

EESSI CI/CD Hackathon 2025

<https://eessi.io/docs/training-events/2025/hackathon-eurohpc-user-days>

Q&A via EESSI Slack

**Please join the #cicd channel
in the EESSI Slack for questions and discussion**

Step 1) **Join the EESSI Slack**,
see “Slack channel” link at <https://eessi.io>

Step 2) **Join #cicd** in EESSI Slack
(direct link: <https://eessi-hpc.slack.com/archives/C096B9JSD0C>)

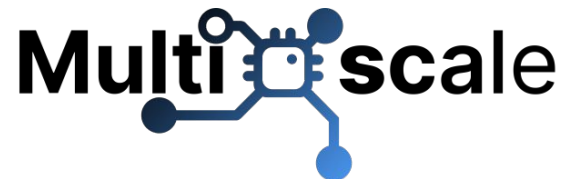
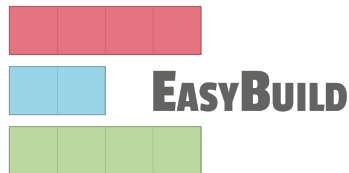


Today's session will start at 09:10 CEST !

Prepared environment

- Small Rocky Linux 9 cluster (in the cloud), for those who cannot install EESSI
- **You need to create an account!**
 - Signup: <https://mokey.tutorial.eessi.science/auth/signup>
 - Accounts will only be approved for access on the day of the tutorial, so please record your username/password !
 - “Reset password” link does **not** work
- Access via SSH or web browser (**pick one and stick to it!**)
 - Shell access: `ssh you@tutorial.eessi.science`
 - Use login node for hands-on, it has 16 cores so should be fine to share
 - Via browser: <https://tutorial.eessi.science>
 - Make sure to change default “Time” to 4 hours
 - Take 4-8 cores
- System will be up until the end of the hackathon

**Today's session
will start at
09:10 CEST !**



EESSI CI/CD hackathon

2 October 2025, Copenhagen

<https://eessi.io/docs/training-events/2025/hackathon-eurohpc-user-days>

Helpful knowledge for this event

- You have watched the episodes from the webinar series:
 - Introduction to EESSI
 - Introduction to EasyBuild
- You have built a software package in some way (e.g., compiled)
 - Nice to have: heard of (and maybe used) CMake/Autotools
- You are somewhat familiar with Git
 - Nice to have: familiar with GitHub/GitLab
- Familiar with environment modules (*Lmod*)

Agenda



[09:15-09:45] **Round table: introduce yourself** [everyone]

[09:45-10:15] **Quick introduction to EESSI + EasyBuild** [Helena + Lara]

[10:15-10:45] **EESSI for Continuous Integration (CI)** [Alan]

[10:45-11:00] (coffee break)

[11:00-11:30] **EESSI for Continuous Integration (CI) - continued** [Alan]

[11:30-12:00] **EESSI for Continuous Deployment (CD)** [Alan]

[12:00-12:30] (lunch break)

[12:30-12:45] **Kickstart hands-on** [Kenneth]

[12:30-16:00] **Hands-on: pick your own adventure!** [everyone]

[16:00-16:30] **Show & tell (optional)** [everyone]

[16:30-17:00] **Q&A + end**

Round table

Please introduce yourself, in 1 minute (or less)...

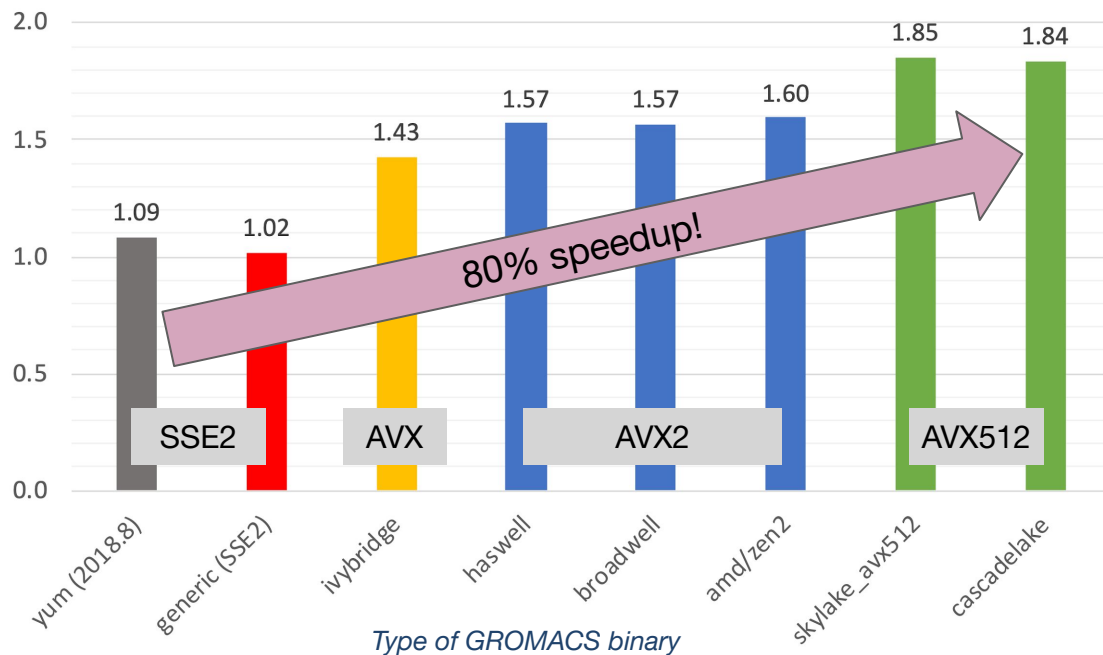
- Who are you? (name + affiliation)
- Where do you work, what are your tasks?
- Why are you here?

What are you hoping to get out of this event?

Keeping the P in HPC

- Software should be **optimized** for the system it will be run on
- Impact on **performance** is often significant for scientific software!

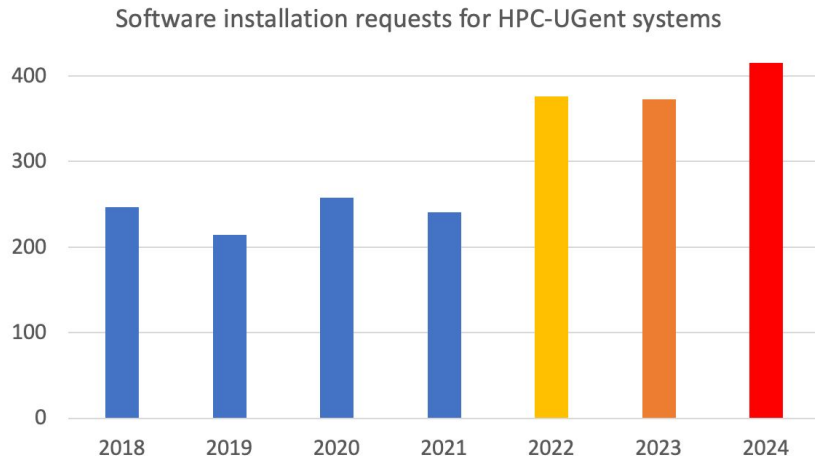
Performance (simulated ns/day) - higher is better



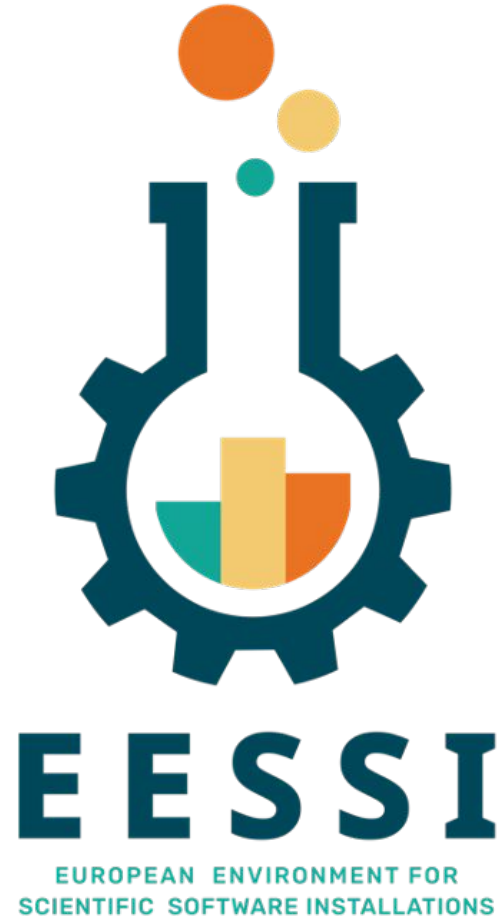
- Example: GROMACS 2020.1 (PRACE benchmark, Test Case B)
- Test system: dual-socket Intel Xeon Gold 6420 (Cascade Lake, 2x18 cores)
- Barplot shows performance of different GROMACS binaries, on exact same hardware/OS

Scientific computing is changing

- **Explosion of available scientific software** (bioinformatics, AI boom, ...)
- Increasing interest in **cloud** for scientific computing (flexibility!)
- **Increasing variety in processor (micro)architectures** beyond Intel & AMD:
Arm is ~~coming~~ already here (see Fugaku, JUPITER, ...), RISC-V is coming (soon?)
- In strong contrast: available (wo)manpower in **HPC support teams** is (still) too limited...



*What if you no longer have to install
a **broad range of scientific software**
from scratch on every laptop, HPC cluster,
or cloud instance you use or maintain,
without compromising on performance?*



EESSI in a nutshell

- European Environment for Scientific Software Installations (EESSI)
- **Shared repository of (optimized!) scientific software installations**
- Avoid duplicate work across (HPC) sites by collaborating on a shared software stack
- Uniform way of providing software to users, regardless of the system they use!
- Should work on any Linux OS and system architecture
- From laptops and personal workstations to HPC clusters and cloud
- Support for different CPUs, interconnects, GPUs, etc.
- **Focus on performance, automation, testing, collaboration**



E E S S I

EUROPEAN ENVIRONMENT FOR
SCIENTIFIC SOFTWARE INSTALLATIONS

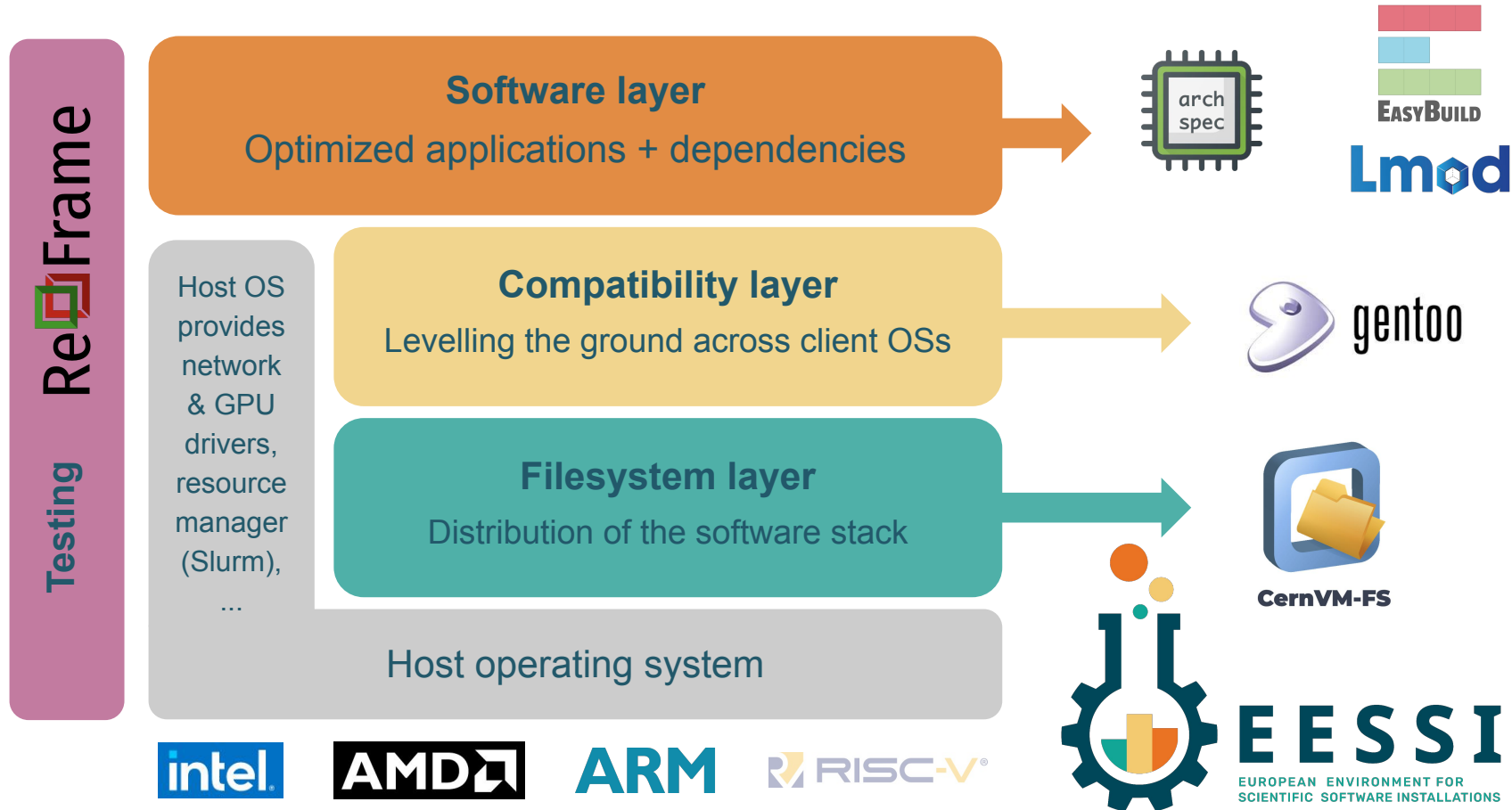
<https://eessi.io> - <https://eessi.io/docs>

Major goals of EESSI



- Providing a truly **uniform software stack**
 - Use the (exact) same software environment everywhere
 - **Without sacrificing performance** for “mobility of compute” (like is typically done with containers/conda)
- **Avoid duplicate work** (for researchers, HPC support teams, sysadmins, ...)
 - Tools that automate software installation process (EasyBuild, Spack) are not sufficient anymore
 - Go beyond sharing build recipes => work towards a shared software stack
- Facilitate HPC training, development of (scientific) software, ...

High-level overview of EESSI



EESSI ingredients



gentoo linux™

Compatibility layer

Abstraction from the
host operating system



Filesystem Layer

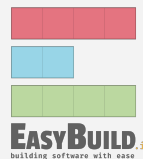
Global distribution of
software installations
via CernVM-FS



E E S S I

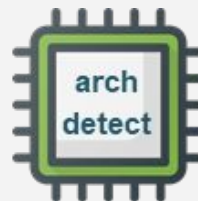
EUROPEAN ENVIRONMENT FOR
SCIENTIFIC SOFTWARE INSTALLATIONS

Software Layer



Optimized software
installations for specific
CPU microarchitectures

Intuitive user interface:
module avail,
module load, ...

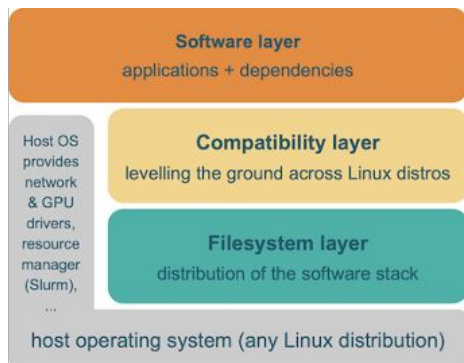


Automatic selection of
best suited part of
software stack for
CPU microarchitectures

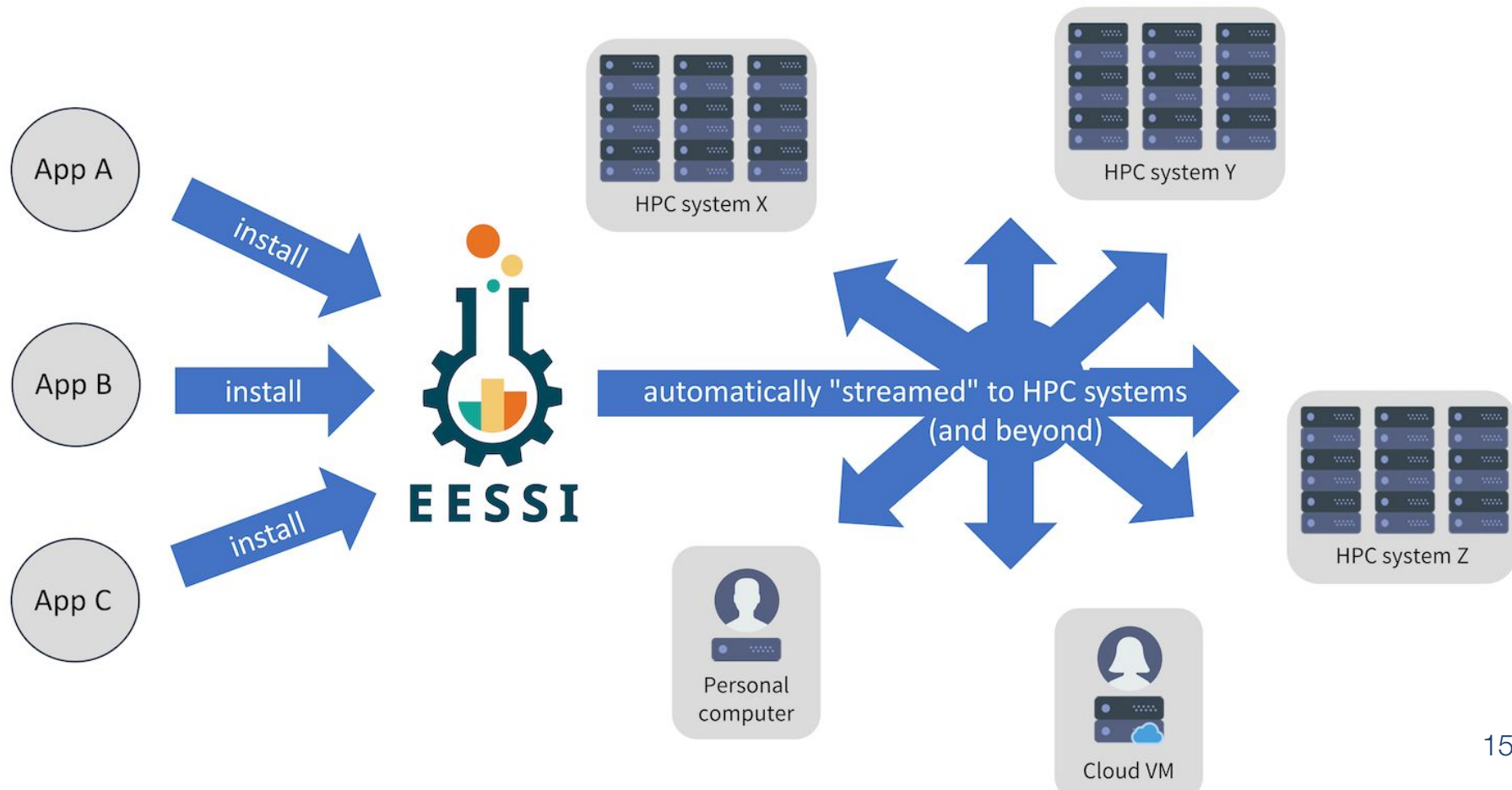
How does EESSI work?



- Software installations included in EESSI are:
 - Automatically “**streamed in**” on demand (via CernVM-FS)
 - Built to be **independent of the host operating system**
“Containers without the containing”
 - **Optimized** for specific CPU generations + specific GPU types
- Initialization script auto-detects CPU + GPU of the system



EESSI as a shared software stack

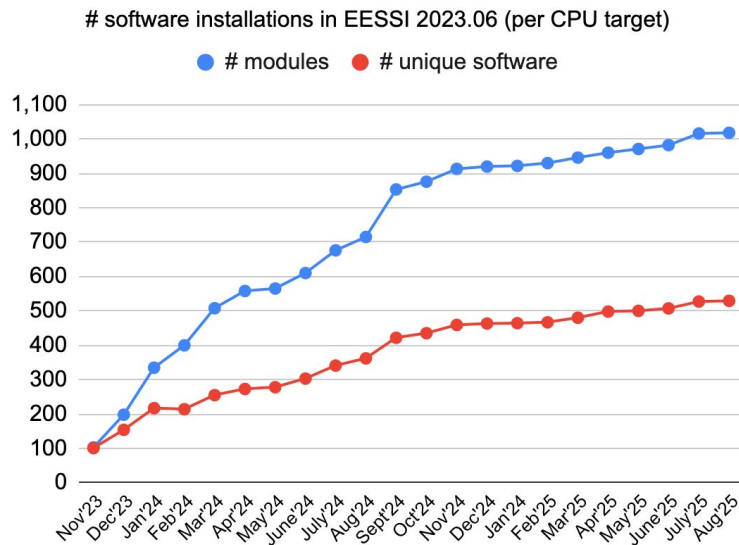


Overview of available software in EESSI



Currently > 1,000 software installations available per supported CPU target via software.eessi.io CernVM-FS repository; increasing every week

- 13 (+1) supported CPU targets (x86_64 + Arm), see https://eessi.io/docs/software_layer/cpu_targets
- Over 500 different software packages, excl. extensions: Python packages, R libraries
- Over 15,000 software installations in total
- Including ESPResSo, GROMACS, LAMMPS, OpenFOAM, PyTorch, R, QuantumESPRESSO, TensorFlow, waLBerla, WRF, ...
- eessi.io/docs/available_software/overview
- `foss/2023a` + `foss/2023b` in EESSI 2023.06, `foss/2024a` + `foss/2025a` in EESSI 2025.06



Getting access to EESSI via CernVM-FS



```
# Native installation
# Installation commands for RHEL-based distros
# like CentOS, Rocky Linux, Almalinux, Fedora, ...

# install CernVM-FS

sudo yum install -y

https://ecsft.cern.ch/dist/cvmfs/cvmfs-release/cvmfs-release-latest.noarch.rpm

sudo yum install -y cvmfs

# create client configuration file for CernVM-FS
# (no proxy, 10GB local CernVM-FS client cache)

sudo bash -c "echo 'CVMFS_CLIENT_PROFILE='single'' > /etc/cvmfs/default.local"
sudo bash -c "echo 'CVMFS_QUOTA_LIMIT=10000' >> /etc/cvmfs/default.local"

# Make sure that EESSI CernVM-FS repository is accessible

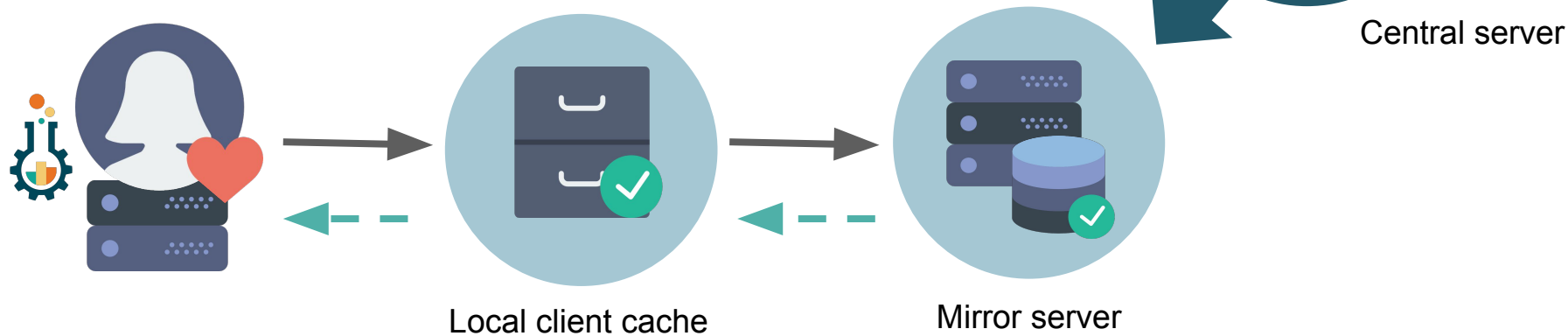
sudo cvmfs_config setup
```

Alternative ways of accessing EESSI are available, via a container image, via cvmfsexec, ...

eessi.io/docs/getting_access/native_installation - eessi.io/docs/getting_access/eessi_container

The EESSI User Experience

```
$ source /cvmfs/software.eessi.io/versions/2023.06/init/bash  
{EESSI 2023.06} $ module load GROMACS/2024.1-foss-2023b  
{EESSI 2023.06} $ gmx mdrun ...
```



EESSI provides **on-demand streaming**
of (scientific) software (like music, TV-series, ...)

Using EESSI (demo)

eessi.io/docs/using_eessi/eessi_demos



```
/cvmfs/software.eessi.io/versions/2023.06/software
```

```
-- linux
|  -- aarch64
|  |  -- a64fx
|  |  -- generic
|  |  -- neoverse_n1
|  |  -- neoverse_v1
|  |  -- nvidia/grace
-- x86_64
|  -- amd
|  |  -- zen2
|  |  -- zen3
|  |  -- zen4
|  -- generic
-- intel
|  -- cascadelake
|  -- haswell
|  -- icelake
|  -- haswell
-- sapphirerapids
|  -- modules
-- software
```

```
$ source /cvmfs/software.eessi.io/versions/2023.06/init/bash
Found EESSI pilot repo @
/cvmfs/software.eessi.io/versions/2023.06!
archdetect says x86_64/amd/zen3
Using x86_64/amd/zen3 as software subdirectory
```

... **Automatically detects CPU microarchitecture**
Environment set up to use EESSI pilot software stack, have fun!

```
{EESSI 2023.06} $ module load R/4.3.2-gfbbf-2023a
```

```
{EESSI 2023.06} $ which R
/cvmfs/software.eessi.io/versions/2023.06/software/linux/x86_64/
amd/zen3/software/R/4.3.2-gfbbf-2023a/bin/R
```

```
{EESSI 2023.06} $ R --version
R version 4.3.2
```

Webinar series: Different aspects of EESSI

5 Mondays in a row May-June 2025

<https://eessi.io/docs/training/2025/webinar-series-2025Q2>

- Introduction to EESSI
- Introduction to CernVM-FS
- Introduction to EasyBuild
- EESSI for CI/CD
- Using EESSI as the base for a system stack

Slides + recordings available



What is EasyBuild?



- **EasyBuild is a software build and installation framework**
- Strong focus on scientific software, performance, and HPC systems
- Open source (GPLv2), implemented in Python
- Brief history:
 - Created in-house at HPC-UGent in 2008
 - First released publicly in Apr'12 (version 0.5)
 - EasyBuild 1.0.0 released in Nov'12 (during SC12)
 - Worldwide community has grown around it since then! (>1,000 members on EasyBuild Slack)

<https://easybuild.io>

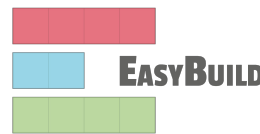
<https://docs.easybuild.io>

<https://blog.easybuild.io>

<https://github.com/easybuilders>

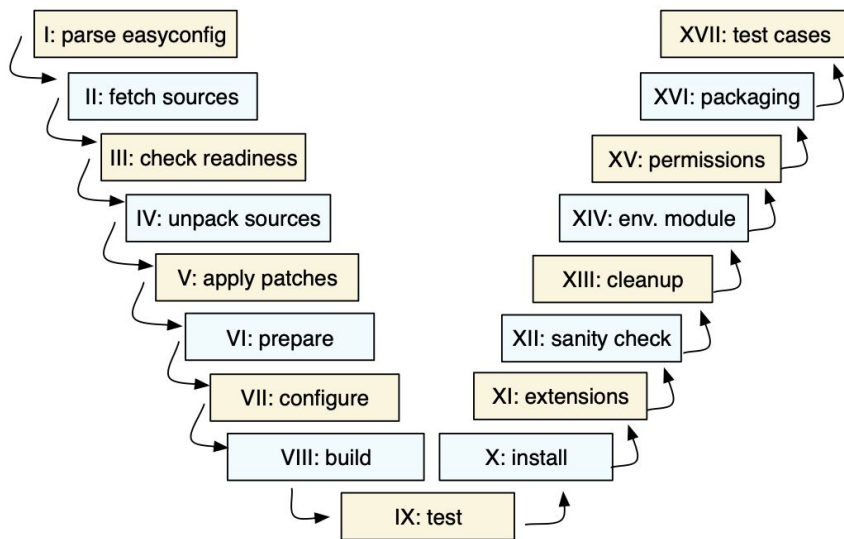
<https://easybuild.io/join-slack>

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 scientific researchers to self-manage their software stack
- A **platform for collaboration** among HPC sites worldwide
- Has become an “expert system” for installing scientific software

Step-wise installation procedure



- EasyBuild framework defines **step-wise installation procedure**, leaves some steps unimplemented
- *Easyblock* completes the implementation, override or extends installation steps where needed
- *Easyconfig file* provides the details (software version, dependencies, toolchain, ...)

EasyBuild terminology



The EasyBuild **framework** (API) leverages **easyblocks** (Python scripts) to automatically build and install (scientific) software, potentially including additional **extensions** (Python pkgs, ...), using a particular compiler **toolchain** (incl. MPI/BLAS/LAPACK/FFT libraries), as specified in **easyconfig files** (“recipes”) which 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.

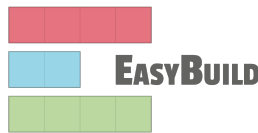
An **easystack** file can be used to specify a collection of software to install with EasyBuild.

Configuring EasyBuild



- There are 3 different configuration levels in EasyBuild:
 - **Configuration files** (see `eb --show-default-configfiles`)
 - **Environment variables** (`$EASYBUILD_XYZ`)
 - **Command line options to the `eb` command**
- Each configuration setting can be specified via each “level” (no exceptions!)
- Hierarchical configuration:
 - Configuration files override default settings
 - Environment variables override configuration files
 - `eb` command line options override environment variables

Inspecting the current configuration



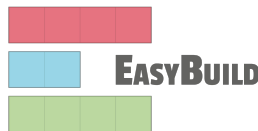
- It can be difficult to remember how EasyBuild was configured
- Output produced by `eb --show-config` is very useful to remind you
- Shows configuration settings that are different from default
- Always shows a couple of key configuration settings
- Also shows on which level each configuration setting was specified
- Full current configuration: `eb --show-full-config`

Basic usage of EasyBuild



- **Use `eb` command to run EasyBuild**
- Software to install is usually specified via name(s) of easyconfig file(s), or easystack file
- `--robot` (`-r`) option is required to also install missing dependencies (and toolchain)
- Typical workflow:
 - Find or create easyconfig files to install desired software
 - Inspect easyconfigs, check missing dependencies + planned installation procedure
 - Double check current EasyBuild configuration
 - Instruct EasyBuild to install software (while you enjoy a coffee... or two)

Searching, copying, installing easyconfigs



- To search for easyconfig files to install (case-insensitive), use `eb --search` (or `-S`)

```
$ eb --search bcftools
```

```
== found valid index for /home/ec2-user/eb-env/easybuild/easyconfigs, so using it...
```

```
...
```

```
* /home/example/eb-env/easybuild/easyconfigs/b/BCFtools/BCFtools-1.18-GCC-12.3.0.eb
```

```
* /home/example/eb-env/easybuild/easyconfigs/b/BCFtools/BCFtools-1.19-GCC-13.2.0.eb
```

```
* /home/example/eb-env/easybuild/easyconfigs/b/BCFtools/BCFtools-1.21-GCC-13.3.0.eb
```

- To copy an easyconfig file, use `eb --copy-ec`

```
$ eb --copy-ec BCFtools-1.18-GCC-12.3.0.eb /tmp
```

- To install an easyconfig file, pass it to `eb` (and maybe also use `--robot`)

```
$ eb BCFtools-1.18-GCC-12.3.0.eb --robot
```

Checking missing dependencies via `eb --missing`



To check which dependencies are still missing, use `eb --missing` (or `eb -M`):

- Takes into account available modules, only shows what is still missing

```
$ eb BCFtools-1.18-GCC-12.3.0.eb --missing
```

```
1 out of 23 required modules missing:
```

```
* BCFtools/1.18-GCC-12.3.0 (BCFtools-1.18-GCC-12.3.0.eb)
```

Using software installed with EasyBuild



To use the software you installed with EasyBuild, load the corresponding module:

```
# inform modules tool about modules installed with EasyBuild
module use $EASYBUILD_INSTALLPATH/modules/all
```

```
# check for available modules for BCFtools
module avail BCFtools
```

```
# load BCFtools module to "activate" the installation
module load BCFtools/1.18-GCC-12.3.0
```

Improved error reporting in EasyBuild v5.x



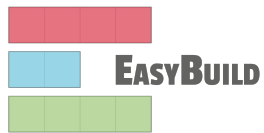
EasyBuild 5.x produces clearer error messages when a shell command failed:

```
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-lynysa6f/out.txt
interactive shell script -> /tmp/eb-i61vle8x/run-shell-cmd-output/make-lynysa6f/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!

Adding support for additional software



- **Every installation performed by EasyBuild requires an easyconfig file**
- Easyconfig files can be:
 - Included with EasyBuild itself (or obtained elsewhere)
 - Derived from an existing easyconfig (manually or automatic)
 - Created from scratch
- Most easyconfigs leverage a generic easyblock
- Sometimes using a custom software-specific easyblock makes sense...

Exercise on creating easyconfig file from scratch



- Step-wise example + exercise of creating an easyconfig file from scratch
- For fictitious software packages: `eb-tutorial` + `py-eb-tutorial`
- Sources available at
<https://github.com/easybuilders/easybuild-tutorial/tree/main/docs/files>

- **Great exercise to work through these yourself!**

```
name = 'eb-tutorial'
```

```
version = '1.0.1'
```

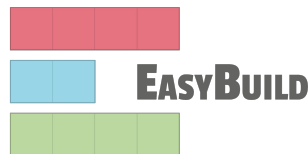
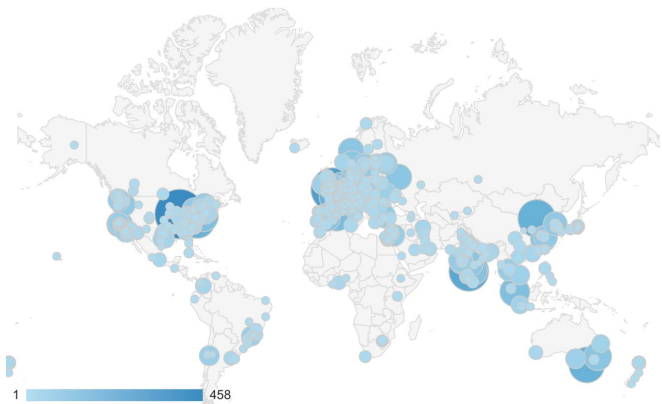
```
homepage = 'https://easybuilders.github.io/easybuild-tutorial'
```

```
description = "EasyBuild tutorial example"
```

The EasyBuild community



EasyBuild User Meeting 2025 (Jülich, Germany)



Documentation is read all over the world

HPC sites, consortia, and companies

Slack: >1000 members,

~180 active members per week

Bi-weekly online conf calls + yearly user meeting



EESSI for CI/CD







- Introducing Continuous Integration (CI)
- Navigating EESSI to build my project
- Building my project with the EESSI GitHub Action
- Navigating EasyBuild to build with **EESSI-extend**
- Building with **EESSI-extend** and the EESSI GitHub Action
- Building my project with the EESSI GitLab Component
- Continuous Deployment and what EESSI can offer there






What is Continuous Integration (CI)?

- Development practice of frequently merging code into a shared repository
- Automated build and test run on each code change
- Helps detect bugs early and improve code quality
- Provides immediate feedback to developers
- Ensures the application is always in a deployable state

Key Components of CI

-  Version Control System (e.g., Git)
-  Automated Build System
-  Automated Testing Suite
-  Notification/Feedback Mechanism

Benefits of Continuous Integration

-  Early detection of integration issues
-  Faster and safer release cycles
-  Encourages small, incremental changes
-  Improved code quality and team collaboration
-  Easier refactoring and code maintenance

Specific Challenges of CI in HPC context

- Need HPC-suitable toolchains and dependencies
- Need a way to deal with MPI
- Typically want to test defined architectures
- Want to have accelerator support
- Performance and scalability (also) matter

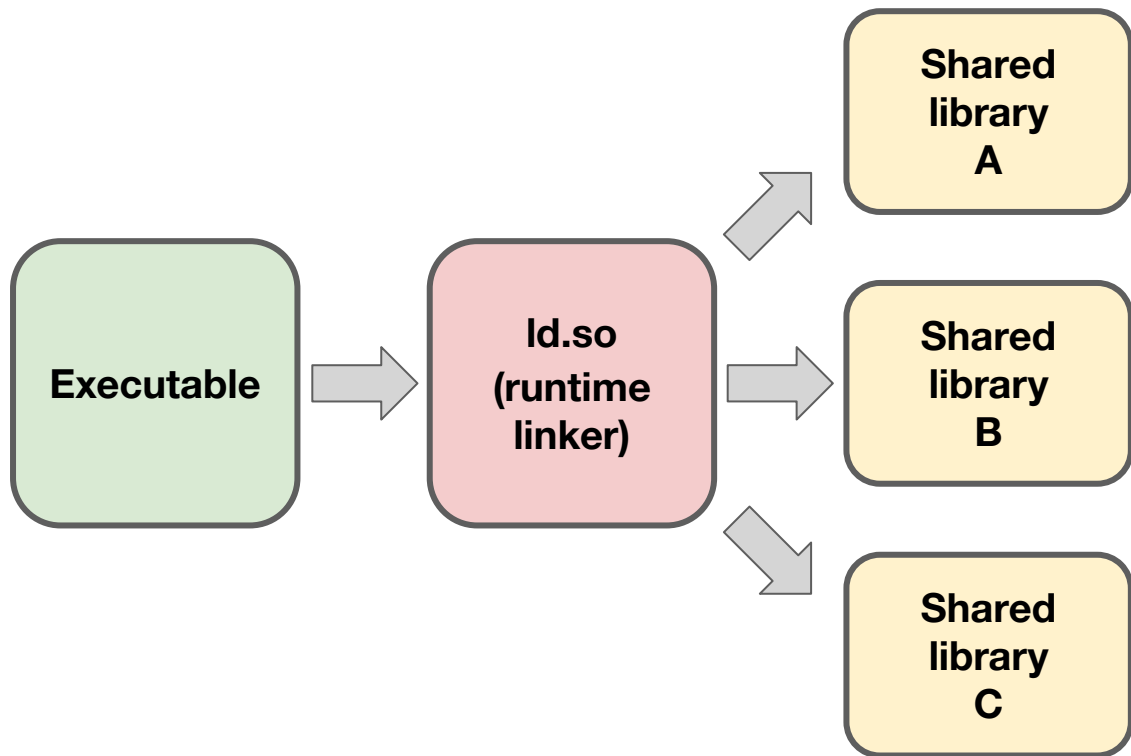
Leveraging EESSI in CI environments

- EESSI can be used in CI environments like:
 - GitHub: github.com/marketplace/actions/eessi
 - GitLab: gitlab.com/explore/catalog/eessi/gitlab-eessi
- EESSI can provide:
 - Different compilers to test your software with
 - Required dependencies for your software
 - Additional tools like ReFrame, performance analysis tools, ...
- Other than CernVM-FS to get access to EESSI, no installations required!
 - Everything is pulled in on-demand by CernVM-FS
- Significantly facilitates also running CI tests in other contexts

Building on top of EESSI

- Load the necessary modules that you need for compilation
 - Compiler, MPI, libraries, dependencies,...
 - Best to try to keep dependencies in the same toolchain
- Load a **builddenv** module for your toolchain
 - Configures typical environment variables for a build (**CC**, **CFLAGS**, **LIBS**, ...)
 - Uses *compiler wrappers* to simplify building with EESSI

Building on top of EESSI



Where does the loader find the libraries?

- Clues in environment variables
- Information in the binary (RPATH)
- It's own defaults

Big long demo:

An example software package

- github.com/EESSI/cicd-demo
- Dependencies: HDF5 and MPI
- Builds with CMake
- Tests via **ctest**

Cheat sheet: <https://hackmd.io/myPkGyj-Rz6pQNMOZW1IA?view>

Continuous Delivery (CD)

- Consider case of “release candidates” and production releases
- Developers commit code → triggers CI
- If tests pass, code can be included in a Release Candidate (RC)
- RC is deployed to a staging environment
- Manual/automated approval takes place
- RC is promoted to production manually or with a controlled deployment

What can EESSI do for CD?

- Production quality software deployed to **software.eessi.io**
- In preparation is **dev.eessi.io**, mostly for release candidates
 - More control for developers
 - Access/deployment to specific architectures
- EESSI is working on getting EuroHPC site support for **EESSI test suite**, could be leveraged for **dev.eessi.io** to remove need for direct access to resources for development

Hands-on: Choose your own adventure!

1. Easy:
Repeat what was covered
in the demo for yourself
2. Medium:
Go through the same workflow
for a *different* application
3. Advanced:
Try your own application



Easy level:

Repeat demo for yourself

- Good exercise for everyone to get the concepts straight
 - Fork github.com/EESSI/cicd-demo
 - Build locally with EESSI first
 - Transfer build to GitHub (or GitLab)
 - Port to EasyBuild (this is an investment for EESSI CD)
 - Transfer port to GitHub (or GitLab)
- Cheat sheet: <https://hackmd.io/myPkGyj-Rz6pQNMOZW1IA?view>

Medium level:

Same workflow, different application

- Select something without too many complications...
 - <https://github.com/samtools/bcftools>
- Git tag 1.19 has an easyconfig ([BCFtools-1.19-GCC-13.2.0.eb](#))
 - Toolchain is supported by EESSI/2023.06
 - All dependencies exist
 - Find the commit that matches the tag and start from there
- **Not trivial**, remember to read the build documentation!

Advanced level:

Try your own application!

- Select toolchain
 - Restricted to what your EESSI version supports
- Check dependencies can be satisfied
 - Restricted to version(s) available for selected toolchain
- Try a local build with dependencies using EESSI
- Add CI
- Port to EasyBuild
 - Local first, then in CI



Website: <https://eessi.io>

Join our Slack channel (see join link on website)

Documentation: <https://eessi.io/docs>

Blog: <https://eessi.io/docs/blog>

GitHub: <https://github.com/eessi>

Paper (open access): <https://doi.org/10.1002/spe.3075>

[EESSI YouTube channel](#)

[Bi-monthly online meetings](#)

(first Thursday of every other month, 14:00 CEST)

[EESSI Happy Hour](#)

(every Monday, 14:00 CEST)

MultiXscale

Web page: multixscale.eu

Facebook: [MultiXscale](https://www.facebook.com/MultiXscale)

Twitter: [@MultiXscale](https://twitter.com/MultiXscale)

LinkedIn: [MultiXscale](https://www.linkedin.com/company/multixscale)

BlueSky: [MultiXscale](https://bsky.app/profile/multixscale)



Co-funded by
the European Union



EuroHPC
Joint Undertaking



UNIVERSITAT DE
BARCELONA



Universität
Stuttgart



SORBONNE
UNIVERSITÉ



Université
de Toulouse



Consiglio Nazionale
delle Ricerche



MAX-PLANCK-GESELLSCHAFT

