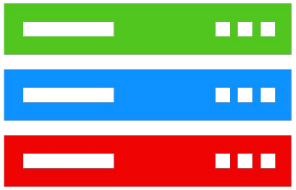


EasyBuild project proposal to join HPSF

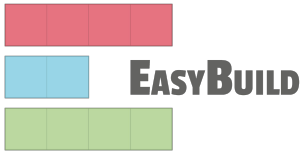
HPSF TAC meeting

1 October 2026

Kenneth Hoste (Ghent University, Belgium)

kenneth.hoste@ugent.be

What is EasyBuild?



- **EasyBuild is a software build and installation framework**
- Strong focus on scientific software, performance, HPC systems, Linux
- Open source (GPLv2), implemented in Python
- Brief history:
  - Created in-house at HPC-UGent in 2008
  - Released under GPLv2 license in April 2012 (version 0.5)
  - EasyBuild 1.0.0 released in November 2012 (during SC12)
  - **Over 110 stable releases** since then,
    latest release is EasyBuild 5.4.0 (9 Aug 2026)
  - **Worldwide community** has grown around it
    (over 1,150 members on EasyBuild Slack)

https://easybuild.io

https://pypi.org/project/easybuild

https://docs.easybuild.io

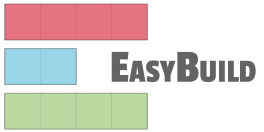
https://tutorial.easybuild.io

https://blog.easybuild.io

https://github.com/easybuilders

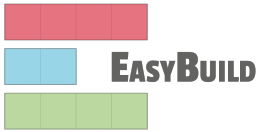
https://easybuild.io/join-slack

EasyBuild in a nutshell



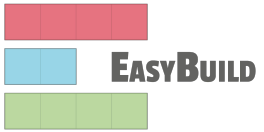
- **Tool** to provide a ***consistent and well performing*** scientific software stack
- **Uniform interface** for installing scientific software on HPC systems
- Saves time by ***automating*** tedious, boring and repetitive tasks
- Can **empower researchers** to self-manage their software stack
- **A platform for collaboration among HPC sites worldwide**
- Has become an "**expert system**" for installing scientific software

Key features of EasyBuild (1/2)



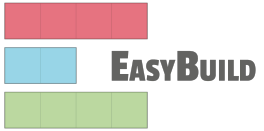
- Supports fully **autonomously** installing (scientific) software, including dependencies, generating environment module files, ...
- **No admin privileges are required** (only write permission to installation prefix)
- **Highly configurable**, easy to extend, support for hooks, easy customisation
- **Detailed logging**, **transparency** via support for “dry runs” and trace mode
- Support for using **custom module naming schemes** (incl. hierarchical)

Key features of EasyBuild (2/2)



- **Integration** with various other tools: Lmod, Slurm, Singularity/Apptainer, FPM, …
- **Actively developed and supported by worldwide community**
- **Frequent stable releases** since 2012 (roughly every 6-8 weeks)
- **Comprehensive testing**: unit tests, testing contributions, regression testing
- **Various support channels** (mailing list, Slack, conf calls) + yearly user meetings

Focus points in EasyBuild



Performance

- Strong preference for building software from source
- Software is optimized for the processor architecture of build host (by default)

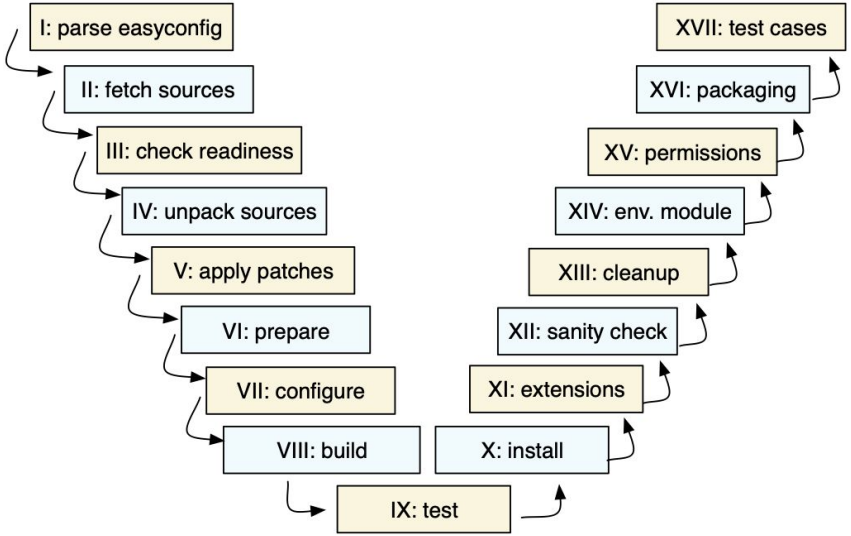
Reproducibility

- Compiler, libraries, and required dependencies are (mostly) controlled by EasyBuild
- Fixed software versions for compiler, libraries, (build) dependencies, ...

Community effort

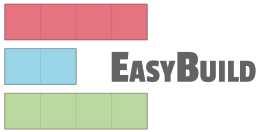
- Development is highly driven by EasyBuild community
- Lots of active contributors, integration with GitHub to facilitate contributions

Step-wise installation procedure



- EasyBuild **framework** specifies a step-wise installation procedure, but leaves some steps unimplemented
- **Easyblock** defines the implementation, can override or extend some installation steps where needed
- **Easyconfig** file provides the details for a particular installation: metadata (homepage, description), version, dependencies, toolchain, patches, configuration options, extensions, ...

EasyBuild terminology

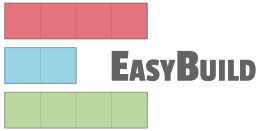


The EasyBuild **framework** leverages **easyblocks** to automatically build and install (scientific) software, potentially including additional **extensions**, using a particular compiler **toolchain**, as specified in **easyconfig files** which each define a set of **easyconfig parameters**.

EasyBuild ensures that the specified **(build) dependencies** are in place, and automatically generates a set of (environment) **modules** that facilitate access to the installed software.

An **easystack** file can be used to specify a collection of software to install with EasyBuild.

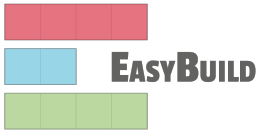
Installing & configuring EasyBuild



- EasyBuild is packaged as a **standard Python package** in PyPI, so can use **`pip install`**, `venv`, etc.
  - Or install EasyBuild with (a temporary installation of) EasyBuild (a.k.a. bootstrapping)
- Required dependencies:
  - **Python 3.6+** (only standard library, no required 3rd party Python packages required)
  - An **environment modules tool**: Lua-based Lmod or Tcl-based Environment Modules
- Python packages can be leveraged as *optional* dependencies for some specific features:
  - *archspec* for CPU detection
  - *Rich* for colors, progress bars, ...
  - *keyring* for securely storing GitHub token, required for GitHub integration features
  - *PyYAML* for parsing of easystack files
- EasyBuild can be configured via configuration files (INI), environment variables (`$EASYBUILD_*`), or command line options
- Customization of EasyBuild functionality can be done via various hooks (Python functions)

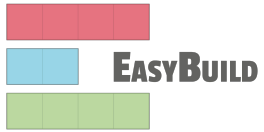
https://docs.easybuild.io/installation https://docs.easybuild.io/configuration

Dry run of software install procedure



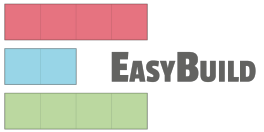
- EasyBuild can quickly unveil how it *would* install an easyconfig file, without actually installing it
- Via `eb --extended-dry-run` (or `eb -x`)
- Produces detailed output in a matter of seconds
- Software is not actually installed, all shell commands and file operations are skipped!
- Some guesses and assumptions are made, so output may not be 100% accurate...
- Any errors produced by the easyblock are reported as being ignored
- Very useful to evaluate changes to an easyconfig file or easyblock!
- **This makes EasyBuild a lot less of a black box for people new to it**

Dry run of software install procedure: example




```
$ eb Boost-1.82.0-GCC-12.3.0.eb -x
...
Defining build environment...
  ...
  export CXX='g++'
  export CXXFLAGS='-O2 -ftree-vectorize -march=native -fno-math-errno -fPIC'
  ...

configuring... [DRY RUN]

[configure_step method]
  running shell command "./bootstrap.sh --with-toolset=gcc
          --prefix=/software/Boost/1.82.0-GCC-12.3.0 --without-libraries=python,mpi"
  (in /tmp/build/Boost/1.82.0/GCC-12.3.0)
  ...
```

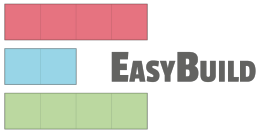
Stacking software installations



- It's easy to "stack" software installed in different locations
- EasyBuild doesn't care much *where* software is installed
- If required environment modules are available to load, EasyBuild can pick up on them up
- End users can easily manage a software stack on top of what's installed centrally!

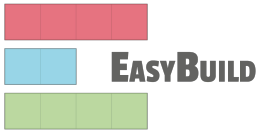
```
module use $HOME/easybuild/modules/all

eb --installpath $HOME/easybuild my-software.eb
```

Troubleshooting failing installations



- Sometimes stuff still goes wrong...
- Being able to troubleshoot a failing installation is a useful/necessary skill
- Problems that occur include (but are not limited to):
  - Missing source files
  - Missing dependencies (perhaps overlooked required dependencies)
  - Failing shell commands (non-zero exit status)
  - Running out of memory or storage space
  - Compiler errors (or crashes)
- EasyBuild keeps a thorough log for each installation which is very helpful

Easier Troubleshooting with EasyBuild v5



- **EasyBuild v5.0 makes troubleshooting failing installations significantly easier**
- When a shell command run by EasyBuild fails:
  - The problem will be reported in a more **user-friendly** way
  - You can quickly inspect (only) the output of that command
  - A script is generated to start an **interactive shell session** to debug “in context”: in the correct working directory + prepared build environment
- This was made possible by switching to the new `run_shell_cmd` function

Easier Troubleshooting with EasyBuild v5

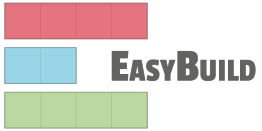
Since version 5.0, EasyBuild produces clearer error messages when a shell command failed:

```
ERROR: Shell command failed!
       full command               ->  make  -j 8 LDFLAGS='-lfast'
       exit code                  ->  2
       called from                ->  'build_step' function in /.../easyblocks/generic/configuremake.py (line 357)
       working directory          ->  /tmp/ec2-user/kenneth/easybuild/build/BCFtools/1.18/GCC-12.3.0/bcftools-1.18
       output (stdout + stderr)   ->  /tmp/eb-i61vle8x/run-shell-cmd-output/make-1ynysa6f/out.txt
       interactive shell script   ->  /tmp/eb-i61vle8x/run-shell-cmd-output/make-1ynysa6f/cmd.sh
```

- Colors to draw attention to the most important parts of the error message
- File with (only) command output + path to build directory are easy to find
- **Auto-generated `cmd.sh` script starts interactive subshell in correct build environment!**

This is powered by the new `run_shell_cmd` function that EasyBuild uses to run shell commands, which took a lot of effort, partially because all ~240 easyblocks has to be updated to use `run_shell_cmd`.

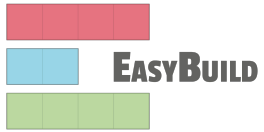
Troubleshooting: integration with LLMs (PoC)



- **Proof-of-concept integration with LLMs**, implemented as an EasyBuild hook (May 2025)
- https://github.com/boegel/easybuild-llm
- Feed output of failing shell command to a (local) LLM,

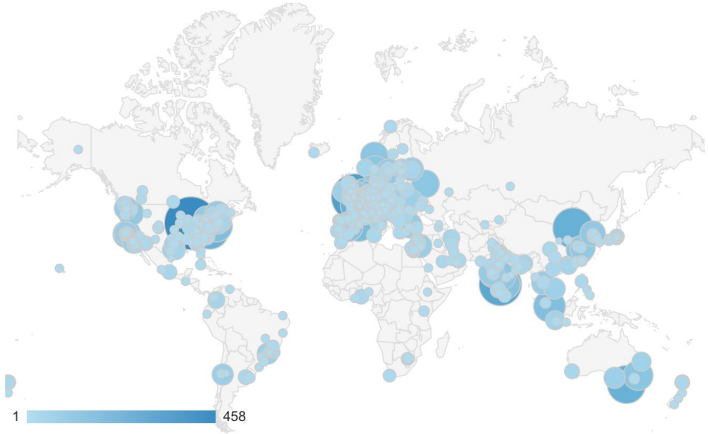
  ask it to provide a concise explanation of what went wrong.
- Next steps could be:
  - Using a more capable (local) LLM
  - Ask to explain to a human how to fix the problem
  - Agentic loop to change easyconfig file until installation works (autofix/test loop)
- Takes into account the EasyBuild AI Policy (don't force LLM-based features on everyone)

Troubleshooting: integration with LLMs (PoC)




```
$ export EB_LLM_MODEL='gemma3:1b'
$ eb --hooks eb-llm-hooks.py example.eb
...
Shell command 'make  -j 8' failed! (exit code 127)
Large Language Model 'gemma3:1b' explains it as follows:

> The error message "command not found" indicates that the shell cannot locate the
> `make` executable. This is because the shell is trying to execute a command,
> but it doesn't know where to find it.
>
> The issue is that the `make` executable is not in the system's `PATH`
> environment variable. This means the shell isn't aware of the location where
> `make` is located.

*** NOTE: the text above was produced by an AI model, it may not be fully accurate! ***
(time spent querying LLM: 3.885 sec | tokens used: input=149, output=89)

ERROR: Shell command failed!
    full command              ->  make  -j 8
    exit code                 ->  127
    called from               ->  'build_step' function in …/easybuild/easyblocks/generic/configuremake.py
(line 382)
    working directory         ->  /tmp/build/example/1.2.3/system-system/example-1.2.3
    output (stdout + stderr)  ->  /tmp/eb-d3adb33f/run-shell-cmd-output/make-c0ff33/out.txt
    interactive shell script  ->  /tmp/eb-d3adb33f/run-shell-cmd-output/make-c0ff33/cmd.sh
```

The EasyBuild community



EASYBUILD

- EasyBuild documentation is read all over the world
- Used by HPC sites, consortia, and companies
- EasyBuild mailing list (mostly dormant now)
- **EasyBuild Slack**: over 1,150 members,
  ~200 active members per week
- **Bi-weekly online conference calls** (via Zoom)
- **Yearly EasyBuild User Meeting** (EUM) since 2016

GitHub

slack

HPC sites using EasyBuild

EasyBuild is used by various (large) HPC sites to manage their central software stack

- Members of the **Flemish Supercompute Centre** in Belgium: UGent, UAntwerp, KU Leuven, VUB
- **Jülich Supercomputing Centre** (JSC) in Germany, which was an early adopter outside of Flanders (since ~2013)
- **EuroHPC supercomputers** like Deucalion (Portugal), MeluXina (Luxembourg), Karolina (Czechia), Vega (Slovenia), …
- The distributed **LUMI User Support Team** (LUST) for LUMI in Finland
- **European HPC sites**, SURF in Netherlands, EMBL + TU Dresden in Germany, various HPC sites in Sweden, Norway, …
- Univ. Birmingham, Imperial College London, Univ. of York, Univ. Oxford, Francis Crick Institute in **United Kingdom**
- Fred Hutch Cancer Research Centre in Seattle, Texas A&M University, Michigan State University, … in **United States**
- **Digital Research Alliance of Canada** (formerly known as Compute Canada)
- **Other HPC sites** beyond Europe & North America, incl. University of Hong Kong, Univ. of Melbourne in Australia, …
- **Consultancy companies** incl. Do IT Now, Red Oak Consulting, E4

GitHub integration features

- EasyBuild has strong integration with GitHub since 2016, which facilitates contributions
- Some additional Python packages required for this: GitPython, keyring
- Also requires some additional configuration, incl. providing a GitHub token
- **Enables creating, updating, reviewing pull requests using `eb` command!**
- Makes testing contributions very easy: over ~2,700 easyconfig pull requests in 2025!
- PRs created via this feature are also better “formatted”, which makes review easier
- Extensively documented: docs.easybuild.io/integration-with-github

Opening a pull request in 1, 2, 3

```
$ mv sklearn.eb scikit-learn-1.4.2-gfbf-2023a.eb
$ mv scikit-learn*.eb easybuild/easyconfigs/s/scikit-learn
$ git checkout develop && git pull upstream develop
$ git checkout -b scikit_learn_142_gfbf_2023a
$ git add easybuild/easyconfigs/s/scikit-learn
$ git commit -m "{data}[gfbf/2023a] scikit-learn v1.4.2"
$ git push origin scikit_learn_142_gfbf_2023a
```

+ log into GitHub to actually open the pull request (clickety, clickety...)

one single `eb` command

no git commands

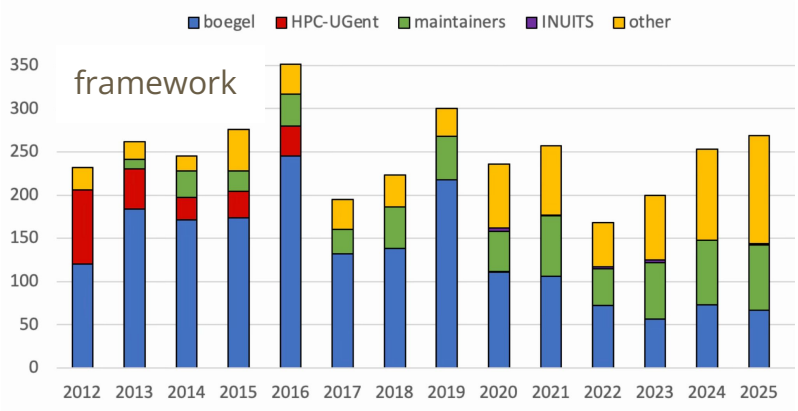
no GitHub interaction

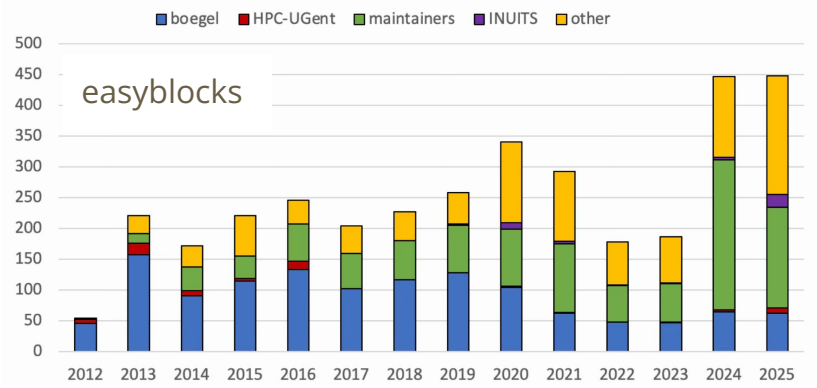
metadata is automatically derived from easyconfig

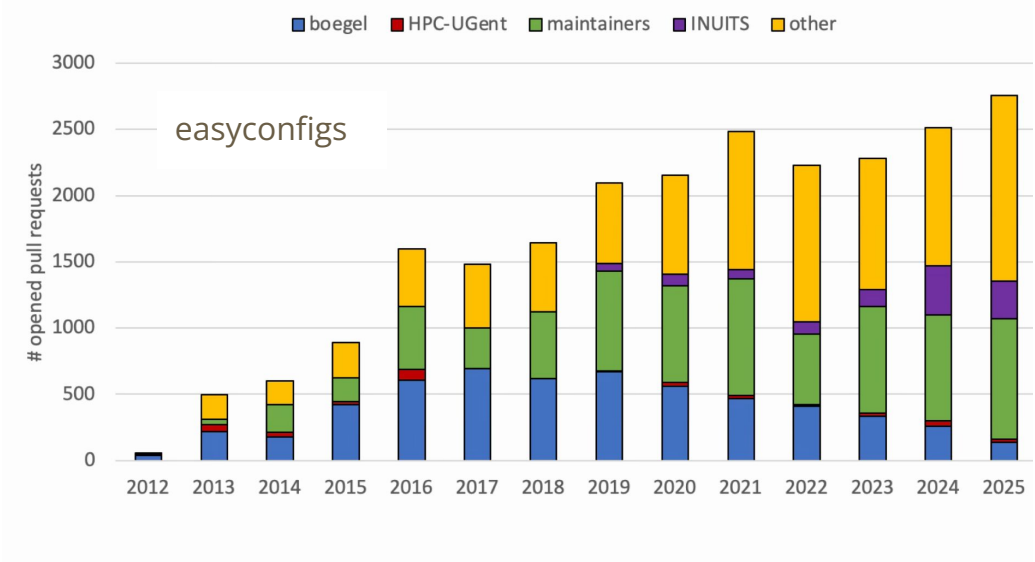
saves a lot of time!

`eb --new-pr sklearn.eb`

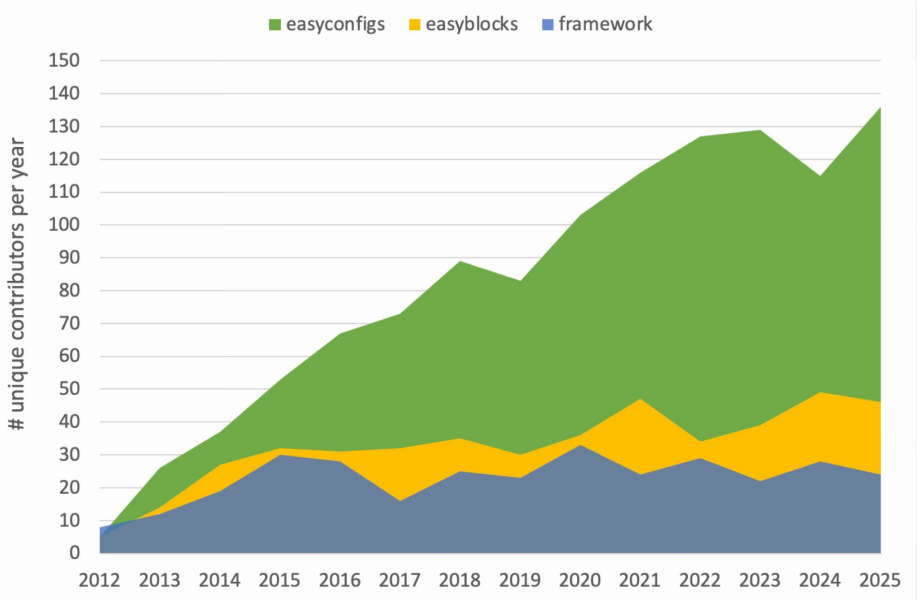
Contributions to EasyBuild (per repo/year)

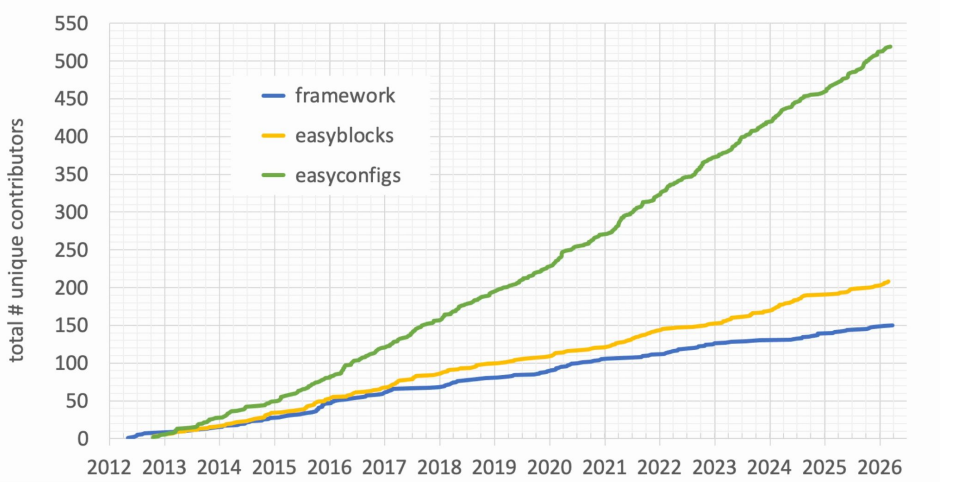






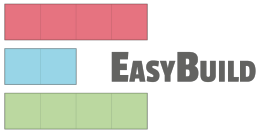
EasyBuild Contributors (per year + since dawn of time)





- Over 130 unique contributors to easyconfigs in 2025, 46 to easyblocks, 24 to framework
- Recent milestones: **500+ unique contributors** in total in easyconfigs, 200+ in easyblocks, 150+ in framework

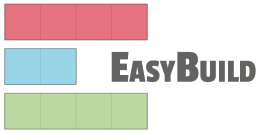
EasyBuild Maintainers



Since 2016, EasyBuild has an **international group of maintainers** *[note: "committers" in HPSF terminology]*

- Number of contributions per year exploded in 2016, so a cry for help was done in EasyBuild community
- Selected frequent contributors were invited to help with EasyBuild maintenance
- EasyBuild currently has 20+ maintainers
    - 10+ different institutions, 8 different countries across Europe & Canada
- Criteria that should be met before you can be *invited* to become a maintainer are documented at https://docs.easybuild.io/maintainers
- Only EasyBuild maintainers can *merge* pull requests
- Handful of trusted contributors have triage-only rights, and/or can trigger our bot to do test installs
- Yearly (virtual) EasyBuild Maintainers Summit to discuss current status & future developments

Testing & CI in EasyBuild



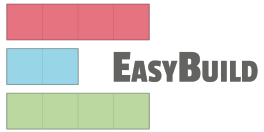
- Contributions to EasyBuild are thoroughly tested before they are accepted:
  - Unit test suites + end-to-end tests are run in GitHub Actions for every PR update
    - Over 1,000 tests in EasyBuild framework, run in multiple config. (Python/Lmod versions)
  - Actual installation of easyconfig files (build recipes) in pull requests on at least 1 system:

    `eb --from-pr 12345 --upload-test-report --robot`
  - Likewise for changes to easyblocks (build scripts):

    `eb example.eb --include-easyblocks-from-pr 12345 --upload-test-report`
- A test bot can launch test installations on request via PR comments in GitHub
- Extensive pre-release regression testing of release branch
  - 1000s of installations are performed to catch regressions prior to a new EasyBuild release

Development & release process

https://github.com/easybuilders



- All development is done in `develop` branches in GitHub (under `easybuilders` org)
  - 3 main repos: `easybuild-framework`, `easybuild-easyblocks`, `easybuild-easyconfigs`
  - Separate GitHub repo for documentation, easybuild metapackage on PyPI, EasyBuild blog, ...
- On a regular basis (every ~6-8 weeks) an stable EasyBuild version is released
- Release branch (`like 5.4.x`) is created from `develop`, regression test starts
- Version bump (patch/minor/major) is determined by merged changes
- Release is published once there are no (more) (large) regressions
- Detailed overview of changes via https://docs.easybuild.io/release-notes
- `main` branch corresponds with latest release
- Backwards-compatibility is a core principle (except for major versions)

Governance (1/2)

Interim Steering Committee for EasyBuild started in March 2025

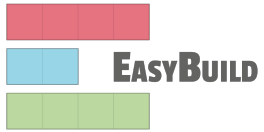
- Consisted of 6 EasyBuild maintainers who volunteered for it
- **Primary task was to agree on formal governance for EasyBuild**
- 5 meetings were held to discuss several aspects of the governance, etc.
- Consensus was to base governance on meritocracy governance model (like EESSI did)
- **Effort completed on 1 July 2026**
- See https://docs.easybuild.io/governance/governance
- Also introduced:
  - Project Charter
  - Code of Conduct (based on Contributor Covenant)
  - EasyBuild AI Policy (already in April 2026)

Governance (2/2)

"Proper" Steering Committee (SC) was voted into existence on 29 June 2026

- https://docs.easybuild.io/governance/steering-committee
- EasyBuild Governance states:
  - Max. 1 SC member with same employer, max. 2 members with same country of residence
  - No more than 30% of SC members employed by commercial entities
  - Ideally odd number of members, composition should reflect EasyBuild community
- Intention is to meet at least once per quarter
- Currently 7 members: 6 EasyBuild maintainers + 1 frequent contributor
  - **Kenneth Hoste** (Ghent University, Belgium) as chair of Steering Committee, **Adam Huffman** (University of Oxford, UK), **Caspar van Leeuwen** (SURF, The Netherlands), **Sebastian Achilles** (Jülich Supercomputing Centre, Germany), **Bart Oldeman** (McGill University, Calcul Québec, Digital Research Alliance of Canada), **Jörg Saßmannshausen** (Imperial College London, UK), **Cintia Willemyns** (Vrije Universiteit Brussel, Belgium)

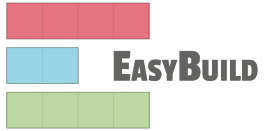
Impact of HPSF on EasyBuild



HPSF has *already* had a positive impact on EasyBuild !

- **Proper governance** in place now, inspired by HPSF
  - From BDFL over dooacry to meritrocacy-based governance
- Adoption of an official **security policy** (formalized on 1 Oct 2026)
  - https://github.com/easybuilders/easybuild-framework/security
  - Security issues for EasyBuild can be reported via GitHub
  - GitHub maintainers strive to respond in 2-3 days, resolve in ASAP (matter of days)
  - 1 CVE in EasyBuild so far (March 2020, leaking of GitHub token in EasyBuild debug logs)
    - Reported internally on 15 March 2020, fixed & patch release out on 16 March 2020

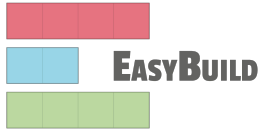
Synergy with other (HPSF) projects



The EasyBuild community has always been very open to collaborating with other projects:

- Support for installing Kokkos, Spack, ReFrame, etc. with EasyBuild
- Support for installing EasyBuild with Spack
- Contributions to EasyBuild from related projects, like Environment Modules
- Ongoing collaborations (and even friendships) with various people (like Lmod)
- Using archspec (fleshed out of Spack) as optional dependency
- Presentations at the EasyBuild User Meeting on HPSF, Spack, flux, ReFrame, Apptainer, Magic Castle, Lmod, XALT, Environment Modules, Open OnDemand, CernVM-FS, LEXIS, Waldur, JUBE, buildtest, ...

HPSF project proposal for EasyBuild



We are confident that EasyBuild meets all requirements to join HPSF at the **Core** stage:

✅ 2 TAC sponsors: Todd Gamblin (Spack), Xavier Delaruelle (Modules)	✅ Defined governing body of 4+ members, no more than 50% of same employer
✅ Aligns with mission of HPSF	✅ Healthy number of committers from at least two organizations
✅ Documented & publicly accessible project governance, decision making, and release process	✅ Explicitly defined security reporting and incident mitigation processes
✅ Used successfully in production by at least 3 independent end user of adequate quality & scope	✅ Explicitly defined project governance and committer process
✅ Demonstrated development processes that lower barriers to contribution + ensure software quality	✅ Evidence of widespread adoption in the HPC ecosystem
✅ Substantial ongoing flow of commits & merged contributions	✅ Willing to list HPSF status prominently on website/README