EasyBuild on MeluXina

11th EasyBuild User Meeting

Tue-Thu 21-23 April 2026 @ Guimarães, Portugal

Xavier Besseron
HPC Software Engineer

Outline

MeluXina and LuxProvide

MUSE: MeluXina User Software Environment

- User view
- Development and Deployment Workflow

Local Customizations

- MUSE-stack
- Parallel build with Slurm backend
- Hooks

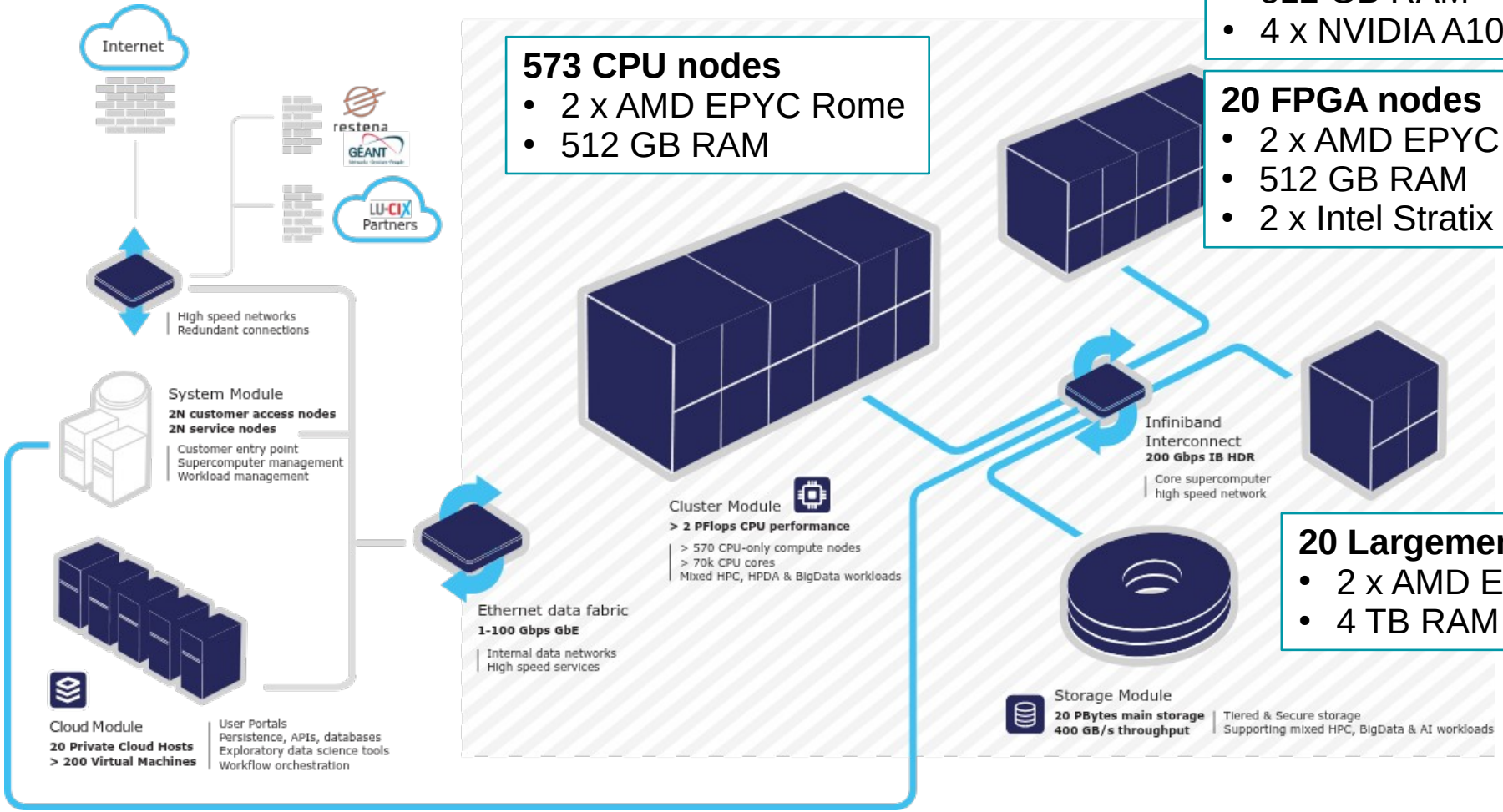
Work In Progress

- AI Coding Assistant for EasyBuild

About MeluXina & LuxProvide

MeluXina Supercomputer

https://docs.lxp.lu/system/overview/#compute-modules



573 CPU nodes
- 2 x AMD EPYC Rome
- 512 GB RAM

200 GPU nodes
- 2 x AMD EPYC Rome
- 512 GB RAM
- 4 x NVIDIA A100 40GB

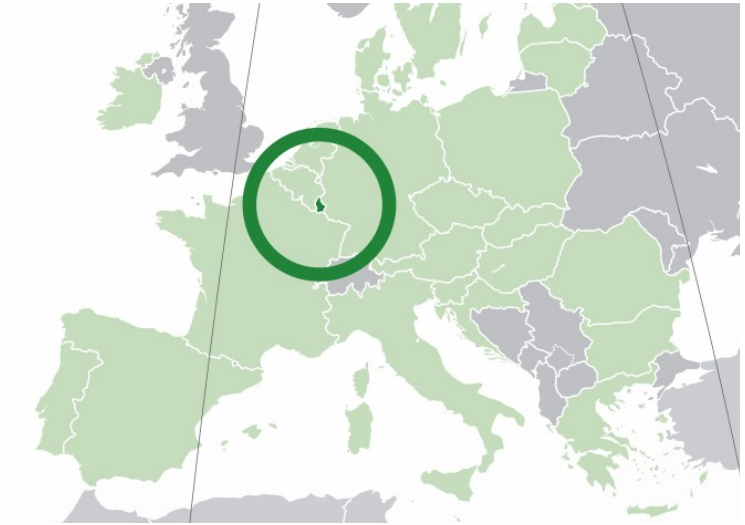
20 FPGA nodes
- 2 x AMD EPYC Rome
- 512 GB RAM
- 2 x Intel Stratix 10MX 16GB

20 Largemem nodes
- 2 x AMD EPYC Rome
- 4 TB RAM

LuxProvide Hosting Entity

LuxProvide operates MeluXina supercomputer

- EuroHPC access
- Luxembourg research institutes
- Commercial access



Supercomputing Application Services (SAS) Team at LuxProvide

- Software installation ~ EasyBuild
- Platform testing ~ ReFrame
- Benchmarking
- Advanced User Support, including EPICURE

...

Melusina statue in Luxembourg
(Photo: Lala La Photo)

Our MUSE:
MeluXina User Software Environment

→ User point of view

MeluXina User Software Environment → MUSE

In LXP Documentation:https://docs.lxp.lu/first-steps/software_env/

Available to users via software environment modules

- New release every year → **`env/release/2025.1`** (current default)
  - 2025.1 based on toolchains 2025a
- Updated during the year → **`env/staging/2025.1`**
  - Additions and modifications to the release
  - Possibly many staging environments per year
- Possibly other releases → **`env/release/2025.2`**

```
------------------------------------------- /apps/USE/system/modules --------------------------------
env/release/2021.3        env/release/2023.1 (S)        env/staging/2021.5    env/staging/2024.1
env/release/2021.5        env/release/2024.1 (S)        env/staging/2022.1    env/staging/2025.1 (D)
env/release/2022.1 (S)    env/release/2025.1 (S,L,D)    env/staging/2023.1
```

MUSE Statistics

Number of software per release

Software Stack	Nb of Software
env/release/2022.1	385
env/staging/2022.1	541
env/release/2023.1	592
env/staging/2023.1	712
env/release/2024.1	539
env/staging/2024.1	651
env/release/2025.1	526
env/staging/2025.1	527

Toolchains used in release 2025.1

Toolchains used	Nb of Software	Incl. with CUDA
GCCcore/14.2.0	310	6
GCC/14.2.0	30	6
gompi/2025a	25	4
gfbf/2025a	24	4
foss/2025a	53	14
iimpi/2025a	7	
intel/2025a	2	

+ a few project-specific modules

(Image: Comingio Merculiano in Jatta Giuseppe, Public Domain)

Our MUSE:
MeluXina User Software Environment

→ Deployment point of view

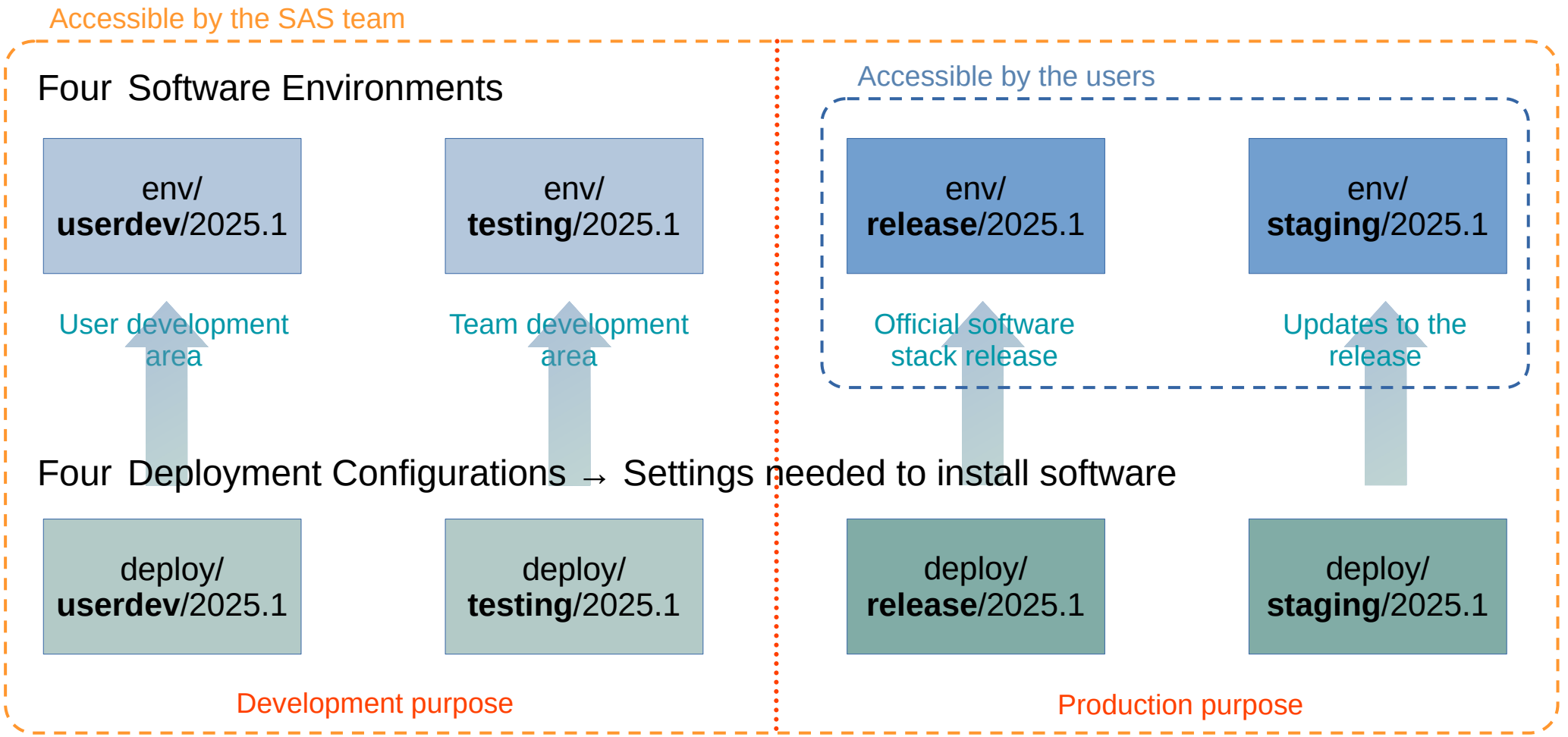
- Deployment configurations
- Development & deployment workflows

MUSE Development

Objectives and Principles

- No development or manual steps in user **`apps`** → Deploy config. userdev and testing
  - Development and testing as a user → Stop using/sharing the **`apps`** account
  - Only final installations (release/staging) with user **`apps`** → Submission script
- Reproducible builds → **All settings** in configuration files saved in Git (deploy modules)
- Automatic builds (as much as possible) → Submission script + Aiming for CI/CD
- Keep track of installed software → MUSE-stack files
- Parallel deployment → EasyBuild Slurm backend and MUSE-stack files
- Automatic selection between CPU/GPU nodes for builds → EasyBuild hook

Underneath the MeluXina User Software Environments



Purpose of the Different Deployment Configurations

For development

deploy/ **userdev**/2025.1 → SAS User development

- Development, debugging and experimentation by each SAS user in their own space.
- No influence from/to other user.
- ➔ Used as a normal user.

deploy/ **testing**/2025.1 → SAS Team development

- Development between the SAS users.
- Common location to share the progress and have a preview of the future release/staging.
- ➔ Used as a normal user.

For users

deploy/ **release**/2025.1 → Software stack release

- Deployment of the software stack release, build from scratch with all dependencies.
- Once deployed, no modifications allowed.
- ➔ Used as the **`apps`** user.

deploy/ **staging**/2025.1 → Updates to the release

- Deployment of the software stack updates.
- Updated regularly until a new software stack release is built from scratch.
- ➔ Used as the **`apps`** user.

Example of Deployment Configuration: `deploy/testing/2025.1`

```
-- Base toolchain version  -> TO BE SET MANUALLY
local toolchain_version = "2025a"
-- User environment module -> TO BE SET MANUALLY
local user_env_module  = "env/testing/2025.1"
-- EasyBuild module        -> TO BE SET MANUALLY
append_path("MODULEPATH", "/project/home/lxp/software_stack/installs/modules/all")
local easybuild_module = "EasyBuild/5.1.2-lxp"
-- Base install directory  -> TO BE SET MANUALLY
local eb_install_dir = "/project/home/lxp/software_stack/testing/2025.1"
```

```
-- Setup EasyBuild configuration
setenv("EASYBUILD_CONFIGFILES",              muse_dir .. "/easybuild-configs/common_" .. deploy_version .. ".cfg")
setenv("EASYBUILD_ROBOT_PATHS",              muse_dir .. "/easybuild-easyconfigs/" .. toolchain_version .. "/:" ..
                                             muse_dir .. "/easybuild-easyconfigs/SYSTEM/:")
setenv("EASYBUILD_INCLUDE_EASYBLOCKS",       muse_dir .. "/easybuild-easyblocks/" .. toolchain_version .. "/*.py")
setenv("EASYBUILD_HOOKS",                    muse_dir .. "/easybuild-hooks/hooks.py")
setenv("EASYBUILD_BUILDPATH",                "/dev/shm/" .. os.getenv("USER"))
setenv("EASYBUILD_PREFIX",                   eb_install_dir)
setenv("EASYBUILD_TMP_LOGDIR",               eb_install_dir .. "/eb_logs/")
setenv("EASYBUILD_JOB_OUTPUT_DIR",           eb_install_dir .. "/eb_logs/")
setenv("EASYBUILD_FAILED_INSTALL_LOGS_PATH", eb_install_dir .. "/eb_logs/")
setenv("EASYBUILD_SOURCEPATH",               eb_install_dir .. "/sources:/apps/sources")
```

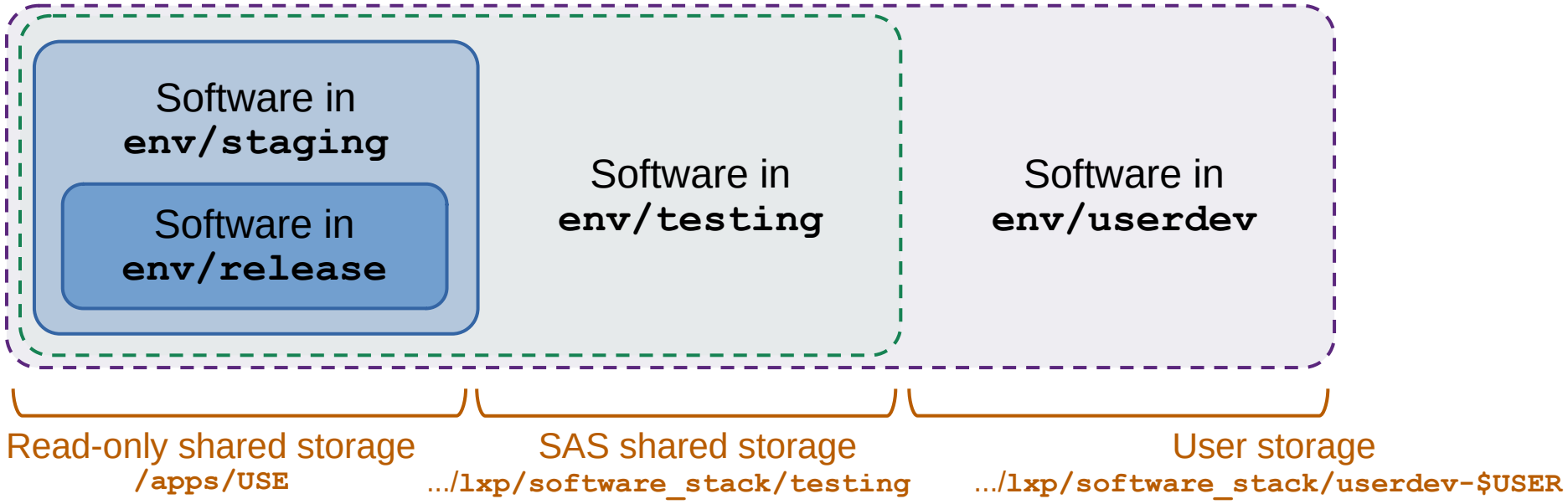
```
-- Other configuration settings
setenv("MUSE_AUTOMATIC_PARTITION", "1")
setenv("PRTE_MCA_rmaps_default_mapping_policy", ":oversubscribe")
prepend_path("PATH", muse_dir .. "/musestacks/bin")
prepend_path("PYTHONPATH", muse_dir .. "/easybuild-hooks")
```

Dependencies between the Software Environments

`env/staging` also includes **`env/release`**

`env/testing` also includes **`env/release + env/staging`**

`env/userdev` also includes **`env/release + env/staging + env/testing`**



Two types of usage: Interactive Build and Batch Build

Interactive Build

- Start an interactive Slurm job
- Load the settings
- Using EasyBuild interactively

→ Behaves like a standard EasyBuild session

Batch Build

- Submit the Slurm script
  - With the deploy module name
  - List of MUSE-stack or EasyConfig files

→ Fully automated parallel build

```
# Load the settings
$ module use ./modules
$ module load deploy/userdev/2025.1

# Search with EasyBuild
$ eb -search HPL.*2025a

# Install with EasyBuild
$ eb HPL-2.3-foss-2025a.eb --robot
```

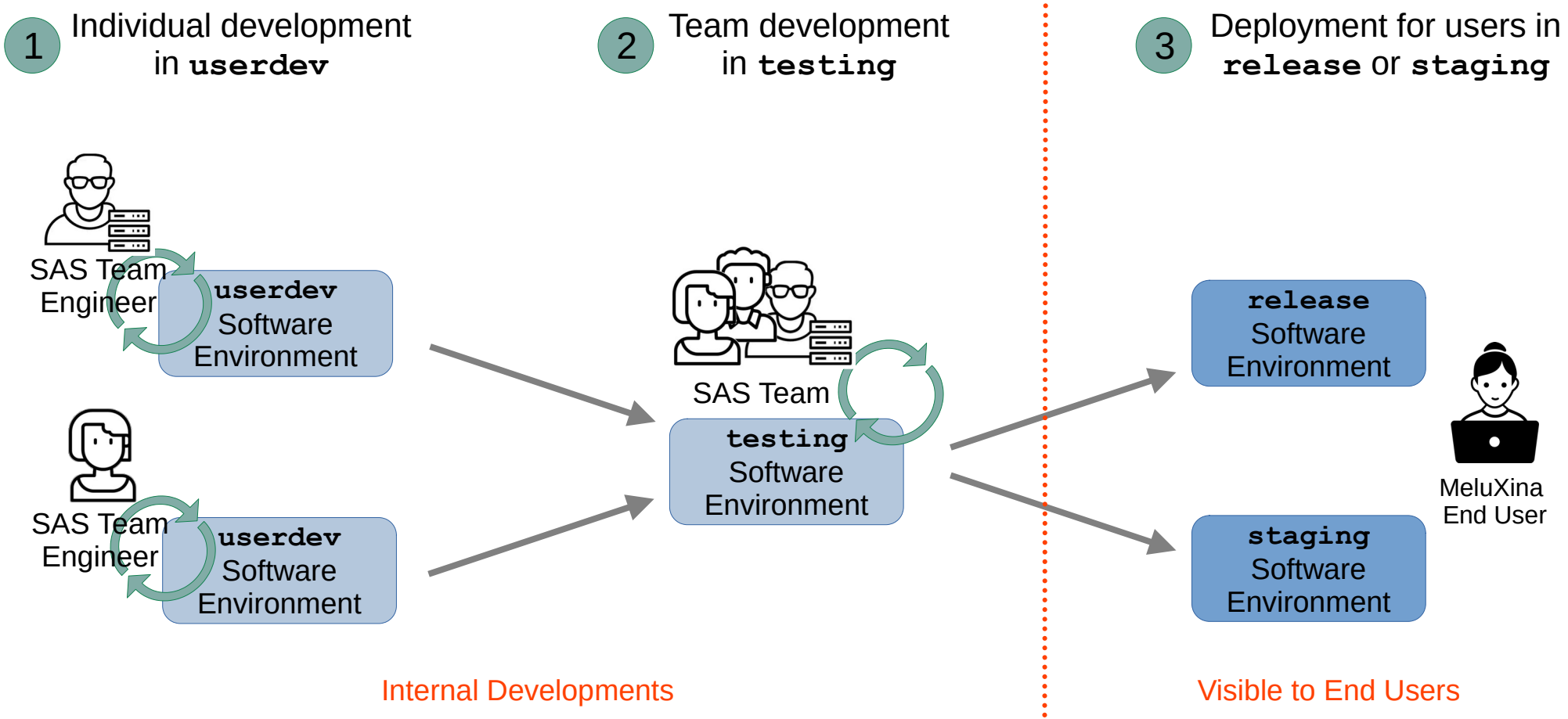
```
# Submit the build job
# - submit-eb-job.sh: provided script
# - arg1: deploy module
# - arg2: musestack file [optional]

$ sbatch submit-eb-job.sh        \
    deploy/release/2025.1        \
    musestack-release-2025.1.txt
```

Organization of the repository

Internal Git repository

`modules/`	→ **`env/`**	→ software environment modules
	→ **`deploy/`**	→ deployment configuration modules
`build-scripts/`		→ Slurm script for automated parallel builds
`musestacks/`		→ Location for MUSE-stack files
`easybuild-configs/`		→ Configuration files for EasyBuild
`easybuild-easyblocks/`		→ Additional or customized EasyBlocks
`easybuild-easyconfig/`		→ Additional or customized EasyConfig files
`easybuild-hooks/`		→ Hooks for EasyBuild
`easybuild-tools/`		→ Additional files for EasyBuild, i.e., the improved Slurm backend
`lmod-config/`		→ Lmod configuration files

Global Workflow



Individual Development

Principles

- Develop locally
- Share when tested and working
- Standard usage of EasyBuild

(Interactive) Workflow

- Develop, debug in your own space
  - Run any EasyBuild command
  - Support for MUSE-stack with **`parse_musestack`**
- Iterate until working
- Save working changes:
  - Add new EasyConfig files if needed
  - **Update MUSE-stack with working software to be installed**
- Push to Gitlab to share your work

1. Install MUSE-NG

```
# Setup MUSE-ng
$ git clone -b 2025a-ng \
    ssh://git@gitlab.lxp.lu:8822/lxp-hpc/meluxina-use.git
$ cd meluxina-use
```

2. Develop & debug installations with EasyBuild

```
# Load the settings
$ module load deploy/userdev/2025.1

# Search with EasyBuild
$ eb -search HPL.*2025a

# Build with EasyBuild
$ eb HPL-2.3-foss-2025a.eb -robot

# Build a full MUSE-stack
$ eb `parse_musestack ./musestack/musestack-testing-2025.1.txt`
```

3. Save working changes, commit+push to share with the team

```
# Save working EasyConfig files
$ git add easybuild-easyconfigs/2025a/h/HPL/HPL-2.3-foss-2025a.eb

# Update MUSE-stack file
$ vi musestacks/musestack-testing-2025.1.txt
# Save MUSE-stack file
$ git add musestacks/musestack-testing-2025.1.txt

# Share with others
$ git commit
$ git push
```

Deployment for users

Principles

- No modifications, no local changes
- Only content from the Git repository
- (Almost) no manual step

Automated build using a Slurm job

- Pull the changes from Git
- Submit the build script
  - Set the deploy configuration
  - Set the list of software
- Check the results

1. Install MUSE-NG in **apps** user

```
# Connect as the apps user
$ ssh apps@meluxina

# Setup MUSE-NG (if needed)
$ git clone … && cd meluxina-use
# Or get the last changes
# git pull
```

2. Submit the build

```
# Submit the build job
# - submit-eb-job.sh: provided script
# - arg1: deploy module
# - arg2: musestack file [optional]

$ sbatch submit-eb-job.sh        \
    deploy/release/2025.1
```

3. Check the log files

```
# Check the log file
$ less log_apps_3758505_MUSEbuild.out
```

Software Deployment for projects

Principles

- Specific deployment configuration: **`deploy/project-p20XXXX/2025.1`**
- Specific MUSE-stack file: **`musestack-project-p20XXXX-2025.1.txt`**
- Everything saved in the Git
  → Repeatable and status understandable by other team members

Same workflow as for general users

- Interactive development in userdev first
- Installatio with automated script

```
# Submit the build job
# - submit-eb-job.sh: provided script
# - arg1: deploy module
# - arg2: musestack file

$ sbatch submit-eb-job.sh                    \
    deploy/project-p20XXXX/2025.1            \
    musestack-project-p20XXXX-2025.1.txt
```

Local Customizations

MUSE-stack files

Objectives

- Maintain a list of software installed or to be installed
- Alternative to the EasyStack (because not working with --job build)

In Practice

- Simple text format, trackable in Git
- Support for comments
- Script **`parse_musestack`** to turn it into a list of EasyConfig files

musestack-testing-2025.1.txt

```
520nmx-20.4.eb
Advisor-2025.0.0.eb
AMD-uProf-5.1.701.eb
ant-1.10.14-Java-11.eb
AOCC-5.0.0-GCCcore-14.2.0.eb
Apptainer-1.4.2-GCCcore-14.2.0.eb
archspec-0.2.5-GCCcore-14.2.0.eb
Arrow-19.0.0-gfbf-2025a.eb
Check-0.15.2-GCCcore-14.2.0.eb
# Blender-4.3.2-linux-x86_64-CUDA-12.6.0.eb
Check-0.15.2-GCCcore-14.2.0.eb
# MISSING Clang
# MISSING Coccinelle
CP2K-2025.2-foss-2025a-CUDA-12.8.0.eb
CP2K-2025.2-foss-2025a.eb
# CubeGUI-4.9 REQUIRES on Qt6
cuDNN-9.10.1.4-CUDA-12.8.0.eb
cuQuantum-25.03.0.11-CUDA-12.8.0.eb
cuTENSOR-2.0.2.5-CUDA-12.8.0.eb
# MISSING dask-cuDF
# MISSING dask-labextension
# MISSING deal.II
# MISSING Elbencho
# MISSING Extrae
ffnvcodec-13.0.19.0.eb
FFTW.MPI-3.3.10-gompi-2025a.eb
# MISSING flatbuffers
...
```

Improved Slurm backend for EasyBuild

Current functionalities

- Parallel build based on Slurm → `--job Slurm`
- Submit each build as an individual Slurm job
- Handle dependencies between Slurm jobs

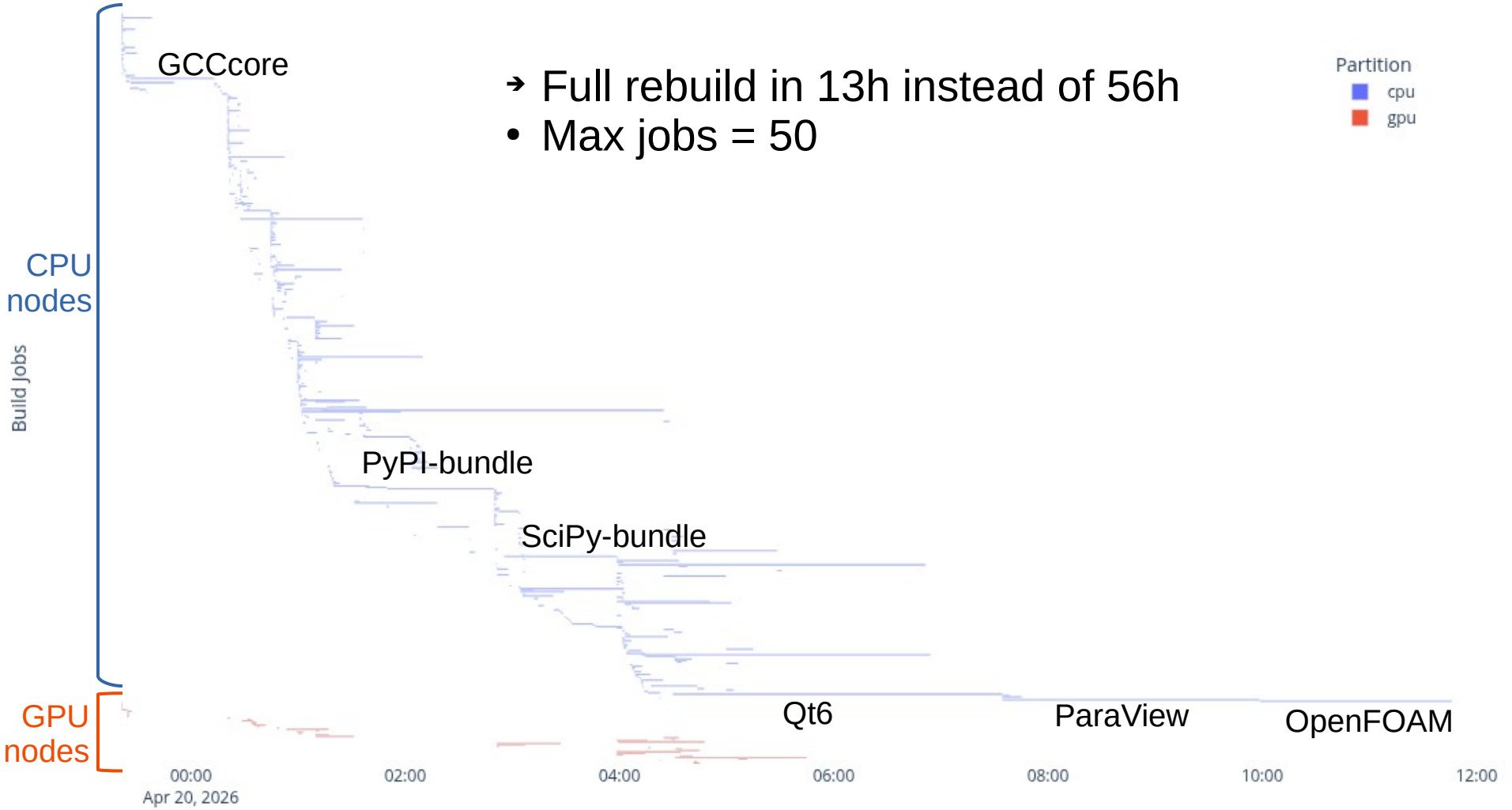
New functionalities

- Respect the **`--job-max-jobs`** setting, and manage a queue accordingly
- Track the jobs and print status according to **`--job-polling-interval`**
- Print a summary of all build jobs status at the end

Freshly opened merge request

- https://github.com/easybuilders/easybuild-framework/pull/5177
- Feel free to try, comment, and review

Improved Slurm backend for EasyBuild



- ➔ Full rebuild in 13h instead of 56h
- Max jobs = 50

Local customization with hooks

Automatic selection of the node type for each build job

- **`pre_run_shell_cmd_hook`** to update the sbatch command
  - CUDA-based software → **`--partition gpu`**
  - Other jobs → **`--partition cpu`**
  - Require to run Easybuild with **`--job Slurm`**

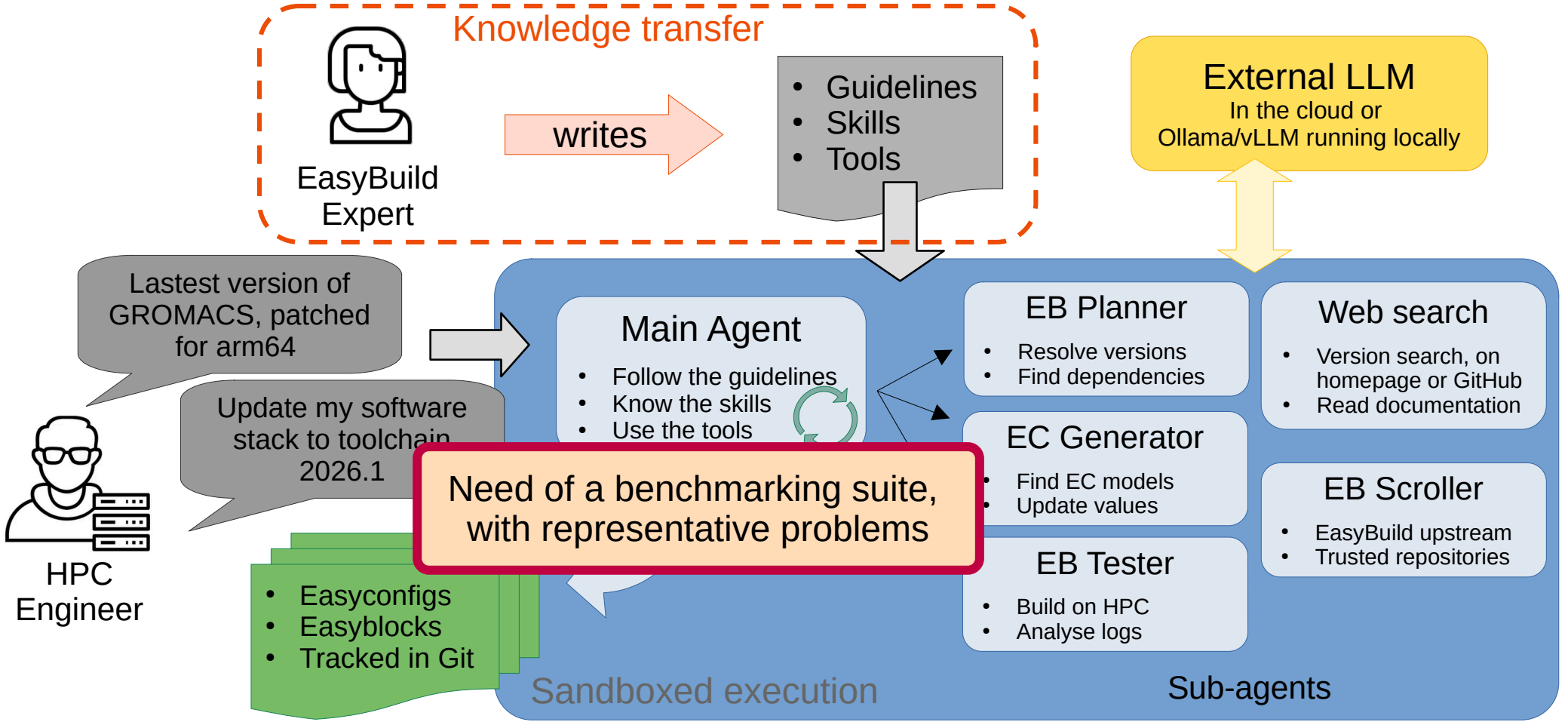
Static Application Security Testing (SAST) for installed software

- **`post_patch_hook`** to archive the patched source
  - Checksum of the archive
  - SAST analysis with semgrep or other tools (later on)

Work In Progress

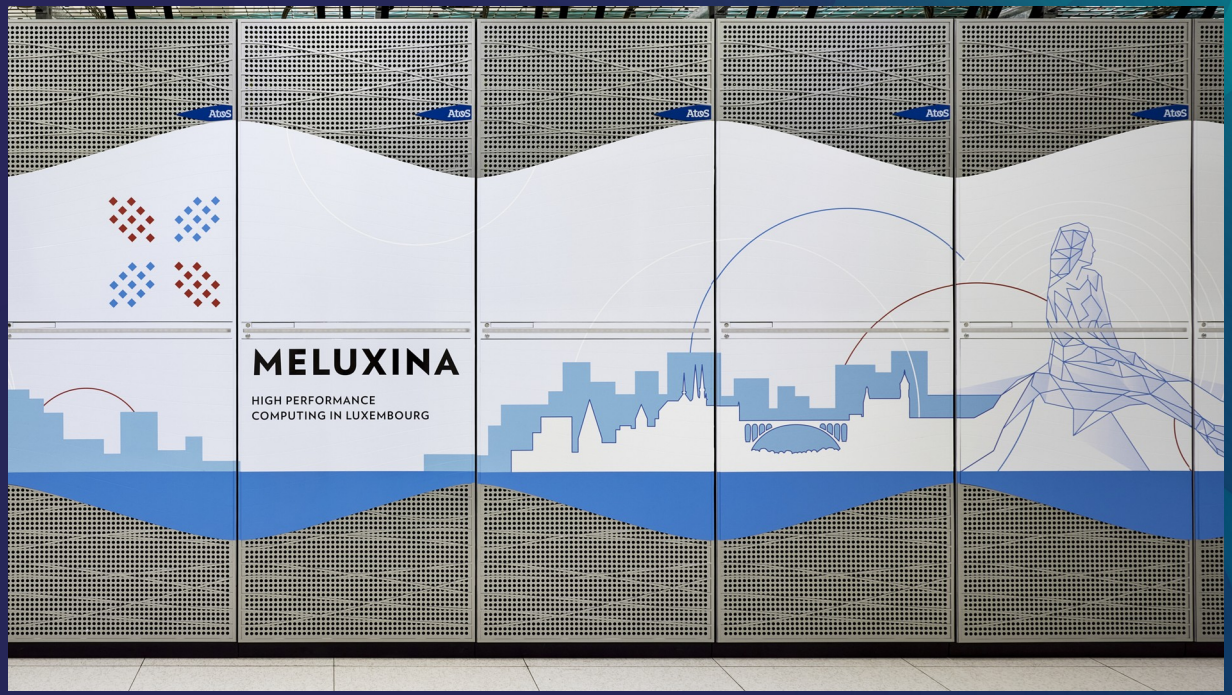
AI Coding Assistant for EasyBuild

Can we teach an AI Coding Assistant to code like an EasyBuild Human Expert?



Thank you!

Any questions?



powered by

EasyBuild.io

building software with ease