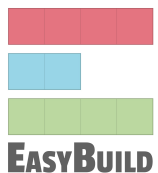




Installing
software
on top of

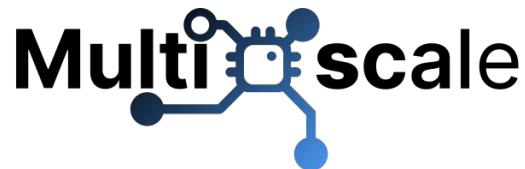
E E S S I

EUROPEAN ENVIRONMENT FOR
SCIENTIFIC SOFTWARE INSTALLATIONS

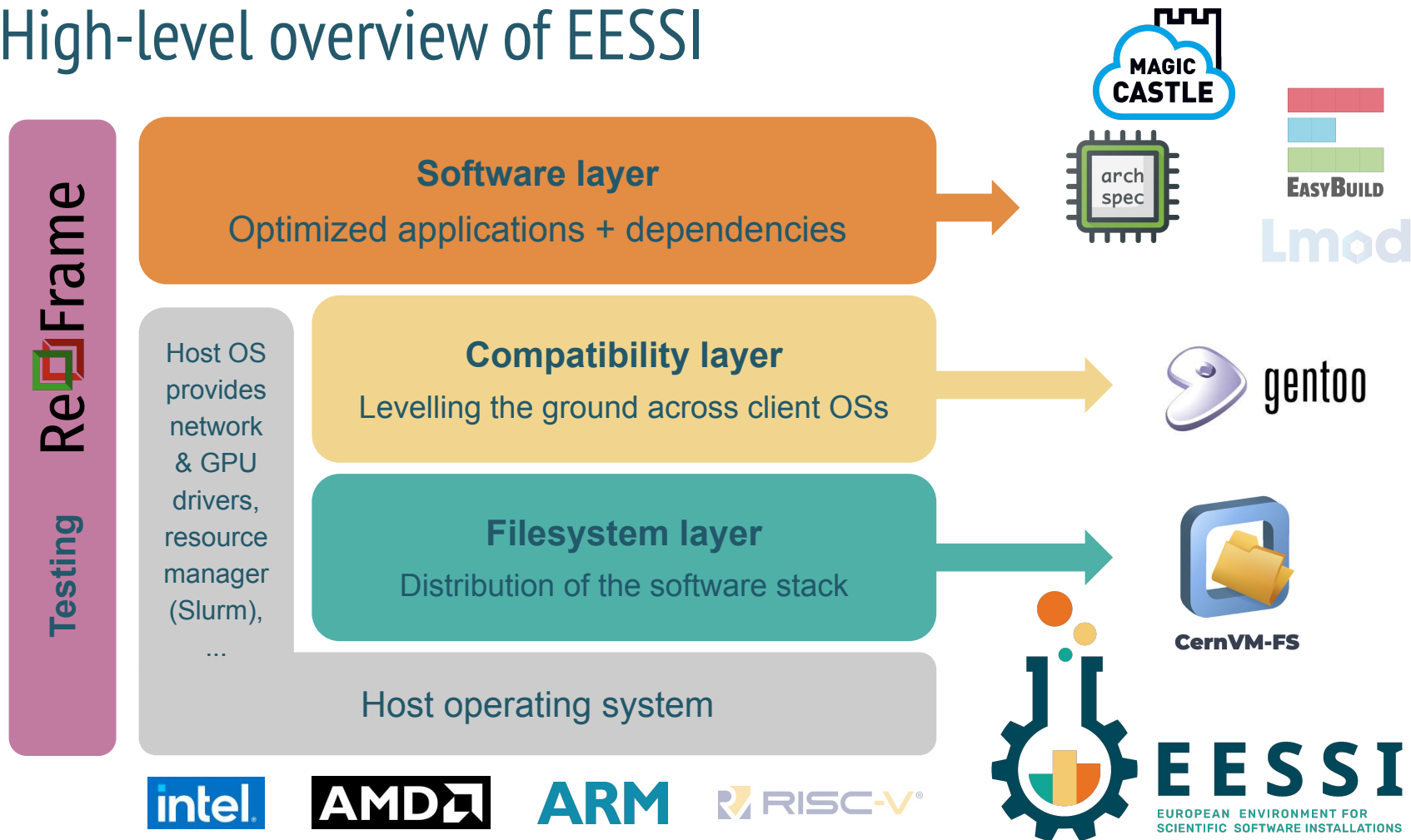


Wed 20 May 2026

Kenneth Hoste (Ghent University)



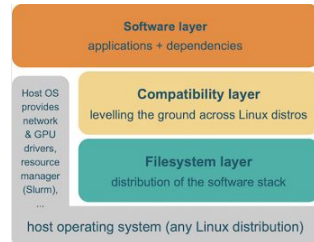
High-level overview of EESSI



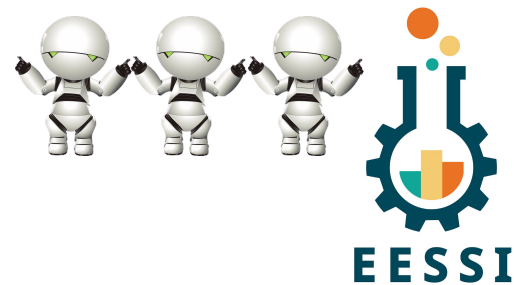
The trick behind the EESSI software layer

Software installations only link to **libraries from compatibility layer or other installations in software layer**, *not* from host (Linux) operating system.

- Because of this, these software installations work on any Linux distribution
- This is ensured via *RPATH linking*
- Build tools (like CMake) need to be made aware of alternative system root (sysroot)
- Some exceptions need to be made: GPU driver libraries must be provided by host OS
- “Containers without the containing”



Automated procedure via EESSI build-deploy bots



- Software installations deployed in EESSI are built by the EESSI build-deploy bots (using EasyBuild), see also <https://eessi.io/docs/bot>
- Semi-automatic installation procedure (human in the loop)
- Via pull request to [EESSI/software-layer](#) repository
- Contributors do not need access to all CPU targets supported by EESSI

Example: `gmx` binary for GROMACS in EESSI

- Let's inspect the `gmx` binary in a GROMACS installation up close...
- We'll use standard tools like `readelf` and `ldd` to show that:
 - RPATH section of binary specifies hardcoded list of paths for libraries
 - `$LD_LIBRARY_PATH` is irrelevant (actually, only used as a fallback)
 - All libraries used by `gmx` binary come from EESSI compatibility layer or dependencies provided via EESSI software layer
- Everything shown is standard on Linux, no tricks up our sleeve

Preparing environment to inspect `gmx` binary for GROMACS



- Initialize shell for using EESSI version 2023.06:

```
source /cvmfs/software.eessi.io/versions/2023.06/init/lmod/bash
```

- Find and load a GROMACS module:

```
module avail GROMACS
```

```
module load GROMACS/2024.4-foss-2023b
```

- Check path to `gmx` binary:

```
command -v gmx (or which gmx)
```

Inspect RPATH section of `gmx` binary with `readelf`

`readelf -d` command reveals details for a specific binary:

```
readelf -d $(command -v gmx)
```

- (RPATH) section specifies which paths should be considered first when loader looks for libraries that are required by `gmx` binary
- (NEEDED) entries specify which libraries are required, which themselves may require additional libraries in turn

Inspect RPATH section of `gmx` binary with `readelf`

There are different types of paths in (RPATH) section:

- Paths to **subdirectories of `host_injections`** (`/rpath_overrides/`)
which allow to inject specific libraries (via a path controlled by system administrators)
- Paths for **libraries in software installation directory** itself (like `/lib`)
- Paths **relative to path of binary** itself (`$ORIGIN/...`)
- Paths to libraries provided by **dependencies in EESSI software layer** (`/software/`)
- Paths to libraries provided by **EESSI compatibility layer** (`/compat/`)

Inspect libraries used by `gmx` binary with `ldd`

`ldd` command reveals libraries that are used when launching `gmx` binary:

```
ldd $(command -v gmx)
```

All libraries are provided either by EESSI compatibility layer or software layer.

Some noteworthy ones:

GROMACS' own library (`libgromacs.so`), provided by the GROMACS software installation directory:

```
libgromacs.so.10 => /cvmfs/software.eessi.io/.../software/GROMACS/.../lib/libgromacs.so.10 (0x000014f646e00000)
```

MPI library (`libmpi.so`), provided by OpenMPI dependency in EESSI software layer:

```
libmpi.so.40 => /cvmfs/software.eessi.io/.../software/OpenMPI/.../lib64/libmpi.so.40 (0x000014f646a00000)
```

GNU C library (`glibc`, `libc.so`), provided by EESSI compat layer (*not* by host OS):

```
libc.so.6 => /cvmfs/software.eessi.io/.../compat/.../lib/.../lib64/libc.so.6 (0x000014f646425000)
```

Building software on top of EESSI



To build software on top of EESSI, we have to make sure that:

- No libraries from host OS are relied on, so that binaries are “portable” across Linux systems
- Path to required libraries is “hardcoded” into binaries and libraries, using RPATH linking.

This is a technical challenge, and is different from how people usually build software:

- At build time, specify library paths via `-L` option in compiler command or via `$LIBRARY_PATH`
- At run time, rely on standard system paths (`/usr/lib`) or `$LD_LIBRARY_PATH`

This can't be relied on when building software to be used in or on top of EESSI (too brittle).

Building software on top of EESSI: options

There are different options to build software on top of EESSI:

- Using **EasyBuild**
 - Should be configured correctly for building on top of EESSI
- Using a “**manual**” build procedure
 - Need to make sure that compatibility layer is used and that RPATH linking is done
- Using **Spack**
 - Can use use software provided by EESSI as “external packages”,
for compiler, dependencies

Building software on top of EESSI with EasyBuild

Over the last couple of years, significant enhancements have been made in EasyBuild to make it easier to build software such that it can be included in EESSI, including:

- Improve support for RPATH linking, and make it the default in EasyBuild v5.0
- `--sysroot` configuration option, and making various easyblocks aware of it
- Support for [easystack files](#)
- `--sanity-check-only` option
- `--from-commit` option, which is more reproducible than `--from-pr`
- CUDA sanity check

For a recent introduction to EasyBuild, see [*Introduction to EasyBuild webinar \(spring 2025\)*](#)



Building software on top of EESSI with `EESSI-extend`



- An `EESSI-extend` module is provided by EESSI which loads + pre-configures EasyBuild
- Very useful when doing local installations on top of EESSI with EasyBuild, or testing
- Basic usage:
 - Initialize EESSI
 - (optional) Set `$EESSI_*_INSTALL` environment variable to control installation path
 - Load `EESSI-extend` module
 - Check EasyBuild configuration (`eb --show-config`)
 - Provide one or more `easyconfig` files to EasyBuild to install software on top of EESSI
- The EESSI build-deploy bots also use `EESSI-extend`, so this mechanism is very well tested!
- Detailed documentation available at https://eessi.io/docs/using_eessi/building_on_eessi

Building software on top of EESSI with EESSI-extend



Configuring EESSI-extend before loading via `$EESSI_*_INSTALL` env. variables

- `$EESSI_USER_INSTALL` to specify that **user-specific** installation prefix that should be used
 - Permissions are set such that directories are only readable by user (umask 077)
- `$EESSI_PROJECT_INSTALL` to specify that **project** installation prefix that should be used
 - Permissions set accordingly: GID bit, group writable installation directories, umask 002, ...
- `$EESSI_SITE_INSTALL` can be set to help with **site-wide** installations on top of EESSI
- `$EESSI_CVMFS_INSTALL` can be set to specify that installation should be done in **CernVM-FS repo**

See also https://eessi.io/docs/using_eessi/building_on_eessi/#using-the-eessi-extend-module

Building software on top of EESSI with EESSI-extend



Hands-on demo (EESSI 2025.06): installing imageio with `foss/2024a`

`imageio-2.36.1-gfbf-2024a.eb` is already available in EasyBuild 5.3.0, so:

```
source /cvmfs/software.eessi.io/versions/2025.06/init/lmod/bash
```

```
export EESSI_USER_INSTALL=$HOME/EESSI-2025.06
```

```
mkdir -p $EESSI_USER_INSTALL
```

```
module load EESSI-extend
```

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

```
eb imageio-2.36.1-gfbf-2024a.eb
```

Building software on top of EESSI with EESSI-extend



Hands-on demo (EESSI 2025.06): installing IQ-TREE 3.1.2

- No easyconfig file for IQ-TREE 3.1.2 available yet in EasyBuild 5.3.0, but a pull request was merged for it recently...
- Merged pull request to EasyBuild: <https://github.com/easybuilders/easybuild-easyconfigs/pull/25981>
- Can test using EESSI-extend and `eb --from-pr:`

```
source /cvmfs/software.eessi.io/versions/2025.06/init/lmod/bash
```

```
export EESSI_USER_INSTALL=$HOME/EESSI-2025.06
```

```
module load EESSI-extend
```

```
eb --from-pr 25981 --missing
```

```
eb --from-pr 25981 --robot
```

Building software on top of EESSI manually, using `buildenv`



Using EasyBuild to build/install software on top is not required, there are other options.

However, you should make sure that:

- RPATH linking is used, or you need to set `$LD_LIBRARY_PATH`
- Compilers and build tools are aware of the alternate sysroot (compatibility layer)

To facilitate this, EESSI provides a `buildenv` module for each (foss) toolchain which:

- Sets up the build environment similar to what EasyBuild uses (`$CC`, `$CFLAGS`, etc.)
- Injects RPATH wrappers for the compiler and linker commands (`gcc`, `g++`, `gfortran`, `ld`, ...)

Detailed documentation available at https://eessi.io/docs/using_eessi/building_on_eessi

Building software on top of EESSI manually



Hands-on demo: Manual build, with the help of `buildenv`

```
source /cvmfs/software.eessi.io/versions/2025.06/init/lmod/bash
```

```
module load buildenv/default-foss-2025b
```

```
module load libpng/1.6.50-GCCcore-14.3.0
```

```
git clone https://github.com/R-Gerard/libpng-demo
```

```
cd libpng-demo
```

```
make
```

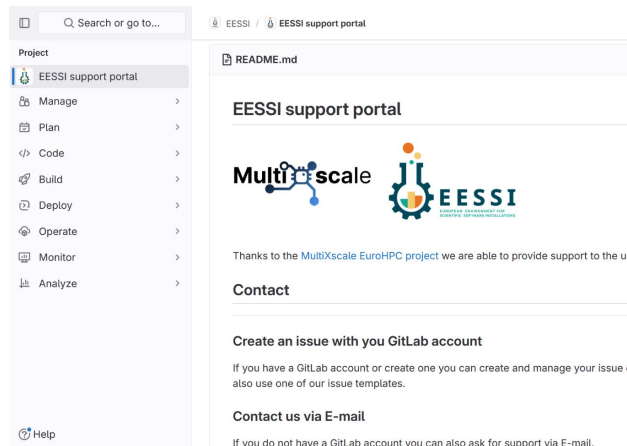
```
readelf -d target/pngdemo
```

```
ldd target/pngdemo
```

Support for installing, using, contributing to EESSI

- Via GitLab, or via email: support@eessi.io
- Report problems
- Ask questions
- Request additional software
- Get help with contributing to EESSI
- Suggest enhancements, additional features, ...
- Confidential tickets possible (security issues, ...)

eessi.io/docs/support



Dedicated support team, thanks to EuroHPC Centre-of-Excellence

