtk2} to $PYTHONPATH for wxPython (#1018)
  - only hard inject RPATH for /usr/lib* directories when building binutils with dummy toolchain (#1026)
  - make HDF5 easyblock handle --filter-deps correctly (#1027)
  - update Travis config w.r.t. changes framework config defaults and required Lmod version (#1029)
  - be more patient when running Mathematica Q&A installer (#1036)

easyconfigs

- backwards incompatible changes:
  - archive easyconfigs using old inactive toolchains
    - see #3725, #3728, #3729, #3730, #3731, #3732, #3733, #3735, #3736, #3737, #3738
    - only taken into account by EasyBuild if --consider-archived-easyconfigs is enabled
    - no easyconfigs available outside of archive for QLogicMPI + 15 toolchains:

        - `ClangGCC`, `cgmpich`, `cgmpolf`, `cgmvapich2`, `cgmvolf`, `cgompi`, `cgoolf`, `gmacml`, `goalf`, `gpsmpi`, `gpsolf`, `iiqmpi`, `intel-para`, `ipsmpi`, `iqacml`
    - fix name in PyTables easyconfigs (was 'pyTables') (#3785)
- added example easyconfig files for 32 new software packages:
    - 3to2 (#3655), Anaconda2 (#3337), Anaconda3 (#3337), ART (#3724), atools (#3631), awscli (#3645), behave (#3751), Blosc (#3785), bokeh (#3790), Cantera (#3655), Cargo (#3764), dadi v1.7.0, distributed (#3786), ea-utils (#3634), Elk (#3644), FGSL (#3638), gencore_variant_detection (#3337), help2man (#3768), lbzip2 (#3791), Log-Log4perl (#3574), Minimac2 (#3783), mypy (#3694), OBITools (#3573), perl-app-cpanminus (#3337), PGDSpider (#3625), prokka (#3755), Reads2snp (#3609), spglib-python (#3620), SUNDIALS (#3654, #3655), SelEstim (#3626), XMLStarlet (#3797), x265 (#3090)
- added easyconfigs for new '`intelcuda`' toolchain (#3750)
- added new easyconfigs for existing toolchains:
    - `goolfc/2016.08` (#3796), `goolfc/2016.10` (#3666, #3775), `intel/2017.00` (#3543), `intel/2017.01` (#3757), `iomkl/2016.09-GCC-4.9.3-2.25` (#3680), `iomkl/2016.09-GCC-5.4.0-2.26` (#3772)
- added additional easyconfigs for various supported software packages, including:
    - Advisor 2017 update 1, Amber 16, ATLAS 3.10.2, GROMACS 2016, Octave 4.0.3, OpenFOAM 3.0.1, PyTables 3.3.0, QuantumESPRESSO 6.0, Rust v1.12.1
- various other enhancements, including:
    - STREAM builds using ~56GiB and ~111GiB (#3670)
- various bug fixes, including:
    - fix source spec in VASP easyconfig, ensure static linking with Intel MKL (#3381)
    - fix source URL in GCCcore 6.2.0 easyconfig (#3608)
    - correct STAMP dependency in i-cisTarget, must be 1.3 (#3613)
    - consistently specify to use `-fgnu89-inline` flag in M4 1.4.17 easyconfigs (#3623)
    - fix source URLs for Cython (#3636)
    - add Bison as build dep and M4 as runtime dep for flex 2.6.0 (#3656)
    - enable parallel building of flex 2.6.0 (#3630)
    - add zlib and bzip2 dependencies to X11 bundle (#3662)
    - use '`letter_dir_for`' function rather than just grabbing 1st letter of software name in easyconfigs tests (#3664)
    - add patch to fix typo in GRIT 2.0.5 (#3675)
    - fix typo in patch for WRF 3.8.0 (#3702)
    - use `$CC`, `$CXX` rather than `$I_MPI_CC`, `$I_MPI_CXX` in patch for OpenFOAM 4.0 (#3703)
    - patch FLTK to fix '`undefined symbol`' issue when building Octave (#3704)
    - include Pillow as a proper dep for scikit-image rather than as extension, since it has deps itself (#3723)
    - update Travis config w.r.t. changes framework config defaults and required Lmod version (#3773)
    - don't limit parallellism to 4 in recent GCC easyconfigs (#3776, #3777, #3778)
    - include M4 as dependency in flex 2.5.39 easyconfigs + fix consistency issues (#3782)

- consistently apply patch for ncurses 6.0 (#3792)
- eliminate dependency on `mpi-mic-rt` in ifort (#3793)
- include Autotools as build dependency in all beagle-lib and MrBayes easyconfigs (#3794)
- make OpenBLAS use the LAPACK version specified in the easyconfig (v0.2.18 & v0.2.19) (#3795)
- include original download URL for ISL source tarball in GCC easyconfigs (#3798)
- disable installing `libiberty` for binutils built with intel toolchain (#3802)

11.12.45 EasyBuild v2.9.0 (September 23rd 2016)

feature release

framework

- note: vsc-base 2.5.4 or more recent is now required
- various small enhancements, including:
  - change option `--color` choices to auto/always/never (#1701, #1898, #1911)
  - add support for 'hidden' easyconfig parameter (#1837)
  - add support for using `ccache` and `f90cache` compiler caching tools (#1844, #1912)
    - see `--use-ccache` and `--use-f90cache`
  - update Cray metadata for 16.06 CrayPE release (#1851)
  - also include patch files in `--new-pr` and `--update-pr` (#1852)
  - handle deleted files in `--new-pr` (#1853)
  - add support for `--install-latest-eb-release` (#1861)
  - add support for hiding toolchains, see `--hide-toolchains` and 'hidden' key in 'toolchain' spec (#1871)
  - add template for GitHub source URL (#1872)
  - add support for combining `--new-pr`/`--update-pr` and `--robot` (#1881)
  - add support for `--list-software` and `--list-installed-software` (#1883, #1910, #1917)
  - print message on which extension is being installed, incl. progress counter (#1886, #1914)
  - add support for `--github-org` to specify GitHub organisation rather than GitHub user (#1894)
  - add support for running Lmod in debug mode (#1895)
  - avoid needless use of deepcopy, speed up support for templating in easyconfigs (#1897)
  - convert `all_dependencies` to a property in `EasyConfig` class (#1909)
  - add support for `--mpi-cmd-template` (#1918)
  - add support for `--disable-mpi-tests` (#1920)
- various bug fixes, including:
  - merge with develop when using `--from-pr` (#1838, #1867)
  - ensure `--new-pr` doesn't open empty pull requests (#1846)
  - better error handling for outdated `GitPython` module in `--check-github` (#1847)

- fix formatting for generated easyblocks documentation (#1860)
- make sure the robot ignores filtered dependencies when creating toolchain cache (#1862)
- honor `--filter-deps` under `--minimal-toolchains` (#1863)
- correct format for '`param`' and '`author`' tags in docstrings (#1866)
- ignore failing bootstrap test in Travis config file (#1870)
- make sure all output of executed command is included in generated temporary log file (#1873, #1874)
- ensure `--show_hidden` is used in the correct location for '`avail`' with Lmod (#1875)
- make sure `self.path` is passed down in copy method of `EasyConfig` object (#1884)
- take into account possible multi-line modloadmsg in `ModuleGeneratorLua` (#1885)
- fix extracting `.bz2` source files (#1889)
- don't resolve path to Lmod command (#1892)
- fix skipping of stage 0 in bootstrap script (#1893)
- fix function signature of `log.deprecated` compared to `fancylogger.deprecated` (#1896, #1899)
- apply patch to Tcl/C environment modules tool for Tcl 8.6 support in Travis config (#1901)
- fix combining `--extended-dry-run` with `--from-pr` (#1902)
- also template dict keys (#1904)
- don't pass '`--try-*`' command-line options to EB instance running within job script (#1908)
- add workaround for incorrectly passing command line arguments with `--job` (#1915)
- fix issues with `--module-only` (#1919, #1924, #1925)
  - correctly deal with specified start_dir
  - do not remove installation directory when build-in-installdir is enabled
- make sure '`which`' function returns path to a file (#1921)
- fix `:param:`, `:return:` tags in docstrings & add test for it (#1923)

easyblocks

- new easyblocks for 6 software packages that require customized support:
  - cppcheck (#983), HEALPix (#982), IMOD (#847), IronPython (#321), Mono (#321), MyMediaLite (#321)
- various enhancements, including:
  - extend OpenFoam-Extend sanity check for decomp libraries (#784)
  - enhance Java easyblock to support installing Java 6.x (#940)
  - make QuantumESPRESSO easyblock aware of multithreaded FFT (#954)
  - extend PSI easyblock to use PCMSolver and CheMPS2 (#967)
  - make Boost easyblock add definition for `$BOOST_ROOT` to generated module file (#976)
  - add support to Bundle easyblock to install list of components (#980)
  - enhance & clean up libxml2 easyblock to also enable installing without Python bindings (#984)
  - update Libint easyblock for Libint 2.1.x (#985)

  - update sanity check for OpenFOAM to support OpenFOAM 4.x (#986)
  - make easyblocks that run MPI tests aware of '`mpi_tests`' build option (#993)
- various bug fixes, including:
  - fix compatibility of OpenFOAM easyblock with `--module-only` (#784)
  - fix testing of `--module-only` compatibility for OpenFOAM and IMOD easyblocks (#784)
  - add '`include/libxml2`' to `$CPATH` in libxml2 easyblock (#981)
  - fix compatibility of IntelBase generic easyblock with `--module-only` (#994)
  - make sure correct config script is used for Tcl/Tk deps of R (#995)

easyconfigs

- added example easyconfig files for 88 new software packages:
  - ADMIXTURE (#3359), angsd (#3593), ASHS (#3429), AutoDock (#3465), AutoGrid (#3466), BayeScan (#2748, #3356), BayPass (#3451), Bazel (#3379), Blender (#3553, #3558), bwakit (#3567), BXH_XCEDE_TOOLS (#3410), CastXML (#3403), CHASE (#3304), configparser (#3368, #3424), configurable-http-proxy (#3380), cppcheck (#3508), CRPropa (#779), DicomBrowser (#3432), DMTCP (#3422), entrypoints (#3368, #3424), f90cache (#3570), fastPHASE (#3343), fastQValidator (#3192), FFindex (#1135), FragGeneScan (#1198), gdc-client (#3399), gflags (#3417), glog (#3417), GRIT (#3561), H5hut (#3431), HAPGEN2 (#3344), HEALPix (#779), IMOD (#1187, #3347), IronPython (#607), jhbuild (#3476), jupyterhub (#3380), Keras (#3581), khmer (#1158), LeadIT (#3345, #3599), LevelDB (#3417), libbitmask (#3481), libcpuset (#3481), LMDB (#3417), log4cplus (#1136), MACH (#3346), Mako (#3460), Maq (#3428), MetaGeneAnnotator (#3307), Metal (#3324), Mono (#607), MyMediaLite (#607), nco (#2575), nose-parameterized (#3579), OpenEXR (#3553), OpenImageIO (#3553), path.py (#3368, #3424), PCRE2 (#3325), pftoolsV3 (#3317), PHASE (#3385), PLAST (#3288), PLINKSEQ (#3402), POV-Ray (#3551), ProbABEL (#3108), prompt-toolkit (#3368, #3424), protobuf-python (#3563), PSORTb (#3317), py (#3403, #3482), pygccxml (#3403, #3482), pyGIMLi (#3403, #3482), pyplusplus (#3403, #3482), PyQt5 (#3533), Pyro4 (#3527), pytest (#3403, #3482), QCA (#3595), RDMC (#1137), S.A.G.E. (#3427), SDL2 (#3551), SHORE (#3531), SimVascular (#3555), SortMeRNA (#3326), SUMACLUST (#3316), SUMATRA (#3316), Text-CSV (#3323), Triangle (#3403), VEGAS (#3457), VirSorter (#3307), wcwidth (#3368, #3424), X11 (#3340)
- added new easyconfigs for existing toolchains:
  - CrayGNU + CrayIntel 2016.06 (#3377)
  - foss 2016.07 (#3517) + 2016.09 (#3523)
  - iomkl 2016.07 (#3458)
  - pomkl 2016.09 (#3516)
- added additional easyconfigs for various supported software packages, including:
  - FFTW 3.3.5, GCC 4.9.4 + 6.2.0, GROMACS 5.1.4, IPython 5.1.0, LLVM 3.9.0, Mesa 12.0.1, OpenCV 3.1.0, OpenFOAM 4.0, OpenMPI 2.0.1, ParaView 5.1.2, PGI 16.7, QuantumESPRESSO 5.4.0, Qt5 5.7.0, R-bundle-Bioconductor 3.3, VTK 7.0.0, Yade 2016.06a
- various enhancements, including:
  - adjust PSI4 easyconfigs for updated easyblock (#3312)
  - clean up libxml2 easyconfigs according to updated libxml2 easyblock (#3479, #3509)
  - significantly speed up verifying of dumped easyconfig by resorting to 'shallow' parsing (#3520)
  - include sanity checks for all MATIO config files (#3528)

  - remove `--with-tcl-config`/`--with-tk-config` from R easyconfig, already done in R easyblock (#3580)

- various bug fixes, including:
  - disable testing in all ParaView 4.4.0 easyconfigs, required download is too much of a PITA (#3178)
  - add SQLite as dep to GDAL 2.1.0 easyconfigs (#3342)
  - add zlib/SQLite/LibTIFF as dep to R 3.3.1 easyconfigs (#3342)
  - add bzip2 as a dependency of freetype (#3464)
  - specify correct MPI target in FDS easyconfigs (#3488)
  - add tcsh as OS dep in NAMD easyconfigs (#3491)
  - statically link ncurses/libreadline in Lua easyconfig with '`dummy`' toolchain (#3545)
  - add M4 as dep for flex 2.6.x (#3542, #3550)
  - add bzip2 and libxcb dependencies to FFmpeg 3.x easyconfigs (#3548)
  - make sure & check that Graphviz does not install Tcl bindings in Tcl install prefix (#3556)
  - add missing patches for extensions in Python 3.x easyconfigs (#3557)
  - add missing XZ dependency to libxml2 2.9.4 easyconfigs, change gettext dep of XZ to build-only dep (#3568)
  - enable running of tests for HPCG (#3578)
  - fix `buildopts` in tabix easyconfigs (#3584)

11.12.46 EasyBuild v2.8.2 (July 13th 2016)

bugfix release

framework

- various small enhancements, including:
  - add support for rst output for `--list-*` and `--avail-*` (#1339)
  - add support for '`eb --check-conflicts`' (#1747, #1807, #1833)
  - ensure nice error message when non-existing path is passed to `apply_regex_substitutions` (#1788)
  - add check for module output, empty stdout is a sign of trouble with Lmod (#1793)
  - add multi-threaded FFT to toolchain (#1802)
  - avoid special characters like '[', ']' in path to temporary directory (#1808)
  - add support for `--zip-logs` (#1820)
  - add support for `--extra-modules` (#1821)
  - add type conversion for 'checksums' and 'patches' parameter in .yeb easyconfigs (#1826, #1840)
  - add support for filtering tests by name (#1828)
  - add support for `--avail-toolchain-opts` (#1830, #1839)
  - use absolute path for robot and easyconfig files (#1834)
  - add backup URL for tarballs hosted on SourceForge in `install_eb_dep.sh` script (#1843)

- various bug fixes, including:
  - fix installation of Lua in `install_eb_dep.sh` script (#1789)
  - fix OpenMP flag for Cray compiler wrappers (#1794)
  - only reset `$MODULEPATH` before loading a module if environment was reset (#1795)
  - include vsc-install as dependency in `setup.py` (#1805)
  - cache `$PATH` & `$PYTHONPATH` in test setUp, restore them in tests where '`eb`' is used (#1806)
  - don't reset `$MODULEPATH` in stage 2 of bootstrap script, support forced installation during stage 2 (#1810)
  - fix issue with templates defined by deps being required while still parsing deps (#1812)
  - skip unneeded unuse/use commands on tail of `$MODULEPATH` in `check_module_path` (#1813)
  - fix auto-convert for all `*dependencies` params in `.yeb` easyconfigs, ensure version is a string (#1818)
  - fix `keyring` version in Travis config (#1819)
  - fix dumping of `.yeb` easyconfig files in easyconfigs archive (#1822)
  - fix format of supported easyconfig templates in help output (#1825)
  - stick to `pydot` 1.1.0 for Python 2.6 in Travis config (#1827)

easyblocks

- new easyblocks for 5 software packages that require customized support:
  - Amber (#958), Extrae (#955), Gurobi (#962), Paraver (#956), Tau (#887)
- various enhancements, including:
  - add support for building & installing old GROMACS versions (#569, #960)
  - add support for building Boost with Cray toolchain (#849)
  - libxsmm support for CP2K (#942)
  - pick up specified components for imkl (#943)
  - add support for building GROMACS with double precision (#946, #960)
  - add support for building GROMACS with CUDA support and using dynamic libraries using `Cray` toolchains (#951, #960)
  - also install vsc-install in `EasyBuildMeta` easyblock, if tarball is provided (#957)
  - enhance PSI easyblock to support PSI4 1.0 (#965)
- various bug fixes, including:
  - also install scripts with MRtrix 0.3.14 (#941)
  - enhance Qt easyblock to support Qt3 (#944)
  - create '`release`' symlink in MRtrix install dir (#947)
  - fix `make_installdir` in Inspector & VTune easyblocks (#952)
  - make `Binary` and `MakeCp` easyblocks aware of '`keepsymlinks`' (#959)
  - correctly define `$G4*` environment variables in Geant4 easyblock (#961, #970)
  - prepend tmp install path to `$PYTHONPATH` in numpy test step, move to build dir when removing '`numpy`' subdir (#963)
  - correct full path to ALADIN config file & patch it to use right Fortran compiler flags (#964)

  - ensure correct compiler command/flags are used for SAMtools (#966)

easyconfigs

- added example easyconfig files for 54 new software packages:
  - Amber (#3200), Bullet (#3175), CONTRAlign (#690), Cluster-Buster (#3191), damageproto (#3222, #3308), DCA++ (#3219), EIGENSOFT (#3147, #3163), Extrae (#507), fdstools (#3237), ffnet (#3273), GP2C (#3257), Gurobi (#3239), gc (#3202, #3261), gputools (#546), IMa2p (#3300), IOzone (#3253), i-cisTarget (#3191, #3194), icmake (#3243), io_lib (#3255), Kent_tools (#3191), libcmaes (#3256), libpsortb (#3259), libxsmm (#3099), MEGACC (#3263), MM-align (#1428), MOSAIK (#880), MView (#1345), MySQL-python (#3172, #3189), magma (#3219), mrFAST (#862), mrsFAST (#862), mysqlclient (#3172, #3232), NTL (#3183), PARI-GP (#3257), Paraver (#508), psutil (#3171, #3231), PSI4 (#3293), Qwt (#3157), RMBlast (#3142), STAMP (#3191), Seqmagick (#3264), splitRef (#946), TAU (#509), TRF (#3141), TVB (#3053, #3247, #3251), TVB-deps (#3053, #3247, #3251), tvb-data (#3053, #3247, #3251), tvb-framework (#3053, #3247, #3251), tvb-library (#3053, #3247, #3251), VampirTrace (#509), Voro++ (#3174), wheel (#3235), wxPropertyGrid (#508), xonsh (#3159)
- added easyconfigs for update of common toolchains: `foss/2016b` (#3271), `intel/2016b` (#3270)
- added new easyconfigs for existing toolchains: `CrayGNU/2016.03` & `CrayGNU/2016.04` (#3291), `foss/2016.06` (#3184), `intel/2016.03-GCC-5.4` (#3185)
- added additional easyconfigs for various supported software packages, including:
  - Boost 1.61.0, GCC 5.4.0, GROMACS 3.3.3, HDF5 1.8.17, netCDF 4.4.1, numpy 1.11.0, Perl 5.24.0, PETSc 3.7.2, Python 2.7.12, Python 3.5.2, Qt 3.3.8, R 3.3.1
- various enhancements, including:
  - use `check_conflicts` function in easyconfigs tests (#2981)
  - also include vsc-install in list of sources for recent EasyBuild easyconfigs, to support offline installation (#3203)
  - enable building of `libmysqld.*` in MariaDB easyconfigs (#3230)
  - add ALDEx2, phyloseq to bundles for Bioconductor 3.2 (#3211, #3241)
  - add biom, geepack, lubridate, pim to list of R 3.2.3 extensions (#3186, #3211, #3275)
- various bug fixes, including:
  - add patch for Boost 1.60.0 to fix bug resulting in `TypeError` (#3162)
  - add fftw dependency to CP2K 2.6.0 easyconfigs using CrayGNU (#3176)
  - fix location of `libelf.h`, only (also) installed as `include/libelf.h` is there's no `/usr/include/libelf.h` (#3201)
  - fix software name for Guile & GnuTLS (was 'guile' & 'gnutls') (#3207)
  - added missing space in Geant4 configopts to specify `-DGEANT4_INSTALL_DATA` (#3209)
  - fix Cython download URL in Python 2.7.11 easyconfigs (#3212)
  - add missing build deps for X stack in easyconfigs using `foss/2016a` or `intel/2016a` (#3222, #3308)
  - fix overruling of exts_list in Perl 5.22.2 easyconfig (#3224)
  - add missing dependency on GMP in R 3.2.3 easyconfigs (#3226)
  - don't hard specify toolchain for binutils build dep in likwid easyconfig, since it matches parent toolchain (#3240)
  - fix `homepage` & `source_urls` in HMMER easyconfigs (#3246)

- stick to `pydot` 1.1.0 for Python 2.6 in Travis config (#3282)
- add `python-dev(el)` to OS deps in GC3Pie easyconfigs (#3310)

11.12.47 EasyBuild v2.8.1 (May 30th 2016)

bugfix release

framework

- various bug fixes, including:
  - fix error message on missing module command in bootstrap script (#1772)
  - expand '~' in paths specified to `--include-*` (#1774)
  - break after deleting cache entry to avoid attempt to delete cache entry again (#1776)
  - avoid changing `$MODULEPATH` when prepending with symlink of path already at head of `$MODULEPATH` (#1777)
  - filter out duplicates in `find_flexlm_license` (#1779)
  - stick with GitPython < 2.0 with Py2.6 in Travis configuration (#1781)
  - don't use `LooseVersion` to define `version_major`/`version_minor` (#1783)

easyblocks

- various enhancements, including:
  - update MRtrix easyblock for version 0.3.14 (#932)
  - update Inspector easyblock for recent versions (#934)
  - update VTune easyblock for recent versions (#935)
  - add debug message to IntelBase easyblock w.r.t. switching to 'exist_lic' (#936)

easyconfigs

- added example easyconfig files for 13 new software packages:
  - drFAST (#906), git-lfs (#2478), grabix (#3127), JWM (#3007), libcroco (#3007), librsvg (#3007), MaCH (#3136), mayavi (#3106), OpenMM (#2762), Pysam (#3080), SeqPrep (#3097), vt (#3128), wkhtmltopdf (#3098)
- added new easyconfigs for existing toolchains: `intel/2016.03-GCC-4.9` (#3088)
- added additional easyconfigs for various supported software packages, including:
  - Boost 1.61.0, ESMF 7.0.0, Inspector 2016 update 3, IPython 4.2, netCDF-C++4 4.3.0, netCDF-Fortran 4.4.4, Perl 5.22.2, VTune 2016 update 3
- various bug fixes, including:
  - apply libreadline patch to fix bug triggering segmentation fault (#3086)

11.12.48 EasyBuild v2.8.0 (May 18th 2016)

feature + bugfix release

framework

- significant speedup improvements of EasyBuild itself, thanks to:

  - stop creating `ModulesTool` instances over and over again (#1735)
  - cache result of '`module avail`' calls (#1742)
- add support for using PGI as toolchain compiler (#1342, #1664, #1759, #1761, #1764)
  - incl. new toolchain definitions `pompi` and `pomkl` (#1724)
- add test configuration for Travis (#1733, #1737, #1743, #1767)
- various other enhancements, including:
  - add `get_total_memory()` function in `systemtools` module (#1623)
  - ignore `__init__.py` files in `--include-*` (#1704)
  - use `-fopenmp` rather than `-openmp` for Intel compilers, since `-openmp` is deprecated (#1718)
  - add modules to metadata for Cray modules (#1721)
  - make sure user write permissions are set after failed removal attempt of installation directory (#1722)
  - escape special characters in software name in `find_related_easyconfigs` (#1726)
  - add support for `CrayPGI` compiler toolchain (#1729)
  - ensure read permission to all installed files for everybody (unless other options specify otherwise) (#1731)
  - also consider `$LMOD_CMD` in bootstrap script (#1736)
  - translate PyPI download URL to alternate URL with a hash (#1749)
  - make `get_software_libdir` compatible with `-x` (#1750)
  - set `$LMOD_REDIRECT` to '`no`' when initialising Lmod (#1755)
  - add test for broken modules tool setup affecting '`module use`' (#1758)
- various bug fixes, including:
  - isolate '`options`' tests from easyblocks other than the ones included in the tests (#1699)
  - don't run '`module purge`' in tests, since EasyBuild may be made available through a module (#1702)
  - avoid rehandling `--include-*` options over and over again during `--show-config` (#1705)
  - remove useless `test_cwd` (#1706)
  - fix bootstrap script: make sure setuptools installed in stage0 is still available at end of stage1 (#1727)
  - forcibly create target branch in `--update-pr` (#1728)
  - remove check whether '`easybuild`' is being imported from dir that contains `easybuild/__init__.py` (#1730)
  - (re)install vsc-base during stage1 using `--always-copy` in bootstrap script, if needed (#1732)
  - use `os.path.realpath` in `test_wrong_modulepath` to avoid symlinked path breaking the test (#1740)
  - unset `$PYTHONPATH` in before tested bootstrapped EasyBuild module (#1743)
  - take into account that paths in modulepath may be symlinks in `test_module_caches` (#1745)
  - change to install dir rather than buildpath in sanity check of extension, latter may not exist (#1746, #1748)
  - only load modules using short module names (#1754)
  - (re)load modules for build deps in extensions_step (#1762)
  - fix `modpath_extensions_for method`: take into account modules in Lua syntax (#1766)

  - fix broken link to VSC website in license headers (#1768)

easyblocks

- add test configuration for Travis (#895, #897, #900, #926)
- new easyblocks for 4 software packages that require customized support:
  - binutils (#907), libQGLViewer (#890), SuperLU (#860), wxPython (#883)
- various other enhancements, including:
  - update SuiteSparse easyblock for version >= 4.5 (#863)
  - enhance imkl easyblock to install on top of PGI (#866, #916)
  - enable runtime logging of install cmd in `IntelBase` (#874)
  - enhance Qt easyblock to support installing with `dummy` toolchain (#881)
  - delete libnuma symbolic links in PGI installation directory (#888)
  - enhance PDT easyblock to support installing with `dummy` toolchain (#894)
  - add support for building Clang with OpenMP support (#898)
  - update Score-P easyblock for additional compilers, MPI libraries & dependencies (#889)
  - drop deprecated '`testrb`' from sanity check in Ruby easyblock (#901)
  - enhance WRF easyblock to support versions >= 3.7 (#902)
  - update QuantumESPRESSO easyblock for version 5.3.0 (#904)
  - add support in PythonPackage easyblock to use '`setup.py develop`' (#905)
  - update Qt easyblock for Qt 5.6.0 (#908)
  - extend bzip2 easyblock to also build dynamic libraries (#910)
  - make threading an explicit option rather than relying on MPI library in SCOTCH easyblock (#914)
  - update PGI easyblock to install siterc file so PGI picks up `$LIBRARY_PATH` (#919)
  - enhance sanity check paths for compiler commands in PGI easyblock (#919)
  - also filter out `-ldl` from $LIBBLAS & co for Intel MKL in numpy easyblock (#920)
  - define `$MIC_LD_LIBRARY_PATH` for impi (#925)
- various bug fixes, including:
  - don't hardcode Python version in `test_make_module_pythonpackage` (#876)
  - make PythonPackage easyblock compatible with `--module-only` (#884, #906)
  - remove check whether '`easybuild`' is being imported from dir that contains `easybuild/__init__.py` (#891)
  - fix passing compiler configure option in PDT easyblock (#894)
  - fix bug in Score-P easyblock w.r.t. `--with-libbfd` (#889)
  - fix extension filter for Ruby (#901)
  - fix `ACTIVATION_TYPES` list in IntelBase + minor style change (#913)
  - correctly define `$MIC_LD_LIBRARY_PATH` in imkl 11.3.x and newer (#915)
  - fix broken link to VSC website in license headers (#927)

easyconfigs

- added example easyconfig files for 69 new software packages:
  - ALPS (#2888), annovar (#3010), BayeScEnv (#2765), BayesAss (#2870), BerkeleyGW (#2925), Blitz++ (#2784, #3004), bam-readcount (#2850), Commet (#2938), CrossTalkZ (#2939), cuDNN (#2882), DBus (#2855), DFT-D3 (#2107), DIAL (#3056), dask (#2885), dbus-glib (#2855), FFLAS-FFPACK (#2793), FLAC (#2824), FLANN (#3015, #3029), FLEUR (#3043), GConf (#2855), GROMOS++ (#1297), GST-plugins-base (#2855), GStreamer (#2855), GTOOL (#2805), Givaro (#2793), gdist (#2935), gromosXX (#1297), HISAT2 (#2809), i-PI (#2940), Kraken (#3037, #3041), LAME (#2823), LASTZ (#3002), LinBox (#2793), Loki (#2839), libQGLViewer (#2923, #3008), libXxf86vm (#2855), MDSplus (#2787, #2838, #3027), MRIcron (#2831), Mawk (#2732), minieigen (#2839), mpmath (#3058), NBO (#3047, 3048), NGS (#2803), NGS-Python (#2810), ncbi-vdb (#2808), OptiX (#2795), PCL (#3024), PEAR (#2731), PLplot (#2990), Postgres-XL (#2891), PyGTS (#2969), RSeQC (#2788), Rust (#2920, #2943), rainbow (#2730), SHAPEIT (#2806), SIONlib (#2908), Saxon-HE (#2773), Singularity (#2901), SoX (#2825), Subread (#2790), SuperLU (#2665), travis (#2953), VASP (#2950), Wannier90 (#2906, #3042), wget (#3041), wxPython (#2855), xf86vidmodeproto (#2855), Yade (#2839), Yambo (#2932)
- add test configuration for Travis (#2942, #2944, #2954, #3061)
- added easyconfigs for new PGI-based toolchains
  - `pomkl/2016.03` (#2899, #2900, #3046), `pomkl/2016.04` (#3044), `CrayPGI/2016.04` (#2927)
- added new easyconfigs for existing toolchains:
  - `foss/2016.04` (#3013), `intel/2016.02-GCC-5.3` (#2523), `intel/2016.03-GCC-5.3` (#3009)
- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...
  - incl. CGAL 4.8, Clang 3.8.0, icc/ifort 2016.2.181 & 2016.3.210, imkl 11.3.2.181 & 11.3.3.210, impi 5.1.3.181, LLVM 3.8.0, OpenCV 2.4.12, pandas 0.18.0, Qt 5.6.0, Scalasca 2.3, Score-P 2.0.1, SuiteSparse 4.5.2, WRF 3.8
- various other enhancements, including:
  - enhance ORCA easyconfig for compatibility with SLURM (#1819)
  - enable `-fPIC` in GraphicsMagick easyconfig, required by Octave (#2764)
  - clean up binutils easyconfigs to use binutils easyblock (#3006)
  - add `include/GraphicsMagick` to `$CPATH` in GraphicsMagick easyconfigs (#3034)
  - update SuiteSparse easyconfigs according to updated SuiteSparse easyblock (#3050)
- various bug fixes, including:
  - fix Perl extensions download urls (#2738)
  - add Autoconf as build dep for `GCCcore` (#2772)
  - fix versions of extensions in Bioconductor 3.2 bundles (#2769)
  - fix (build) deps for `intel/2016a` easyconfigs of cairo, libXext, libXrender (#2785, #2874)
  - use '`env`' wherever preconfig/build/installopts is used to set environmental variables (#2807, #2811, #2812)
  - add zlib as explicit dep in Tk easyconfigs (#2815)
  - consistently specify to use `-fgnu89-inline` flag in M4 1.4.17 easyconfigs (#2774, #2779, #2816)
  - fix homepage and description in Pygments easyconfigs (#2822)

- include pkg-config as build dependencies for libXau, libXdmcp, libxcb (#2827)
- consistently use `XORG_*_SOURCE` constants (#2829, #2830, #2848)
- update source URLs in ScientificPython easyconfig files (#2847)
- add checksums in SuiteSparse easyconfigs (#2849)
- fix build deps for GObject-Introspection (#2852)
- correctly specify Perl location in git easyconfig (#2866)
- fix bitstring 3.1.3 download URL in Python easyconfigs, source tarball on PyPI disappeared (#2880)
- fix Perl dependency in worker easyconfigs, it requires non-standard Perl modules (#2884)
- add XZ as dependency in Python 3.5.1 easyconfigs, required for lzma (#2887)
- fix download URL for packmol (#2902)
- drop `usempi` toolchain in numexpr easyconfigs, not needed (#2937)
- fix use of `resolve_dependencies` in tests according to changes in framework (#2952)
- add dependency extensions for MarkupSafe and jsonscheme in IPython 3.2.3 easyconfigs (#2967)
- add patch for matplotlib 1.5.1 to fix Tcl/Tk library paths being used (#2971)
- add xproto build dependency for makedepend v1.0.5 (#2982)
- disable parallel build for Doxygen (#2986)
- fix source URLs for `FreezeThaw` and `Tie::Function` extensions for Perl v5.22.1 (#2988)
- add `sed` command in worker easyconfig files to fix module_path in conf/worker.conf (#2997, #3000)
- drop toolchainopts from Eigen easyconfigs, since it is headers-only (#3025)
- clean up dummy bzip2 easyconfig, define buildopts rather than defining `$CC` and `$CFLAGS` via `os.environ` (#3036)
- use `%(pyshortver)s` template rather than hardcoding 2.7 in VTK easyconfigs (#3052)
- correct install location of OpenCV Python bindings (#3054)
- include XZ as dependency for libunwind (#3055)
- add patch to fix broken OpenSSL tests due to expired certificates (#3057)
- fix broken link to VSC website in license headers (#3062)

11.12.49 EasyBuild v2.7.0 (March 20th 2016)

feature + bugfix release

framework

- stabilize Cray support
  - enable '`dynamic`' toolchain option by default for `Cray*` toolchains (#1581)
  - remove FFTW from the Cray toolchains definition (#1585)
  - add external modules metadata for Cray systems (#1638)
  - fix independency of Cray toolchains w.r.t. toolchain build environment (#1641, #1647)
  - remove requirement to use `--experimental` for Cray toolchains (#1663)

- enable Python optimization mode in 'eb' (#1357)
- improved GitHub integration
  - improve error handling on git commands + better logging for --new-pr/--update-pr (#1590)
  - use git rather than https in --new-pr/--update-pr (#1602)
  - add -u as shorthand for --upload-test-report (#1605)
  - fix --from-pr for PRs that include renamed/deleted files (#1615)
  - add support for --install-github-token and --check-github (#1616)
  - fix fetch_easyconfigs_from_pr w.r.t. duplicate files in PRs (#1628)
- various other enhancements, including:
  - add support for --search-filename and --terse (#1577)
  - support complete bash completion (#1580)
  - add support for %(*ver)s and %(*shortver)s templates (#1595, #1604)
    - incl. %(javaver)s, %(javashortver)s, %(perlver)s, %(perlshortver)s, %(pyver)s, %(pyshortver)s, %(rver)s, %(rshortver)s
  - define HOME constant that can be used in easyconfig files (#1607)
  - implement support for generating 'swap' statements in module files (#1609)
  - add support for --show-config (#1611, #1620)
  - simplified support for --minimal-toolchains (#1614, #1619, #1622, #1625, #1646)
  - add support for --dump-env-script (#1624)
  - enhance ModulesTool.exist to also recognize partial module names (#1630)
  - improve error message for toolchain definition errors (#1631)
  - make default is_short_modname_for check less strict to support versionless external modules as deps (#1632)
  - mention hostname in comment made by --upload-test-report (#1635)
  - support providing additional relative path for prefix in external module metadata (#1637)
  - add ThematicModuleNamingScheme (#1645)
  - enhance logging format: remove logger name, mention location instead (#1649, #1654)
  - update kernel versions for SLES12 (#1659)
  - raise EasyBuildError rather than ImportError in only_if_module_is_available decorator (#1662)
- various bug fixes, including:
  - fix Lmod spider output in generated modules (#1583)
  - correctly define 'easybuild' namespaces (#1593, #1666, #1680)
    - this change requires that the setuptools Python package is available (at runtime)
    - using custom easyblocks by adding them in the Python search path ($PYTHONPATH) may require adjustments, i.e. also using pkg_resources.declare_namespace in the __init__.py files; *we highly recommend to use* --include-easyblocks *instead*, see http://easybuild.readthedocs.org/en/latest/Including_additional_Python_modules.html

    - note: this has the side-effect of not being able anymore to reliably use 'eb' in the parent directory of the easybuild-framework repository (#1667)
  - fix template for savannah.gnu.org source URL (#1601)
  - stop running 'module purge', only restore environment (#1608)
  - fix license headers: Hercules foundation is now FWO (#1629)
  - avoid that fancylogger tries to import mpi4py to determine MPI rank (#1648)
  - fix error in tests when 'file' backend is not available in Python keyring (#1650)
  - update develop install script (#1651)
  - handle allowed system deps during prepare_step rather than during parsing of easyconfig (#1652)
  - add function to find FlexLM licenses: find_flexlm_license (#1633, #1653)
  - fix availability check for external modules with partial module name (#1634, #1643)
  - fix bootstrap script to ensure setuptools is also installed (#1655)
  - fix issue in bootstrap script with vsc-base being picked up from the OS (#1656)
  - fix bootstrap script for environment where 'python' is Python 3.x (#1660)
  - remove --experimental for tests related to --package (#1665)
  - ensure path to setuptools is included in $PYTHONPATH being used to test scripts (#1671)
  - sanitize environment before initializing easyblocks (#1676)
  - remove reload statements in include.py, since they are not required and break --include-toolchains (#1679)

easyblocks

- new easyblocks for 6 software packages that require customized support:
  - ADF (#826), MPICH (#844, #852, #868), mutil (#859), pplacer (#835), psmpi (#852), SNPhylo (#865)
- various other enhancements, including:
  - implement support for 'use_pip' in PythonPackage easyblock (#719, #831)
  - add support in CUDA easyblock to install wrappers for host compilers (#758)
  - update sanity check for picard version 1.124 and above (#796)
  - use 'module swap' for all components in CrayToolchain (#823)
  - update PSI4 easyblock to cope with changed name of PSI4 data dir (#824)
  - use find_flexlm_license function and avoid defining $CPATH in PGI easyblock (#837)
  - use find_flexlm_license function in IntelBase generic easyblock (#839)
  - add unit test to check module file generated by PythonPackage easyblock (#841)
  - rework MVAPICH2 easyblock on top of new MPICH easyblock (#844)
  - add CUDA support in CP2K easyblock (#850)
  - also define $LD in buildopts for GATE (#855)
  - use find_flexlm_license function in TotalView easyblock (#839)
  - enhance MakeCp easyblock to also support renaming of files while copying them (#859)
  - hunt for usable 'python' command in PythonPackage easyblock when system Python is used (#861)

  - add sanity check in `easybuild/__init__.py` w.r.t. current working dir (#869)
  - change suffix of original file to `.easybuild` when using `fileinput` in impi easyblock (#870)
- various bug fixes, including:
  - make sure Python unicode settings match that of the system Python (#817)
  - remove FFTW related statements in HPL easyblock, since HPL doesn't require FFTW at all (#822)
  - use `pkg_resources.declare_namespace` rather than `pkgutil.extend_path` to declare '`easybuild`' namespaces (#827)
  - fix license headers: Hercules foundation is now FWO (#836)
  - fix check for non-empty lib dirs in `PythonPackage` easyblock (#840)
  - consider all Python lib dirs in sanity check for libxml2 (#842)
  - correctly handle deprecated configure options (`--with-hwloc`/`--enable-mpe`) in MVAPICH2 easyblock (#853)
  - use correct configure option for checkpoint/restart in MVAPICH2 easyblock (#854)
  - ensure list of Python lib dirs always has a '`lib/...`' entry (#858)
  - check whether `rpm`/`rpmrebuild` commands are available using '`which`', rather than checking for OS deps (#864)
  - fix `test_step` in UFC easyblock (#872)

easyconfigs

- added example easyconfig files for 63 new software packages:
  - ATSAS (#616, #2587), astropy (#2724, #2727), attr (#2706), BamUtil (#2654), BBMap (#2322), BH (#2508), CheMPS2 (#2445), CosmoloPy (#2723, #2727), csvkit (#2639), Firefox (#2648), FreeXL (#2422), GL2PS (#2667), Glade (#2631), htop (#2538), IGV (#2019), IGVTools (#2019), ImageMagick (#2438), jModelTest (#2529), KEALib (#2420), libcerf (#2656), libgcrypt (#2201), libglade (#2631), libgpg-error (#2201), libspatialite (#2431), LittleCMS (#2438), MAST (#2542), MLC (#2577), MPJ-Express (#2529), mutil (#2201), neon (#758), NextClip (#2544), npstat (#2686, #2703), Octopus (#2643), QuickFF (#2721), p4vasp (#2328), PCMSolver (#2445), PFFT (#2643), PHYLIP (#2694), pkgconfig (#2475, #2476), Platypus (#2618), pplacer (#1056), PRINSEQ (#2437, #2444, #2585), PyFFmpeg (#2501, #2519), PyGObject (#2443), PyGTK (#2443), PyOpenGL (#2628), pyringe (#2533), qrupdate (#2675), rgeos (#2635), rpmrebuild (#2402), shift (#2201), SNAPE-pooled (#2688), SNPhylo (#2701), sratoolkit (#2715), STAR-Fusion (#2463), statsmodels (#2719), StringTie (#2527), synchronicity (#2508), testpath (#2461), USEARCH (#2537), VarScan (#2464), vsc-install (#2165), Whoosh (#2725), xprop (#2645)
- added new easyconfigs for existing toolchains:
  - `intel/2016.02-GCC-4.9` (#2620), `gmpolf/2016a` & `gmvolf/2016a` (#2589)
- stable Cray-specific easyconfigs
  - delete deprecated Cray toolchains and easyconfig files (#2400)
  - don't hardcode `PrgEnv` version, remove `craype` and `fftw` components in Cray toolchains (#2554)
  - remove `-XC` versionsuffix for stable definitions for `Cray*` toolchains (#2714)
  - support for various software packages with `CrayGNU` and `CrayIntel` toolchains: CP2K, GROMACS, WRF
- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...

  - including BWA 0.7.13, CMake 3.4.3, GATE 7.2, GROMACS 5.1.2, Mesa 11.1.2, netCDF 4.4.0, Perl 5.22.1, Python 3.5.1, R 3.2.3, R-bundle-Bioconductor 3.2, scipy 0.17.0, SuiteSparse 4.5.1

- various other enhancements, including:
  - copy `contrib` dir in Velvet easyconfigs so scripts are also available (#2456)
  - redefine matplotlib 1.5.1 easyconfig as a bundle, also include `cycler` extension (dep for matplotlib) (#2470)
  - add bitstring extension to Python 2.7.11 easyconfigs (#2471)
  - enable building of MetaVelvet in Velvet 1.2.10 easyconfigs (#2473)
  - add custom sanity check for libjpeg-turbo (#2480)
  - add Velvet easyconfigs that include BioPerl dependency, so VelvetOptimizer can use it (#2495, #2729, #2733)
  - add source URL in RAxML 7.2.6 easyconfigs (#2536)
  - update MPICH easyconfigs to use new MPICH easyblock (#2589)
  - free libX11 & co from unneeded Python dependency/versionsuffix (#2549, #2563, #2605, #2664)
  - add '`--enable-utf --enable-unicode-properties`' configure options in PCRE easyconfigs (#2561) * required for latest R versions
  - add HCsnip, metagenomeSeq in Bioconductor 3.1 bundles (#2553, #2578)
  - add additional extensions in R 3.2.x easyconfigs that are required for extra Bioconductor extensions (#2547, #2556)
  - update psmpi easyconfig files to use the new psmpi easyblock (#2619)
  - add easyconfig for Python 2.7.11 on top of X11-enabled Tk (#2614, #2621)
  - add virtualenv as extension in Python 2.7.11 easyconfigs (#2660)

- various bug fixes, including:
  - fix software name for GTK+ (was 'gtk+'), PyCairo (was 'pycairo') and Gdk-Pixbuf (was 'gdk-pixbuf') (#2468)
  - don't hardcode `CC`/`CXX` in OpenMPI easyconfigs (#2472)
  - remove Google Code source URL for mpi4py (#2474)
  - rename ffmpeg to FFmpeg (#2425, #2481)
  - use available easyblock for flex (#2486)
  - fix determining list of easyconfigs in unit test suite, don't assume locations are correct (#2530)
  - fix specifying DB dependency in DB_File easyconfigs (#2539)
  - remove hard-coded `-xSSE4.2` for numpy/scipy with Intel compilers (#2546)
  - fix license headers: Hercules foundation is now FWO (#2550)
  - add `--with-zlib` configure argument in libxml easyconfigs (#2555)
  - don't hardcode `optarch=True` in xextproto/xtrans easyconfigs (#2601)
  - change toolchain version to '' in easyconfigs that use `dummy` toolchain and include dependencies (#2612)
  - GLib doesn't require libxml2 with Python bindings (#2632)

  - add patch file to imkl 10.2.6.038 32-bit easyconfig to fix installer not being able to deal with '`--`' in build path (#2634)
  - add missing 'pkgconfig' dependency for h5py (#2476, #2650)
  - correct software name in FastQC easyconfigs (was 'fastqc'), use '`dummy`' toolchain for all FastQC version (#2657, #2666)
  - add missing libxml2 dependencies in GLib easyconfigs (#2658)
  - fix Xerces-C++ download location (#2668)
  - enable `XML::Bare` extension in all Perl easyconfigs (#2672)
  - update dead link for SuiteSparse (#2679)
  - remove custom `exts_filter` in easyconfigs used `PythonPackage` easyblock (#2683, #2685)
  - add M4 as build dep for binutils & flex (#2681)
  - add missing dependencies in Python 3.5.x easyconfigs: SQLite, Tk, GMP (#2704)
  - fix (OS) deps, add checksums, remove parameter definition with default values in MVAPICH2 easyconfigs (#2707)
- style cleanup in various easyconfigs (#2378, #2387, #2395, #2396, #2488-#2493, #2496-#2500, #2502-#2504, #2602)
  - working towards automated style review of pull requests

11.12.50 EasyBuild v2.6.0 (January 26th 2016)

feature + bugfix release

framework

- add (experimental) support for opening/updating (easyconfigs) pull requests (`--new-pr`, `--update-pr`) (#1528)
- sanitize environment before each installation by undefining `$PYTHON*` (#1569, #1572)
- various other enhancements, including:
  - allow user-local modules with hierarchical naming schemes (`--subdir-user-modules`) (#1472)
  - enhance `--extended-dry-run` output to include paths for requirements in `make_module_req` (#1520)
  - rewrite read_file to use '`with`' (#1534)
  - add support for `eb --last-log` (#1541)
  - support using fixed install dir scheme (`--fixed-installdir-naming-scheme`) (#1546)
  - add edge attributes for build dependencies in `--dep-graph` output (#1548)
  - check whether dependencies marked as external module are hidden (#1552)
  - implement support for `--modules-header` (#1558)
  - add support to specify '`else`' body for conditional statements in modules (#1559)
  - add extra test for `--include-easyblocks` for generic easyblocks (#1562)
  - allow user to define the default compiler optimization level (`--default-opt-level`) (#1565)
  - make `toolchain.get_variable` more robust w.r.t. dummy toolchain (#1566)

- various bug fixes, including:
  - fix missing '`yaml`' module check in tests (#1525)
  - fix 'develop' install script (#1529)
  - correctly quote FPM option values in packagin support (#1530)
  - correctly handle '`.`' in software name w.r.t. `$EB*` environment variables (#1538)
  - exclude logs and test reports from packages (#1544)
  - also pass down `--job-cores` for `pbs_python` job backend (#1547)
  - skip dependencies marked as external modules when packaging (#1550)
  - fix syntax for `set_alias` statement in Lua syntax (#1554)
  - handle the case of all 'offline' nodes correctly for `--job` (#1560)
  - fix `test_modules_tool_stateless` unit test for stateless ModulesTool with Lmod as modules tool (#1570)

easyblocks

- add generic easyblock for Cray toolchains (#766)
- new easyblocks for 2 software packages that require customized support:
  - EggLib (#811), PGI (#658)
- various other enhancements, including:
  - update BamTools easyblock for versions 2.3.x and newer: some shared libraries are now static) (#785)
  - don't hardcode `.so`, use `get_shared_lib_ext` instead (#789, #790, #791, #793, #794, #803, #815)
  - enhance CPLEX easyblock by adding more subdirs to `$PATH`, define `$LD_LIBRARY` and `$CPLEXDIR` (#797)
  - make sanity check for netcdf4-python work with both egg and non-egg installs (#799)
  - update sanity check in PETSc/SLEPc easyblocks for v3.6.x (#800)
  - update Trinity easyblock for 2.x versions (#802)
  - update DOLFIN easyblock for v1.6.0 (#804)
  - check for `libkokkoscore.a` rather than `libkokkos.a` for Trilinos 12.x (#805)
  - add an option to skip the sanitizer tests of Clang (#806)
  - update Molpro easyblock to support binary installs and 2015 version (#807)
  - make `ConfigureMake` more robust w.r.t. custom easyconfig parameters (#810)
- various bug fixes, including:
  - add back support for Eigen 2.x in Eigen easyblock (#798)
  - fix for vsc-base being picked up from OS in EasyBuildMeta easyblock (#813)
  - remove `setuptools.pth` if it includes absolute paths after installing EasyBuild (#813)

easyconfigs

- add easyconfigs for `foss/2016a` and `intel/2016` common toolchains (#2310, #2311, #2339, #2363)
  - incl. easyconfigs for Boost, CMake, Python, Perl using these toolchains
- added example easyconfig files for 21 new software packages:

  - BLASR (#922), BioKanga (#2247), BoltzTraP (#2365), basemap (#2221), CppUnit (#2271), EggLib (#2335), FLASH (#2281), GLM (#2288), hub (#2249), MACS2 (#1983), MotEvo (#843), numba (#2243), PGI (#1833, #2367), PLY (#2305), PaStiX (#2319, #2326), patchelf (#2327), pip (#2284), RSEM (#2316), RcppArmadillo (#2289), SCDE (#2289), slepc4py (#2318)
- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...
  - including BamTools 2.4.0, Boost 1.60.0, Clang 3.7.1, DOLFIN/FFC/FIAT/Instant/UFL 1.6.0, GATE 7.0, GCC 5.3.0, LLVM 3.7.1, pandas 0.17.1, PETSc 3.6.3, SAMtools 1.3, scipy 0.16.1, SLEPc 3.6.2, Trilinos 12.4.2, Trinity 2.1.1, VTK 6.3.0
- various other enhancements, including:
  - added new `Cray*` toolchain versions with pinned dependency versions (#2222)
  - don't hardcode `.so`, use `SHLIB_EXT` constant instead (#2245)
  - add custom sanity check in GEOS easyconfigs (#2285)
- various bug fixes, including:
  - add Autotools (M4) as a build dependency in GMP v6.x easyconfigs (#2096)
  - remove argparse from list of extensions in Python 3.2+ easyconfigs, since it became part of stdlib (#2323)
- various style fixes, including:
  - get rid of tabs (#2302)
  - remove trailing whitespace (#2341)

11.12.51 EasyBuild v2.5.0 (December 17th 2015)

feature + bugfix release

framework

- add support for IBM XL compilers on Power7 and PowerPC (BlueGene) (#1470)
- add support for generic compilation using `--optarch=GENERIC` (#1471)
  - see also *Controlling compiler optimization flags*
- update experimental support for `.yeb` easyconfigs (#1515)
  - support clean way to specify toolchain + dependencies in `.yeb` easyconfigs
- various other enhancements, including:
  - add support for '`whatis`' easyconfig parameter (#1271)
  - add support for SLES 12 and kernel 3.12.x (#1412)
  - add GCCcore toolchain definition (#1451)
  - use '`diff --git`' lines to determine patched files in pull request with `--from-pr` (#1460)
  - add proper option parser to bootstrap script (#1468)
  - add `get_gcc_version()` function in systemtools module (#1496)
  - don't load fake module in sanity_check_step during a dry run (#1499)
  - allow string values to be passed in `make_module_req` by hoisting them into a list (#1502)
  - add support for listing build dependencies as hidden dependencies (#1503)
  - also consider `lib32/pkgconfig` and `lib64/pkgconfig` for `$PKG_CONFIG_PATH` (#1505)

  - add support to `make_module_dep` to specify module to unload before loading a dependency module (#1506)
  - add support to `make_module_extra` to specify alternative root/version for `$EBROOT/$EBVERSION` (#1508)
  - packaging support is no longer considered experimental (#1510)
- various bug fixes, including:
  - also consider `lib64` in sanity check performed during EasyBuild bootstrap (#1464)
  - also add description/homepage to packages created with FPM (#1469)
  - fix develop setup script to install EasyBuild-develop module in subdirectory (#1480)
  - don't create a whole set of temporary '`minimal-easyconfigs`' subdirs with `--minimal-toolchains` (#1484)
  - only keep polling if exit code is `None` in `run_cmd_qa`, to correctly deal with negative exit codes (#1486)
  - fix bootstrap script for missing sys_platform by using newer distribute 0.6.49 in stage 0 (#1490)
  - make sure that extra custom easyconfig parameters are known for extensions (#1498)
  - add missing import for `EasyBuildError` in `easybuild/toolchains/linalg/libsci.py` (#1512)
  - isolate tests from possible system-wide configuration files (#1513)
  - only use `glob` in `make_module_req` on non-empty strings (#1519) * this fixes the problem where `$CUDA_HOME` and `$CUDA_PATH` are not defined in module files for CUDA

easyblocks

- update easyblocks for Intel tools to support 2016 versions (#691, #745, #756, #777)
  - IntelBase easyblock has been enhanced to support specifying which components to install
- new easyblocks for 3 software packages that require customized support:
  - Intel Advisor (#767), DIRAC (#778), MRtrix (#772)
- various other enhancements, including:
  - update numpy and SuiteSparse easyblock to use scikit-umfpack (#718)
  - add an option to allow removal of the `-Dusethreads` flag in Perl easyblock (#724)
  - update Doxygen easyblock for 1.10.x (CMake) (#734)
  - update sanity check in Qt easyblock for Qt 5.x (#740)
  - add support for multilib build of GCC on PowerPC (#741)
  - add support to OpenFOAM and SCOTCH easyblocks to support 64-bit integers, via 'i8' toolchain option (#744)
  - fix sanity check to support numpy 1.10 (dropped _dotblas.so) (#757, #761, #762)
  - update IPP easyblock for v9.x (#759)
  - cleaner output for PythonPackage under dry run, make numpy easyblock dry-run aware (#760, #671)
  - add support for using netCDF-Fortran as dependency in ALADIN easyblock (#764)
  - add support for tbb 4.4.x in tbb easyblock (#769)
  - add support for specifying altroot/altversion in Bundle easyblock (#773)

  - update OpenFOAM easyblock for OpenFOAM-Extend 3.2 + use apply_regex_substitutions (#770)
- various bug fixes, including:
  - fix module path extension of system compiler in HMNS setup (#742)
  - only restore `$PYTHONPATH` if it was defined in EasyBuildMeta easyblock (#743)
  - make sure `$PYTHONPATH` is defined correctly in module file for Python packages created with `--module-only` (#748)
  - fix WRF easyblock to produce correct module under `--module-only --force` (#746, #752)
  - don't hardcode '`openPBS`' in GATE easyblock, use value for `default_platform` easyconfig parameter (#753)
  - avoid adding lib subdirs to `$*LIBRARY_PATH` if no libraries are there in `PythonPackage` easyblock (#755)
  - fix installing Python bindings for libxml2 to correct installation prefix (#765)

easyconfigs

- add GCCcore easyconfig that can be used as base for all compilers (without getting in the way) (#2214)
  - along with easyconfig for `GCC/4.9.3-2.25`: bundle of GCCcore 4.9.3 and binutils 2.25
  - intended to replace the GNU toolchain
- added example easyconfig files for 39 new software packages:
  - DIRAC (#2212), GeoIP (#2172, #2185), GeoIP-C (#2172, #2185), graph-tool (#1591), gtkglext (#2217), Intel Advisor (#2210), InterProScan (#2225, #2227, #2234), intltool (#2136), kallisto (#2173), LibUUID (#1930), LuaJIT (#2153), libXcursor (#2136), libXrandr (#2136), libXtst (#2143), libdap (#1930), libtasn1 (#2208), libxkbcommon (#2136), MRtrix (#2217, #2218), MultiNest (#2166, #2168), Nipype (#2150), PPfold (#2183, #2187), p11-kit (#2208), pangox-compat (#2217), Qt5 (#2136), randrproto (#2136), rhdf5 (#2175), Stampy (#2180, #2182), scikit-umfpack (#2061), scp (Python pkg) (#2196), sleuth (#2175), traits (#2150), vincent (#2169, #2185), XKeyboardConfig (#2136), xcb-util (#2136), xcb-util-image (#2136), xcb-util-keysyms (#2136), xcb-util-renderutil (#2136), xcb-util-wm (#2136), zlibbioc (#2175)
- **added new easyconfigs for existing toolchains:** `intel/2015.08` (#2194), `intel/2016.00` (#2209), `intel/2016.01` (#2219), `iomkl/2015.03` (#2155)
- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...
  - including CMake 3.4.1, HDF5 1.8.16, netCDF 4.3.3.1, netCDF-Fortran 4.4.2, numpy 1.10.1, Octave 4.0.0, OpenFOAM 3.0.0, OpenFOAM-Extend 3.2, Python 2.7.11
- various other enhancements, including:
  - add tidyr to R 3.2.1 easyconfigs (#2174)
  - enable C++ support in MIGRATE-N (#2178)
  - also installed shared libraries for AMD and UMFPACK in SuiteSparse (#2061)
  - fix software name for ParaView (was: Paraview) (#2132)
  - enable building of shared libraries for binutils (#2133)
  - harden binutils built with dummy toolchain by linking to system libraries via `RPATH` (#2228)
  - enhance easyconfig unit tests to check that each easyconfig file is in the right subdirectory (#2232)
- various bug fixes, including:

- fix ALADIN patch file to not use relative paths, and adjust list of ALADIN sources accordingly ((#2207), (#2213))
- rename patch files for OpenFOAM to be in line with other patches (#2226)
- fix typo in bzip2 source URLs (#2204)
- force linking of ncurses in libreadline (#2206)
- enable `-fPIC` in all zlib 1.2.8 easyconfigs (#2220)
- move Net-LibIDN/SRA-Toolkit/bbftpPRO/o2scl easyconfigs to right location (#2232)
- restrict parallel build in OpenFOAM-Extend easyconfigs via '`maxparallel`', not '`parallel`' (#2233)

11.12.52 EasyBuild v2.4.0 (November 10th 2015)

feature + bugfix release

framework

- add support for `--extended-dry-run`/`-x` (#1388, #1450, #1453, #1455)
  - detailed documentation is available at *Extended dry run*
- fix checking of sanity check paths w.r.t. discriminating between files and directories (#1436)
  - this impacts several easyconfig files where `sanity_check_paths` was not 100% correct
- make '`eb`' script aware of Python v3.x, fall back to using `python2` if required (#1411)
- add experimental support for parsing .yeb easyconfig files in YAML syntax (#1447, #1448, #1449)
  - see also easyconfig_yeb_format
- add experimental support for resolving dependencies with minimal toolchains (#1306)
  - see also *Using minimal toolchains for dependencies*
- various other enhancements, including:
  - refactor `extract_cmd` function to get rid of if/elif/else spaghetti blob (#1382)
  - add support for `--review-pr` (#1383)
  - add `apply_regex_substitutions` function to perform runtime patching from easyblocks (#1388, #1458)
  - add support for specifying alternate name to be part of generated module name (#1389) * via '`modaltsoftname`' easyconfig parameter
  - support overriding # used cores via `--parallel` (#1393)
  - also define `$FC` and `$FCFLAGS` in build environment (#1394)
  - add support extracting for `.tar.Z` files (#1396)
  - include easybuild/scripts in installation (#1397)
  - ignore hidden directories in find_base_dir (#1413, #1415)
  - add `only_if_module_is_available` decorator function to guard functionality that uses optional dependencies (#1416)
  - give easyblocks the possibility to choose `maxhits` for `run_cmd_qa` (#1417)
  - use class name (string) rather than License instances as values for software license constants (#1418)

  - support controlling recursive unloading of dependencies via '`recursive_module_unload`' easyconfig parameter (#1425)
  - implement basic support for type checking of easyconfig parameters (#1427)
  - support auto-converting to expected value type for easyconfig parameters (enabled by default) (#1428, #1437)
  - add support for `--rebuild` command line option, alternative for `--force` which doesn't imply `--ignore-osdeps` (#1435)
  - add support for Mercurial easyconfig repository (#979, #1446)
  - add dedicated class for psmpi toolchain MPI component, and use it in gpsmpi and ipsmpi toolchains (#1454)
- various bug fixes, including:
  - fix extracting of comments from an easyconfig file that includes 'tail' comments (#1381)
  - fix dev version to follow PEP-440, as required by recent setuptools versions (#1403)
    - required to avoid that setuptools transforms the version itself
    - see also https://www.python.org/dev/peps/pep-0440/#developmental-releases
  - allow `get_cpu_speed` to return `None` if CPU freq could not be determined (#1421)
  - relax `sanity_check_paths` in EasyBuild bootstrap script to deal with possible zipped .egg (#1422)
  - use empty list as default value for src/patches in Extension class (#1434)
  - skip symlinked files in `adjust_permissions` function (#1439)
  - fix HierarchicalMNS to always use full version number (#1440)

easyblocks

- 3 new generic easyblocks: OCamlPackage (#467), SCons (#689, #700), Waf (#722)
- new easyblocks for 2 software packages that require customized support:
  - OCaml (#467), Samcef (#678)
- various other enhancements, including:
  - add support for installing OpenFOAM with external METIS, CGAL and Paraview (#497)
  - update netCDF easyblock updated for netCDF v4.3.3.1 (#674)
  - update Rosetta easyblock for recent Rosetta versions (#677)
  - make unpacked source dir detection in easyblock for VSC-tools a little bit more flexible (#679)
  - add support for building with Plumed support enabled in CP2K easyblock (#681)
  - update Go easyblock for Go v1.5 (#683)
  - use `apply_regex_substitutions` function in WRF easyblock (#685)
  - update MUMPS easyblock for 5.x (#686)
  - implement runtime patching of `$WM_*` and compiler variables for OpenFOAM (#688)
  - specify sequential compiler to use in compiler command that gets injected in OpenFOAM easyblock (#692)
  - make `PythonPackage` and WRF easyblocks dry-run aware (#696)
    - see also *Guidelines for easyblocks*

  - add support in `PythonPackage` for installing with `easy_install` + installing zipped eggs (#698, #711, #715)
  - update Bowtie easyblock for recent Bowtie versions (#707)
  - update CUDA easyblock for CUDA 7.x(#708)
  - also consider `config/make.sys.in` for want in QuantumESRESSO easyblock (#714)
  - define `$NWCHEM_LONG_PATH` if needed in NWChem easyblock (#720)
  - remove custom post-install step in PDT easyblock (#723)
    * no longer needed now that `adjust_permissions` functions ignores symlinks
  - use `$LIBS` in HPL easyblock (#727, #736)
- various bug fixes, including:
  - also define `$MCRROOT` for MCR in module (#687)
  - add missing '`super`' call in `configure_step` of easyblock for python-meep (#694)
  - only prepend existing non-empty paths to `$PYTHONPATH` in `PythonPackage` easyblock (#697)
  - fix extra_options definition in `CMakePythonPackage` easyblock (#698)
  - fix dev version to follow PEP-440, as required by recent setuptools versions (#702, #703, #704)
    * required to avoid that setuptools transforms the version itself
    * see also https://www.python.org/dev/peps/pep-0440/#developmental-releases
  - consider both `lib` and `lib64` in sanity check paths for flex (#705)
  - also copy signature file and don't copy CMake files in Eigen easyblock (#709)
  - fix directory names in `make_module_req_guess` of ANSYS easyblock (#713)
  - fix imports for `set_tmpdir` in easyblock unit tests after function was moved in EasyBuild framework (#726)
  - use `--with-tcltk*` configure options for Python to point to ensure Tcl/Tk deps are picked up (#729)
  - fix order of subdirs for QuantumESPRESSO binaries (#730)
  - correctly handle having both `$FC`/`$FCFLAGS` and `$F90`/`$F90FLAGS` defined when building MVAPICH2 (#732)
  - fix OpenSSL sanity check paths: lib/engines is a directory (#731, #733)
  - fix sanity check paths for netcdf-python (#735)

easyconfigs

- added example easyconfig files for 45 new software packages:
  - animation (#2007), ANSYS CFD (#1969), ANTLR (#1191, #1980), APR (#1970), APR-util (#1970), Aspera Connect (#2005), ChIP-Seq (#2119), deap (#2082), DISCOVARdenovo (#1932), FastQC (#1984), fontsproto (#1618, #2038), GraphicsMagick (#2007), HBase (#1990), ISIS (#1972), libedit (#293), libfontenc (#1618, #2038), libGLU (#1627), libXdamage (#1618, #2038), libXfont (#1618, #2038), LLVM (#1620, #1989, #2031), MIGRATE-N (#1944), MIRA (#1938), mympingpong (#2049), MySQLdb (#2011), NCO (#1191, #1980), NIPY (#2064), Nilearn (#2064), NiBabel (#2064), PBZIP2 (#1038), PIL (#2062), PhyloCSF (#2018), pycairo (#2085), pydicom (#2063), Salmon (#2051), Samcef (#1941), scikit-image (#1974, #2006), Serf (#1970), SSAHA2 (#1039), Subversion (#1970), SWASH (#2059), time (#1954), Trim_Galore (#1984), Trimmomatic (#1987), WEKA (#1986), x264 (#2017)
- added new easyconfigs for existing toolchains: `gimkl/2.11.5` (#2093)

- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...
  - including Clang + LLVM 3.7.0, CMake 3.3.2, CUDA 7.5.18, hanythingondemand v3.0.1, Mesa 11.0.2, mpi4py v2.0.0, ncurses 6.0, OpenFOAM 2.4.0, Paraview 4.4.0, Python 3.5.0, QuantumESPRESSO v5.2.1
- various other enhancements, including:
  - enable '`pic`' toolchain option in libxml2 easyconfigs (#1993)
  - extend list of R libraries included in R v3.2.1 easyconfigs (#2042, #2046, #2067, #2072)
  - add Rsubread in Bioconductor easyconfigs (#1971)
- various bug fixes, including:
  - fix software name for `BEEF` (was '`libbeef`') (#1679)
  - add patch to install `qhull.pc` (pkgconfig) file with Qhull (#1975)
  - don't enable experimental nouveau API in libdrm easyconfigs (#1994)
  - fix dev version to follow PEP-440, as required by recent setuptools versions (#1997)
    - required to avoid that setuptools transforms the version itself
    - see also https://www.python.org/dev/peps/pep-0440/#developmental-releases
  - correct homepage in Cufflinks easyconfigs (#2060)
  - fix imports for `set_tmpdir` in easyblock unit tests after function was moved in EasyBuild framework (#2097)
  - add patch for Tk 8.6.4 to fix problem with `tk.tcl` not being found (#2102)
  - don't use `%(version)s` template in toolchain version, causes problems with HierarchicalMNS (#2104)
  - fix sanity check paths in several easyconfig (#2109, #2120, #2121, #2125)
    - required because of bug fix in `sanity_check_step` implementation
    - CVXOPT, h5py, LIBSVM, libunistring, MDP, monty, PhyloCSF, Pyke, pandas, pycosat, pyhull, pymatgen, python-dateutils, Seaborn, Theano, XML-LibXML, XML-Simple

11.12.53 EasyBuild v2.3.0 (September 2nd 2015)

feature + bugfix release

framework

- requires vsc-base v2.2.4 or more recent (#1343)
  - required for `mk_rst_table` function in `vsc.utils.docs`
- various other enhancements, including:
  - add support for generating documentation for (generic) easyblocks in `.rst` format (#1317)
  - preserve comments in easyconfig file in `EasyConfig.dump()` method (#1327)
  - add `--cleanup-tmpdir` option (#1365)
    - enables to preserve the used temporary directory via `--disable-cleanup-tmpdir`
  - enhance `EasyConfig.dump()` to reformat dumped easyconfig according to style guidelines (#1345)
  - add support for extracting `.iso` files using 7z (p7zip) (#1375)
- various bug fixes, including:

  - correctly deal with special characters in template strings in `EasyConfig.dump()` method (#1323)
  - rework `easybuild.tools.module_generator` module to avoid keeping state w.r.t. fake modules (#1348)
  - fix dumping of hidden deps (#1354)
  - fix use of `--job` with hidden dependencies: include `--hidden` in submitted job script when needed (#1356)
  - fix `ActiveMNS.det_full_module_name()` for external modules (#1360)
  - fix `EasyConfig.all_dependencies` definition, fix tracking of job dependencies (#1359, #1361)
  - fix `ModulesTool.exist()` for hidden Lua module files (#1364)
  - only call `EasyBlock.sanity_check_step` for non-extensions (#1366)
    * this results in significant speedup when installing easyconfigs with lots of extensions, but also results in checking the default sanity check paths if none were defined for extensions installed as a module
  - fix using module naming schemes that were included via `--include-module-naming-schemes` (#1370)

easyblocks

- new easyblocks for 2 software packages that require customized support:
  - MCR (#623), Molpro (#665)
- various other enhancements, including:
  - enhance BWA easyblock to also install man pages (#650)
  - enhance tbb easyblock to consider lib dirs in order and also define `$CPATH`, `$LIBRARY_PATH`, `$TBBROOT` (#653, #654)
  - call `PythonPackage.configure_step` in `ConfigureMakePythonPackage.configure_step` (#668)
  - add '`foldx3b6`' as possible binary name in FoldX easyblock (#671)
  - enhance/cleanup MATLAB easyblock (#672)
  - move preparing of '`intel`' subdir in `$HOME` to `configure_step` in `IntelBase` easyblock (#673)
- various bug fixes, including:
  - add missing super call in `post_install_step` of imkl easyblock (#648, #660)
  - fix regex used to correct `I_MPI_ROOT` in impi `mpivars.sh` scripts (#662)
  - fix regex used to patch `.mk` file in configure step of SuiteSparse easyblock (#666)
  - correctly specify installation prefix via `$GEM_HOME` in `RubyGem` easyblock (#667)
  - add custom sanity check in scipy easyblock (#669)
  - specify to always use the bfd linker for OpenFOAM, to stay away from using `ld.gold` (#670)

easyconfigs

- added example easyconfig files for 19 new software packages:
  - ATK (#1780), Atkmm (#1780), cairomm (#1780), GLibmm (#1780), GlobalArrays (#1868), gdk-pixbuf (#1780), gtk+ (#1780), Gtkmm (#1780), libbeef (#1827), libsigc++ (#1780), libsodium (#1876), MACS (#1869), MCR (#1677), Molpro (#1880), NFFT (#1921), p7zip (#1931), Pangomm (#1780), pygraphviz (#1861), pycosat (#1859)

- added new easyconfigs for existing toolchains: GNU/4.9.3-2.25 (#1836), foss/2015b (#1695), intel/2015b (#1696)
  - add easyconfigs using this toolchain for BLAST+ 2.2.31, Boost 1.58.0, CP2K 2.6.1, OpenFOAM 2.3.1, Perl 5.20.2 + 5.22.0 (bare), Python 2.7.10, R 3.2.1
- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...
  - including Boost 1.59.0, CP2K 2.6.1, GCC 5.2.0
- various other enhancements, including:
  - enhance texinfo easyconfig w.r.t. `texmf`, only use it as a build dependency (#1840)
  - enable building of `ld.gold` in binutils 2.25 (#1885)
- various bug fixes, including:
  - fix enabling MPI support for h5py 2.5.0 (#1825)
  - fix versions of Bioconductor packages + add a couple extra (#1828, #1852, #1895, #1917)
  - put dummy values in place for `builddir`/`installdir` templates in easyconfigs unit tests (#1835)
  - fix easyconfigs unit tests w.r.t. changes made in framework (#1853, #1870, #1874, #1875)
  - add GMP as missing dep in Python 2.7.10 easyconfigs, required for pycrypto extension (#1858)
  - specify installation prefix for SIP (#1888, #1892)
  - add custom sanity check paths in various easyconfigs (#1889, #1894, #1897 - #1909)
    - required because of fix in EasyBuild framework, causing default sanity check paths to be considered for extensions that are installed as a module
    - affected easyconfigs include: AnalyzeFMRI, Biggus, bibtexparser, DB_File, DBD-Pg, DBD-SQLite, DBD-mysql, evmix, fmri, FPM, GraphViz, gsl, GSSAPI, MDP, mpi4py, ncdf, ncdf4, netifaces, NetLibIDN, networkx, ordereddict, Parallel-ForkManager, paycheck, PyQuante, Pyke, PyQt, r2py, rjags, runjags, scikit-learn, SOBAcl, vsc-processcontrol, vsc-mympirun-scoop, XML, XML-Dumper, XML-Parser, XML-Twig, YAML-Syck
  - don't enable '`static`' toolchain option in SuiteSparse 4.4.3 easyconfig (#1911)
  - add –exclude unpack options for OpenFOAM 2.3.1 to avoid cyclic symlink causing problems when unpacking (#1925)

11.12.54 EasyBuild v2.2.0 (July 15th 2015)

feature + bugfix release

framework

- add support for using GC3Pie as a backend for `--job` (#1008)
  - see also *Submitting jobs using –job*
- add support for `--include-*` configuration options to include additional easyblocks, toolchains, etc. (#1301)
  - see *Including additional Python modules (–include-*)*
- add (experimental) support for packaging installed software using FPM (#1224)
  - see *Packaging support*
- various other enhancements, including:
  - use https for PyPI URL templates (#1286)

  - add GNU toolchain definition (#1287)
  - make bootstrap script more robust (#1289, #1325):
    * exclude 'easyblocks' pkg from `sys.path` to avoid that `setup.py` for easybuild-easyblocks picks up wrong version
    * undefine `$EASYBUILD_BOOTSTRAP*` environment variables, since they do not correspond with known config options
  - improve error reporting/robustness in `fix_broken_easyconfigs.py` script (#1290)
  - reset keep toolchain component class 'constants' every time (#1294)
  - make `--strict` also a build option (#1295)
  - fix purging of loaded modules in unit tests' setup method (#1297)
  - promote `MigrateFromEBToHMNS` to a 'production' MNS (#1302)
  - add support for `--read-only-installdir` and `--group-writable-installdir` configuration options (#1304)
  - add support for *not* expanding relative paths in `prepend_paths` (#1310)
  - enhance `EasyConfig.dump()` method to use easyconfig templates where possible (#1314), #1319), #1320), #1321)
- various bug fixes, including:
  - fix issue with cleaning up (no) logfile if `--logtostdout`/`-l` is used (#1298)
  - stop making `ModulesTool` class a singleton since it causes problems when multilple toolchains are in play (#1299)
  - don't modify values of 'paths' list passed as argument to prepend_paths in `ModuleGenerator` (#1300)
  - fix issue with `EasyConfig.dump()` + cleanup (#1308), #1311)
  - re-enable (and fix) accidentally disabled test (#1316)

easyblocks

- modified easybuild.easyblocks package declaration to support giving preference to custom easyblocks (#617)
- 2 new generic easyblocks: RubyGem (#565), SystemCompiler (#633)
- new easyblocks for 5 software packages that require customized support:
  - NEMO (#564), pbdMPI (#612), #620), pbdSLAP (#620), PDT (#624), Ruby (#565)
- various other enhancements, including:
  - update CUDA easyblock for v6.x (#476)
  - make METIS easyblock take into account configopts (#494)
  - enable building of EOMIP and EOMEA for NWChem versions 6.5 and up (#508)
  - make out-of-source build with CMake truly out-of-source (#615)
  - add support in Bundle easyblock to run full sanity check (#627)
  - also take platform-specific Python lib dirs into account in PythonPackage easyblock (#628)
  - fix mpivars scripts in Intel MPI installation for versions where the installation is moved (#629)
  - don't restrict `det_pylibdir` function to only EasyBuild-provided Python (#631), #641)
  - support snappy and other optional native libs in Hadoop easyblock (#632), #638), #640), #642)

- various bug fixes, including:
  - fix Xmipp easyblock, use provided `install.sh` script (#630)
  - update Clang easyblock to disable tests that may fail when unlimited stack size is used (#622)
  - fix setting of `$INTEL_LICENSE_FILE` for port@server values (#635)

easyconfigs

- added example easyconfig files for **62** new software packages:
  - ADF (#899), AutoDock_Vina (#808), bibtexparser (#1726), Biggus (#1770), Bismark (#990), blasr (#1662), BSMAP (#1171), Check (#811), Circuitscape (#1222), CONTRAfold (#689), cramtools (#1741), DBD-Pg (#1066), DendroPy (#995), EMAN2 (#1737), ETSF_IO (#727), eudev (#1578), fastqc (#1636), FDS (#814), #1617), #1625), FPM (#1440), frealign (#1619), g2log (#1035), GC3Pie (#1692), #1756), #1768), GenotypeHarmonizer (#1672), gensim (#1762), GraphViz (#1658), hisat (#1674), IDBA-UD (#1045), IMa2 (#828), IMPUTE2 (#824), JUBE (#1396), LAMARC (#760), libXScrnSaver (#1653), MATIO (#1004), MuTect (#1483), ncdf (#617), NEMO (#1640), ngspice (#1116), ordereddict (#1774), OSU Micro-Benchmarks (#1777), Parallel-ForkManager (#847), pBWA (#1009), PeakSeq (#1412), Pillow (#1702), Pindel (#1126), PLUMED (#1596), #1665), PostgreSQL (#1066), PROJ (#1006), PyAMG (#1222), Pyke (#1776), rpy2 (#1775), Sailfish (#1035), SCANMS (#1386), Seaborn (#1763), snpEff (#1680), SOBAcl (#1658), SPIDER (#1624), #1723), STAR (#1043), #1676), system GCC (#1778), tabix (#1059), tecplot360ex (#1100), Vampir (#512), VampirServer (#512), verifyBamID (#1675)
- added easyconfigs for 4 new software bundles:
  - R-bundle-Bioconductor (#1573), #1795), R-bundle-devtools (#1621), #1759), R-bundle-extra (#1387), #1759), R-bundle-pbd (#1659)
- added easyconfigs for new GNU toolchain (#1346), #1669)
- added new easyconfigs for existing toolchains: goolf/1.5.16, intel/2014.06
- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...
  - including BLAST 2.2.31+, Clang 3.6.1, CUDA 6.x, GCC 4.9.3, GROMACS 5.0.5, HDF5 1.8.15 + 1.8.15-patch1, Python 2.7.10, R 3.2.0 + 3.2.1, WRF 3.6.1
- various other enhancements, including:
  - update all ncurses easyconfigs to enable ncursesw and use ConfigureMake easyblock (#1337)
  - update PDT easyconfigs to use PDT easyblock (#1687)
  - add custom `sanity_check_paths` in libxml2 easyconfigs (#1690)
  - enhance unit tests to also cover EasyConfig.dump() method on all easyconfigs (#1761)
  - include snappy as dependency in Hadoop easyconfigs (#1758), #1773)
  - enable SSL support in CMake v3.2.3 easyconfigs (#1784)
  - add additional extensions in R easyconfigs (#1637)
- various bug fixes, including:
  - add patch file required for correct CUDA-aware OpenMPI v1.7.3 build (#631)
  - fix issue with OpenMPI dependency in ECore easyconfigs (#777)
  - don't run the Bloom tests for Jellyfish, they can randomly fail (#1016)
  - fix source URLs in BioPerl easyconfigs (#1075)
  - patch out svnversion command in Python 2.5.6 to fix build on recent systems (#1576)

- consistently use https for PyPI URLs in homepage/source_urls (#1616), #1722)
- include Tcl and Tk as dependencies in R easyconfig (#1623)
- add patch for installing paycheck as Py3 extension (#1629)
- add Perl dependency in git 2.x easyconfigs (#1631)
- fix easyconfig for GMP 6.0.0, don't use 6.0.0a sources (#1635)
- fix source_urls in QuantumESPRESSO 5.0.2 easyconfigs (#1652)
- include Tk as dependency in Python 2.7.9 easyconfigs (#1654)
- include tk-devel is list of OS deps for Python 2.7.9 Cray easyconfigs, make sure '`import Tkinter`' works (#1655)
- drop gpfs versionsuffix and stop using no longer existing `--enable-gpfs` configopt for recent HDF5 versions (#1657)
- include missing libxml2 dep in GLib easyconfigs (#1666)
- fix source URLs in Qt easyconfigs (#1673)
- fix source URLs for argparse Python extension (#1697)
- fix source URLs for deap Python extension (#1699)
- fix easyconfigs unit tests after making ModulesTool a non-singleton class (#1708)
- fix names for Xmipp easyconfigs and patches (#1719)
- add patch for Qt 4.8.6 to fix build issue on RHEL6 with intel/2015a (#1734)
- add M4 as build dep for GCC 5.1.0 (#1735)
- fix Bioconductor extension versions in R 3.1.3 easyconfigs (#1748)
- remove custom exts_filter definition from Python 3.4.3 easyconfig (#1765)
- fix source_urls in netCDF easyconfigs (#1766)
- fix source_urls in netCDF-C++ and netCDF-Fortran easyconfigs (#1767)

11.12.55 EasyBuild v2.1.1 (May 18th 2015)

bugfix release

framework

- fix issue with missing load statements when `--module-only` is used, don't skip ready/prepare steps (#1276)
- enhance `--search`: only consider actual filename (not entire path), use regex syntax (#1281)
- various other bug fixes, including:
  - fix `generate_software_list.py` script w.r.t. dependencies marked as external modules (#1273)
  - only use `$LMOD_CMD` value if `lmod` binary can't be found in $PATH (#1275)
  - fix location of `module_only` build option w.r.t. default value (#1277)
  - fix combined use of `--hide-deps` and `hiddendependencies` (#1280)
  - remove log handlers that were added during tests, to ensure effective cleanup of log files (#1282)
    - this makes the unit test suite run ~3x faster!

  - define `$CRAYPE_LINK_TYPE` if '`dynamic`' toolchain option is enabled for Cray compiler wrappers (#1283)

easyblocks

- fix compatibility of easyblocks with `--module-only` + dedicated unit test module (#610)
- minor enhancements, including:
  - update GROMACS easyblock for version 5 (#513)
- various other bug fixes, including:
  - only check compiler being used if FFTW interfaces are being built in Intel MKL easyblock (#606)

easyconfigs

- added example easyconfig files for **3** new software packages:
  - networkx (#1592), Platanus (#1597), SaguaroGW (#1600)
- added new easyconfigs for existing toolchains: `ictce/7.3.5`, `CrayCCE/5.2.40`, `CrayGNU/5.2.40`, `CrayIntel/5.2.40`
- added easyconfigs using `CrayGNU/5.2.25` and `CrayGNU/5.2.40` toolchains (#1610, #1611)
- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...
  - including Boost 1.58.0, GROMACS 5.0.4, Python 3.4.3
- various bug fixes, including:
  - enable usempi in GROMACS easyconfig using CrayGNU toolchain (as required by GROMACS easyblock) (#1590)
  - use system-provided tcsh when building WRF on Cray systems, to avoid hanging build (#1595)
  - only use '`dynamic`' toolchain option, not '`shared`', in easyconfigs using Cray toolchain (#1609)

11.12.56 EasyBuild v2.1.0 (April 30th 2015)

feature + bugfix release

framework

- requires vsc-base v2.2.0 or more recent
  - added support for LoggedException (vsc-base#160, vsc-base#163, vsc-base#165, vsc-base#167)
  - added support for add_flex action in GeneralOption (vsc-base#162)
  - added support to GeneralOption to act on unknown configuration environment variables (vsc-base#162)
- add support for only (re)generating module files: `--module-only` (#1018)
  - module naming scheme API is enhanced to include det_install_subdir method
  - see *Only (re)generating (additional) module files using –module-only*
- add support for generating module files in Lua syntax (note: requires Lmod as modules tool) (#1060, #1255, #1256, #1270)
  - see `--module-syntax` configuration option and *Module files syntax (–module-syntax)*
- deprecate `log.error` method in favor of raising `EasyBuildError` exception (#1218)
  - see *error and exception log methods no longer raise an exception*

- add support for using external modules as dependencies, and to provide metadata for external modules (#1230, #1265, #1267)
  - see *Using external modules*
- add experimental support for Cray toolchains on top of PrgEnv modules: `CrayGNU`, `CrayIntel`, `CrayCCE` (#1234, #1268)
  - see https://github.com/easybuilders/easybuild/wiki/EasyBuild-on-Cray
- various other enhancements, including:
  - clear list of checksums when using `--try-software-version` (#1169)
  - sort the results of searching for files (e.g., `--search` output) (#1214)
  - enhance test w.r.t. use of templates in cfgfile (#1217)
  - define '`%(DEFAULT_REPOSITORYPATH)s`' template for cfgfiles (see `eb --avail-cfgfile-constants`) (#1220)
  - also reset `$LD_PRELOAD` when running module commands, in case module defined `$LD_PRELOAD` (#1222)
  - move location of '`module use`' statements in generated module file (*after* '`module load`' statements) (#1232)
  - add support for `--show-default-configfiles` (#1240)
    - see *Default configuration files*
  - report error on missing configuration files, rather than ignoring them (#1240)
  - clean up commit message used in easyconfig git repository (#1248)
  - add `--hide-deps` configuration option to specify names of software that must be installed as hidden modules (#1250)
    - see *Installing dependencies as hidden modules using –hide-deps*
  - add support for appending/prepending to `--robot-paths` to avoid overwriting default robot search path (#1252)
    - see *Prepending and/or appending to the default robot search path*
  - enable detection of use of unknown `$EASYBUILD`-prefixed environment variables (#1253)
    - see *Environment variables*
  - add `--installpath-modules` and `--installpath-software` configuration options (#1258)
    - see *Software and modules install path (–installpath, –installpath-software, –installpath-modules)*
  - use dedicated subdirectory in temporary directory for each test to ensure better cleanup (#1260)
  - get rid of `$PROFILEREAD` hack when running commands, not needed anymore (#1264)
- various bug fixes, including:
  - make bootstrap script robust against having `vsc-base` already available in Python search path (#1212, #1215)
  - set default value for unpack_options easyconfig parameter to `''`, so `self.cfg.update` works on it (#1229)
  - also copy rotated log files (#1238)
  - fix parsing of `--download-timeout` value (#1242)

  - make `test_XDG_CONFIG_env_vars` unit test robust against existing user config file in default location (#1259)
  - fix minor robustness issues w.r.t. `$XDG_CONFIG*` and `$PYTHONPATH` in unit tests (#1262)
  - fix issue with handling empty toolchain variables (#1263)

easyblocks

- replace '`log.error`' with '`raise EasyBuildError`' in all easyblocks (#588)
- one new generic easyblock: `ConfigureMakePythonPackage` (#540)
- new easyblocks for 2 software packages that require customized support:
  - TINKER (#578), Xmipp (#581)
- various other enhancements, including:
  - enhance WIEN2k easyblock for recent versions + cleanup (#486)
  - define `$PYTHONNOUSERSITE` in PythonPackage easyblock so user-installed packages are not picked up (#577)
  - add support in CP2K easyblock for building on top of MPICH/MPICH2 (#579)
  - enhance Hadoop easyblock to support parallel builds (#580)
  - drop `-noroot` for recent FLUENT versions, honor `installopts`, enable `-debug` (#582)
  - include `prebuildopts` in build command for Python packages (#585)
  - also install `rosetta_tools` subdirectory for Rosetta (#586)
  - update SLEPc easyblock for v3.5 + style cleanup (#593)
  - minor fix in HPL easyblock w.r.t. checking defined environment variables (#595)
  - tweak CP2K easyblock w.r.t. LAPACK/FFTW support (#596)
  - minor update to GCC easyblock to support GCC v5.x (#598)
  - update sanity check in R easyblock for version 3.2.0 (#602)
- various bug fixes, including:
  - fix Score-P easyblock for compiler-only toolchains, include Qt as optional dependency (#552)
  - fix definition of `$MKLROOT`, should be set to '`mkl`' subdir of install dir (#576)
  - add `-libmpichf90` to list of MPI libraries in NWChem easyblock (#584)
  - stop using '`$root`' to make easyblocks compatible with module files in Lua syntax (#590)
  - also set `$PYTHONPATH` before installing Python package in temporary directory in `test_step` (#591)
  - unset `buildopts`/`installopts` before installing Python extensions in Python easyblock (#597)
  - allow not including vsc-base sources when installing EasyBuild (gets installed with easybuild-framework) (#600)
  - use `self.initial_environ` rather than `self.orig_environ` in EasyBuildMeta easyblock (#600)
  - make GCC easyblock compatible with `--module-only` by setting default value for `self.platform_lib` in constructor (#603)

easyconfigs

- added example easyconfig files for **27** new software packages:

  - AFNI (#1322, #1521), BCFtools (#1492), getdp (#1518), gmsh (#1518), gtest (#1244), hanythingondemand (#1530), mawk (#1369), Minimac (#815), Minimac3 (#1502), monty (#1548), Octave (#1563), pbs_python (#1530), pigz (#1036), Pygments (#1536), pyhull (#1539), pymatgen (#1549), PyQt (#1322, #1521), Ray (#1494), requests (#1536), seqtk (#1524), SIP (#1322, #1521), S-Lang (#1369), Spark (#1554), spglib (#1549), TINKER (#1465), tmux (#1369), Xmipp (#1489)

- added easyconfigs for new (Cray-specific) toolchains (#1538): `CrayGNU`, `CrayIntel`, `CrayCCE`
  - initially supported software (using CrayGNU toolchains): CP2K, GROMACS, HPL, Python + numpy/scipy, WRF (#1558)
  - see also https://github.com/easybuilders/easybuild/wiki/EasyBuild-on-Cray
- added new easyconfigs for existing toolchains: `goolf/1.5.16`, `intel/2014.06`
- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...
  - including GCC v5.1.0, OpenFOAM v2.3.1, R v3.1.3 and v3.2.0, PETSc/SLEPc v3.5.3, WIEN2k v14.1
- various other enhancements, including:
  - include pbr dependency for lockfile Python extension in Python v2.7.9 easyconfigs + mock/pytz/pandas (#1462, #1540)
  - include SQLite as dependency in Python v2.7.9 easyconfigs (#1468)
  - set `$LD_PRELOAD` for Hoard and jemalloc (#1470)
  - fix homepage in SCOTCH easyconfigs (#1485)
  - adding missing six/ecdsa dependencies for dateutil/paramiko Python packages in Python easyconfigs (#1504, #1505, #1506, #1507, #1508, #1509, #1510)
  - enable `pic` toolchain option in expat easyconfigs (#1562)
  - extend list of source URLs for Bioconductor packages in R easyconfigs to include 'release' source URLs (#1568)
- various bug fixes, including:
  - adding missing zlib dependency in all Tcl easyconfig files (#1344)
  - fix homepage in FLUENT easyconfigs (#1472)
  - use `--with-verbs` rather than deprecated `--with-openib` in OpenMPI configure options (#1511)
  - stop relying on `OS_NAME` constant to specify OS dependencies in OpenMPI easyconfigs (#1512)
  - replace use of '`$root`' with '`%(installdir)s`' to ensure compatibility with module files in Lua syntax (#1532)
  - stop relying on '`$MKLROOT`' in ROOT easyconfigs (#1537)
  - use proper Bundle easyblock for biodeps/PRACE (#1566)
  - make source_urls in Cube and Scalasca easyconfigs compatible with –try-software-version (#1574)
  - add patch for Cube to fix configure script w.r.t. Qt dependency, add –without-java-reader configure option (#1574)

11.12.57 EasyBuild v2.0.0 (March 6th 2015)

feature + bugfix release

framework

- requires vsc-base v2.0.3 or more recent
  - avoid deprecation warnings w.r.t. use of `message` attribute (vsc-base#155)
  - fix typo in log message rendering `--ignoreconfigfiles` unusable (vsc-base#158)
- removed functionality that was deprecated for EasyBuild version 2.0 (#1143)
  - see *Removed functionality*
  - the fix_broken_easyconfigs.py script can be used to update easyconfig files suffering from this (#1151, #1206, #1207)
  - for more information about this script, see *fix_broken_easyconfigs.py*
- stop including a crippled copy of vsc-base, include vsc-base as a proper dependency instead (#1160, #1194)
  - vsc-base is automatically installed as a dependency for easybuild-framework, if a Python installation tool is used
  - see *Required Python packages*
- various other enhancements, including:
  - add support for Linux/POWER systems (#1044)
  - major cleanup in `tools/systemtools.py` + significantly enhanced tests (#1044)
  - add support for '`eb -a rst`', list available easyconfig parameters in ReST format (#1131)
  - add support for specifying one or more easyconfigs in combination with `--from-pr` (#1132)
    - see from_pr_specifying_easyconfigs
  - define `__contains__` in EasyConfig class (#1155)
  - restore support for downloading over a proxy (#1158)
    - i.e., use `urllib2` rather than `urllib`
    - this involved sacrificing the download progress report (which was only visible in the log file)
  - let `mpi_family` return `None` if MPI is not supported by a toolchain (#1164)
  - include support for specifying system-level configuration files for EasyBuild via `$XDG_CONFIG_DIRS` (#1166)
    - see *Default configuration files*
  - make unit tests more robust (#1167, #1196)
    - see *Unit tests*
  - add hierarchical module naming scheme categorizing modules by `moduleclass` (#1176)
  - enhance bootstrap script to allow bootstrapping using supplied tarballs (#1184)
    - see bootstrap_advanced_options
  - disable updating of Lmod user cache by default, add configuration option `--update-modules-tool-cache` (#1185)
    - for now, only the Lmod user cache can be updated using `--update-modules-tool-cache`
  - use available `which()` function, rather than running '`which`' via `run_cmd` (#1192)
  - fix `install-EasyBuild-develop.sh` script w.r.t. vsc-base dependency (#1193)
  - also consider robot search path when looking for specified easyconfigs (#1201)

    * see *Specifying builds*

- various bug fixes, including:
  - stop triggering deprecated/no longer support functionality in unit tests (#1126)
  - fix `from_pr` test by including dummy easyblocks for HPL and ScaLAPACK (#1133)
  - escape use of '`%`' in string with command line options with `--job` (#1135)
  - fix handling specified patch level 0 (+ enhance tests for fetch_patches method) (#1139)
  - fix formatting issues in generated easyconfig file obtained via `--try-X` (#1144)
  - use `log.error` in `tools/toolchain/toolchain.py` where applicable (#1145)
  - stop hardcoding `/tmp` in `mpi_cmd_for` function (#1146, #1200)
  - correctly determine variable name for `$EBEXTLIST` when generating module file (#1156)
  - do not ignore exit code of failing postinstall commands (#1157)
  - fix rare case in which used easyconfig and copied easyconfig are the same (#1159)
  - always filter hidden deps from list of dependencies (#1161)
  - fix implementation of `path_matches` function in `tools/filetools.py` (#1163)
  - make sure plain text keyring is used by unit tests (#1165)
  - suppress creation of module symlinks for `HierarchicalMNS` (#1173)
  - sort all lists obtained via `glob.glob`, since they are in arbitrary order (#1187)
  - stop modifying `$MODULEPATH` directly in `setUp`/`tearDown` of toolchain tests (#1191)

easyblocks

- one new generic easyblock for installing a bundle of modules: `Bundle` (#550)
  - and let the `Toolchain` generic easyblock derive from `Bundle`
- new easyblocks for 2 software packages that require customized support:
  - GAMESS-US (#470, #544, #558), Hadoop (#563)
- various other enhancements, including:
  - move support for `staged_install` from CPLEX easyblock to generic `Binary` easyblock (#502)
  - fix sanity check in picard easyblock for v1.119 that doesn't include `sam.jar` (#522)
  - log warning message when unlinking jellyfish symlink fails in Trinity easyblock (#534)
  - revamp `EB_libint2` easyblock into `EB_Libint` that works for both Libint v1x and v2.x (#536)
  - update CP2K easyblock for recent versions (no more '`fes`') (#537)
  - update SuiteSparse easyblock for recent versions (#541)
  - fix easyblock unit tests after dropping support for deprecated behaviour in framework (#543)
  - rework PSI easyblock to support future releases (#545)
  - enable always generating a 'verbose' Makefile in the generic CMakeMake easyblock (#546)
  - remove functionality in (generic) easyblocks that was deprecated for EasyBuild v2.0 (#547)
  - enhance generic RPackage easyblock to support installing extensions in a separate prefix (#551)
  - deprecate GenomeAnalysisTK easyblock, since it's basically equivalent to Tarball (#557)

  - update SAMtools easyblock for v1.2 (#562)
  - take `preconfigopts` easyconfig parameter into account in ROOT easyblock (#566)
  - update easyblock for installing EasyBuild
    * to support bootstrapping with provided source tarballs (#559)
    * to also cover vsc-base dependency, and verify `easy-install.pth` (#567)
  - update disabling sanitizer tests for Clang 3.6 (#570)
- various bug fixes, including: * fix handling of LTO in GCC easyblock (#535) * relocate FDTD RPM to fix installation on SL6 (#538)

easyconfigs

- added example easyconfig files for **29** new software packages:
  - bsoft (#1353), Coot (#1400), Cuby (#1258), DSRC (#1242), Exonerate (#568), fastqz (#1242), FSA (#568), fqzcomp (#1242), GAMESS-US (#1153, #1406), Grep (#1308), Hadoop (#1418), Hoard (#1415), IMB (#1284), ISL (#1389), jemalloc (#1416), libdwarf (#1283), libelf (#1283), MPC (#1310), multitail (#1327), Pmw (#1403), Quip (#1242), rCUDA (#720), SCALCE (#1242), SMALT (#568), STREAM (#1332), worker (#1307), Xerces-C++ (#1370), XQilla (#1370), ZPAQ (#1242)
- added easyconfigs for new (common) toolchains
  - `foss/2015a` (#1239), `gompi/1.5.16` (#1380), `gmvolf/1.7.20` (#1397), `goolf/1.7.20` (#1294), intel/2015a (#1238), `intel/2015.02` (#1393), `iomkl/2015.01` (#1325), `iomkl/2015.02` (#1401)
- added new software bundle: `Autotools` (#1385)
- various other enhancements, including:
  - don't define `$LDSHARED` in zlib easyconfigs (#1350)
    * this fixes the long-standing "`no version information available`" issue with zlib
    * see also framework#108
  - add unit test to check that all `extra_options` keys are defined in `EasyConfig` instance (#1378)
  - add source MD5 checksums in all GCC easyconfigs (#1391)
  - speeding up the unit tests by avoiding rereading of same easyconfig file (#1432)
  - fix conflict detection in unit tests by considering build deps separately from runtime deps (#1447)
  - fix toolchain for Bison build dep in `MVAPICH2-1.9-iccifort-2011.13.367.eb` easyconfig (#1448)
  - use `Bundle` generic easyblock for HPCBIOS bundles and fix `moduleclass` (#1451)
- various bug fixes, including:
  - revert version of Libint dependency to 1.1.4 in CP2K v2.5.1 easyconfig (#1144)
  - added Java dependencies to EMBOSS easyconfigs (#1167)
  - don't list '`lto`' as a language in GCC easyconfigs (#1286)
    * related to the fixes in the GCC easyblock, see easyblocks#535
  - rename libint2 easyconfigs as Libint v2 easyconfigs (#1287)
  - fix mpi4py `source_urls` in Python easyconfigs (#1306)
  - consistently use CLooG 0.18.0 for GCC 4.8 series (#1392)

  - rename GenomeAnalysisTk easyconfigs to GATK (#1399)
  - include `openssl-devel` SLES11 OS dependency in cURL/MySQL/Python easyconfigs (#1422)
  - add missing Perl dependency in parallel easyconfigs (#1430)
  - correct name in BLAST+ easyconfigs (#1443)
  - fix name for sparsehash easyconfigs (#1452)

11.12.58 EasyBuild v1.16.2 (March 6th 2015)

bugfix release

framework

(no changes compared to v1.16.1, simple version bump to stay in sync with easybuild-easyblocks)

easyblocks

- make `EB_EasyBuildMeta` easyblock aware of vsc-base to make upgrading to EasyBuild v2.0.0 possible (#573)

easyconfigs

(no changes compared to v1.16.1, simple version bump to stay in sync with easybuild-easyblocks)

11.12.59 EasyBuild v1.16.1 (December 19th 2014)

bugfix release

framework

- fix functionality that is broken with `--deprecated=2.0` or with `$EASYBUILD_DEPRECATED=2.0`
  - don't include easyconfig parameters for `ConfigureMake` in `eb -a`, since fallback is deprecated (#1123)
  - correctly check software_license value type (#1124)
  - fix `generate_software_list.py` script w.r.t. deprecated fallback to `ConfigureMake` (#1127)
- other bug fixes
  - fix logging issues in tests, sync with vsc-base v2.0.0 (#1120)

easyblocks

- fix EasyBuild versions for which `$EASYBUILD_DEPRECATED=1.0` is set in EasyBuild sanity check (#531)

easyconfigs

- set default easyblock to ConfigureMake in TEMPLATE.eb (#1277)

11.12.60 EasyBuild v1.16.0 (December 18th 2014)

feature + bugfix release

framework

- deprecate automagic fallback to `ConfigureMake` easyblock (#1113)

  - easyconfigs should specify `easyblock = 'ConfigureMake'` instead of relying on the fallback mechanism
  - **note: automagic fallback to `ConfigureMake` easyblock will be removed in EasyBuild v2.0**
  - see also Automagic fallback to ConfigureMake
- stop triggering deprecated functionality, to enable use of `--deprecated=2.0` (#1107, #1115, #1119)
  - see Deprecated functionality for more information
- various other enhancements, including:
  - add script to clean up gists created via `--upload-test-report` (#958)
  - also use `-xHost` when using Intel compilers on AMD systems (as opposed to `-msse3`) (#960)
  - add Python version check in `eb` command (#1046)
  - take `versionprefix` into account in `HierarchicalMNS` module naming scheme (#1058)
  - clean up and refactor `main.py`, move functionality to other modules (#1059, #1064, #1075, #1087)
  - add check in `download_file` function for HTTP return code + show download progress report (#1066, #1090)
  - include info log message with name and location of used easyblock (#1069)
  - add toolchains definitions for `gpsmpi`, `gpsolf`, `impich`, `intel-para`, `ipsmpi` toolchains (#1072, #1073)
    * support for Parastation MPI based toolchains
  - enforce that `hiddendependencies` is a subset of `dependencies` (#1078)
    * this is done to avoid that site-specific policies w.r.t. hidden modules slip into contributed easyconfigs
  - enable use of `--show_hidden` for `avail` subcommand with recent Lmod versions (#1081)
  - add `--robot-paths` configure option (#1080, #1093, #1095, #1114)
  - support use of `%(DEFAULT_ROBOT_PATHS)s` template in EasyBuild configuration files (#1100)
    * see also Controlling the robot search path
  - use `-xHost` rather than `-xHOST`, to match Intel documentation (#1084)
  - update and cleanup `README` file (#1085)
  - deprecate `self.moduleGenerator` in favor of `self.module_generator` in `EasyBlock` (#1088)
  - also support `MPICH` MPI family in `mpi_cmd_for` function (#1098)
  - update documentation references to point to http://easybuild.readthedocs.org (#1102)
  - check for OS dependencies with *both* `rpm` and `dpkg` (if available) (#1111)
- various bug fixes, including:
  - fix picking required software version specified by `--software-version` and clean up `tweak.py` (#1062, #1063)
  - escape `$` characters in module load message specified via `modloadmsg` easyconfig parameter) (#1068)
  - take available hidden modules into account in dependency resolution (#1065)
  - fix hard crash when using patch files with an empty list of sources (#1070)
  - fix Intel MKL BLACS library being used for MPICH/MPICH2-based toolchains (#1072)

  - fix regular expression in `fetch_parameter_from_easyconfig_file` function (#1096)
  - don't hardcode queue names when submitting a job (#1106)
  - fix affiliation/mail address for Fotis in headers (#1105)
  - filter out `/dev/null` entries in patch file in `det_patched_files` function (#1108)
  - fix `gmpolf` toolchain definition, to have `gmpich` as MPI components (instead of `gmpich2`) (#1101)
    * 'MPICH' refers to MPICH v3.x, while MPICH2 refers to MPICH(2) v2.x (MPICH v1.x is ancient/obsolete)
    * **note**: this requires to reinstall the `gmpolf` module, using the updated easyconfig from easybuild-easyconfigs#1217

easyblocks

- new easyblocks for **2** software packages that requires customized support:
  - Chimera (#524), GATE (#518)
- fix use of deprecated functionality, enhance unit tests to check for it (#523)
- various other enhancements, including:
  - update PETSc easyblock for recent versions (v3.5) (#446)
  - only include major/minor version numbers for FLUENT subdir (#480)
  - factor out 'move after install' code from impi easyblock to `IntelBase`, use it for itac (#487)
  - support custom easyconfig parameters in `EB_impi` easyblock to correct behavior of MPI wrapper commands (#493)
  - consider both `lib` and `lib64` in sanity check for GROMACS (#501)
  - take `preinstallopts` and `installopts` into account in `Binary` easyblock (#503)
  - add sanity check command `abaqus information=all` for ABAQUS (#504)
  - update and clean up `README`, refer to http://easybuild.readthedocs.org documentation (#505, #516)
  - rename deprecated `self.moduleGenerator` to `self.module_generator` (#506)
    * see also easybuild-framework#1088
  - check whether specified `type` value is a known value (`psmp` or `popt`) in CP2K easyblock (#509)
  - add support to `SAMtools` easyblock for installing recent versions (v1.x) (#512)
  - fix version check + sanity check in Geant4 easyblock + style fixes (#517)
  - added `$root/modlib` to `$PYTHONPATH` guesses in Modeller easyblock (#525)
  - mark `license` custom easyconfig parameter as deprecated in `IntelBase` (#527)
- various bug fixes, including:
  - fix Libxc version check in CP2K easyblock (#478)
  - correctly specify `mkl_libs` when building *numpy* with GCC and imkl (#485)
  - extend noqa for OpenFOAM-Extend in build_step (#488, #520)
  - correctly set `$LD_LIBRARY_PATH`, `$LIBRARY_PATH` and `$PKG_CONFIG_PATH` for `R` (#495)
  - fix default value for `files_to_copy` in `MakeCp` easyblock (#500)
  - treat `MPICH` MPI family as MPICH v3.x instead of MPICH v1.x (#519)

    - see also easybuild-framework#1112
  - fix affiliation/mail address for Fotis in headers (#521)
  - clean up in `extra_options` methods, avoid casting return value to `dict` or returning via parent (#528)

easyconfigs

- added example easyconfig files for **39** new software packages:
  - ANTs (#1232), BEOPS (#1264), Chhimera (#1255), ctffind (#1249), DBD-SQLite (#1064), DBD-mysql (#1063), DIALIGN-TX (#668), ffmpeg (#1088), GObject-Introspection (#1079), GTS (#1079), Graphviz (#1079), GraphViz2 (#1079), grace (#1131), HarfBuzz (#1079), HTSlib (#1161), GSSAPI (#1048), Kerberos_V5 (#1048), libevent (#1063), libXdmcp (#1129), libXft (#1017), libXinerama (#1017), libXrender (#1017), Maven (#1094), MySQL (#1063), Net-LibIDN (#1060), OpenCV (#1088), OpenMD (#1105), Qhull (#1105), Pango (#1079), psmpi (#1245, #1246), RELION (#1017), renderproto (#1017), rjags (#1125), runjags (#1125), SPRNG (#1138, #1141), xineramaproto (#1017), XML-Dumper (#1061), XML-Parser (#1061), XML-Twig (#1061)
- added easyconfigs for new toolchains
  - `intel/2014.10` & `intel/2014.11` (#1219), `intel-para/2014.12` (#1246), `gpsolf/2014.12` (#1245), `iompi/6.6.4` (#1215)
- include `easyblock = 'ConfigureMake'` in relevant easyconfigs to deal with deprecation of automagic fallback to `ConfigureMake` (#1248)
  - see also easybuild-framework#1113 and Automagic fallback to ConfigureMake
- clean up use of deprecated functionality in existing easyconfigs (#1252, #1259)
  - stop using deprecated `makeopts`, `premakeopts` and `shared_lib_ext`
  - check for use of deprecated functionality in easyconfigs unit tests
  - see also http://easybuild.readthedocs.org/en/latest/Deprecated-functionality.html#easyconfig-parameters
- various other enhancements, including:
  - also build `fftw3_threads` libraries, and enhance sanity checks (#1013)
  - add unit test to verify specified `sanity_check_paths` (#1119)
  - update and clean up `README`, refer to http://easybuild.readthedocs.org documentation (#1184, #1224)
- various bug fixes, including:
  - fix unit tests w.r.t. changes in framework (#1146)
  - remove unnecessary build dependencies for OpenMPI (#1168)
  - remove duplicate line in OpenMPI easyconfigs (#1207)
  - fix affiliation/mail address for Fotis in headers (#1237)
  - fix permissions of easyconfig files for consistency (#1210)
  - disable symbol lookup feature in cairo to fix build on SL6 (#1241)
  - fix easyconfig `gmpolf` toolchain w.r.t. MPICH software name (#1217)
    - see also easybuild-framework#1112
  - fix `source_urls` for WRF and WPS (#1225)
  - fix and clean up GATE easyconfigs (#1228)
  - fix broken CLHEP builds by including `-gcc` in `$CXXFLAGS` (#1254)

  - add patch to fix broken test in Go (#1257)
  - fix name of GMAP easyconfigs, should be GMAP-GSNAP (#1268)
  - fix easyconfig filenames, enhance unit test to check easyconfig filenames (#1271)

11.12.61 EasyBuild v1.15.2 (October 7th 2014)

bugfix release

framework

- fix `$MODULEPATH` extensions for Clang/CUDA, to make `goolfc`/`cgoolf` compatible with `HierarchicalMNS` (#1050)
- include `versionsuffix` in module subdirectory with `HierarchicalMNS` (#1050, #1055)
- fix unit tests which were broken with bash patched for ShellShock bug (#1051)
- add definition of gimpi toolchain, required to make gimkl toolchain compatible with `HierarchicalMNS` (#1052)
- don't override `COMPILER_MODULE_NAME` obtained from `ClangGCC` in Clang-based toolchains (#1053)
- fix wrong code in `path_to_top_of_module_tree` function (#1054)
  - because of this, load statements for compilers were potentially included in higher-level modules under `HierarchicalMNS`

easyblocks

- only disable sanitizer tests for recent Clang versions (#481, #482)
- pass down installopts to CUDA install command (#483)

easyconfigs

- disable parallel build for slalib (#968)
- fix compatibility of goolfc with HierarchicalMNS by using GCC toolchain for installing CUDA (#1106, #1115)
- fix zlib OS dependency spec for Debian in Boost easyconfigs (#1109)
- fix compatibility of gimkl with HierarchicalMNS by using gimpi subtoolchain (#1110)
- make both GCC and Clang first-class members in Clang-based toolchains to fix compatibility with HierarchicalMNS (#1113)

11.12.62 EasyBuild v1.15.1 (September 23rd 2014)

bugfix release

framework

- take into account that multiple modules may be extending `$MODULEPATH` with the same path, when determining path to top of module tree (see #1047)
  - this bug caused a load statement for either `icc` or `ifort` to be included in higher-level modules installed with an Intel-based compiler toolchain, under the `HierarchicalMNS` module naming scheme
- make `HierarchicalMNS` module naming scheme compatible with `cgoolf` and `goolfc` toolchain (#1049)
- add definition of `iompi` (sub)toolchain to make `iomkl` toolchain compatible with `HierarchicalMNS` (#1049)

easyblocks

(no changes compared to v1.15.0, simple version bump to stay in sync with easybuild-framework)

easyconfigs

- minor bug fixes, including:
  - use SHLIB_EXT in GMP/MPFR easyconfigs (#1090)
  - fix TopHat homepage and source_urls since page moved (#1092)
  - make iomkl toolchain compatible with HierarchicalMNS (#1099)

11.12.63 EasyBuild v1.15.0 (September 12th 2014)

feature + bugfix release

framework

- various other enhancements, including:
  - fetch extension sources in fetch_step to enhance `--stop=fetch` (#978)
  - add `iimpi` toolchain definition (#993)
  - prepend robot path with download location of files when `--from-pr` is used (#995)
  - add support for excluding module path extensions from generated modules (#1003)
    - see `include_modpath_extensions` easyconfig parameter
  - add support for installing hidden modules and using them as dependencies (#1009, #1021, #1023)
    - see `--hidden` and `hiddendependencies` easyconfig parameter
  - stop relying on `conflict` statement in module files to determine software name of toolchain components (#1017, #1037)
    - instead, the `is_short_modname_for` method defined by the module naming scheme implementation is queried
  - improve error message generated for a missing easyconfig file (#1019)
  - include path where tweaked easyconfigs are placed in robot path (#1032)
  - indicate forced builds in `--dry-run` output (#1034)
  - fix interaction between `--force` and `--try-toolchain --robot` (#1035)
  - add `--software` option, disable recursion for `--try-software(-X)` (#1036)
- various bug fixes, including: * fix `HierarchicalMNS` crashing when MPI library is installed with a dummy toolchain (#986) * fix list of FFTW wrapper libraries for Intel MKL (#987) * fix stability of unit tests (#988, #1027, #1033) * make sure `$SCALAPACK_INC_DIR` (and `$SCALAPACK_LIB_DIR`) are defined when using `imkl` (#990) * fix error message on missing FFTW wrapper libs (#992) * fix duplicate toolchain elements in `--list-toolchains` output (#993) * filter out load statements that extend the `$MODULEPATH` to make the module being installed available (#1016) * fix conflict specification included in module files (#1017) * avoid `--from-pr` crashing hard unless `--robot` is used (#1022) * properly quote GCC version string in archived easyconfig (#1028) * fix issue with `--repositorypath` not honoring `--prefix` (#1031) * sync with latest vsc-base version to fix log order (#1039) * increase # commits per page for `--from-pr` (#1040)

easyblocks

- added support for **2** new software packages that requires customized support:

  - Modeller (#392), NAMD (#397)
- various enhancements, including:
  - fix locale used for running Perl unit tests (#425)
  - fix Rmpi easyblock to correctly configure for Intel MPI (#435)
  - add support in generic Rpackage easyblock for handling patches (#435)
  - enhance OpenFOAM easyblock: fix building MPI build of Pstream and (pt)scotchDecomp + overall cleanup (#436)
  - enhance NWChem easyblock for version 6.3, clean up running of test cases (#441)
  - enhance noqa for interactive configuration of Qt build procedure (#447)
  - add custom sanity check for R in easyblock (#449)
  - make perlmodule take into account `(pre){config,build,install}opts` (#450)
  - add support for specifying an activation key after installing Mathematica (#452)
  - consider both `lib` and `lib64` directories in netCDF sanity check (#453)
  - fix sanity check for ANSYS for v15.0.x (#458)
  - fix Trilinos easyblock for recent version (#462)
  - enhance impi easyblock to handle install subdir for impi v5.0.1 and future version (#465, #472)
  - include $CFLAGS in linker flags for HPL (#466)
  - update sanity test checks for GROMACS 5.x (#471)
- various bug fixes:
  - fix building of FFTW wrappers for Intel MKL v11.1.x + cleanup of imkl easyblock (#445)

easyconfigs

- added example easyconfig files for **13** new software packages:
  - Circos (#780), DB_File (#913), Emacs (#970), evmix (#1077), GD (#780), gsl (#1077), IOR (#949), JAGS (#1076), libgd (#780), MethPipe (#1070), Modeller (#825), NAMD (#835), netCDF-C++4 (#1015)
- added easyconfigs for new toolchains (#986, #1051):
  - gimkl/1.5.9, ictce/7.1.2
- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...
  - including Python 2.7.8/3.4.1, Perl 5.20.0, R 3.1.1, NWChem 6.3, OpenFOAM-Extend 3.1, GCC 4.9.1, Clang 3.4.2, ...
- various enhancements, including:
  - make existing ictce/intel toolchains compatible with HierarchicalMNS (#1069)
    - this involves installing impi with an iccifort toolchain, and imkl with an iimpi toolchain
- various bug fixes, including:
  - download link for Perl modules changed to use cpan.metapan.org
  - fix missing MPI-based OpenFOAM libraries (`Pstream`, `(pt)scotchDecomp`), make sure provided SCOTCH is used (#957)

11.12.64 EasyBuild v1.14.0 (July 9th 2014)

feature + bugfix release

framework

- important changes
  - required Lmod version bumped to v5.6.3 (#944)
    - required due to enhancements and bug fixes in Lmod, e.g. making `--terse avail` significantly faster, and correctly handling a `prepend-path` statement that includes multiple directories at once
  - required Tcl/C environment modules version set to 3.2.10 (
    - hard requirement due to fixed `modulecmd` segmentation fault bug, that only tends manifests itself when making a large amount of changes in the environment (e.g. `module load <toolchain>`)
  - renamed `EasyBuildModuleNamingScheme` to `EasyBuildMNS`
- enhanced custom module naming schemes functionality to support hierarchical module naming schemes (#953, #971, #975)
  - extended API for custom module naming schemes to allow tweaking different aspects of module naming
    - see `easybuild/tools/module_naming_scheme/mns.py` for abstract `ModuleNamingScheme` class
    - an example hierarchical module naming scheme is included, see `HierarchicalMNS`
  - split up full module names into a module subdirectory part, which becomes part of `$MODULEPATH`), and a 'short' module name (is exposed to end-users)
    - example: `GCC/4.7.2` in `Core` subdir, `OpenMPI/1.6.5` in `Compiler/GCC/4.7.2` subdir
  - make `ModuleNamingScheme` class a singleton, move it into `easybuild.tools.module_naming_scheme.mns` module
  - implement `ActiveMNS` wrapper class for querying active module naming scheme
  - implement toolchain inspection functions that can be used in a custom module naming scheme
    - `det_toolchain_compilers`, `det_toolchain_mpi` in `easybuild.tools.module_naming_scheme.toolchain`
  - significant code cleanup & enhanced unit tests
- enhance & clean up `tools/modules.py` (#944, #953, #963, #964, #969)
  - make `ModulesTool` a singleton to avoid repeating module commands over & over again needlessly
  - use `module use`, `module unuse` rather than fiddling with `$MODULEPATH` directly
  - improve debug logging (include full stdout/stderr output of module commands)
  - remove deprecated functionality (`add_module`, `remove_module`, indirect module load)
- various other enhancements, including:
  - added toolchain definitions for 'common' toolchains: `intel` and `foss` (#956)
  - implement caching for easyconfig files, parsed easyconfigs and toolchains (#953)
  - enable `--ignore-osdeps` implicitly when `-D`, `--dry-run` or `--dep-graph` are used (#953)
  - flesh out `use_group` and `det_parallelism` function, include them in `easybuild.tools.systemtools` (#953)

  - make symlinking of module files part of module naming scheme API (#973)
    * list of symlinks paths can be controlled using `det_module_symlink_paths()` method
  - added support for new configuration options:
    * tweaking compiler flags triggered by `optarch` toolchain options using `--optarch` (#949)
    * filtering out dependencies from easyconfig files using `--filter-deps` (#957)
    * filtering environment included in test reports with `--test-report-env-filter` (#959) e.g. `--test-report-env-filter='^SSH|USER|HOSTNAME|UID|.*COOKIE.*'`
    * made suffix used for module files install path configurable, using `--suffix-modules-path` (#973)
  - added support for additional easyconfig parameters: * define aliases in module files (`modaliases`) (#952) * add print message on module load (`modloadmsg`) and Tcl footer (`modtclfooter`) in module files (#954, #974)
- various bug fixes, including:
  - don't try to tweak generated easyconfigs when using `--try-X` (#942)
  - currently create symlinks to module files modules/all under a custom module naming scheme (#953)
  - restore traceback error reporting on hard crashes (#965)

easyblocks

- added **one** new *generic* easyblock: CmdCp (#395)
- added support for **2** new software packages that requires customized support:
  - ANSYS (#398), HPCG (#408)
- various enhancements, including:
  - updated ABAQUS easyblock so that it works for version 13.2 (#406)
  - enhance BLAT easyblock by using `super`'s `build_step` and `prebuildopts`/`buildopts` (#423)
  - enhance Mothur easyblock to guess correct start dir for recent versions (#424)
  - replace use of deprecated (pre)makeopts with (`pre`)``buildopts`` in all easyblocks (#427)
  - fix poor mans version of toolchain compiler detection in imkl easyblock (#429)
- various bug fixes:
  - only check for `idb` for older versions of icc (#426)
  - fix issues with installing RPMS when `rpmrebuild` is required (#433)
  - correctly disable parallel build for ATLAS (#434)
  - fix sanity check for Intel MPI v5.x (only provides bin64) (#432)
  - add `$MIC_LD_LIBRARY_PATH` for MKL v11.x (#437)

easyconfigs

- added example easyconfig files for **17** new software packages:
  - ANSYS (#836), Beast (#912), ELPH (#910, #911), FastTree (#900, #947), GEM-library (#858), HPCG (#853), mdtest (#925), ncview (#648), PRANK (#917), RDP-Classifier (#903), SDPA (#955), SIBELia (#886), SOAPaligner (#857), SPAdes (#884), stemming (#891), WHAM (#872), YAXT (#656)
- added easyconfigs for new toolchains (#935, #944, #948):

  - foss/2014b, ictce/6.3.5, intel/2014b
- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...
- various enhancements, including:
  - replace use of deprecated `(pre)makeopts` with `(pre)buildopts` in all easyblocks (#954)
  - disable running embossupdate on installation of EMBOSS (#963)
- various bug fixes, including: * really enable OpenMP support in FastTree easyconfigs (#947) * fix easyconfigs unit tests after changes in framework (#958)

11.12.65 EasyBuild v1.13.0 (May 29th 2014)

feature + bugfix release

framework

- make `--try-X` command line options work recursively (i.e. collaborate with `--robot`) (#922)
  - EasyBuild will first build a full dependency graph of the specified easyconfigs, and then apply the `--try` specifications
    - the elements of the dependency graph for the used toolchain and its dependencies are left untouched
  - this makes `eb foo-1.0-goolf-1.4.10.eb --try-toolchain=ictce,5.5.0 --robot` also work when `foo` has dependencies
  - caveat: the specified easyconfig files must all use the same toolchain (version)
- add support for testing easyconfig pull requests from EasyBuild command line (#920, #924, #925, #932, #933, #938)
  - add `--from-pr` command line option for downloading easyconfig files from pull requests
  - add `--upload-test-report` command line option for uploading a detailed test report to GitHub as a gist
    - this requires specifying a GitHub username for which a GitHub token is available, using `--github-user`
    - with `--dump-test-report`, the test report can simply be dumped to file rather than being uploaded to GitHub
    - see also https://github.com/easybuilders/easybuild/wiki/Review-process-for-contributions#testing-result
- the `makeopts` and `premakeopts` easyconfig parameter are deprecated, and replaced by `buildopts` and `prebuildopts` (#918)
  - both `makeopts` and `premakeopts` will still be honored in future EasyBuild v1.x versions, but should no longer be used
- various other enhancements, including:
  - add `--disable-cleanup-builddir` command line option, to keep the build dir after a (successful) build (#853)
    - the build dir is still cleaned up by default for successful builds, i.e. `--cleanup-builddir` is the default
  - also consider lib32 in paths checked for `$LD_LIBRARY_PATH` and `$LIBRARY_PATH` (#912)
  - reorganize support for file/git/svn repositories into `repository` package, making it extensible (#913)

  - add support for `postinstallcmds` easyconfig parameter, to specify commands that need to be run after the install step (#918)
  - make `VERSION=` part in version of C environment modules tool optional, which is required for older versions (#930)
- various bug fixes, including:
  - fix small issues in bootstrap script: correctly determine EasyBuild version and make sure modules path exists (#915)
  - fix github unit tests (#916)
  - disable useless debug logging for unit tests (#919)
  - fix unit test for `--skip` (#929)
  - make sure `start_dir` can be set based on location of unpacked first source file (#931)
  - the `vsc` package shipped with easybuild-framework is synced with vsc-base v1.9.1 (#935)
    * fancylogger (used for logging in EasyBuild) is now robust against strings containing UTF8 characters
    * the `deprecated` logging function now does a non-strict version check (rather than an erroneous strict check)
    * the `easybuild.tools.agithub` module is removed, `vsc.utils.rest` now provides the required functionality
  - fix support for unpacking gzipped source files, don't unpack `.gz` files in-place in the source directory (#936)

easyblocks

- added support for **1** new software package that requires customized support:
  - Go (#417)
- various enhancements, including:
  - enhance OpenFOAM easyblock so OpenFOAM-Extend fork can be built too + style fixes (#253, #416)
  - enhance CPLEX easyblock by adding support for staged installation (#372)
  - include support for `configure_cmd_prefix` easyconfig parameter in ConfigureMake generic easyblock (#393)
  - enhance GCC easyblock for building v4.9.0 and versions prior to v4.5 (#396, #400)
  - enhance easyblocks for Intel ipp and itac to support recent versions (#399)
  - enhance Clang easyblock: install static analyzer (#402), be more robust against changing source dir names (#413)
  - enhance FoldX easyblock, update list of potential FoldX binaries to support recent versions (#407)
  - add runtime patching in Boost easyblock to fix build issue with old Boost versions and recent glibc (> 2.15) (#412)
  - enhance `MakeCp` generic easyblock: use location of 1st unpacked source as fallback base dir for `files_to_copy` (#415)
- various bug fixes:
  - fix installing Mathematica when X forwarding is enabled (make sure `$DISPLAY` is unset) (#391)
  - fix permissions of installed files in SAMtools easyblock, ensure read/execute for group/other (#409)

  - fix implementation of `det_pylibdir` function in PythonPackage generic easyblock (#419, #420)
    * determine Python lib dir via `distutils` Python, which works cross-OS (as opposed to hardcoding `lib`)

easyconfigs

- added example easyconfig files for **32** new software packages:
  - APBS (#742), BayesTraits (#770), bc (#888), BitSeq (#791), CEM (#789), CVS (#888), eXpress (#786), file (#888), GEMSTAT (#861), GMAP (#594), Go (#887), iscp (#602), IsoInfer (#773), Jellyfish (#868), less (#888), libcircle (#883), mcpp (#602), MMSEQ (#795), MUSTANG (#800), OpenFOAM-Extend (#437), popt (#759), pscom (#759), psmpi2 (#759), QuadProg++ (#773), rSeq (#771), RSEQtools (#870), Ruby (#873), segemehl (#792), SOAPec (#879), SOAPdenovo2 (#874), SRA-Toolkit (#793), texinfo (#888)
- added easyconfig for new toolchain goolfc/1.4.10
- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...
  - e.g., older versions of Boost (1.47.0), GCC (4.1-4.4), & recent versions of Clang, GCC, Lmod, etc.
- various enhancements, including:
  - add OpenSSL dependency for cURL, to enable HTTPS support (#881)
  - also install esl-X binaries for HMMER (#889)
- various bug fixes, including:
  - properly pass down compiler flags for ParMGridGen (#437)
  - specify proper make options for PLINK, fixing the build on SL6 (#594, #772)
  - fix netloc version (0.5 rather than 0.5beta) (#707)
  - remove Windows-style line ending in netCDF patch file (#796)
  - bump version of OpenSSL dep for BOINC (#882)

11.12.66 EasyBuild v1.12.1 (April 25th 2014)

bugfix release

framework

- return to original directory after executing a command in a subdir (#908)
- fix bootstrap with Lmod, fix issue with module function check and Lmod (#911)

easyblocks

(no changes compared to v1.12.0, simple version bump to stay in sync with easybuild-framework)

easyconfigs

(no changes compared to v1.12.0, simple version bump to stay in sync with easybuild-framework)

11.12.67 EasyBuild v1.12.0 (April 4th 2014)

feature + bugfix release

framework

- various enhancements, including:

  - completed support for custom module naming schemes (#879, #904)
    - a fully parsed easyconfig file is now passed to the `det_full_module_name` function
    - this does require that an easyconfig file matching the dependency specification is available
  - added more features to better support using a shared install target with multiple users (#902, #903, #904)
  - further development on support for new easyconfig format (v2.0) (#844, #848)
    - not considered stable yet, so still requires using `--experimental`
  - enhanced bootstrap script to also support Lmod and `modulecmd.tcl` module tools (#869)
  - added support to `run_cmd_qa` function to supply a list of answers (#887)
  - detect mismatch between definition of `module` function and selected modules tool (#871)
    - allowing mismatch now requires `--allow-modules-tool-mismatch`; an empty `module` function is simply ignored
  - provide lib64 fallback option for directories in default sanity check paths (#896)
  - add support for adding JAR files to `$CLASSPATH` (#898)
  - enhanced and cleaned up unit tests (#877, #880, #884, #899, #901)
  - code cleanup and refactoring
    - get rid of global variable for configuration settings in `config.py`, use singleton instead (#874, #888, #890, #892)
    - track build options via singleton in `config.py` rather than passing them around all over (#886, #889)
    - avoid parsing easyconfig files multiple times by passing a parsed easyconfig to the easyblock (#891)
    - deprecate list of tuples return type of `extra_options` static method (#893, #894)
    - move OS dependency check to `systemtools.py` module (#895)
- bug fixes, including:
  - fix linking with `-lcudart` if CUDA is part of the toolchain, should also include `-lrt` (#882)

easyblocks

- various enhancements, including:
  - also run `Perl Build build` for Perl modules (usually not required, but sometimes it is) (#380)
  - added glob support in generic makecp block (#367, #384)
  - enhance sanity check in GCC easyblock such that it also passes/works on OpenSuSE (#365)
  - add multilib support in GCC easyblock (#379)
- various bug fixes: * Clang: disable sanitizer tests when vmem limit is set (#366) * make sure all libraries are installed for recent Intel MKL versions (#375) * fix appending Intel MPI include directories to `$CPATH` (#371) * statically link readline/ncurses libraries in Python and NWChem easyblocks (#370, #383, #385) * fix easyblock unit tests after changes in framework (#376, #377, #378)

easyconfigs

- added example easyconfig files for **6** new software packages:
  - CLooG (#653), ELPA (#738), LIBSVM (#788), netaddr (#753), netifcas (#753), slalib-c (#750)
- added easyconfigs for new toolchains:

  - ClangGCC/1.3.0 (#653), goolf/1.4.10-no-OFED (#749), goolf/1.5.14(-no-OFED) (#764, #767), ictce/6.2.5 (#726), iomkl (#747)
- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...
- various enhancements, including:
  - tweak BOINC easyconfig to make use of `glob` support available for `files_to_copy` (#781)
  - enable `-fPIC` for libreadline, so it can be linked into shared libs (e.g. `libpython2.x.so`) (#798)
- various bug fixes, including: * fix Qt source_urls (#756) * enable `-fPIC` in ncurses 5.9 ictce/5.2.0 easyconfig, just like in the others (#801) * fix unit tests after changes to framework (#763, #766, #769)

11.12.68 EasyBuild v1.11.1 (February 28th 2014)

bugfix release

framework

- various bug fixes, including:
  - fix hard crash when `$LMOD_CMD` specified full path to lmod binary, but `spider` binary is not in `$PATH` (#861, #873)
  - fix bug in initialisation of repositories, causing problems when a repository subdirectory is specified (#852)
  - avoid long wait when dependency resolution fails if `--robot` is not specified (#875)

easyblocks

(no changes compared to v1.11.0, simple version bump to stay in sync with easybuild-framework)

easyconfigs

(no changes compared to v1.11.0, simple version bump to stay in sync with easybuild-framework)

11.12.69 EasyBuild v1.11.0 (February 16th 2014)

feature + bugfix release

framework

- various enhancements, including:
  - add checksum support for extensions (#807)
  - make checksum functionality more memory efficient by reading in blocks (#836)
  - rewrite of dependency solving for speed and better reporting of missing dependencies (#806, #818)
  - refactoring of `main.py` (#815, #828)
    - function/method signatures to pass down build options
    - move functions from main.py into easybuild.framework.X or easybuild.tools
  - provide better build statistics (#824)
  - add –experimental, `--deprecated` and `--oldstyleconfig` command line options (#838)
    - with `--experimental`, new but incomplete (or partially broken) features are enabled
    - with `--deprecated`, removed of deprecated functionality can be tested (anything deprecated will fail hard)

    - with `--disable-oldstyleconfig`, support for the old style configuration is disabled
  - define `$LIBRARY_PATH` in generated module files (#832)
  - more constants for source URLs (e.g. for downloads from bitbucket) (#831)
  - prefer `$XDG_CONFIG_HOME` and `~/.config/easybuild` over `~/.easybuild` for configuration files (#820)
  - add support for specifying footers to be appended to generated module files (#808)
    - see `--modules-footer` command line option
  - track version of modules tool + cleanup of `modules.py` (#839)
  - move actual `run_cmd` and `run_cmd_qa` implementations from `tools.filetools` into `tools.run` (#842, #843)
  - add support for generating modules that support recursive unloading (#830)
    - see `--recursive-module-unload` command line option
  - add flexibility support for specifying OS dependencies (#846)
    - alternatives can be specified, e.g. (`openssl-devel`, `libssl-dev`)
  - initial (incomplete) support for easyconfig files in new format (v2.0) (#810, #826, #827, #841)
    - requires `--experimental` to be able to experiment with format v2 easyconfig files
- various bug fixes, including:
  - fix problems with use of new-style configuration file (#821)
  - fix removal of old build directories, unless `cleanupoldbuild` easyconfig parameter is set (#809)
  - fix support for different types of repository path specifications (#814)
  - fix unit tests sensitive to `$MODULEPATH` and available easyblocks (#845)

easyblocks

- added **one** new *generic* easyblock: `VSCPythonPackage` (#348)
- added support for **1** new software package that requires customized support:
  - netcdf4-python (#347)
- various enhancements, including:
  - add support for installing recent tbb versions (#341)
  - use `makeopts` in the build step of the generic `PythonPackage` easyblock (#352)
  - define the `$CMAKE_INCLUDE_PATH` and `$CMAKE_LIBRARY_PATH` in the generic `CMakeMake` easyblock (#351, #360)
  - update Clang easyblock to support v3.4 (#346)
  - make sure Python is built with SSL support, adjust Python easyblock to pick up OpenSSL dep (#359)
    - note: providing OpenSSL via an OS package is still recommended, such that security updates are picked up
  - add support for recent netCDF versions (#347)
  - update SuiteSparse easyblock for new versions, and clean it up a bit (#350)
- various bug fixes:

  - fix name of `VersionIndependentPythonPackage` easyblock, deprecate `VersionIndependendPythonPackage` easyblock (#348)
  - fix detection of machine architecture in FSL easyblock (#353)
  - fix bug in NWChem easyblock w.r.t. creating local dummy `.nwchem` file (#360)
  - make sure `$LIBRARY_PATH` is set for Intel compilers and Intel MPI, fix 64-bit specific paths (#360)

easyconfigs

- added example easyconfig files for **30** new software packages:
  - argtable (#669), Clustal-Omega (#669), Coreutils (#582), GMT (#560), gperftools (#596), grep (#582), h4toh5 (#597), libunwind (#596), Lmod (#600, #692), Lua (#600, #692), MAFFT (#654), Molekel (#597), NEdit (#597), netcdf4-python (#660), nodejs (#678), OCaml (#704), patch (#582), PhyML (#664), PRACE Common Production Environment (#599), protobuf (#680), python-dateutil (#438), sed (#582), sickle (#651), Tesla-Deployment-Kit (#489), TREE-PUZZLE (#662), VCFtools (#626), Vim (#665), vsc-mympirun-scoop (#661), vsc-processcontrol (#661), XZ (#582)
- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...
  - OpenSSL with ictce toolchain (#703)
- various enhancements, including:
  - using more constants and templates (#613, #615)
  - specify OS dependency for SSL support, with OpenSSL dependency as fallback (#703)
- various bug fixes, including:
  - fix unit tests after (internal) framework API changes (#667, #672)
  - fix homepage in vsc-mympirun easyconfig file (#681)
  - align OpenMPI easyconfigs (#650)
  - add additional source URL in Qt easyconfigs (#676)
  - specify correct $PATH specification and define `$CHPL_HOME` for Chapel (#683)

11.12.70 EasyBuild v1.10.0 (December 24th 2013)

feature + bugfix release

framework

- various enhancements, including:
  - set unique default temporary directory, add `--tmpdir` command line option (#695)
  - add support for computing and verifying source/patch file checksums (#774, #777, #779, #801, #802)
    - cfr. `checksums` easyconfig parameter
  - add support for *eb –configshelp*, which prints out an example configuration file (#775)
  - add initial support for `eb` tab completion in bash shells (#775, #797, #798)
    - see also https://github.com/easybuilders/easybuild/wiki/Setting-up-tab-completion-for-bash
    - note: may be quite slow for now
  - enhancements for using Lmod as modules tool (#780, #795, #796):
    - ignore Lmod spider cache by setting `$LMOD_IGNORE_CACHE` (requires Lmod 5.2)

    - bump required Lmod version to v5.2
    - get rid of slow workaround for detecting module directories (only required for older Lmod versions)
    - fix version parsing for Lmod release candidates (rc)
    - improve integration with *lmod spider* by adding `Description: ``` prefix to ```module-whatis` field of module
  - add `--dry-short-short`/`-D` and `--search-short`/`-S` command line options (#781)
  - add toolchain definition for 'gompic', intended for using with CUDA-aware OpenMPI (#783)
  - add support for specifying multiple robot paths (#786)
  - add various source URL constants, add support for `%(nameletter)s` and `%(nameletterlower)s` templates (#793)
  - add `buildininstalldir` easyconfig parameter (#794)
  - add `--ignore-osdeps` command line option (#799, #802)
- various bug fixes, including:
  - enable `-mt_mpi` compiler flag if both `usempi` and `openmp` toolchain options are enabled (#771)
  - only use `libmkl_solver*` libraries for Intel MKL versions prior to 10.3 (#776)
  - fix toolchain support for recent Intel tools (#785)
  - code style fixes in `main.py` to adhere more to PEP8 (#789)
  - make sure `easyblock` easyconfig parameter is listed in `eb -a` (#791)
  - fix error that may pop up when using `skipsteps=source` (#792)

easyblocks

- added **one** new *generic* easyblock: `VersionIndependendPythonPackage` (#329, #334)
- added support for **2** new software packages that require customized support:
  - Charmm (#318), GROMACS (#335, #339)
- various enhancements, including:
  - fix support for recent SAMtools version (>= 0.1.19) (#320)
  - fix support for recent Intel tools (icc, ifort, imkl, impi) (#325, #327, #338)
  - enhance generic easyblock for installing RPMs (#332)
  - pick up `preinstallopts` in generic `PythonPackage` easyblock (#334)
  - enhance CP2K easyblock (libxc support, new versions) + style cleanup (#336)
- various bug fixes:
  - use unwanted env var functionality to unset `$MKLROOT` rather than failing with an error (#273)
  - always include `-w` flag for preprocessor to suppress warnings that may break QuantumESPRESSO configure (#317)
  - link with multithreaded LAPACK libs for ESMF (#319)
  - extend `noqanda` list of patterns in CUDA easyblock (#328, #337)
  - add `import _ctypes` to Python sanity check commands to capture a common build issue (#329)
  - bug fixes in generic `IntelBase` easyblock (#331, #333, #335)

    - remove broken symlink `$HOME/intel` if present
    - fix use of gettempdir to obtain a common (user-specific) tmp dir to symlink `$HOME/intel` to
  - fix build of recent scipy versions with GCC-based toolchain (#334)
  - fix use of gettempdir to obtain common (user-specific) tmp dir for `$HOME/.nwchemrc` symlink (#340, #342)
  - extend `noqanda` list in Qt easyblock (#342)

easyconfigs

- added example easyconfig files for **18** new software packages:
  - BEDTools (#579, #632, #635), CAP3 (#548), CHARMM (#584), cutadapt (#620), ErlangOTP (#556, #630), Ghostscript (#547, #632), HTSeq (#554, #632), Jansson (#545), libjpeg-turbo (#574), Molden (#566), netloc (#545), o2scl (#633), packmol (#566), PP (#405), qtop (#500), TAMkin (#566), vsc-base (#621), vsc-mympirun (#621)
- added easyconfigs for new toolchains (#545, #609, #629):
  - gcccuda/2.6.10, gompic/2.6.10, goolfc/2.6.10, ictce/6.0.5, ictce/6.1.5
- added additional easyconfigs for various supported software packages: version updates, different toolchains, ...
  - new versions of icc, ifort, imkl, impi (#609, #629)
  - large collection of ictce/5.3.0 easyconfigs (#627)
- various enhancements, including:
  - extended list of Python packages as extensions to Python (#625)
  - add MPI-enabled version of GROMACS + easyconfigs using ictce (#606, #636)
  - clean up templating of `source_urls` (#637)
- various bug fixes, including:
  - provide alternative download URL for Mesa (#532)
  - add Python versionsuffix in OpenBabel filenames (#566)
  - apply no-gets patch in all M4 v1.4.16 easyconfigs (#623)
  - fix patching of Python w.r.t. `libffi`/`_ctypes` issues (#625, #642)
  - bug fixes in GROMACS easyconfigs (#606)
    - change versionsuffix for non-MPI GROMACS easyconfigs to `-mt`
    - stop using 'CMakeMake' easyblock for GROMACS now that there's a dedicated GROMACS easyblock, which correctly specifies building against external BLAS/LAPACK libraries
  - fix Qt dependency for CGAL (#642)
  - fix libctl, libmatheval, Meep, PSI build issues caused by full paths in `guile-config`/`python-config` shebang (#642)
  - make sure HDF easyconfigs specify dedicated `include/hdf` include dir (#642)
    - this is required to avoid build issues with NCL, because HDF ships it's own `netcdf.h`
    - this also triggered removal of patch files for NCL that rewrote `include/hdf` to `include`
  - fix WPS v3.5.1 patch file after upstream source tarball was changed, supply checksum for verification (#642)

11.12.71 EasyBuild v1.9.0 (November 17th 2013)

feature + bugfix release

framework

- add support for Tcl environment modules (`modulecmd.tcl`) (#728, #729, #739)
  - special care was taken to make sure also the DEISA variant of `modulecmd.tcl` can be used
- code refactoring to prepare for supporting two formats for easyconfig files (#693, #750)
  - this prepares the codebase for supporting easyconfig format v2.0
  - some initial work on adding support for the new easyconfig format is included, but it's by no means complete yet
  - the current easyconfig format (now dubbed v1.0) is still the default and only supported format, for now
  - for more details, see https://github.com/easybuilders/easybuild/wiki/Easyconfig-format-two
- various other enhancements, including:
  - include a full version of vsc-base (see the `vsc` subdirectory) (#740)
    - this is a first step towards switching to using vsc-base as a proper dependency
  - implement get_avail_core_count function in systemtools module that takes cpusets and co into account (#700)
    - the `get_core_count` function is now deprecated
  - add `impmkl` toolchain definition (#736)
  - make regtest more robust: put holds on jobs without dependencies, release holds once all jobs are submitted (#751)
  - add support for specifying multiple alternatives for sanity check paths (#753)
  - add `get_software_libdir` function to modules.py (along with unit tests) (#758)
  - add support for more file extensions and constants w.r.t. sources (#738, #760, #761)
  - add MPICH2 support in `mpi_cmd_for` function (#761)
- various bug fixes, including:
  - fix checking of OS dependencies on Debian/Ubuntu that have `rpm` command available (#732)
  - make unit tests more robust w.r.t. non-writeable `/tmp` and loaded modules prior to starting unit tests (#752, #756)
  - also call `EasyBlock`'s sanity check in `ExtensionEasyblock` if paths/commands are specified in easyconfig (#757)
  - set compiler family for dummy compiler, add definition of toolchain constant for dummy (#759)
- other
  - add build status badges for master/develop branches to `README` (#742)
  - add scripts for installing EasyBuild develop version or setting up git development environment (#730, #755)

easyblocks

- added support for **8** new software packages that require customized support:

  - Allinea DDT/MAP (#279), ARB (#291), GenomeAnalysisTK (#278), OpenBabel (#305, #309), picard (#278), PyQuante (#297), Scalasca v1.x (#304), Score-P (#304)
    * the Score-P easyblock is also used for Cube 4.x, LWM2, OTF2, and Scalasca v2.x
- various enhancements, including:
  - add support building ScaLAPACK on top of MPICH2, required for gmpolf toolchain (#274)
  - add support to ConfigureMake easyblock to customize configure `--prefix` option (#287)
  - add support for specifying install command in Binary easyblock (#288)
  - enhance CMakeMake easyblock to specify srcdir via easyconfig parameter, and to perform out-of-source builds (#303)
- various bug fixes:
  - use correct configure flag for Szip in HDF5 easyblocks, should be `--with-szlib` (#286, #301)
  - add support for serial HDF5 builds (#290, #301)
  - enhance robustness of Qt easyblock w.r.t. interactive configure (#295, #302)
  - enhance support for picking up license specification via environment variables (#298, #307)
  - extend list of values for `$CPATH` in libint2 easyblock (#300)
  - fix `extra_options` `super` call in Clang easyblock (#306)
  - add support in Boost easyblock to specify toolset in easyconfig file (#308)
- other:
  - add build status badges for master/develop branches to README (#289)

easyconfigs

- added example easyconfig files for **58** new software packages:
  - Allinea (#468), ARB + dependencies (#396, #493, #495), arpack-ng (#451, #481), CDO (#484, #521), Cube (#505), ed (#503), FLTK (#503), GenomeAnalysisTK (#467), GIMPS (#527), GTI (#511), IPython (#485, #519), LWM2 (#510), MPICH2 (#460), MUST (#511), ncdf (#496, #522), OPARI2 (#505), OpenBabel (#504, #524), OTF (#505), OTF2 (#505), PandaSEQ (#475), ParaView (#498, #514), ParFlow (#483, #520), PCC (#486, #528), PDT (#505), picard (#467), PnMPI (#511), PyQuante (#499, #523), pysqlite (#519), Scalasca (#505), Score-P (#505), SDCC (#486, #528), Silo (#483, #520), Stride (#503), SURF (#503), TCC (#486, #528)
  - ARB dependencies (23): fixesproto, imake, inputproto, kbproto, libICE, libSM, libX11, libXau, libXaw, libXext, libXfixes, libXi, libXmu, libXp, libXpm, libXt, lynx, motif, printproto, Sablotron, xbitmaps, xextproto, xtrans
- added easyconfigs for new gmpich2/1.4.8, gmpolf/1.4.8 and goolf/1.6.10 toolchains (#460, #525, #530)
- added additional easyconfigs for various software packages (list too long to include here)
  - version updates, different toolchains, ...
- various bug fixes, including:
  - enable building of shared libraries for MPICH (#482)
  - fix HDF configure option for Szip, should be `--with-szlib` (#533)
    * for HDF5, this issue is fixed in the HDF5 easyblock
- other

  - add build status badges for master/develop branches to README (#490)

11.12.72 EasyBuild v1.8.2 (October 18th 2013)

bugfix release

framework

- fix regular expression used for obtaining list of modules from `module avail` (#724)
  - modules marked as default were being hidden from EasyBuild, causing problems when they are used as dependency

easyblocks

- fix installing of EasyBuild with a loaded EasyBuild module (#280)
  - this is important to make upgrading to the latest EasyBuild version possible with a bootstrapped EasyBuild

easyconfigs

- port thread pool patch to PSI 4.0b4 and include it to avoid hanging tests (#471)

11.12.73 EasyBuild v1.8.1 (October 14th 2013)

bugfix release

- various bug fixes, including:
  - fix bugs in regtest procedure (#713)
    - force 2nd and 3rd attempt of build in case 1st attempt failed
  - fix copying of install log to install directory (#716)
  - only create first source path if multiple paths are specified (#718)
  - detect failed PBS job submission by checking obtained job ID for `None` value (#713, #717, #719, #720)

easyblocks

- various bug fixes:
  - fix problems in PSI easyblock causing build to fail (#270)
  - fix issues with running NWChem test cases, fail early in case broken symlink is present (#275)

easyconfigs

- added additional easyconfigs for various software packages (#457):
  - Boost, bzip2, libreadline, ncurses, PSI, Python, zlib
- various bug fixes, including:
  - fix faulty easyconfig file names for HPCBIOS_Math, MUSCLE, XML-LibXML and YAML-Syck (#459, #462)
  - stop (re)specifying default maximum ratio for failed tests in NWChem easyconfig (#457)

11.12.74 EasyBuild v1.8.0 (October 4th 2013)

feature + bugfix release

framework

- add support for using alternative module naming schemes (#679, #696, #705, #706, #707)
  - see https://github.com/easybuilders/easybuild/wiki/Using-a-custom-module-naming-scheme for documentation
  - module naming scheme classes that derive from the 'abstract' `ModuleNamingScheme` class can be provided to EasyBuild
    - the Python module providing the class must be available in the `easybuild.tools.module_naming_scheme` namespace
    - a function named `det_full_module_name` must be implemented, that determines the module name in the form of an string based on the supplied dictionary(-like) argument
  - the active module naming scheme is determined by EasyBuild configuration option `--module-naming-scheme`
  - for now, only the `name`/`version`/`versionsuffix`/`toolchain` easyconfig parameters are guaranteed to be provided
    - consistently providing all easyconfig parameters (i.e., also for dependencies) requires more work (see #687)
  - implementing this involved a number of intrusive changes:
    - the API of the modules.py module needed to be changed, breaking backward compatibility
      - the function for which the signatures were modified are considered to be internal to the framework, so this should have very minor impact w.r.t. easyblocks not included with EasyBuild
      - affected functions include: `available`, `exists`, `show`, `modulefile_path`, `dependencies_for`
    - the format for specifying dependencies was extended, to allow for specifying a custom toolchain
      - this allows to fix inaccurate dependency specifications, for example: `('OpenMPI', '1.6.4-GCC-4.7.2')" to "('OpenMPI', '1.6.4', '', ('GCC', '4.7.2'))`
      - see also easyconfigs#431
    - the recommended version for Lmod was bumped to v5.1.5
      - using earlier 5.x version still works, but may be very slow when 'available' is used, due to bugs and a missing feature in Lmod versions prior to v5.1.5 on which we rely
- various other enhancements, including:
  - only (try to) change group id if it is different from what is wanted (#685)
  - added toy build unit test (#688)
  - support for specifying a list of source paths in EasyBuild configuration (#690, #702)
  - add function to determine CPU clock speed in `systemtools.py` (#694, #699)
- various bug fixes, including:
  - avoid importing toolchain modules over and over again to make toolchain constants available in toolchain module (*#679 <https://github.com/easybuilders/easybuild-framework/pull/679>*)

  - only change the group id if current gid is different from what we want in `adjust_permissions` function (*#685 <https://github.com/easybuilders/easybuild-framework/pull/685>*)
  - restore original environment after running 'module purge' (*#698 <https://github.com/easybuilders/easybuild-framework/pull/698>*)
    * important sidenote: this results in resetting the entire environment, and has impact on the sanity check step;
    * any environment variables that are set before the EasyBlock.sanity_check_step method fires, are no longer there when the sanity check commands are run (cfr. easyblocks#268)

easyblocks

- added **one** new *generic* easyblock: `BinariesTarball` (#255)
- added support for **5** new software packages that require customized support:
  - DB (#226), FDTD Solutions (#239), FoldX (#256), Mathematica (#240), MUMPS (#262)
- various enhancements, including:
  - support optionally running configure in generic `MakeCp` easyblock (#252)
  - enhanced Clang easyblock to support building Clang 3.3 (#248)
  - add support for `$INTEL_LICENSE_FILE` specifying multiple paths (#251)
  - enhanced ATLAS easyblock to support building ATLAS 3.10.1 (#258)
- various bug fixes:
  - add zlib lib dir in link path dirs for WPS (#249)
  - stop using deprecated `add_module` function in imkl easyblock (#250)
  - fixed PSI easyblock w.r.t. support for building plugins (#254, #269)
  - move OS check for Clang to `check_readiness_step`, to allow a build job to be submitted from SL5 (#263, #264)
  - enable verbose build for DOLFIN, to allow for proper debugging if the build fails (#265)
  - make sure `$LDFLAGS` and `$INSTANT_*_DIR` env vars are set for DOLFIN sanity check commands (#268)
    * this is required after resetting the environment after running module purge (see framework release notes)
  - don't try to load module in LAPACK test-only build (#264, #266)

easyconfigs

- added example easyconfig files for **9** new software packages:
  - BOINC (#436), DB (#343, #449), fastahack (#374), FDTD Solutions (#387), FoldX (#440, #442), Mathematica (#394), Mesquite (#447), MUMPS (#447), ParMGridGen (#447)
- added additional easyconfigs for goalf, gompi, ClangGCC, cgmvapich2, cgmvolf toolchains (#350, #441)
- added additional easyconfigs for various software packages:
  - ATLAS, Bison, bzip2, Clang, CMake, cURL, EasyBuild, expat, FFTW, GDB, gettext, git, HPL, LAPACK, libreadline, M4, METIS, MVAPICH2, Mercurial, ncurses, OpenBLAS, OpenMPI, ParMETIS, Python, ScaLAPACK, SCOTCH, Valgrind, zlib
- various 'bug' fixes, including:

  - fix source URL for lockfile in Python easyconfigs (#428)
  - correct dependency specifications w.r.t. versionsuffix and toolchain (#431)
    * this is required to support building the affected easyconfigs with a custom module naming scheme
  - correct PSI patch file to avoid errors w.r.t. memcpy not being in scope (#446)
  - fix gettext build with adding `--without-emacs` configure options, add gettext as dependency for a2ps (#448)
  - exclude EMACS support in a2ps because of build failures (#452)

11.12.75 EasyBuild v1.7.0 (September 2nd 2013)

feature + bugfix release

framework

- various enhancements, including:
  - also search for patch files in directory where easyconfig file is located (#667)
  - reduce false positives in reporting of possible errors messages (#669)
  - make module update `$ACLOCAL` if a share/aclocal subdir is found (#670)
  - add `unwanted_env_vars` easyconfig parameter to list environment variables to unset during install procedure (#673)
  - add support for updating list easyconfig values (next to string values) (#676)
  - add `--dry-run` command line option which prints installation overview for specified easyconfig files (#677)
- various bug fixes, including: * ensure that all extensions are listed in `$EBEXTSLISTX` set by module, also when using `--skip` (#671) * report reason for failed sanity check for extensions (#672, #678) * fix `--list-toolchains` command line option (#675)

easyblocks

- added support for **3** new software packages that require customized support:
  - Libint2 (#236), Qt (#210), Rosetta (#218)
- various enhancements, including:
  - allow building OpenFOAM without 3rd party tools (#230)
  - also add sitelib path to `$PERL5LIB`, refactor code to add generic `get_site_suffix` function (#232, #233)
  - support building imkl FFT wrappers using MVAPICH2 MPI library (#234)
  - remove libnpp from CUDA sanity check to support installing CUDA v5.5 (#238)
  - pick up `$INTEL_LICENSE_FILE` for Intel tools, if it is set (#243) * note: gets preference over `license_file` easyconfig parameter
- various bug fixes:
  - call WRF build script with '`tcsh <script>` to ensure that tcsh available in `$PATH` is used (#231)
  - make sure some environment variables that may disrupt the GCC install procedure are unset (#237) * e.g., `$CPATH`, `$C_INCLUDE_PATH`, `$CPLUS_INCLUDE_PATH`, `$OBJC_INCLUDE_PATH`, `$LIBRARY_PATH`

  - code cleanup in GEANT4 easyblock: use `self.version` (instead of `self.get_installversion()`) (#242)
  - enhance list of `noqanda` patterns for CUDA, to get less failing installations (#244)

easyconfigs

- added example easyconfig files for **15** new software packages:
  - Glib (#294, #400), GLPK (#400), horton (#413), libint2 (#413), molmod (#413), Rosetta (#336), SCons (#336), Stacks (#367, #377), sympy (#413), Qt (#294), XML-LibXML (#397), XML-Simple (#397), yaff (#413), YAML-Syck (#380), zsh (#376)
- added additional easyconfigs for various software packages:
  - BLAST, BamTools, BioPerl, Bison, Boost, bzip2, CMake, Cython, CUDA, FFTW, FIAT, GCC, GMP, gettext, git, h5py, HDF5, libffi, libreadline, libxc, matplotlib, METIS, ncurses, Oases, Python, RAxML, ScientificPython, Szip, tcsh, imkl, MVAPICH2, TotalView, VTune, WRF, zlib
- added toolchain easyconfig files for HPCBIOS policies (#402, #407)
  - HPCBIOS_BioInfo, HPCBIOS_Debuggers, HPCBIOS_LifeSciences, HPCBIOS_Math, HPCBIOS_Profilers
- various enhancements, including:
  - added more XML Perl modules to non-bare Perl easyconfigs (#375)
- various 'bug' fixes, including:
  - fix website/description in scipy easyconfigs (#399)
  - specify OpenMPI libibverbs-dev(el) OS dependency in an OS-dependent way (#403)
  - add patch file for M4 to fix building on systems with recent glibc (>=2.16) (#406)
  - align moduleclass in R easyconfigs (#411)
  - fixed filename of Biopython/CD-HIT easyconfig files (#407)
  - disable parallel building of otcl (#419)

11.12.76 EasyBuild v1.6.0 (July 11th 2013)

feature + bugfix release

framework

- added support for using Lmod as module tool (#645)
- various other enhancements, including:
  - allow prepending to/appending to/overwriting list easyconfig parameters using `--try-amend-X` (#658, #664)
- various bug fixes, including:
  - add salt to temporary log file name (#656, #665)
  - fix determining CPU architecture on Raspberry Pi (ARM) systems (#655, #662)
  - fix support for determining base path of tarballs containing a single file (#660)

easyblocks

- added support for **2** new software packages that require customized support:

  - BamTools (#224), BLAT (#214)
- various enhancements, including:
  - update impi easyblock to allow installing impi v4.1.1, which features a slight change in build procedure (#217)
  - enhance `PackedBinary` easyblock to copy both files and directories (#212)
  - added get `sitearch_suffix` to Perl search path and use it in `PerlModule` easyblock (#209)
- various bug fixes:
  - make sure EasyBuild configuration is initialized when running unit tests (#220)
  - make Boost easyblock pick up configopts easyconfig parameter (#221)
  - add `-DMPICH_IGNORE_CXX_SEEK` compiler flag for Mothur when MPI support is enabled (#222)
  - fix Boost sanity check, only check for `libboost_python.so` if Python module is loaded (#223)
  - enhance Trinity support w.r.t. jellyfish (#225, #227)
  - fix checking for `beagle-lib` dep (deprecate checking for BEAGLE) for MrBayes (#228)

easyconfigs

- added example easyconfig files for **26** new software packages:
  - ALLPATHS-LG (#359), AutoMake (#347), BamTools (#319, #338), BLAT (#340), Biopython (#356), cairo (#361), CCfits (#327), CD-HIT (#344), CFITSIO (#327), Diffutils (#347), FASTA (#358, #361), findutils (#347), fontconfig (#361), gawk (#347), gettext (#361), GLIMMER (#357, #361), libidn (#361), LibTIFF (#347), libungif (#347), make (#355), MUSCLE (#339), Oases (#354), pixman (#361), PLINK (#352), RCS (#347), SQLite (#347)
- added additional easyconfigs for various software packages:
  - ant, Bash, Bison, bzip2, cURL, expat, GCC, EasyBuild, freetype, FFTW, GDB, git, HMMER, JUnit, libreadline, libpng, libtool, libxml2, libxslt, M4, makedepend, Mothur, MVAPICH2, Mercurial, ncurses, OpenBLAS, Python, ScaLAPACK, Tcl, tcsh, TopHat, Trinity, Valgrind, Velvet, VTune, zlib (see #169, #297, #298, #301, #309, #323, #331, #332, #341, #347, #349, #351, #355, #361)
- various enhancements, including:
  - added easyconfigs for `ictce/5.4.0`, `ictce/5.5.0` and `gmvolf/1.7.12` toolchain modules (#297, #332, #349)
  - added a template sanity_check_paths as 'MUST' in TEMPLATE.eb (#329)
  - introduced biodeps 'toolchain' to ease keeping common dependencies for bio* software in sync (#309)
  - added collection of easyconfigs for ictce/5.3.0 (#309, #323)
    - bam2fastq, bbFTP, BLAST, Boost, DL_POLY Classic, EMBOSS, FFTW, libharu, libxml2, libxslt, libyaml, lxml, Mercurial, Mothur, mpi4py, ncurses, ns, orthomcl, otcl, PAML, Perl, PyYAML, pandas, problog, scikit-learn, TiCCutils, TiMBL, TinySVM, TopHat, tclcl, YamCha
  - added missing dependencies for various software packages (#323, #328, #348, #361)
  - style fixes in various easyconfigs (#309, #323, #345, #349, #355, #361)
- various 'bug' fixes, including:
  - added `pic` toolchain option for Perl goolf easyconfig (#299)
  - fixed source URLs for R (use correct template `%(version_major)s`) (#302)
  - synced libreadline easyconfigs w.r.t. ncurses dependency (#303)

  - make sure EasyBuild configuration is initialized when running unit tests (#334)
  - specify `lowopt` (-O1) optimization level for OpenIFS, to avoid floating-point related issues (#328)
  - fix naming of 'beagle-lib' (used to be 'BEAGLE'), to avoid name clashes with other software package(s) (#346)

11.12.77 EasyBuild v1.5.0 (June 1st 2013)

feature + bugfix release

framework

- various enhancements, including:
  - define `SHLIB_EXT` constant for shared library extension (`.so`, `.dylib`), deprecate `shared_lib_ext` global var (#630)
  - enhance support for sanity checking extensions (#632, #649)
  - add support for `modextrapaths` easyconfig parameter (#634, #637)
  - allow `source_urls` to be templated for extensions (#639, #646, #647)
  - set `OMPI_*` environment variables for OpenMPI (#640)
  - make BLACS optional as toolchain element, depending on ScaLAPACK version (#638)
- various bug fixes, including:
  - fixed `--list-toolchains`, avoid listing toolchains multiple times (#628)
  - fix templating dictionary after parsing easyconfig file (#633)
  - fix support for ACML as compiler toolchain element (#632)
  - make unit tests clean up after themselves more thoroughly (#641, #642, #643)

easyblocks

- added **one** new *generic* easyblock: `MakeCp` (#208)
- added support for **5** new software packages that require customized support:
  - CBLAS (#192), FreeSurfer (#194), Mothur (#206), OpenIFS (#200), PSI (#191)
- various enhancements, including:
  - add support for building ScaLAPACK 2.x on top of QLogic MPI (#195)
  - support newer BWA versions (#199)
  - explicitly list license server type in ABAQUS install options, required for correct installation of v6.12 (#198)
  - update SCOTCH and OpenFOAM easyblock for recent versions (#201)
- various bug fixes:
  - fix numpy easyblock to get an optimal build (w.r.t. `numpy.dot` performance) (#192)
  - correct build procedure for MUMmer to yield a complete installation (#196, #197)
  - make unit tests clean up after themselves more thoroughly (#203, #204)
  - fix getting Perl version for extensions (#205)

easyconfigs

- added example easyconfig files for **23** new software packages:
  - bam2fastq (#287), CBLAS (#263), EMBOSS (#265, #290), FCM (#272), FRC_align (#273), freeglut (#271), FreeSurfer (#271), FSL (#271), GATK (#287), libharu (#290), libxslt (#235), MariaDB (#292), Mothur (#265) mpi4py (#276), OpenIFS (#272), orthomcl (#265), PAML (#287), pandas (#262), phonopy (#235), problog (#277), PSI (#258), PyYAML (#235), RAxML (#277)
- added additional example easyconfig files for:
  - ABINIT (#235), ACML (#267), BLAST (#275), Boost (#273), BWA (#270), bzip2 (#263), Chapel (#240), CMake (#290), FFTW2 (#247, #267), flex (#267), freetype (#235), grib_api (#272), gzip (#265), Java (#279), libpng (#240, #279), libreadline (#267), libxml2 (#235), libxml (#235), matplotlib (#235), MCL (#265), MUMmer (#265), ncurses (#267), numpy (#267), OpenFOAM (#267), Perl (#265), Python (#276, #263), R (#240, #279), SCOTCH (#267), ScaLAPACK (#267), TopHat (#289), Valgrind (#255), zlib (#267)
- various enhancements, including:
  - enhance unit test suite, include testing for module conflicts (#256) and presence of patch files (#264)
  - use provided constants and templates in easyconfig files where appropriate (#248, #266, #281)
- various 'bug' fixes, including:
  - get rid of hardcoded license_file paths for VTune, Inspector (#253)
  - assign proper moduleclass where they were missing (#278)
  - fix naming for LZO (#280)
  - make unit tests clean up after themselves more thoroughly (#283, #284, #285, #286)
  - fix TopHat dependencies (#289)
  - fix source URLs for XML (#279)
  - fix versions for all listed R extensions (#279)
  - fix CUDA easyconfig file for use on Debian Squeeze (#291)
  - correct easyconfig filename and module name mismatches (bbcp, DL_POLY Classic, ...) (#295)

11.12.78 EasyBuild v1.4.0 (May 2nd 2013)

feature + bugfix release

framework

- the unit tests for easybuild-framework were moved to test/framework, to make things consistent with easybuild-easyblocks and easybuild-easyconfigs (#611, #613, #624)
  - running the framework unit tests should now be using `python -m test.framework.suite`
- various other enhancements, including:
  - extend unit test suite (#593, #599, #603, #618, #620, #622, #624, ...)
  - extended list of constants and templates that can be used in easyconfig files (#566)
  - add support for additional compiler toolchains
    - CUDA-enabled toolchain: `goolfc` (#603, #624)
    - Clang(+GCC)-based toolchains: `cgoolf`, `cgmpolf`, `cgmvolf` (#593, #598, #600)
    - gmvolf (GCC+MVAPICH2+...) (#585)

  - properly decode easyblock to module name using `decode_*` functions (#618)
- various bug fixes, including:
  - fixed default value for `--stop` (#601)
  - remove useless `sleep()` calls in `run_cmd`, `run_cmd_qa` (#599)
  - determine module path based on class name, not software name (#606)
  - remove unwanted characters in build dirs (#591, #607)
  - ignore some error codes spit out by modulecmd that are actually warnings (#609)
  - fix `agithub.py` w.r.t. changes in GitHub API (user-agent string is now obligatory for non-authenticated connections) (#617)
  - fix typo breaking the `adjust_cmd` decorator on SuSE (#615)
  - fix prepending paths with absolute paths in module file (#621)
  - clean up open file handles properly (#620, #624)
  - fix `--search` help and implementation (#622)

easyblocks

- added a unit test suite for easyblocks (#175, #177, #178)
  - every easyblock is parsed and instantiated as a sanity check
- added **one** new *generic* easyblock: `PerlModule` (#183)
- added support for **8** new software packages that require customized support:
  - ABAQUS (#179), Bowtie (#174, #185, #186), Clang (#151), DL_POLY Classic (#118), ESMF (#171), Perl (#183), Intel VTune and Intel Inspector (#180)
- the `CMakeMake.configure_step` parameter `builddir` was renamed to `srcdir`, because the name `builddir` is incorrect (#151)
  - `builddir` will remain supported for legacy purposes up until v2.0
- various enhancements, including:
  - reverted back to hardcoding Python library path, since it's hardcoded by setuptools too (#184)
  - added MPICH support in ScaLAPACK easyblock (#172)
  - enhanced NCL easyblock: add support UDUNITS and ESMF dependencies (#171)
  - enhanced MATLAB easyblock: avoid hardcoding Java options, make sure `$DISPLAY` is unset, extend list of sanity check paths (#181)
  - enhanced TotalView easyblock: add support for license file (#146)

easyconfigs

- added a unit test suite for easyconfigs (#228, #230)
- added example easyconfig files for **20** new software packages: * ABAQUS (#231), BioPerl (#242), Bowtie (#227), Clang (#177), CRF++ (#131), DL_POLY Classic (#132), ESMF, GROMACS (#165), HH-suite (#219), Inspector (#232), likwid (#131), Perl (#242), scikit (#133), TiCCutils (#131), TiMBL (#131), TinySVM (#131), UDUNITS (#167), VTune (#232), YamCha (#131)
- add example easyconfigs for new compiler toolchains (use `eb --list-toolchains` for a full list of supported toolchains):

  - the newly added toolchains only differ in compilers/MPI library, and all feature OpenBLAS, LAPACK, ScaLAPACK and FFTW
  - *goolfc*: GCC, CUDA (co-compiler), OpenMPI (#191)
    - a goolfc easyconfig for GROMACS is included as proof-of-concept (#165)
  - *cgmpolf*: Clang (C/C++ compilers), GCC (Fortran compilers), MPICH (#213)
  - *cgmvolf*: Clang, GCC, MVAPICH2 (#218)
  - *cgoolf*: Clang, GCC, OpenMPI (#213)
  - *gmvolf*: GCC, MVAPICH (#202, #222)
- ported already available easyconfigs to new compiler toolchains:
  - *ictce-5.3.0*: 193 easyconfigs (#229)
  - *cgmpolf*: 11 easyconfigs (#213)
  - *cgmvolf*: 11 easyconfigs (#218)
  - *cgoolf*: 12 easyconfigs (#213)
  - *gmvolf*: 10 easyconfigs (#215)
- added additional example easyconfig files for:
  - CMake (#163), git (#210), Java (#206), #221, Mercurial (#201, #215), ncurses (#225), TotalView (#160)
- various enhancements, including:
  - added ESMF and UDUNITS dependencies to NCL easyconfigs (#211)
  - changed value of source_urls in R easyconfigs, to be generic enough for version 3.0 and possibly beyond (#251)
- various 'bug' fixes, including:
  - add `--enable-mpirun-prefix-by-default` configure option for all OpenMPI easyconfigs (#205)

11.12.79 EasyBuild v1.3.0 (April 1st 2013)

feature + bugfix release

framework

- added script to bootstrap EasyBuild with EasyBuild, see https://github.com/easybuilders/easybuild/wiki/Bootstrapping-EasyBuild (#531)
- reorganize framework/easyconfig.py module into framework/easyconfig package with modules (#574, #580)
- support EasyBuild configuration via command line, environment variables and configuration files (#529, #552, #556, #558, #559)
- various other enhancements, including:
  - extended set of unit tests for eb command line options and EasyBuild configuration (#517, #556, #559, #571)
  - made `--search` also useful when easybuild-easyconfigs package is not installed (#524)
  - extended set of default module classes (#525)
  - add support for license easyconfig parameter (which will be mandatory in v2.x) (#526, #569)

  - added `setup.cfg` for `setup.py` to allow creating RPMs (#528)
  - added support for obtaining system information, see `get_os_*` functions in `easybuild/tools/systemtools.py` (#543, #546, #547)
  - added support for iterated builds that require cycling over multiple sets of configure/build/install options, e.g. FFTW (#549)
  - added support for OpenBLAS as toolchain lib for linear algebra (#537, #565)
  - added support for tweaking prefix and separator for library toolchain variables (`LIB*`) (#572, #576)
  - skip already built modules in regression testing mode, to ease regression testing (#582)
- various bug fixes, including: * added `zip_safe` flag to `setup.py`, to silence multitude of warnings (#521) * only define `LIBFFT` for Intel MKL if FFTW interface libraries are available (#518, #567, #579) * set POSIX group early in build process, make EasyBuild aware of consistent chmod/chown failures (#527) * properly order the name/version keys for the toolchain easyconfig parameter when using `--try-toolchain` (#563) * take the `skipsteps` easyconfig parameter into account in regression testing mode as well (#583)

easyblocks

- added support for **2** new software packages that require customized support:
  - CUDA (#145), Ferret (#160, #163)
- remove `license` easyconfig parameter from `IntelBase`, since it clashes with generic `license` parameter (#153, #158)
  - `license_file` should be used instead (see framework#569)
  - using `license` instead of `license_file` will be supported until v2.x for legacy purposes
- various enhancements, including:
  - stop hardcoding Python site-packages library dir, obtain it via `distutils.sysconfig` instead (#141, #156, #159, #161)
  - stop hardcoding list of libraries for BLAS libs, ask toolchain modules or use `$LIBBLAS` instead (#150, #155)
  - enhanced CP2K easyblock, following Intel guidelines for `ictce` builds (#138)
  - added `setup.cfg` for `setup.py` to allow creating RPMs (#140)
  - clean up specifying BLAS/LAPACK libs for building numpy, check whether requires patch is being used for IMKL builds (#155, #161)
- various bug fixes, including:
  - added `zip_safe` flag to `setup.py`, to silence multitude of warnings (#135)
  - install EasyBuild packages in reversed order to avoid funky setuptools issues (#139)
  - fixed a typo in the ScaLAPACK easyblock, and set CCFLAGS and FCFLAGS for v2.x (#149, #162)
  - fix sanity check for python-meep (#159)
  - exclude Python tests from DOLFIN sanity check, since they hang sometimes (#161)
  - add support in ALADIN easyblock for answering question on location of `libgrib_api.a` (#165)

easyconfigs

- added example easyconfig files for **13** new software packages:
  - Bash, CUDA, ccache, Ferret, gzip, libxc, ns, numactl, OpenBLAS, otcl, Tar, tclcl, tcsh

  - several of these easyconfig files were contributed by attendees of the EasyBuild hackathon in Cyprus!
- added example easyconfigs for goolf toolchain (#158)
- added example easyconfigs for builds with goolf toolchain, i.e. for all goalf easyconfigs (#189)
  - for several software packages, a patch file was needed to get them to build with GCC 4.7:
    * AMOS, BEAGLE, Cufflinks, DOLFIN, GATE, ns, Pasha, Trilinos, Trinity
  - for PETSc, a patch file was required to make it build with a toolchain that doesn't include BLACS
- added additional example easyconfig files for:
  - gompi, hwloc, LAPACK, MVAPICH2, OpenMPI, ScaLAPACK
- various enhancements, including:
  - define a proper module class in *all* easyconfigs, cfr. default module classes defined in framework (#150, #159, #161, #162, #179, #181)
  - extend FFTW easyconfig to 'fat' builds that include single, double, long double and quad precision libraries in the same module
    * quad precision is disabled for Intel compiler based builds (it requires GCC v4.6+)
  - the imkl easyconfigs used for the ictce 3.2.2.u3 toolchain now also enable FFTW interfaces
- various 'bug' fixes, including:
  - fix filename for Mercurial and ROOT easyconfig files
  - fix homepage/description for Hypre
  - fix enabling OpenMP support in OpenMPI: use `--enable-openmp`, not `--with-openmp`
  - use correct configure flag for enabling OpenMPI threading support in v1.6 (#186)
    * `--enable-mpi-thread-multiple` instead of `--enable-mpi-threads`
  - explicitly add `--without-openib --without-udapl` configure options in OpenMPI easyconfig using versionsuffix `-no-OFED` (#168)
    * this avoids that OpenMPI auto-detects that it can enable Infiniband (OpenIB) support, which doesn't fit the -no-OFED versionsuffix
    * Note: this makes goalf-1.1.0-no-OFED effectively not suitable to produce software builds that are IB-capable!
  - remove explicit `--with-udapl` from OpenMPI easyconfigs, does more harm than good (#178)
  - remove libibvers, libibmad, libibumad as explicit dependencies for OpenMPI/MVAPICH2 (#173, #182)
    * leave it up to the OS to provide these, since the required version is too much tied to the version of IB drivers
  - use `license_file` in Intel tools easyconfigs, as opposed to the new generic `license` parameter with a different meaning (#180)
  - modify patch for impi to avoid installation problems due to hardcoded path in /tmp (#185)
    * now uses `$USER-$RANDOM` subdir to avoid clashes between different users on the same system
  - the patch file for numpy was extended to also supporting ATLAS and other BLAS libraries spread across multiple directories
    * the extension for ATLAS is required because we now no longer rely on the automatic numpy mechanism to find the ATLAS libs

  - fixed dependencies:
    * libibumad as dependency for libibmad
    * ncurses as dependency for libreadline
    * ncurses and zlib as dependency for SAMtools (+ enhanced patch)
    * remove explicit FFTW dependency for Meep, ... since toolchain already provided FFTW

11.12.80 EasyBuild v1.2.0 (February 28th 2013)

feature + bugfix release

framework

- new backend module for option parsing: generaloption
- support for using constants and string templates in easyconfig files
  - currently disabled for `exts_filter` and `exts_list` easyconfig parameters, for backward compatibility
- various other enhancements, including:
  - support for iqacml and iiqmpi toolchains (Intel compilers + QLogic MPI based)
  - clearer errors messages when sanity check failed
  - unit tests for (about half of) the `eb` command line options
  - support for specifying build/install steps to skip in easyconfig file (`skipsteps`)
  - support for allowing certain dependencies to be resolved by the system instead of modules (`allow_system_deps`)
  - cache ppn value required by regtest, clean up temporary files let behind by `--regtest`/`--job`
  - make sure MPD is used as process manager for Intel MPI (required for impi v4.1 and later)
  - rename template names `name` and `version` used in exts_filter to `ext_name`, `ext_version`
    * `name` and `version` are still supported for legacy reasons
  - cleaned up module docstrings w.r.t. list of authors
- various bug fixes, including:
  - print correct (lowercase) toolchain names with `--list-toolchains`
  - correct easyconfig parameter name `license_server_port`
  - fix string quoting in develop modules
  - ensure `modulecmd` is run with original `$LD_LIBRARY_PATH` value
    * to avoid breaking `modulecmd`, see https://bugzilla.redhat.com/show_bug.cgi?id=719785
  - remove use of hardcoded files/dirs in unit tests
  - fix various inconsistencies w.r.t. paths considered with `--robot`
  - various cleanup and fixes w.r.t. logging
    * use correct logger instance in main script
    * stop passing logger instances around
    * make module logging variables private

  - get rid of `ModuleGenerator` deconstructor, clean up via `EasyBlock.clean_up_fake_module`
  - fix disabling of `optarch` toolchain option (and extend unit tests to check on this)

easyblocks

- added **one** new *generic* easyblock: `Rpm`
- added support for **6** new software packages that require customized support:
  - EasyBuild, EPD (Enthought Python Distribution), freetype, MATLAB, QLogic MPI (RPM), TotalView
  - support for installing EasyBuild with EasyBuild enables bootstrapping an EasyBuild installation!
- various enhancements, including:
  - corrections in WRF/WPS to also enable building with iqacml toolchain
    - use `mpi_cmd_for` instead of hardcoding test commands, using correct Fortran compilers (F90)
  - fix NCL easyblock to also support v6.1.x
    - use correct Fortran compiler (F90), set correct lib/include paths for dependencies (netCDF-Fortran, GDAL)
  - cleanup sweep of license headers and authors list in easyblock module docstrings
  - use new `ext_name` template name in `exts_filter` in Python and R easyblocks
- various bug fixes, including:
  - general code cleanup
    - don't set `sanityCheckOK` in `Toolchain` easyblock
    - get rid of using `os.putenv`
  - NEURON easyblock: don't hardcode number of processes used in test cases
  - make sure `easybuild.easyblocks.generic` namespace is extendable

easyconfigs

- added example easyconfig files for **41** new software packages:
  - a2ps, AnalyzeFMRI, aria, bbcp, bbFTP, bbftpPRO, binutils, Bonnie++, ccache, cflow, cgdb, Corkscrew, EasyBuild, Elinks, EPD, FLUENT, fmri, GDB, GDAL, gnuplot, gnutls, gperf, Iperf, lftp, libyaml, lzo, MATLAB, mc, nano, NASM, nettle, numexpr, parallel, pyTables, QLogic MPI, Stow, TotalView, Valgrind, VTK, Yasm, zsync
- added example easyconfigs for iqacml and iiqmpi toolchains
- added additional example easyconfig files for:
  - ABINIT, ABySS, ACML, BFAST, Bison, BLACS, Cython, cURL, Doxygen, FFTW, flex, g2clib, g2lib, GHC, h5py, HDF, HDF5, HMMER, JasPer, icc, ictce, ifort, imkl, impi, libpng, libreadline, M4, matplotlib, MCL, MEME, mpiBLAST, NCL, ncurses, netCDF, netCDF-Fortran, NWChem, R, ScaLAPACK, Tcl, Tk, WPS, WRF, zlib
- various enhancements, including:
  - fix version of xtable R library in list of extensions for R, to avoid installation failures
- various 'bug' fixes, including:
  - fix toolchain and file names for ABINIT easyconfigs
  - fix sanity check paths for cURL

  - don't disable `optarch` for WRF, it's not needed (only disable heavy optimizations is required)
  - fix homepage/description for ALADIN

11.12.81 EasyBuild v1.1.0 (January 27th 2013)

feature + bugfix release

framework

- improvements w.r.t. support for software extensions (tested on Python and R, see easyblocks package)
  - cleaned up support for building/installing extensions
  - define `ExtensionEasyblock` class that implements support for installing extensions as stand-alone module as well
  - return to build dir before building/installing each extension
  - define `EBEXTSLIST<NAME>` environment variable in module if `exts_list` was defined
  - make sure sanity check for extensions results in an error if it fails
- various enhancements, including:
  - log both framework and easyblocks versions
  - add support for `gimkl`, `gmacml`, `iccifort`, `iomkl` and `ismkl` toolchains
  - define `*_SEQ compiler` variables for sequential compilers
  - add `--list-toolchains` command line option for listing supported toolchains
  - add support for customizing software and modules install path suffixes
  - support both setuptools and distutils installation methods for finding installed easyconfigs
  - also consider robot path in list of paths searched for patch files
  - allow skipping of default extension sanity check (by setting `modulename` to False in options)
- various bug fixes, including:
  - typos in toolchain Python modules w.r.t. `imkl` support, handling of `i8`/`optarch`/`unroll` options
  - purge before loading 'fake' module, unload 'fake' module before removing it, use original `$MODULEPATH`
  - restore environment after unloading fake module, set variables that were incorrectly unset, i.e., that were defined before as well
  - unset `$TMPDIR` for builds submitted as jobs (required by `IntelBase` easyblock)
  - correctly track easyconfig parse error
  - always run all jobs in regtest, also if dependency jobs failed
  - cosmetic adjustments to default EasyBuild configuration file to avoid confusion between e.g. `build_dir` and `build_path` (only latter matters)
  - fix SuSe hack, only prefix command with sourcing of `/etc/profile.d/modules.sh` if it is there
  - leave build directory before it is removed during cleanup step
  - load generated module file before running test cases

easyblocks

- added **3** new *generic* easyblocks: `CMakePythonPackage`, `JAR`, `RPackage`

- added support for **23** new software packages that require customized support:
  - ACML, ALADIN, ant, Bioconductor (R packages), Chapel, Cufflinks, ESPResSo, FLUENT, Geant4, GHC, Java, NEURON, NWChem, PyZMQ, QuantumESPRESSO, R, Rmpi, ROOT, Rserve, SCOOP, Trinity, VSC-tools, XML
- various enhancements, including:
  - clean up of `python.py` easyblock:
    - merge `EB_DefaultPythonPackage` and `PythonPackage` easyblocks into generic *easyblock* `PythonPackage`, which derives from `ExtensionEasyblock`
    - move `EB_FortranPythonPackage` into dedicated *generic* `FortranPythonPackage` easyblock module
    - split off support for building/installing `nose`, `numpy`, `scipy` into dedicated `EB_*` easyblock modules, which allows them to be built as stand-alone modules as well
  - clean up testing of Python packages (`PythonPackage` easyblock)
  - make sure there is no `site.cfg` in home dir when building Python packages, because e.g. `scipy` will pick it up
  - added support for building Intel MKL wrappers with OpenMPI toolchain
  - cleaning up of fake module that was loaded for running tests
  - move calls to functions that rely on environment up in the chain of steps (mostly for cleanup reasons)
  - use better module name for UFC sanity check, minor change to sanity check paths for UFC
- various bug fixes, including:
  - only call `make ptcheck` for ATLAS when multi-threading support is enabled
  - use a symbolic link for $HOME/intel instead of a randomly suffixed subdirectory in home and patching of Intel install scripts
    - latter does not work anymore with recent versions of Intel tools (2013.x)

easyconfigs

- added example easyconfig files for **48** new software packages:
  - ABINIT, ABySS, ACML, ALADIN, ant, BFAST, BLAST, Chapel, CLHEP, Cufflinks, ESPResSo, GATE, GHC, Geant4, Greenlet, google-sparsehash, grib_api, HMMER, Java, JUnit, libibmad, libibumad, libibverbs, MCL, MDP, MEME, mpiBLAST, NCBI Toolkit, NEURON, NWChem, numpy, MDP, Oger, OpenPGM, paycheck, PyZMQ, QuantumESPRESSO, R, ROOT, SCOOP, scipy, Tophat, Trinity, util-linux, VSC-tools, wiki2beamer, XML, ZeroMQ
- added example easyconfigs for gmacml, gmvapich2, iccifort, ictce, iomkl toolchains
- added additional example easyconfig files for:
  - ATLAS, BLACS, Boost, Bowtie2, bzip2, CP2K, Doxygen, FFTW, GCC, HDF5, hwloc, icc, ifort, imkl, impi, JasPer, Libint, libreadline, libsmm, libxml, ncurses, netCDF, M4, Meep, MVAPICH2, OpenMPI, Python, ScaLAPACK, Szip, zlib
- various enhancements, including:
  - major style cleanup of all example easyconfig file (PEP008 compliance)
  - added setuptools to list of Python extensions
  - get rid of `parallel` versionsuffix for HDF5, as its meaningless (MPI-enabled build is always parallel)

- various 'bug' fixes, i.e. added missing dependencies or replaced OS dependencies with proper dependencies

11.12.82 EasyBuild v1.0.2 (December 8th 2012)

bugfix release

framework

- properly catch failing sanity check commands
- fix bug in toolchain support which cause linking environment variables set by toolchain to include too many libraries
  - elements in toolchain variables were being passed by reference instead of by value
- fix selecting a compiler toolchain for a specified software package (`--software-name`) if only a template is a viable option
- fix passing command line parameters with `--job`
- fix list of valid stops (`-s`/`--stop`)
- fix minor issues in help messages (`-h`/`--help`)

easyblocks

- fix typos in WIEN2k easyblock (missing commas after list elements)

easyconfigs

- fixed source URL for ligtextutils (toolchain refactoring error)

11.12.83 EasyBuild v1.0.1 (November 24th 2012)

bugfix release

framework

- fix support for installing with `distutils` (broken import in `setup.py`)
- fix support for ACML as a toolchain element (`toolchains/linalg/acml.py`)
- add name to aggregated regtest XML so that is parsed correctly by Jenkins
- reorder code in `main.py` so that regtest also works with incomplete easyconfig paths
- add bash script for running regression test and sending a trigger to Jenkins to pull in the XML with results
- get rid of assumption that loaded modules should have name like `foo/bar`, make it more flexible
- retry failed builds in regtest twice to ignore fluke errors
- report leaf nodes in dependency graph when regtest is submitted
  - this is required for setting job dependencies in the regtest script for the Jenkins trigger job
- implement and use rmtree2 as more (NFS) robust replacement for `shutil.rmtree`
- bump max hit count for `run_cmd_qa` from 20 to 50, to make false positives of unasnwered questions less likely

easyblocks

- fix support for installing with `distutils` (broken import in `setup.py`)
- only build GMP/CLooG/PPL libraries during GCC build in parallel, don't install in parallel

  - `make -j N install` doesn't work consistently for GMP
- fix GCC build on OS X
  - location of libraries is slightly different (`lib` vs `lib64` dir)
- add support to `ConfigureMake` easyblock for pre-passing tar options to configure
  - see `tar_config_opts` easyconfig parameter
  - workaround for issue with pax hanging `configure` with an interactive prompt
- enhance Q&A for WRF and WIEN2k by adding entries to `qa` dict and `no_qa` list
- use `rmtree2` from `tools/filetools.py` as more (NFS) robust replacement for `shutil.rmtree`

easyconfigs

- remove patch file for OpenMPI to resolve issue with pax hanging configure
  - `tar_config_opts` should be enabled as needed
- disable parallel build for PAPI, seems to be causing problems

11.12.84 EasyBuild v1.0 (November 13th 2012)

- split up EasyBuild across three repositories: framework, easyblocks and easyconfigs
- packaged EasyBuild, different parts can now be installed easily using `easy_install`

framework

- various changes to both internal and external API:
  - renamed main script to main.py (from `build.py`)
  - file and directory organisation
  - module, class, function and function argument renaming and reorganisation
  - split up `Application` into `EasyBlock` and `ConfigureMake` (see easybuild-easyblocks for the latter)
  - created `EasyConfig` class for handling easyconfig files
  - renaming of EasyBuild configuration parameters (non-camelCase)
  - renaming of various easyconfig parameters (non-camelCase)
  - rename `SOFTROOT` and `SOFTVERSION` environment variables set in generated module files to `EBROOT` and `EBVERSION`
  - use 'extension' as generic terminology for Python packages, R libraries, Perl modules, ...
- added support for building software packages in parallel via PBS jobs
- added unit testing framework and initial set of unit tests for basic functionality
  - and run them in Jenkins continuous integration service, see https://jenkins1.ugent.be/view/EasyBuild/
- implement single-command regression test (e.g. to test building all supported software)
  - eb `--regtest` –robot
- switch to new style Python classes
- replaced `toolkit` module with `toolchain` package (total rewrite), providing modular support for toolchains
- adjust default EasyBuild configuration to only use `$HOME/.local/easybuild` by default

- added support for running EasyBuild without supplying an easyconfig file
  - make EasyBuild search for a matching easyconfig file
  - support automatic rewriting of an existing partially-matching easyconfig file (use this with care!)
  - support for automatically generating an easyconfig file according to given specifications (best effort!)
- add support for looking for easyconfig file in Python search path if it can't be found on specified (relative) path (that way, easyconfig files available in the easybuild-easyconfigs package can be used easily)
- various other enhancements and bug fixes, e.g.:
  - extended sanity check capabilities
  - cleaned up logging
  - creating of devel module which allows to mimic environment that was used by EasyBuild to build the software
  - support for creating dependency graphs for a set of easyconfig files
  - grouped options in help output and categorised available easyconfig parameters
  - more consistent code style

easyblocks

- implement class name encoding scheme, see wiki https://github.com/easybuilders/easybuild/wiki/Encode-class-names
  - (non-generic) easyblock class names are now prefixed with `EB_` and non-alphanumeric characters are escaped
- split off generic easyblocks into separate package `easyblocks.generic`
- added custom support for **39** software packages:
  - Armadillo, BiSearch, Boost, Bowtie2, BWA, bzip2, CGAL, CPLEX, DOLFIN, Eigen, flex, FSL, Hypre, libxml2, MetaVelvet, METIS, MTL4, MUMmer, ncurses, OpenFOAM, OpenSSL, ParMETIS, Pasha, PETSc, Primer3, python-meep, SAMtools, SCOTCH, SHRiMP, SLEPc, SOAPdenovo, SuiteSparse, SWIG, Tornado, Trilinos, UFC, Velvet, WIEN2k, XCrySDen
- various enhancements and bug fixes to existing easyblocks

easyconfigs

- added example easyconfig files for **106** new software packages:
  - AMOS, Armadillo, ASE, Autoconf, BiSearch, Boost, Bowtie2, BWA, byacc, bzip2, CGAL, ClustalW2, CMake, CPLEX, cURL, CVXOPT, Cython, Docutils, DOLFIN, ECore, Eigen, expat, FASTX-Toolkit, FFC, FIAT, freetype, FSL, GEOS, git, glproto, GMP, gmvapich2, gompi, GPAW, GSL, guile, h5py, h5utils, Harminv, hwloc, Hypre, Infernal, Instant, Jinja2, libctl, libdrm, libffi, libgtextutils, libmatheval, libpciaccess, libpthread-stubs, libreadline, libtool, libunistring, libxcb, libxml2, makedepend, matplotlib, Meep, Mercurial, Mesa, MetaVelvet, METIS, MPFR, MTL4, MUMmer, ncurses, OpenFOAM, OpenSSL, ORCA, PAPI, ParMETIS, Pasha, PCRE, PETSc, petsc4py, pkg-config, Primer3, python-meep, RNAz, SAMtools, ScientificPython, SCOTCH, setuptools, Shapely, SHRiMP, SLEPc, SOAPdenovo, Sphinx, SuiteSparse, SWIG, Tcl, Theano, Tk, Tornado, Trilinos, UFC, UFL, Velvet, ViennaRNA, Viper, WIEN2k, xcb-proto, XCrySDen, xorg-macros, xproto
- added additional example easyconfig files (versions, builds) for several software packages
  - Bison, BLACS, Doxygen, flex, GCC, HDF5, icc, ifort, libpng, M4, MVAPICH2, OpenMPI, Szip, tbb, zlib
- replaced GCC/OpenMPI based easyconfig files with equivalents using the *gompi* toolchain * ATLAS, BLACS, FFTW, LAPACK, ScaLAPACK

- enhanced Python example easyconfig files (more dependencies required for features, e.g. libreadline, bzip2, zlib, ...)
- corrected file name of easyconfig files to adher to standard as followed by EasyBuild robot dependency resolver
- style cleanup in existing easyconfig files

11.12.85 EasyBuild v0.8 (June 29th 2012)

- added support for building/installing 17 additional software packages:
  - BEAGLE, Doxygen, g2clib, g2lib, HDF, HDF5, JasPer, libpng, Maple, MrBayes, NCL, netCDF, netCDF-Fortran, Szip, WPS, WRF, zlib
- the build procedure for WRF and WPS includes running the tests available for these packages
- various bug fixes and enhancements:
  - made support for interactive installers (`run_cmd_qa`) more robust
  - fixed Python git package check
  - implemented toolkit functions for determine compiler family, MPI type, MPI run command, ...

11.12.86 EasyBuild v0.7 (June 18th 2012)

- fixed BLACS build
  - diagnostic tools to determine `INTERFACE` and `TRANSCOMM` values are now used
- added support for building Bison, CP2K, flex
  - with regression test enabled for CP2K as part of build process
  - note: BLACS built with EasyBuild prior to v0.7 needs to be rebuilt for CP2K to work correctly
- added `--enable-mpi-threads` to OpenMPI example easyconfigs
  - required for correct CP2K psmp build
- adjusted libsmm example easyconfig for lower build time
  - to make the full regression test finish in a reasonable amount of time
- added script to make porting of easyblocks from old to new EasyBuild codebase easier
  - takes care of refactoring, checks for PyLint warnings/errors, ...
- fixed several small bugs
- prefixed EasyBuild messages with ==
- full regression test passed (58 easyconfigs tested)

11.12.87 EasyBuild v0.6 (May 11th 2012)

- added support for Intel compiler toolkit (ictce)
  - which included the Intel compilers, Intel Math Kernel Library (imkl) and Intel MPI library (impi)
- added support for building Python with nose/numpy/scipy packages
- added simple regression test

- this version is able to build all supplied example easyconfigs

11.12.88 EasyBuild v0.5 (April 6th 2012)

- first public release of EasyBuild
  - after a thorough cleanup of the EasyBuild framework of the in-house version
- supports building HPL with goalf compiler toolkit
  - the goalf toolkit consists of the GCC compilers, and the OpenMPI, ATLAS, LAPACK, FFTW and ScaLAPACK libraries
- also support build and installation of MVAPICH2