LUMI Software Stacks

Kurt Lust
LUMI User Support Team (LUST)
VSC Tier-0 support, University of Antwerp

May 2024

Software stack design considerations

- Very leading edge and inhomogeneous machine (new interconnect, new GPU architecture with an immature software ecosystem, some NVIDIA GPUs for visualisation, a mix of zen2 and zen3)
    - Need to remain agile
- Users that come to LUMI from 12 different channels (not counting subchannels), with different expectations
- Small central support team considering the expected number of projects and users and the tasks the support team has
    - But contributions from local support teams
- Cray Programming Environment is a key part of our system
- Users really want more and more a customised environment
    - Everybody wants a central stack as long as their software is in there but not much more
    - Look at the success of conda, Python virtual environments, containers, ...

The LUMI solution

- Software organised in extensible software stacks based on a particular release of the PE
  - Many base libraries and some packages already pre-installed
  - Easy way to install additional packages in project space
- Modules managed by Lmod
  - More powerful than the (old) Modules Environment
  - Powerful features to search for modules
- EasyBuild is our primary tool for software installations
  - But uses HPE Cray specific toolchains
  - Offer a library of installation recipes
  - User installations integrate seamlessly with the central stack
  - We do have a Spack setup but don’t do development in Spack ourselves

Policies

- Bring-your-own-license except for a selection of tools that are useful to a larger community
  - One downside of the distributed user management is that we do not even have the information needed to determine if a particular userid can use a particular software license
  - Even for software on the system, users remain responsible for checking the license!
- LUST tries to help with installations of recent software, but porting or bug fixing is not our work
  - Not all Linux or even supercomputer software will work on LUMI
  - We're too small a team to do all software installations, so don't count on us to do all the work . The diversity in requested packages is just too high.
- Conda, (large) Python installations need to go in containers
  - We offer cotainr and lumi-container-wrapper to do that

Organisation: Software stacks

LUMI

- **CrayEnv:** Cray environment with some additional tools pushed in through EasyBuild
- **LUMI** stacks, each one corresponding to a particular release of the PE
  - Work with the Cray PE modules, but accessed through a replacement for the PrgEnv-* modules
  - Tuned versions for the 3 ~~4~~ types of hardware: zen2 (login, large memory nodes), zen3 (LUMI-C compute nodes), ~~zen2 + NVIDIA GPU (visualisation partition)~~, zen3 + MI250X (LUMI-G GPU partition)
- **spack:** Install software with Spack using compilers from the PE
  - Offered as-is for users who know Spack, but we do not do development in Spack
- Far future: Stack based on common EB foss toolchain as-is for LUMI-C
  - No plans for EESSI as it is a bad match with LUMI

Accessing the Cray PE on LUMI
3 different ways

- Very bare environment available directly after login
  - What you can expect on a typical Cray system
  - Few tools as only the base OS image is available
  - User fully responsible for managing the target modules
- **CrayEnv**
  - "Enriched" Cray PE environment
  - Takes care of managing the target modules: (re)loading CrayEnv will reload an optimal set for the node you're on
  - Some additional tools, e.g., newer build tools (offered here and not in the bare environment as we need to avoid conflicts with other software stacks)
  - Otherwise used in the way discussed in this course

Accessing the Cray PE on LUMI

3 different ways

- **LUMI** software stack
  - Each stack based on a particular release of the HPE Cray PE
    - Other modules are accessible but hidden from the default view
  - Better not to use the PrgEnv modules but the EasyBuild LUMI toolchains

HPE Cray PE	LUMI toolchain	
PrgEnv-cray	cpeCray	Cray Compiling Environment
PrgEnv-gnu	cpeGNU	GNU C/C++ and Fortran
PrgEnv-aocc	cpeAOCC	AMD CPU compilers (not on LUMI-G)
PrgEnv-amd	cpeAMD	AMD ROCm GPU compilers (LUMI-G only)

  - Environment in which we install most software (mostly with EasyBuild)

Accessing the Cray PE on LUMI
The LUMI software stack

- The LUMI software stack uses two levels of modules
  - LUMI/22.08, LUMI/22.12, LUMI/23.03, LUMI/23.09: Versions of the LUMI stack
  - partition/L, partition/C, partition/G (~~and future partition/D~~): To select software optimised for the respective LUMI partition
    - partition/L is for both the login nodes and the large memory nodes (4TB)
  - Hidden partition/common for software that is available everywhere, but be careful using it for your own installs
  - When (re)loaded, the LUMI module will load the best matching partition module.
  - So be careful in job scripts: When your job starts, the environment will be that of the login nodes, but if you reload the LUMI module it will be that of the compute node!

Installing software on HPC systems

- Software on an HPC system is rarely installed from RPM
  - Generic RPMs often not optimised for the specific CPU
  - Generic RPMs may not work with the specific LUMI environment (Slingshot interconnect, kernel modules, resource manager)
  - Multi-user system so usually no "one version fits all"
  - Need a small system image as nodes are diskless
- Spack and EasyBuild are the two most popular HPC-specific software build and installation frameworks
  - Usually install from sources to adapt the software to the underlying hardware and OS
  - Installation instructions in a way that can be communicated and executed easily
  - Make software available via modules
  - Dependency handling compatible with modules

Extending the LUMI stack with EasyBuild

- Fully integrated in the LUMI software stack
  - Load the LUMI module and modules should appear in your module view
  - EasyBuild-user module to install packages in your user space
  - Will use existing modules for dependencies if those are already on the system or in your personal/project stack
- EasyBuild built-in easyconfigs do not work well on LUMI, not even on LUMI-C
  - GNU-based toolchains: Would give problems with MPI
  - Intel-based toolchains: Intel compilers and AMD CPUs are a problematic cocktail
- Library of recipes that we made in the LUMI-EasyBuild-contrib GitHub repository
  - EasyBuild-user will find a copy on the system or in your installation
  - List of recipes in the LUMI Software Library

EasyBuild recipes - easyconfigs

- Build recipe for an individual package = module
  - Relies on either a generic or a specific installation process provided by an easyblock
- Steps
  - Downloading sources and patches
  - Typical configure – build – (test) – install process
  - Extensions mechanism for perl/python/R packages
  - Some simple checks
  - Creation of the module
- All have several parameters in the easyconfig file

The toolchain concept

- A set of compiler, MPI implementation and basic math libraries
  - Simplified concept on LUMI as there is no hierarchy as on some other EasyBuild systems
- These are the cpeCray, cpeGNU, cpeAOCC and cpeAMD modules mentioned before!

HPE Cray PE	LUMI toolchain	
PrgEnv-cray	cpeCray	Cray Compiling Environment
PrgEnv-gnu	cpeGNU	GNU C/C++ and Fortran
PrgEnv-aocc	cpeAOCC	AMD CPU compilers (not on LUMI-G)
PrgEnv-amd	cpeAMD	AMD ROCm GPU compilers (LUMI-G only)

The toolchain concept (2)

- Special toolchain: SYSTEM to use the system compiler
  - Does not fully function in the same way as the other toolchains when it comes to dependency handling
  - Used on LUMI for CrayEnv and some packages with few dependencies
- It is not possible to load packages from different cpe toolchains at the same time
  - EasyBuild restriction, because mixing libraries compiled with different compilers does not always work
- Packages compiled with one cpe toolchain can be loaded together with packages compiled with the SYSTEM toolchain
  - But we do avoid mixing them when linking

easyconfig names and module names

GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb

Additional information

Toolchain name and version (missing for SYSTEM)

Version of the package

Name of the package

Module: GROMACS/2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU

Installing
Step 1: Where to install

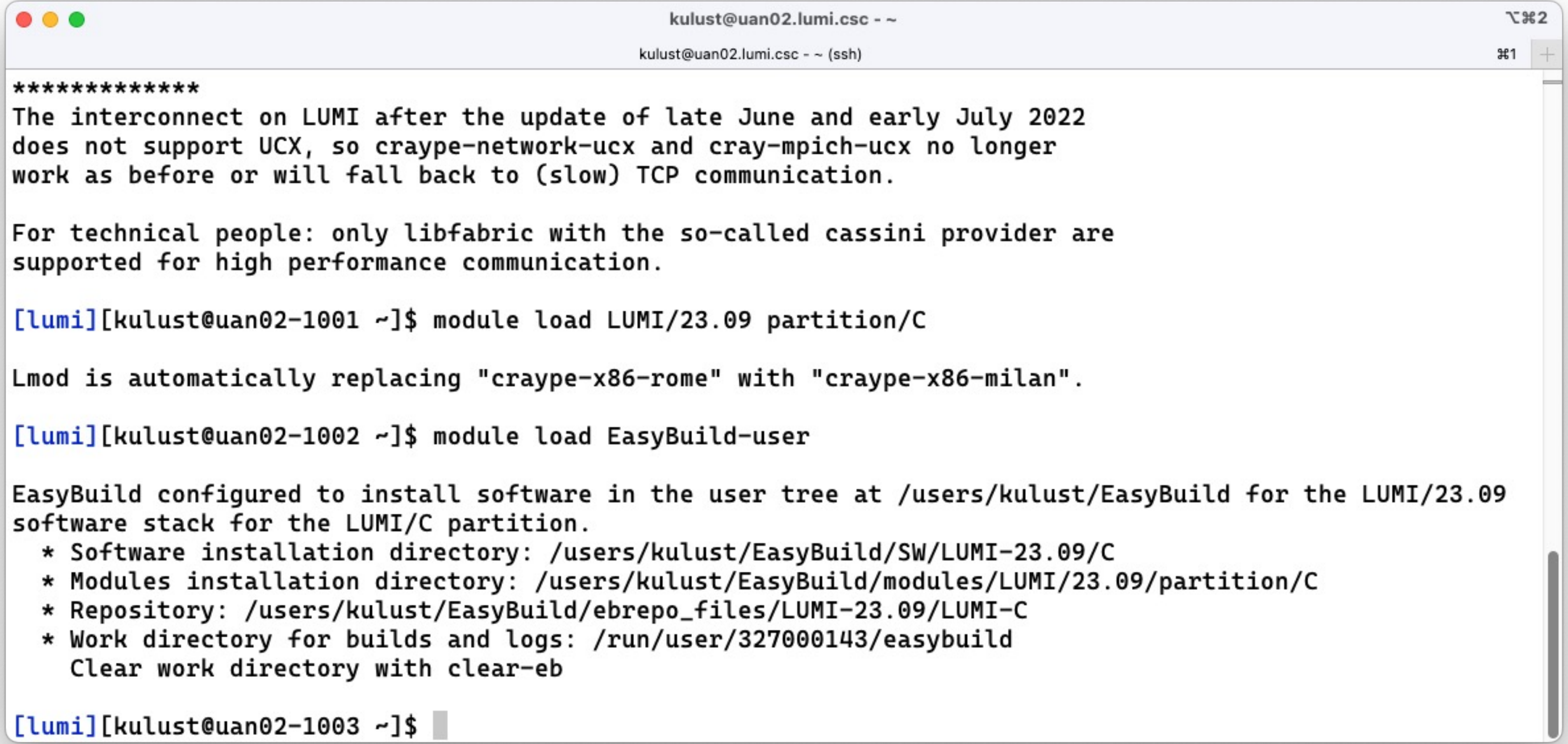
- Default location is `$HOME/EasyBuild`
- But better is to install in your project directory for the whole project
  - `export EBU_USER_PREFIX=/project/project_465000000/EasyBuild`
  - Set this *before* loading the `LUMI` module
  - All users of the software tree have to set this environment variable to use the software tree

Installing
Step 2: Configure the environment

L U M I

- Load the modules for the LUMI software stack and partition that you want to use. E.g.,
  ```
  module load LUMI/23.09 partition/C
  ```
- Load the EasyBuild-user module to make EasyBuild available and to configure it for installing software in the chosen stack and partition:
  ```
  module load EasyBuild-user
  ```
- In many cases, cross-compilation is possible by loading a different partition module than the one auto-loaded by LUMI
  - Though cross-compilation is currently problematic for GPU code

```
module load LUMI/23.09 partition/C
module load EasyBuild-user
```



kulust@uan02.lumi.csc - ~

kulust@uan02.lumi.csc - ~ (ssh)

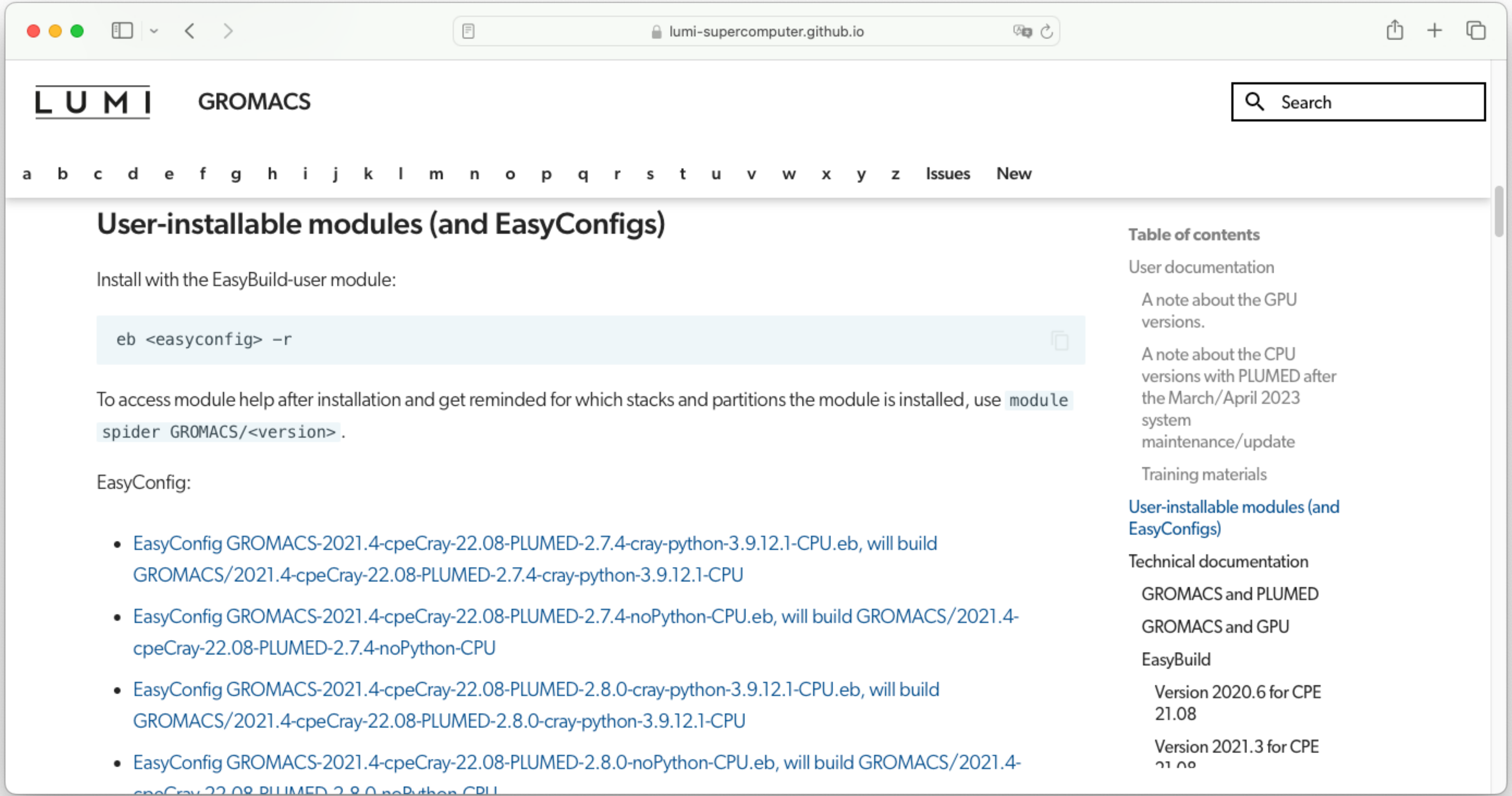
```
*************
The interconnect on LUMI after the update of late June and early July 2022
does not support UCX, so craype-network-ucx and cray-mpich-ucx no longer
work as before or will fall back to (slow) TCP communication.

For technical people: only libfabric with the so-called cassini provider are
supported for high performance communication.

[lumi][kulust@uan02-1001 ~]$ module load LUMI/23.09 partition/C

Lmod is automatically replacing "craype-x86-rome" with "craype-x86-milan".

[lumi][kulust@uan02-1002 ~]$ module load EasyBuild-user

EasyBuild configured to install software in the user tree at /users/kulust/EasyBuild for the LUMI/23.09
software stack for the LUMI/C partition.
  * Software installation directory: /users/kulust/EasyBuild/SW/LUMI-23.09/C
  * Modules installation directory: /users/kulust/EasyBuild/modules/LUMI/23.09/partition/C
  * Repository: /users/kulust/EasyBuild/ebrepo_files/LUMI-23.09/LUMI-C
  * Work directory for builds and logs: /run/user/327000143/easybuild
    Clear work directory with clear-eb

[lumi][kulust@uan02-1003 ~]$
```

Installing
Step 3: Install the software

- Let’s, e.g., install GROMACS
  - Search if GROMACS build recipes are available:
    - Search the LUMI Software Library that lists all available software through EasyBuild.
    - Or on the command line:

      ```
      eb --search GROMACS
      eb –S GROMACS
      ```

LUMI Software Library



lumi-supercomputer.github.io

GROMACS

Search

a b c d e f g h i j k l m n o p q r s t u v w x y z Issues New

User-installable modules (and EasyConfigs)

Install with the EasyBuild-user module:

```
eb <easyconfig> -r
```

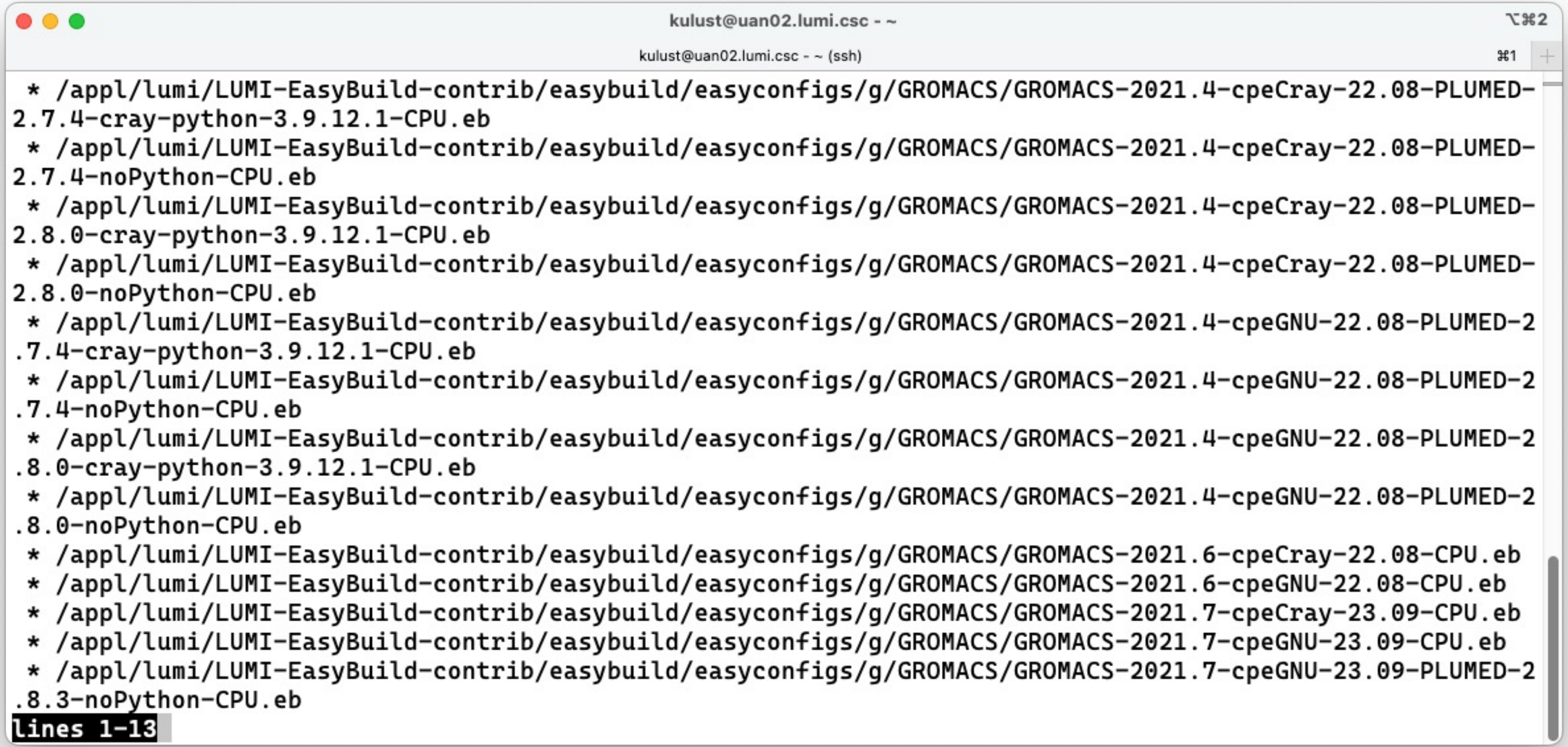
To access module help after installation and get reminded for which stacks and partitions the module is installed, use `module spider GROMACS/<version>`.

EasyConfig:

- EasyConfig GROMACS-2021.4-cpeCray-22.08-PLUMED-2.7.4-cray-python-3.9.12.1-CPU.eb, will build GROMACS/2021.4-cpeCray-22.08-PLUMED-2.7.4-cray-python-3.9.12.1-CPU
- EasyConfig GROMACS-2021.4-cpeCray-22.08-PLUMED-2.7.4-noPython-CPU.eb, will build GROMACS/2021.4-cpeCray-22.08-PLUMED-2.7.4-noPython-CPU
- EasyConfig GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-cray-python-3.9.12.1-CPU.eb, will build GROMACS/2021.4-cpeCray-22.08-PLUMED-2.8.0-cray-python-3.9.12.1-CPU
- EasyConfig GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-noPython-CPU.eb, will build GROMACS/2021.4-cpeCray-22.08-PLUMED-2.8.0-noPython-CPU

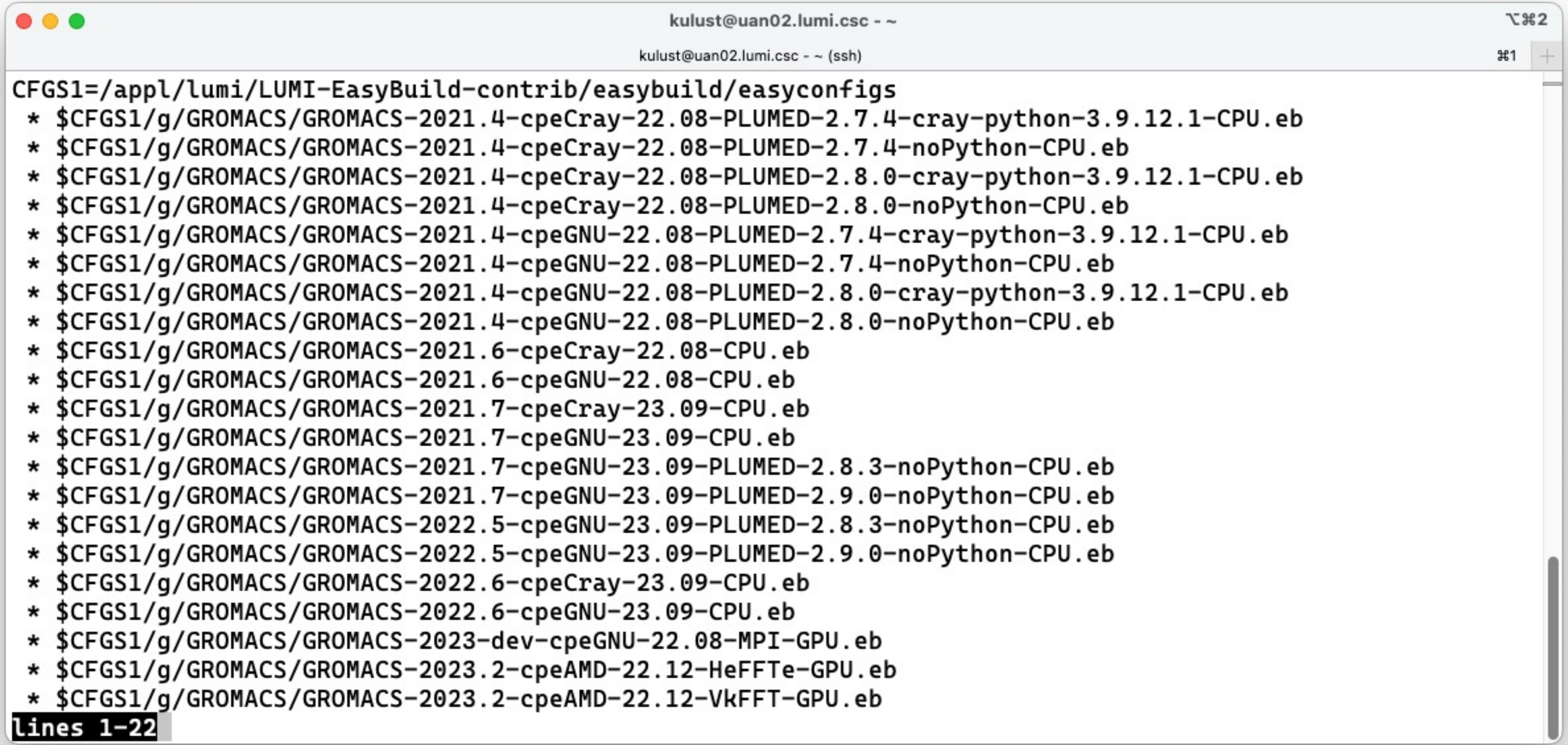
```
eb --search GROMACS | less
```

LUMI



```
kulust@uan02.lumi.csc - ~
kulust@uan02.lumi.csc - ~ (ssh)
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeCray-22.08-PLUMED-
2.7.4-cray-python-3.9.12.1-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeCray-22.08-PLUMED-
2.7.4-noPython-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeCray-22.08-PLUMED-
2.8.0-cray-python-3.9.12.1-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeCray-22.08-PLUMED-
2.8.0-noPython-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeGNU-22.08-PLUMED-2
.7.4-cray-python-3.9.12.1-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeGNU-22.08-PLUMED-2
.7.4-noPython-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeGNU-22.08-PLUMED-2
.8.0-cray-python-3.9.12.1-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeGNU-22.08-PLUMED-2
.8.0-noPython-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.6-cpeCray-22.08-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.6-cpeGNU-22.08-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.7-cpeCray-23.09-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.7-cpeGNU-23.09-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.7-cpeGNU-23.09-PLUMED-2
.8.3-noPython-CPU.eb
lines 1-13
```

eb -S GROMACS | less



```
CFGS1=/appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs
 * $CFGS1/g/GROMACS/GROMACS-2021.4-cpeCray-22.08-PLUMED-2.7.4-cray-python-3.9.12.1-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2021.4-cpeCray-22.08-PLUMED-2.7.4-noPython-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-cray-python-3.9.12.1-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-noPython-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2021.4-cpeGNU-22.08-PLUMED-2.7.4-cray-python-3.9.12.1-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2021.4-cpeGNU-22.08-PLUMED-2.7.4-noPython-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2021.4-cpeGNU-22.08-PLUMED-2.8.0-cray-python-3.9.12.1-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2021.4-cpeGNU-22.08-PLUMED-2.8.0-noPython-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2021.6-cpeCray-22.08-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2021.6-cpeGNU-22.08-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2021.7-cpeCray-23.09-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2021.7-cpeGNU-23.09-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2021.7-cpeGNU-23.09-PLUMED-2.8.3-noPython-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2021.7-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.8.3-noPython-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2022.6-cpeCray-23.09-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2022.6-cpeGNU-23.09-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2023-dev-cpeGNU-22.08-MPI-GPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2023.2-cpeAMD-22.12-HeFFTe-GPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2023.2-cpeAMD-22.12-VkFFT-GPU.eb
lines 1-22
```

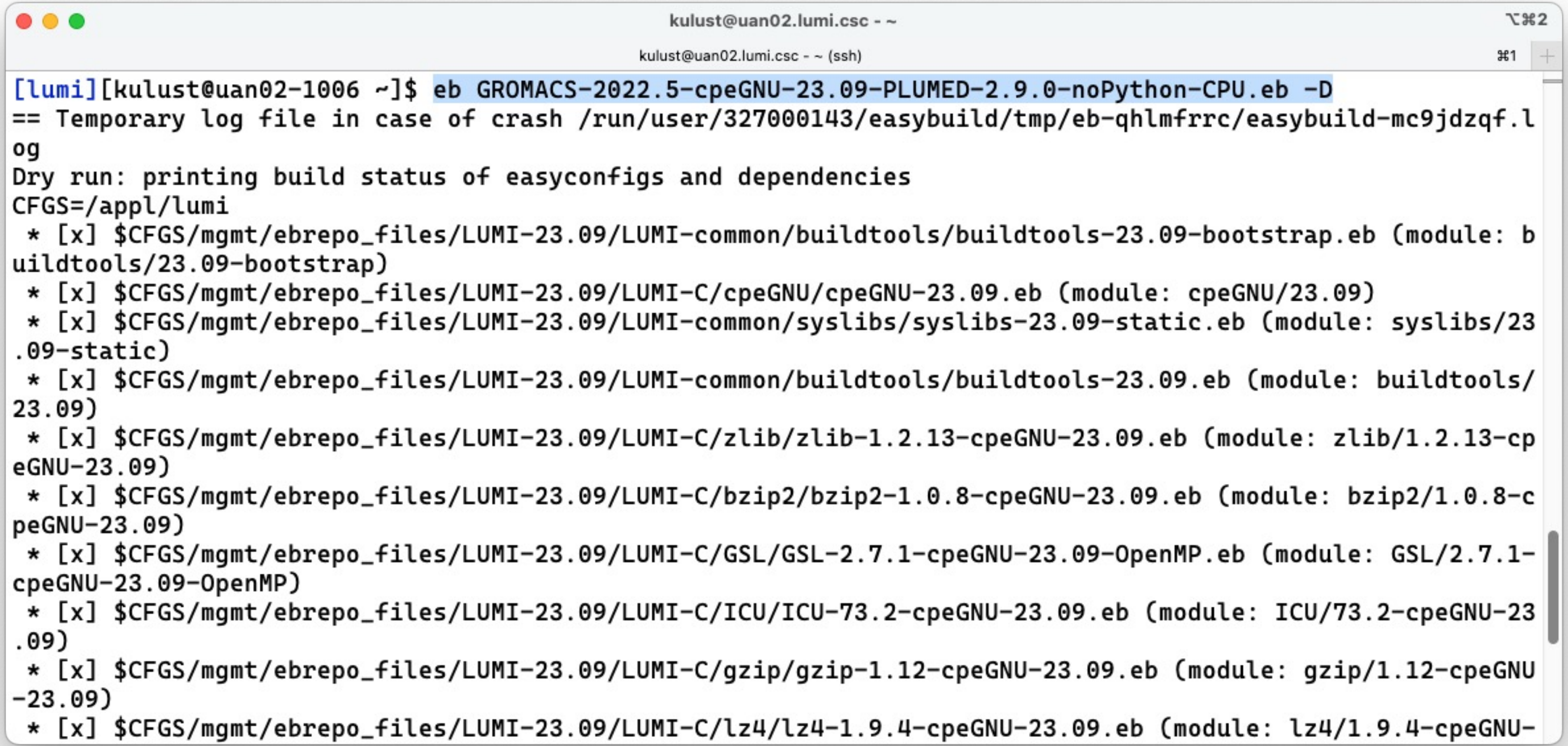
Installing
Step 3: Install the software

- Let’s, e.g., install GROMACS
  - Search if GROMACS build recipes are available:
    - Search the LUMI Software Library that lists all available software through EasyBuild.
    - Or on the command line:

      ```
      eb --search GROMACS
      eb –S GROMACS
      ```
  - Let’s take GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb:

    ```
    eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -D
    ```

eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -D

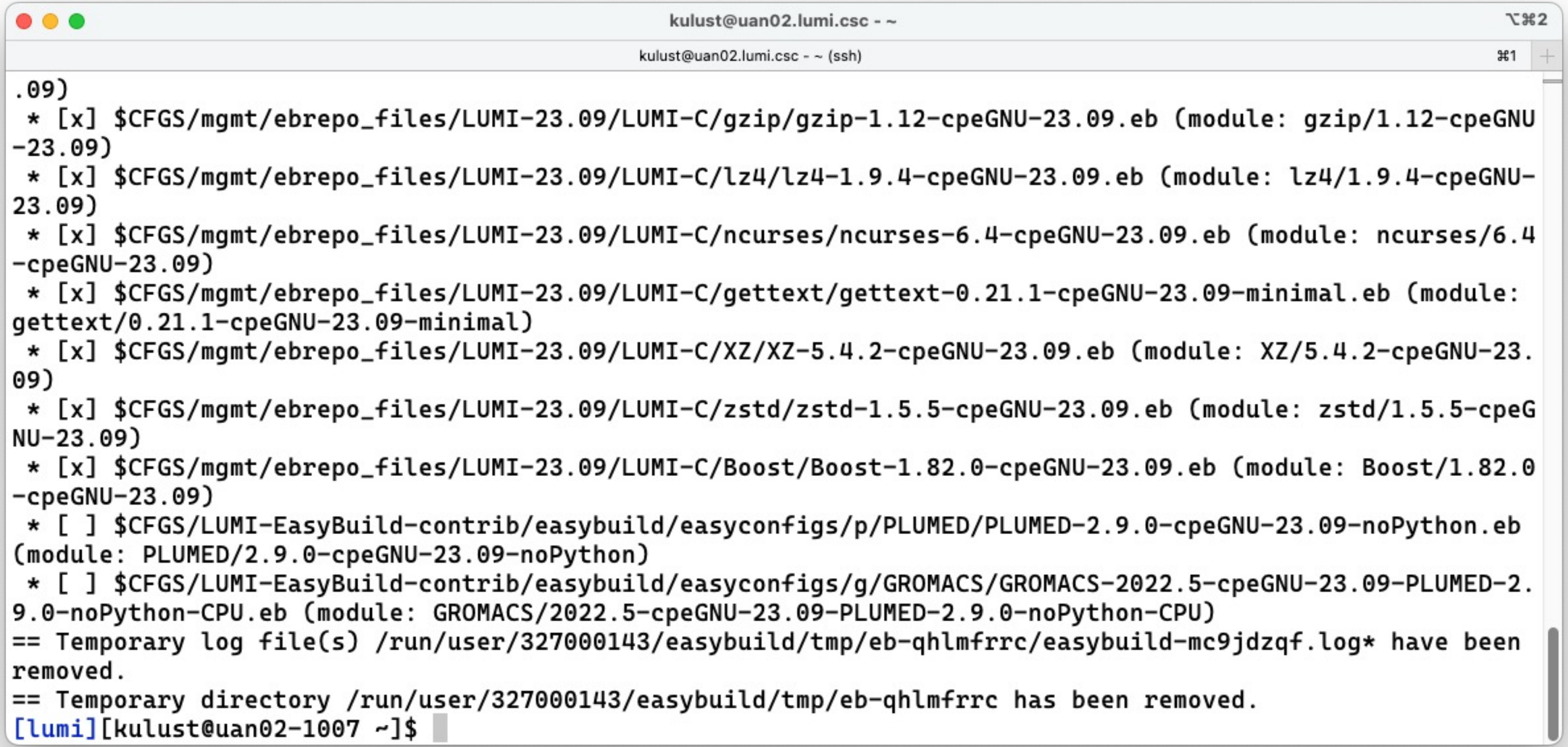


kulust@uan02.lumi.csc - ~

kulust@uan02.lumi.csc - ~ (ssh)

```
[lumi][kulust@uan02-1006 ~]$ eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -D
== Temporary log file in case of crash /run/user/327000143/easybuild/tmp/eb-qhlmfrrc/easybuild-mc9jdzqf.l
og
Dry run: printing build status of easyconfigs and dependencies
CFGS=/appl/lumi
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-common/buildtools/buildtools-23.09-bootstrap.eb (module: b
uildtools/23.09-bootstrap)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-C/cpeGNU/cpeGNU-23.09.eb (module: cpeGNU/23.09)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-common/syslibs/syslibs-23.09-static.eb (module: syslibs/23
.09-static)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-common/buildtools/buildtools-23.09.eb (module: buildtools/
23.09)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-C/zlib/zlib-1.2.13-cpeGNU-23.09.eb (module: zlib/1.2.13-cp
eGNU-23.09)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-C/bzip2/bzip2-1.0.8-cpeGNU-23.09.eb (module: bzip2/1.0.8-c
peGNU-23.09)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-C/GSL/GSL-2.7.1-cpeGNU-23.09-OpenMP.eb (module: GSL/2.7.1-
cpeGNU-23.09-OpenMP)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-C/ICU/ICU-73.2-cpeGNU-23.09.eb (module: ICU/73.2-cpeGNU-23
.09)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-C/gzip/gzip-1.12-cpeGNU-23.09.eb (module: gzip/1.12-cpeGNU
-23.09)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-C/lz4/lz4-1.9.4-cpeGNU-23.09.eb (module: lz4/1.9.4-cpeGNU-
```

eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -D (2)



kulust@uan02.lumi.csc - ~

kulust@uan02.lumi.csc - ~ (ssh)

```
.09)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-C/gzip/gzip-1.12-cpeGNU-23.09.eb (module: gzip/1.12-cpeGNU
-23.09)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-C/lz4/lz4-1.9.4-cpeGNU-23.09.eb (module: lz4/1.9.4-cpeGNU-
23.09)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-C/ncurses/ncurses-6.4-cpeGNU-23.09.eb (module: ncurses/6.4
-cpeGNU-23.09)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-C/gettext/gettext-0.21.1-cpeGNU-23.09-minimal.eb (module:
gettext/0.21.1-cpeGNU-23.09-minimal)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-C/XZ/XZ-5.4.2-cpeGNU-23.09.eb (module: XZ/5.4.2-cpeGNU-23.
09)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-C/zstd/zstd-1.5.5-cpeGNU-23.09.eb (module: zstd/1.5.5-cpeG
NU-23.09)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-23.09/LUMI-C/Boost/Boost-1.82.0-cpeGNU-23.09.eb (module: Boost/1.82.0
-cpeGNU-23.09)
 * [ ] $CFGS/LUMI-EasyBuild-contrib/easybuild/easyconfigs/p/PLUMED/PLUMED-2.9.0-cpeGNU-23.09-noPython.eb
(module: PLUMED/2.9.0-cpeGNU-23.09-noPython)
 * [ ] $CFGS/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.
9.0-noPython-CPU.eb (module: GROMACS/2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU)
== Temporary log file(s) /run/user/327000143/easybuild/tmp/eb-qhlmfrrc/easybuild-mc9jdzqf.log* have been
removed.
== Temporary directory /run/user/327000143/easybuild/tmp/eb-qhlmfrrc has been removed.
[lumi][kulust@uan02-1007 ~]$
```

Installing

Step 3: Install the software

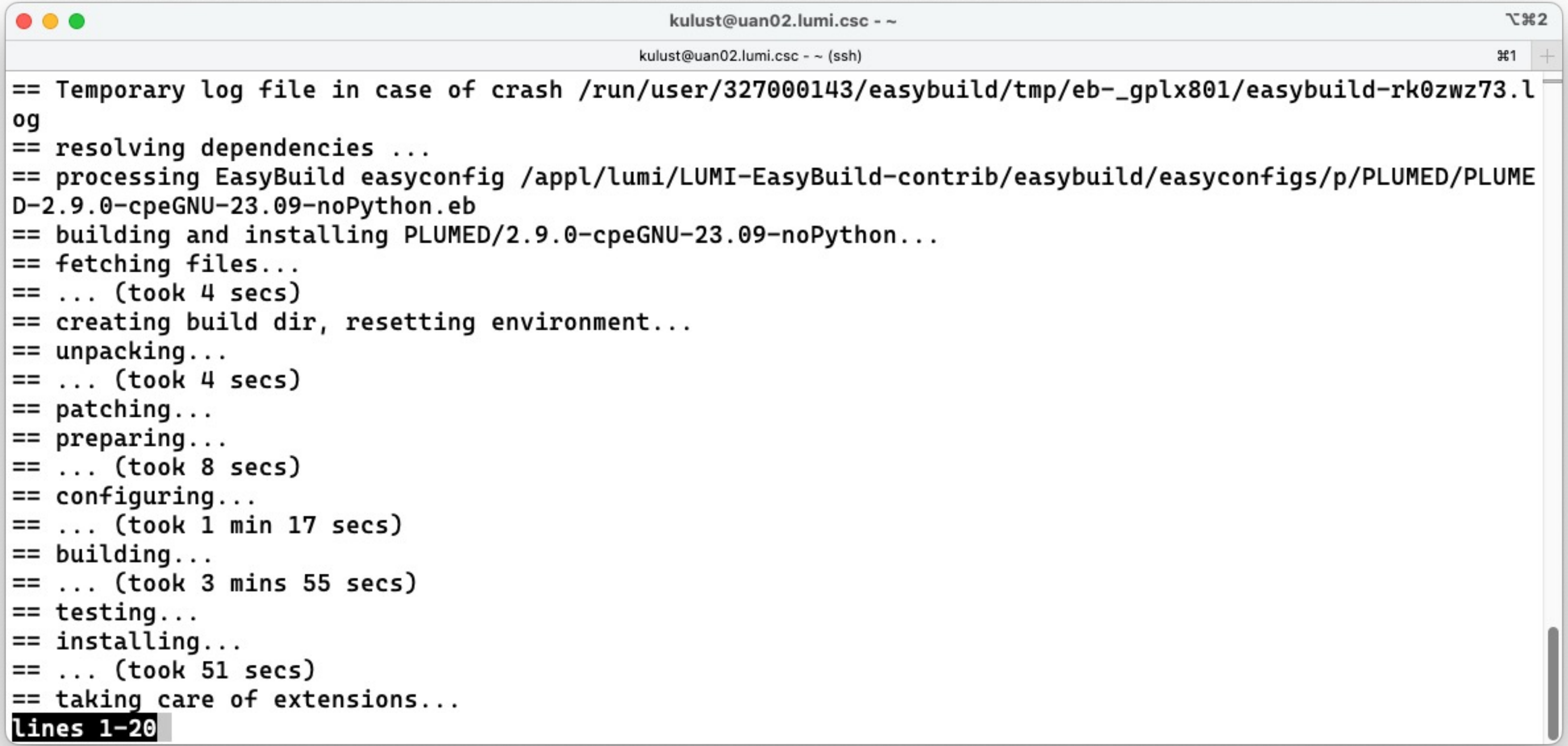
- Let’s, e.g., install GROMACS
  - Search if GROMACS build recipes are available:
    - Search the LUMI Software Library that lists all available software through EasyBuild.
    - Or on the command line:

      ```
      eb --search GROMACS
      eb –S GROMACS
      ```
  - Let’s take GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb:

    ```
    eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -D
    eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -r
    ```

eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -r

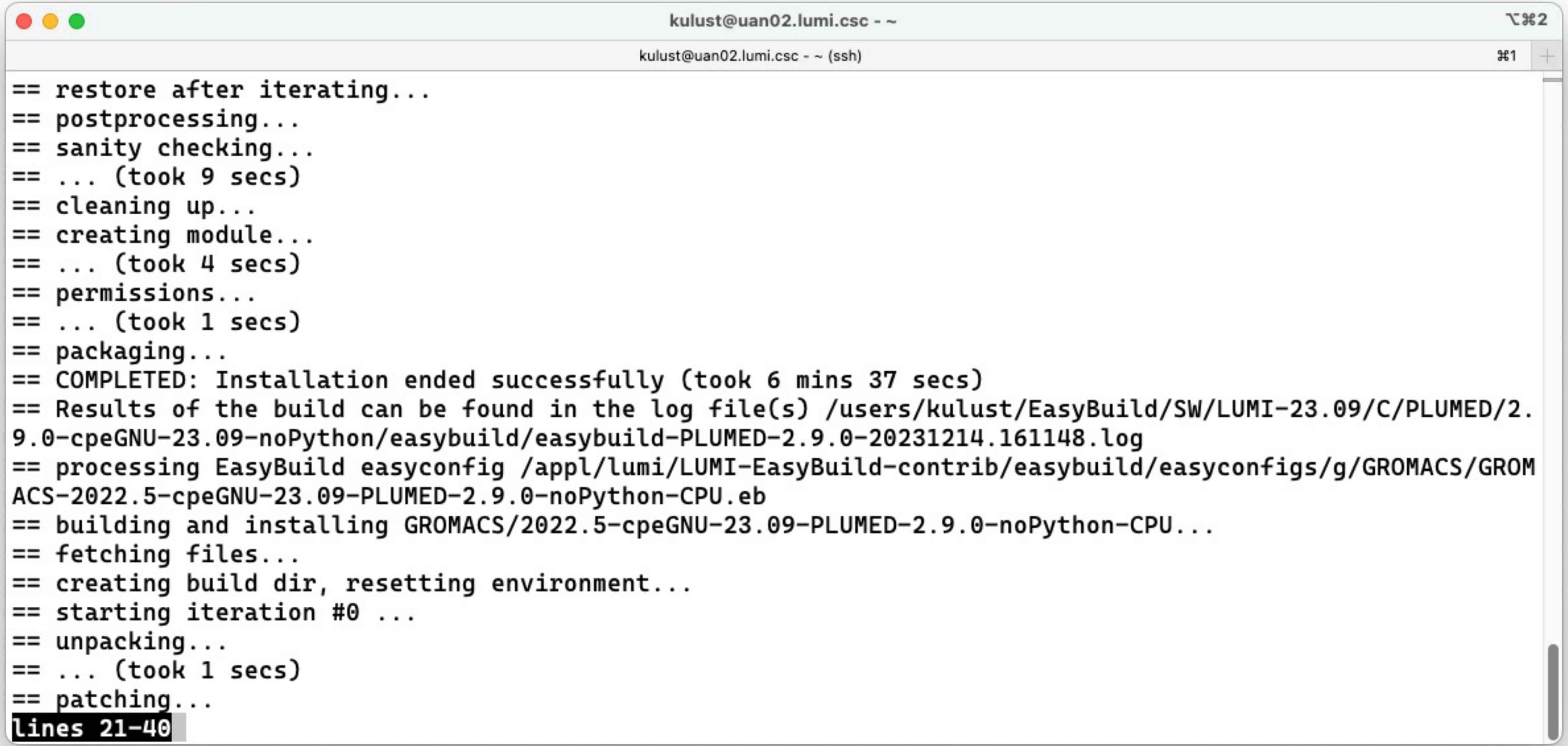
LUMI


kulust@uan02.lumi.csc - ~

kulust@uan02.lumi.csc - ~ (ssh)

```
== Temporary log file in case of crash /run/user/327000143/easybuild/tmp/eb-_gplx801/easybuild-rk0zwz73.l
og
== resolving dependencies ...
== processing EasyBuild easyconfig /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/p/PLUMED/PLUME
D-2.9.0-cpeGNU-23.09-noPython.eb
== building and installing PLUMED/2.9.0-cpeGNU-23.09-noPython...
== fetching files...
== ... (took 4 secs)
== creating build dir, resetting environment...
== unpacking...
== ... (took 4 secs)
== patching...
== preparing...
== ... (took 8 secs)
== configuring...
== ... (took 1 min 17 secs)
== building...
== ... (took 3 mins 55 secs)
== testing...
== installing...
== ... (took 51 secs)
== taking care of extensions...
lines 1-20
```

eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -r (2)

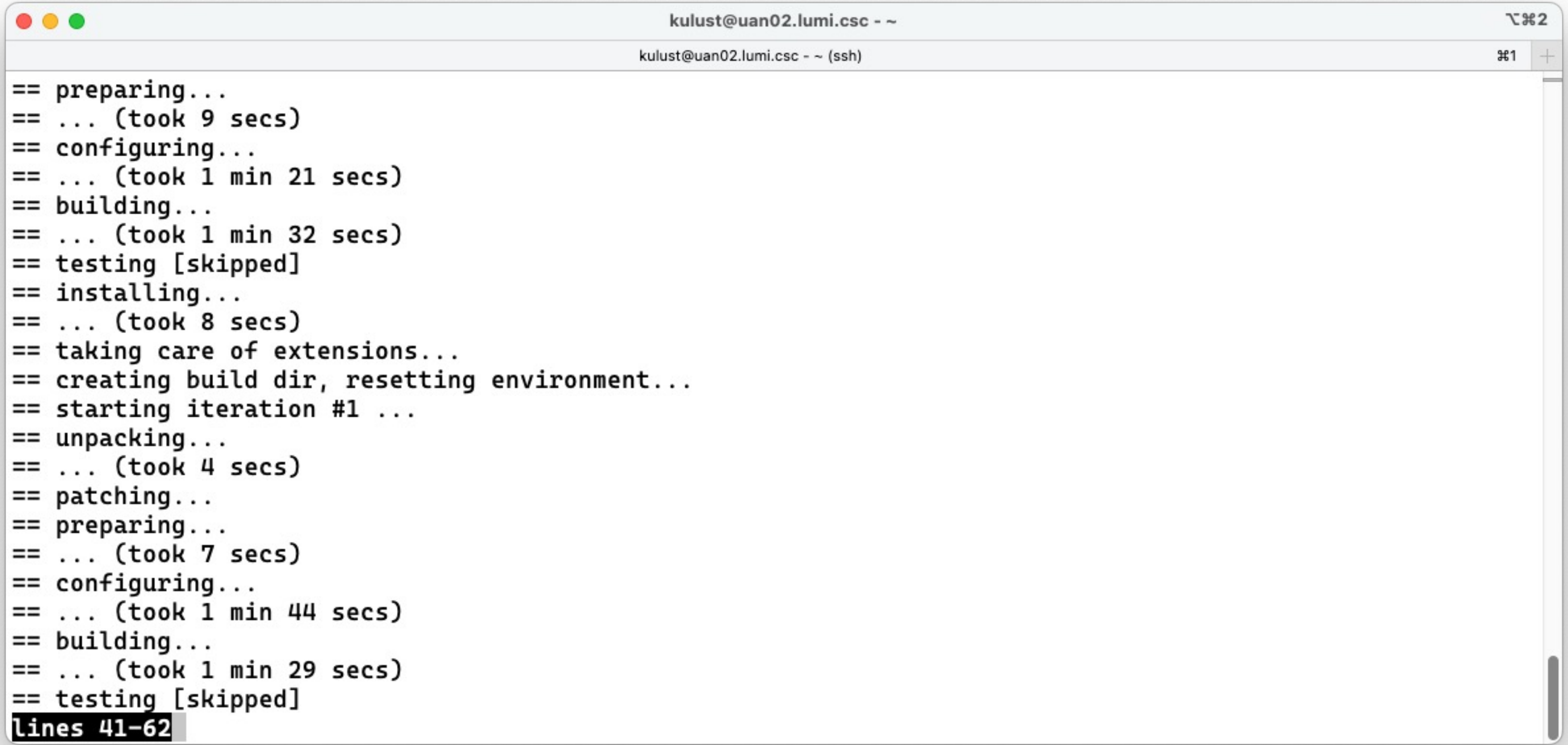

kulust@uan02.lumi.csc - ~

kulust@uan02.lumi.csc - ~ (ssh)

```
== restore after iterating...
== postprocessing...
== sanity checking...
== ... (took 9 secs)
== cleaning up...
== creating module...
== ... (took 4 secs)
== permissions...
== ... (took 1 secs)
== packaging...
== COMPLETED: Installation ended successfully (took 6 mins 37 secs)
== Results of the build can be found in the log file(s) /users/kulust/EasyBuild/SW/LUMI-23.09/C/PLUMED/2.
9.0-cpeGNU-23.09-noPython/easybuild/easybuild-PLUMED-2.9.0-20231214.161148.log
== processing EasyBuild easyconfig /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROM
ACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb
== building and installing GROMACS/2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU...
== fetching files...
== creating build dir, resetting environment...
== starting iteration #0 ...
== unpacking...
== ... (took 1 secs)
== patching...
lines 21-40
```

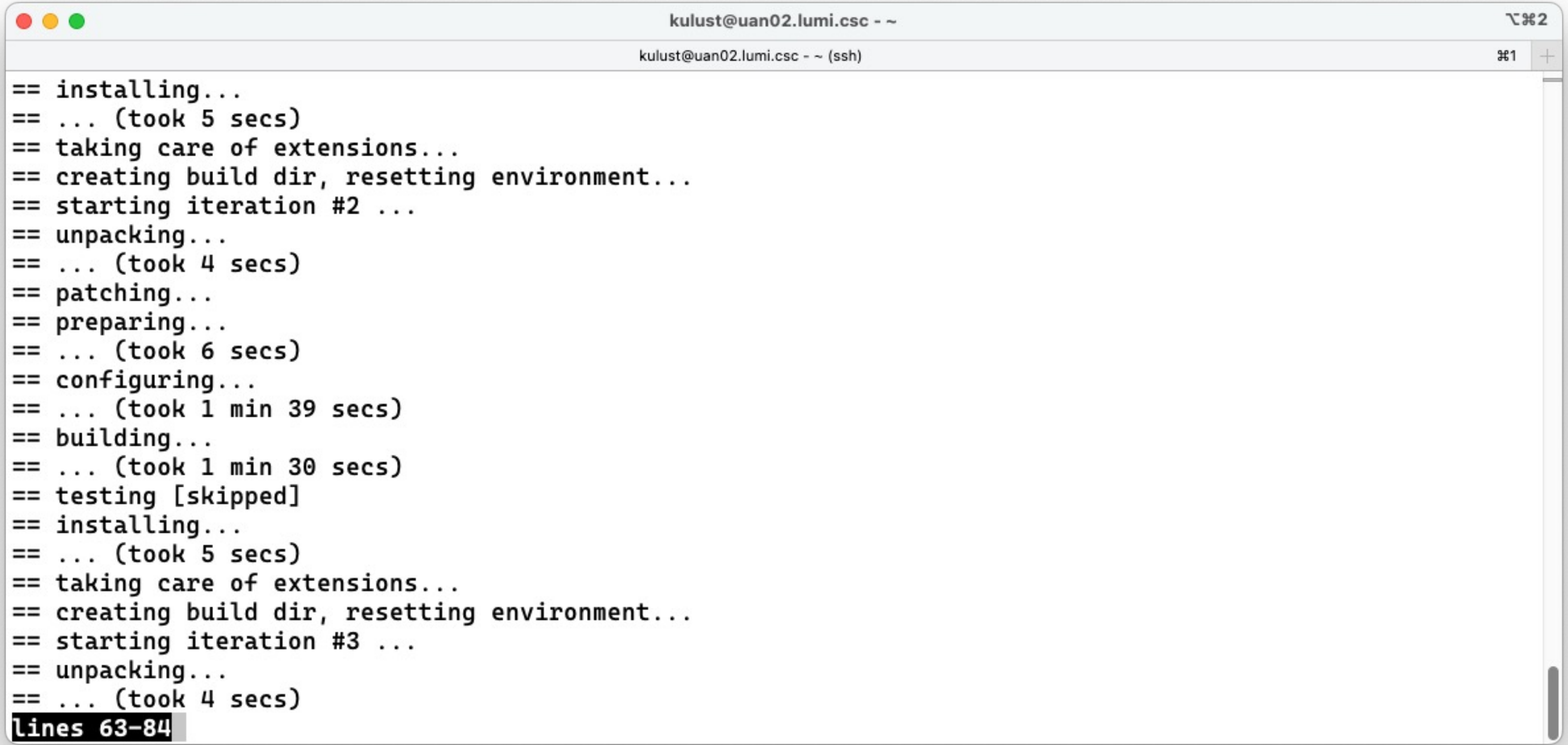
eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -r(3)

LUMI



```
== preparing...
== ... (took 9 secs)
== configuring...
== ... (took 1 min 21 secs)
== building...
== ... (took 1 min 32 secs)
== testing [skipped]
== installing...
== ... (took 8 secs)
== taking care of extensions...
== creating build dir, resetting environment...
== starting iteration #1 ...
== unpacking...
== ... (took 4 secs)
== patching...
== preparing...
== ... (took 7 secs)
== configuring...
== ... (took 1 min 44 secs)
== building...
== ... (took 1 min 29 secs)
== testing [skipped]
lines 41-62
```

eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb –r (4)

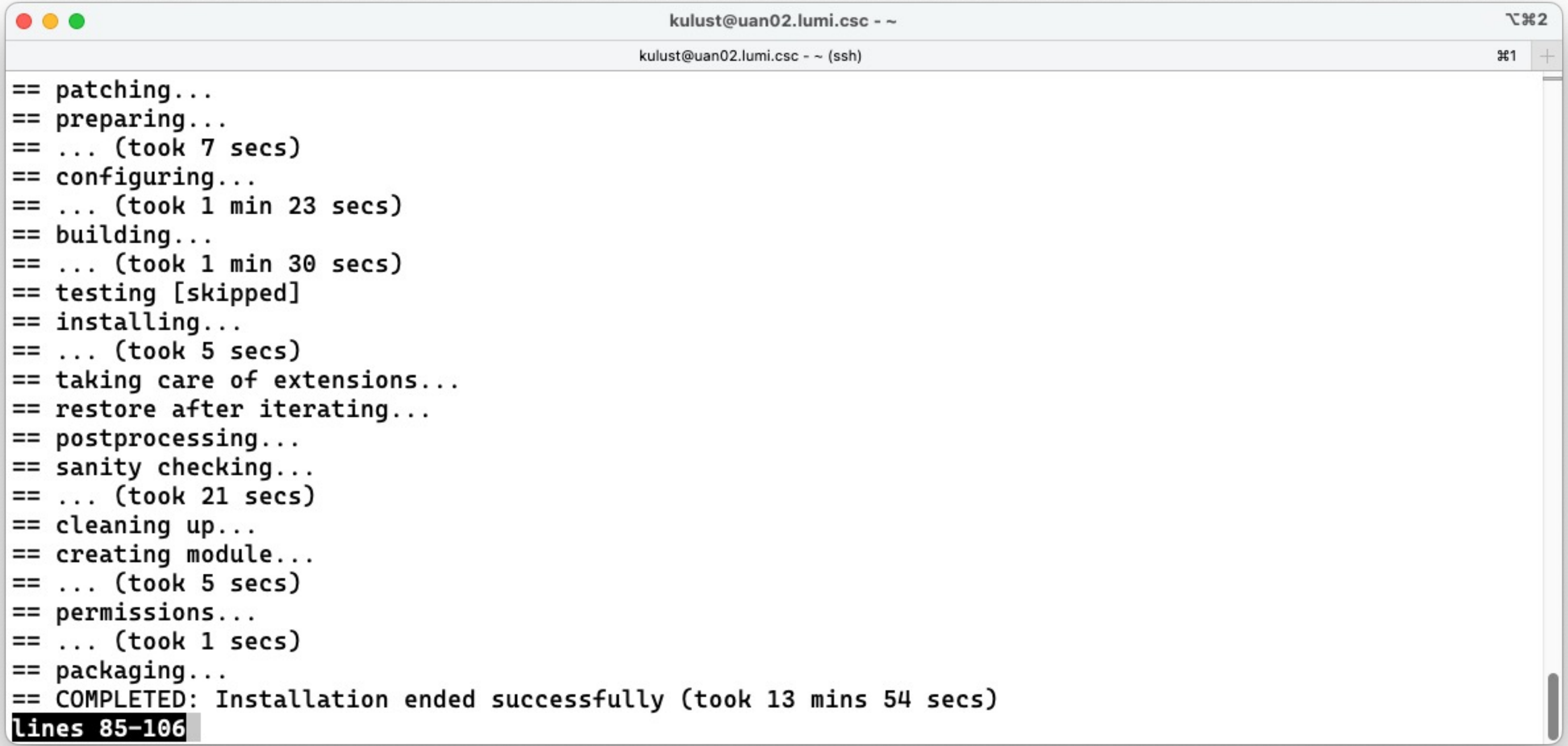

kulust@uan02.lumi.csc - ~

kulust@uan02.lumi.csc - ~ (ssh)

```
== installing...
== ... (took 5 secs)
== taking care of extensions...
== creating build dir, resetting environment...
== starting iteration #2 ...
== unpacking...
== ... (took 4 secs)
== patching...
== preparing...
== ... (took 6 secs)
== configuring...
== ... (took 1 min 39 secs)
== building...
== ... (took 1 min 30 secs)
== testing [skipped]
== installing...
== ... (took 5 secs)
== taking care of extensions...
== creating build dir, resetting environment...
== starting iteration #3 ...
== unpacking...
== ... (took 4 secs)
lines 63-84
```

eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -r (5)

LUMI

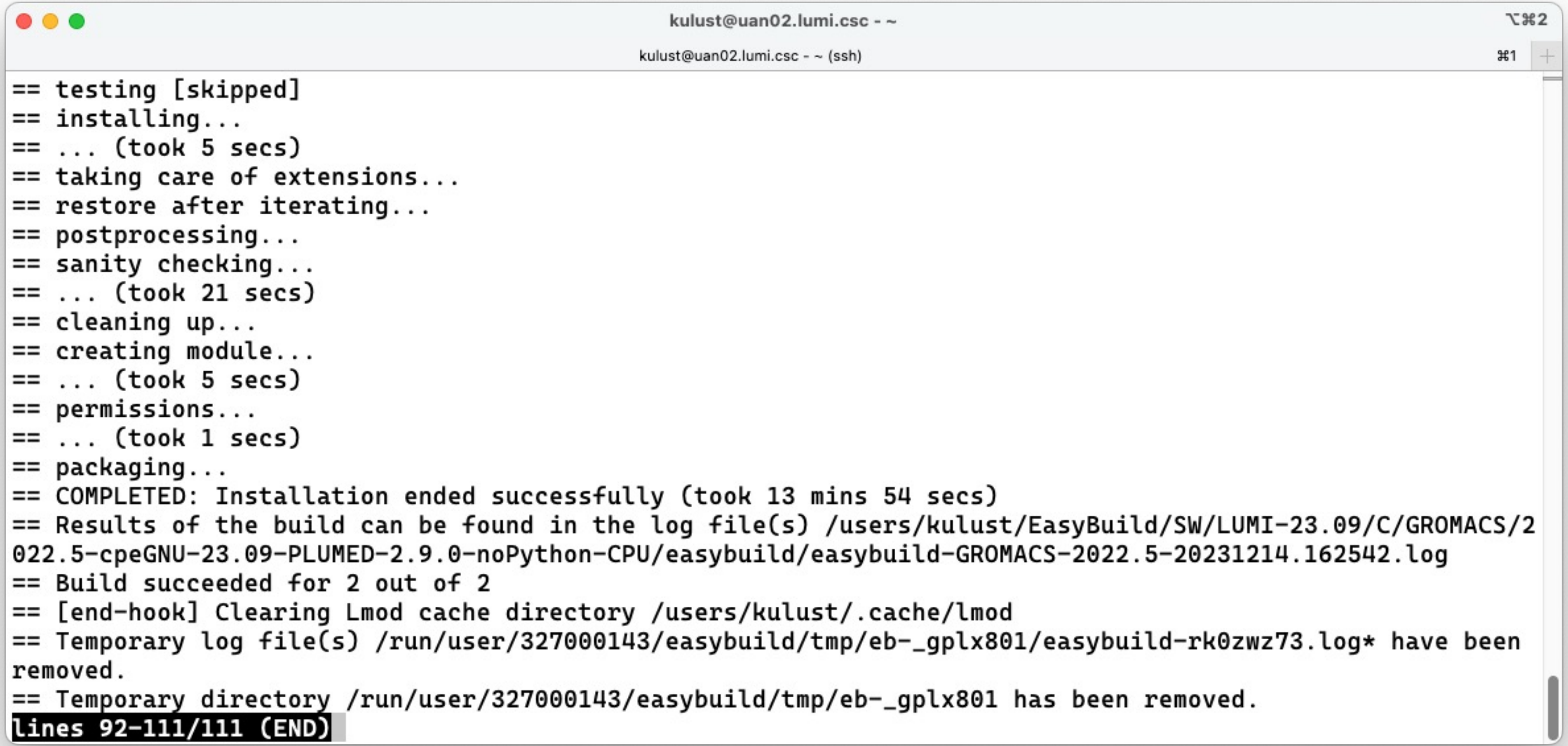

kulust@uan02.lumi.csc - ~

kulust@uan02.lumi.csc - ~ (ssh)

```
== patching...
== preparing...
== ... (took 7 secs)
== configuring...
== ... (took 1 min 23 secs)
== building...
== ... (took 1 min 30 secs)
== testing [skipped]
== installing...
== ... (took 5 secs)
== taking care of extensions...
== restore after iterating...
== postprocessing...
== sanity checking...
== ... (took 21 secs)
== cleaning up...
== creating module...
== ... (took 5 secs)
== permissions...
== ... (took 1 secs)
== packaging...
== COMPLETED: Installation ended successfully (took 13 mins 54 secs)
lines 85-106
```

eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -r (6)

LUMI



kulust@uan02.lumi.csc - ~

kulust@uan02.lumi.csc - ~ (ssh)

```
== testing [skipped]
== installing...
== ... (took 5 secs)
== taking care of extensions...
== restore after iterating...
== postprocessing...
== sanity checking...
== ... (took 21 secs)
== cleaning up...
== creating module...
== ... (took 5 secs)
== permissions...
== ... (took 1 secs)
== packaging...
== COMPLETED: Installation ended successfully (took 13 mins 54 secs)
== Results of the build can be found in the log file(s) /users/kulust/EasyBuild/SW/LUMI-23.09/C/GROMACS/2
022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU/easybuild/easybuild-GROMACS-2022.5-20231214.162542.log
== Build succeeded for 2 out of 2
== [end-hook] Clearing Lmod cache directory /users/kulust/.cache/lmod
== Temporary log file(s) /run/user/327000143/easybuild/tmp/eb-_gplx801/easybuild-rk0zwz73.log* have been
removed.
== Temporary directory /run/user/327000143/easybuild/tmp/eb-_gplx801 has been removed.
lines 92-111/111 (END)
```

Installing

Step 3: Install the software

- Let's, e.g., install GROMACS
  - Search if GROMACS build recipes are available:
    - Search the LUMI Software Library that lists all available software through EasyBuild.
    - Or on the command line:

      ```
      eb --search GROMACS
      eb –S GROMACS
      ```
  - Let's take GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb:

    ```
    eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -D
    eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -r
    ```
- Now the module should be available

  ```
  module avail GROMACS
  ```

Installing
Step 3: Install the software - Note

- Installing this way is 100% equivalent to an installation in the central software tree. The application is compiled in exactly the same way as we would do and served from the same file systems.
  - And you are in control of updates.
- Note: EasyBuild clears the Lmod user cache so in principle newly installed modules should show up without problems after installation.
  - We've seen rare cases where internal Lmod data structures were corrupt and logging out and in again was needed.
- To manually remove the cache: Remove `$HOME/.cache/lmod`

```
rm -rf $HOME/.cache/lmod
```

More advanced work

- You can also install some EasyBuild recipes that you got from support and are in the current directory (preferably one without subdirectories):
  `eb my_recipe.eb -r .`
  - Note the dot after the `-r` to tell EasyBuild to also look for dependencies in the current directory (and its subdirectories)
- In some cases you will have to download the sources by hand, e.g., for VASP, which is then at the same time a way for us to ensure that you have a license for VASP. E.g.,
  - `eb --search VASP`
  - Then from the directory with the VASP sources:
    `eb VASP-6.4.1-cpeGNU-22.12-build01.eb -r .`

More advanced work (2): Repositories

- It is possible to have your own clone of the LUMI-EasyBuild-contrib repo in your `$EBU_USER_PREFIX` subdirectory if you want the latest and greatest before it is in the centrally maintained repository

  -
    ```
    cd $EBU_USER_PREFIX
    git clone https://github.com/Lumi-supercomputer/LUMI-EasyBuild-contrib.git
    ```
- It is also possible to maintain your own repo
  - The directory should be `$EBU_USER_PREFIX/UserRepo` (but of course on GitHub the repository can have a different name)
  - Structure should be compatible with EasyBuild: easyconfig files go in `$EBU_USER_PREFIX/UserRepo/easybuild/easyconfigs`

More advanced work (3): Reproducibility

- EasyBuild will keep a copy of the sources in `$EBU_USER_PREFIX/sources`
- EasyBuild also keeps copies of all installed easyconfig files in two locations:
  - In `$EBU_USER_PREFIX/ebrepo_files`
    - And note that EasyBuild will use this version if you try to reinstall and did not delete this version first!
    - This ensures that the information that EasyBuild has about the installed application is compatible with what's in the module files
  - With the installed software (in `$EBU_USER_PREFIX/SW`) in a subdirectory called easybuild
    This is meant to have all information about how EasyBuild installed the application and to help in reproducing

EasyBuild tips&tricks

- **Updating version:** Often some trivial changes in the EasyConfig (`.eb`) file
  - Checksums may be annoying: Use `--ignore-checksums` with the `eb` command
- **Updating to a new toolchain:**
  - Be careful, it is more than changing one number
  - Versions of preinstalled dependencies should be changed and EasyConfig files of other dependencies also checked
- LUMI Software Library at lumi-supercomputer.github.io/LUMI-EasyBuild-docs
  - For most packages, pointers to the license
  - User documentation gives info about the use of the package, or restrictions
  - Technical documentation aimed at users who want more information about how we build the package

EasyBuild training for advanced users and developers

- EasyBuild web site: easybuild.io
- Generic EasyBuild training materials on tutorial.easybuild.io.
- Training for CSC and local support organisations: Most up-to-date version of the training materials on lumi-supercomputer.github.io/easybuild-tutorial.

Questions?

Why not EESSI?

- Cluster architecture
  - Hardware requirements for CVMFS
    - We have no local disks
    - Cache infrastructure: 1 per 500-1000 nodes?
  - Low-noise OS
    - with reduced daemons so software depending on those will break,
    - Lustre as the only network filesystem directly on the nodes
    - And DVS for the others (doesn't seem to play well with CVMFS even though the CVMFS manual claims it can be done)
  - Specific hardware
    - Slingshot 11 support for MPI and AI software?
    - ROCm?
  - SUSE OS
    - Abstraction layer cannot abstract everything away
    - E.g. CVMFS support on SUSE is mediocre

Why not EESSI? (2)

- Large software stacks have become hard to develop due to version conflicts
  - Not yet as bad as with Python where virtual environments are needed to be able to manage a stack…
  - But the EasyBuild community themselves does (without realising) an excellent job at demonstrating this principle
    - Can’t get a new toolchain out-of-the-door in time
    - Version conflicts for the basic compilers between what they want to offer in FOSS and requirements for CUDA and the Intel compiler
  - What with all the custom installations that people require these days? Building apon a stack that you don’t control is not ideal.

Why not EESSI? (3)

- Management problems 1: How can you even organise support?
  - LUST cannot properly support software they don’t install
  - But the EESSI community can impossibly test properly on LUMI hardware
  - And EESSI suffers from a combinatorial problem just as Spack
    - Different processor architectures and sub-architectures
    - Different GPU architectures (and of course different driver bugs for different CPUs)
    - Different interconnect architectures
    - Daemons that may be lacking on compute nodes of some clusters
- Management problem 2: Needs access to a system directory that is not managed by the user software team

Why not EESSI? (4)

- From a LUMI point of view EESSI gives all the disadvantages of containers on LUMI without any of the advantages
- Basically a cluster needs to be built to run EESSI and LUMI is not
  - Problem overlooked by the EESSI community because most participating have very similar clusters
- If I need to invest in a different way of managing software, it would be one based on proper environments installed in containers that function as the isolation a virtual environment needs
  - So basically Spack on top of a tool that can build containers without too many privileges (and preferably none)

LUMI