Installing Scientific Software on HPC with EasyBuild

14\textsuperscript{th} International SuperComputing Camp
https://sc-camp.org

SC-Camp 2023
Cartagena, Colombia
May 14-20, 2023

Xavier Besseron
University of Luxembourg
LuXDEM Research Team
https://luxdem.uni.lu
Outline

• Part 1: Short Introduction to EasyBuild
• Part 2: EasyBuild at work

This tutorial is based on EasyBuild tutorials available at http://tutorial.easybuild.io/.
Short Introduction to EasyBuild
What is EasyBuild?

- Software **build** and **installation** framework
- Targeted to **scientific software**
- Tailored towards **High Performance Computing** systems
- Focus points: **Performance**, **Reproducibility**, **Collaboration**

**Key features**

- Fully autonomous installation
- No admin privileges are required
- Highly configurable

**Objective**

→ **Empower scientists to self-manage their software stack on HPC**

Alternative to EasyBuild → **Spack** https://spack.io/
## Supported software

EasyBuild supports 2995 different software packages (incl. toolchains, bundles):

<table>
<thead>
<tr>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAQUS</td>
</tr>
<tr>
<td>AlphaFold</td>
</tr>
<tr>
<td>Amber</td>
</tr>
<tr>
<td>ANSYS</td>
</tr>
<tr>
<td>Blender</td>
</tr>
<tr>
<td>Boost</td>
</tr>
<tr>
<td>Caffe</td>
</tr>
<tr>
<td>Cantera</td>
</tr>
<tr>
<td>CGAL</td>
</tr>
<tr>
<td>COMSOL</td>
</tr>
<tr>
<td>Eigen</td>
</tr>
<tr>
<td>FastQC</td>
</tr>
<tr>
<td>GROMACS</td>
</tr>
<tr>
<td>gmsh</td>
</tr>
<tr>
<td>HDF5</td>
</tr>
<tr>
<td>Horovod</td>
</tr>
<tr>
<td>Hypre</td>
</tr>
<tr>
<td>Julia</td>
</tr>
<tr>
<td>JupyterLap</td>
</tr>
<tr>
<td>LAMMPS</td>
</tr>
<tr>
<td>MATLAB</td>
</tr>
<tr>
<td>matplotlib</td>
</tr>
<tr>
<td>METIS</td>
</tr>
<tr>
<td>MOOSE</td>
</tr>
<tr>
<td>MUMPS</td>
</tr>
<tr>
<td>NAMD</td>
</tr>
<tr>
<td>Nektar++</td>
</tr>
<tr>
<td>netCFD</td>
</tr>
<tr>
<td>OpenBLAS</td>
</tr>
<tr>
<td>OpenFOAM</td>
</tr>
<tr>
<td>OSPRay</td>
</tr>
<tr>
<td>pandas</td>
</tr>
<tr>
<td>PAPI</td>
</tr>
<tr>
<td>ParaView</td>
</tr>
<tr>
<td>PETSc</td>
</tr>
<tr>
<td>preCICE</td>
</tr>
<tr>
<td>PyTorch</td>
</tr>
<tr>
<td>R</td>
</tr>
<tr>
<td>RSEM</td>
</tr>
<tr>
<td>Rstudio-Server</td>
</tr>
<tr>
<td>SAGE</td>
</tr>
<tr>
<td>SAMTools</td>
</tr>
<tr>
<td>Scalasca</td>
</tr>
<tr>
<td>scikit-learn</td>
</tr>
<tr>
<td>scipy</td>
</tr>
<tr>
<td>Score-P</td>
</tr>
<tr>
<td>SQLite</td>
</tr>
<tr>
<td>STAR-CCM+</td>
</tr>
<tr>
<td>TensorFlow</td>
</tr>
<tr>
<td>Valgrind</td>
</tr>
<tr>
<td>Vampire</td>
</tr>
<tr>
<td>VASP</td>
</tr>
<tr>
<td>WRF</td>
</tr>
<tr>
<td>Yade</td>
</tr>
</tbody>
</table>

---

Full list at [https://docs.easybuild.io/version-specific/supported-software](https://docs.easybuild.io/version-specific/supported-software)
Toolchains

• Set of **compilers** and **libraries** used to build a software
• Typically
  - compilers for C, C++, and Fortran, and possibly CUDA
  - libraries for MPI, linear algebra, fast fourier transform, etc.
• Combined hierarchically to form bigger toolchains
• Defined/released twice a year → 2019a, 2019b, 2020a, 2020b, 2021a, ...

Common toolchains

• **System** toolchain: OS compilers and libraries, used to bootstrap other toolchains
• **FOSS** toolchain consists of open source components
  → GCC, Open MPI, FlexiBLAS with OpenBLAS, ScaLAPACK, and FFTW
• **Intel** toolchain based on Intel software
  → Intel C, C++, and Fortran compilers, Intel MPI, and Intel MKL libraries
Example of toolchain: FOSS 2022b

Software

- OpenMPI 4.1.4
- GCCcore 12.2.0
- binutils 2.39
- FlexiBLAS 3.2.1
- FFTW 3.3.10
- FFTW.MPI 3.3.10
- ScaLAPACK 2.2.0

Toolchains

- gompi-2022b
- GCC-12.2.0
- gfbf-2022b
- foss-2022b
EasyConfig file → recipe to build a software

Naming convention:  \(<\text{name}>-<\text{version}>-<\text{toolchain}><\text{versionsuffix}>.eb\)

- \(<\text{name}>\) represents the software name;
- \(<\text{version}>\) represents the software version;
- \(<\text{toolchain}>\) represents the toolchain: \(<\text{toolchain name}>-<\text{toolchain version}>\)
  → omitted for the system toolchain
- \(<\text{versionsuffix}>\) is an optional name to represent the variants

→ To keep in mind to search for easyconfig files

Examples

- GCC-12.2.0.eb
- OpenMPI-4.1.4-GCC-12.2.0.eb
- HDF5-1.12.2-gompi-2022a.eb
- PyTorch-1.12.1-foss-2022a.eb
- HDF5-1.12.2-GCC-11.3.0-serial.eb
- PyTorch-1.12.1-foss-2022a-CUDA-11.7.0.eb
Software Installation and Environment Modules

EasyBuild will

- install each software in its own sub-directory
  - under `<install prefix>/software/
- create a module file to use the software
  - under `<install prefix>/modules/

- The installation prefix can be customized using the environment variable `EASYBUILD_INSTALLPATH`
- Default install prefix in `~/.local/easybuild/`
- See [https://docs.easybuild.io/configuration/](https://docs.easybuild.io/configuration/)
Question?
EasyBuild at work

Practical session!

Have a look at EasyBuild documentation https://docs.easybuild.io/
or `eb --help` to understand the meaning of command line parameters.
Getting started

• Connect to Guane

  $ ssh USERNAME@toctoc.sc3.uis.edu.co
  $ ssh guane

• Get a computing node

  $ srun -N 1 -n 2 -c 1 --time 2:0:0 --pty bash
Installing EasyBuild

- Install your own version of EasyBuild
- More details on https://docs.easybuild.io/installation/

```bash
$ pip3 install --user easybuild
```

- Check the installation

```bash
$ which eb
$ eb --version
$ eb --help
```
Configuring EasyBuild

• Installation path

```
$ export EASYBUILD_INSTALLPATH=~/sccamp2023/easybuild
```

➔ Software will be installed under ~/sccamp2023/easybuild/software
➔ Modules will be installed under ~/sccamp2023/easybuild/modules

• Check the configuration

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

➔ Many configuration variables → see `eb --help`
➔ Can be specified via a configuration file, environment variables or command line parameters
➔ See https://docs.easybuild.io/configuration/
Installing Scientific Software with EasyBuild

SC-Camp 2023

Searching for available software

• Search for PyTorch

  $ eb -S PyTorch
  $ eb -S ^PyTorch.*.eb
  $ eb -S ^PyTorch.*foss-2022.*.eb

• Search for Scikit-Learn

  $ eb --search scikit-learn
  $ eb --search scikit-learn.*2022b

• Search for OSU Micro Benchmark

  $ eb -S OSU
  $ eb -S OSU-Micro-Benchmarks.*2022b
Checking dependencies and installations

• What is required to install PyTorch?

```bash
$ eb PyTorch-1.12.1-foss-2022a.eb -D
$ eb PyTorch-1.12.1-foss-2022a.eb -x
```

• What is required to install Scikit-Learn?

```bash
$ eb scikit-learn-1.2.1-gfbf-2022b.eb -D
```

• What is required to install OSU Micro Benchmarks?

```bash
$ eb OSU-Micro-Benchmarks-6.2-gompi-2022b.eb -D
```
Building on top of other installations

- Rely on software installed at another location

```
$ ls -l /home/xbesseron/sccamp-2023/easybuild/
$ ls -l /home/xbesseron/sccamp-2023/easybuild/modules/
$ ls -l /home/xbesseron/sccamp-2023/easybuild/software/

$ module use /home/xbesseron/sccamp-2023/easybuild/modules/all/
```

- Check again what is required to install Scikit-Learn

```
$ eb scikit-learn-1.2.1-gfbf-2022b.eb -D
```

- Check again what is required to install OSU Micro Benchmarks

```
$ eb OSU-Micro-Benchmarks-6.2-gompi-2022b.eb -D
```
Installing a software

- Install OSU Micro Benchmarks

  
  $ \text{eb OSU-Micro-Benchmarks-6.2-gompi-2022b.eb}$

- Install Scikit-Learn

  
  $ \text{eb scikit-learn-1.2.1-gfbf-2022b.eb}$
  $ \text{eb scikit-learn-1.2.1-gfbf-2022b.eb --robot}$
Using an installed software

• Update the MODULEPATH

```bash
$ module use ~/sccamp2023/easybuild/modules/all
$ module avail
```

• Load the module for OSU Micro Benchmarks

```bash
$ module avail OSU
$ module show OSU-Micro-Benchmarks/6.2-gompi-2022b
$ module load OSU-Micro-Benchmarks/6.2-gompi-2022b

$ echo $EBROOTOSUMINMICROMINBENCHMARKS
$ which osu_latency
```

• Load the module for OSU Micro Benchmarks

```bash
$ mpirun -n 2 osu_latency
$ mpirun -n 2 osu_bibw
```