L U M I

LUMI Software Stacks

Kurt Lust
LUMI User Support Team (LUST)
University of Antwerp

Software stack design considerations   LUMI

- 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 11 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
- Need for customised setups
    - 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
- Conda, (large) Python installations need to go in containers
  - We offer lumi-container-wrapper and cotainr to do that

Organisation: Software stacks

- **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 toolchains as-is for LUMI-C

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

- **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
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

L U M I

- The LUMI software stack uses two levels of modules
  - LUMI/22.08, LUMI/22.12, LUMI/23.03: 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 trigger a reload of the LUMI module it will be that of the compute node!

Installing software on HPC systems

L U M I

- Software on an HPC system is rarely installed from RPM
  - Generic RPMs 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 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 install
  - 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
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-2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU.eb

Additional information

Toolchain name and version (missing for SYSTEM)

Version of the package

Name of the package

Module: GROMACS/2021.4-cpeCray-22.08-PLUMED-2.8.0-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
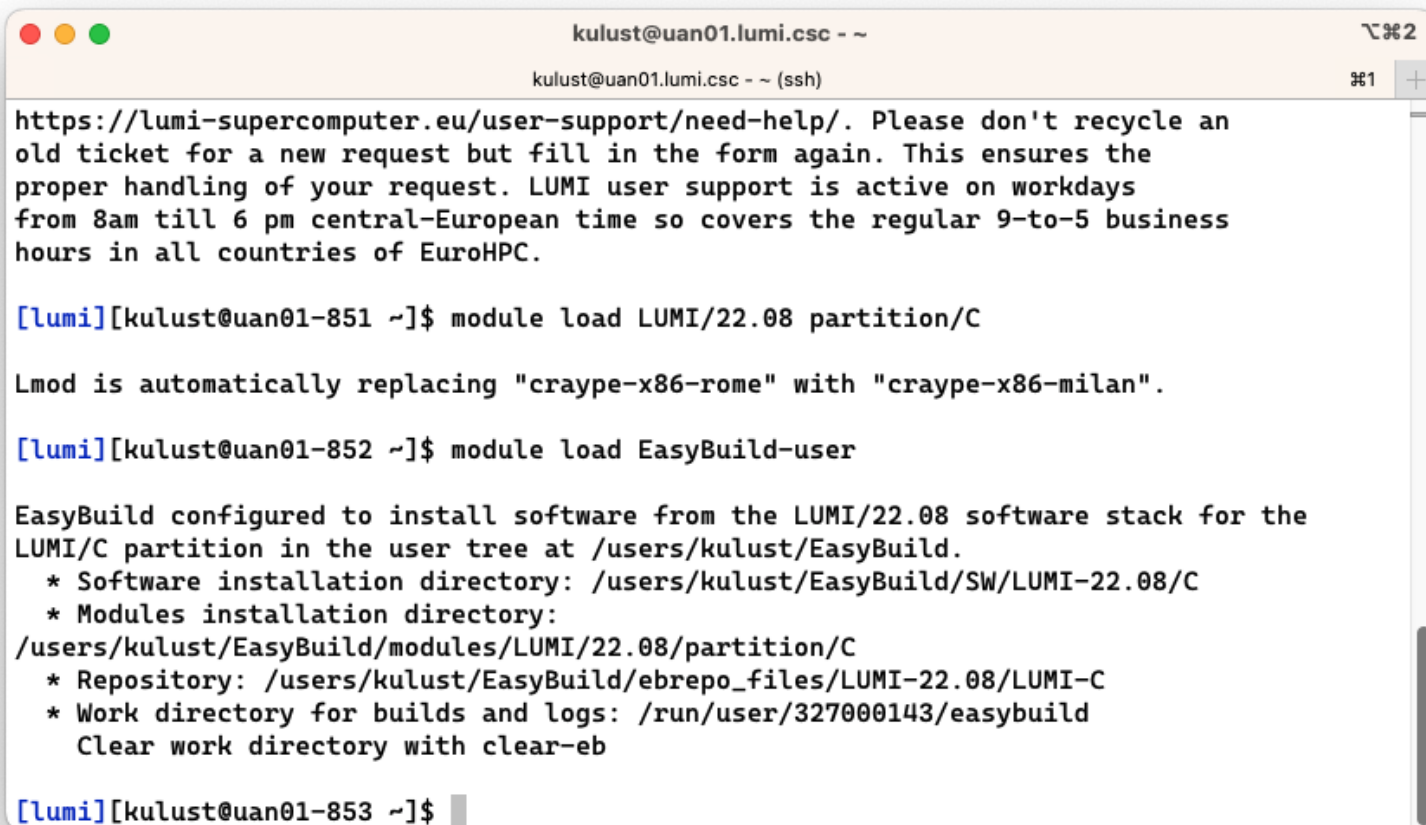
Installing

- Load the modules for the LUMI software stack and partition that you want to use. E.g.,
  `module load LUMI/22.08 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/22.08 partition/C
module load EasyBuild-user
```


LUMI

```
                        kulust@uan01.lumi.csc - ~                      ⌥⌘2
                      kulust@uan01.lumi.csc - ~ (ssh)                   ⌘1   +

https://lumi-supercomputer.eu/user-support/need-help/. Please don't recycle an
old ticket for a new request but fill in the form again. This ensures the
proper handling of your request. LUMI user support is active on workdays
from 8am till 6 pm central-European time so covers the regular 9-to-5 business
hours in all countries of EuroHPC.

[lumi][kulust@uan01-851 ~]$ module load LUMI/22.08 partition/C

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

[lumi][kulust@uan01-852 ~]$ module load EasyBuild-user

EasyBuild configured to install software from the LUMI/22.08 software stack for the
LUMI/C partition in the user tree at /users/kulust/EasyBuild.
  * Software installation directory: /users/kulust/EasyBuild/SW/LUMI-22.08/C
  * Modules installation directory:
/users/kulust/EasyBuild/modules/LUMI/22.08/partition/C
  * Repository: /users/kulust/EasyBuild/ebrepo_files/LUMI-22.08/LUMI-C
  * Work directory for builds and logs: /run/user/327000143/easybuild
    Clear work directory with clear-eb

[lumi][kulust@uan01-853 ~]$
```

Installing
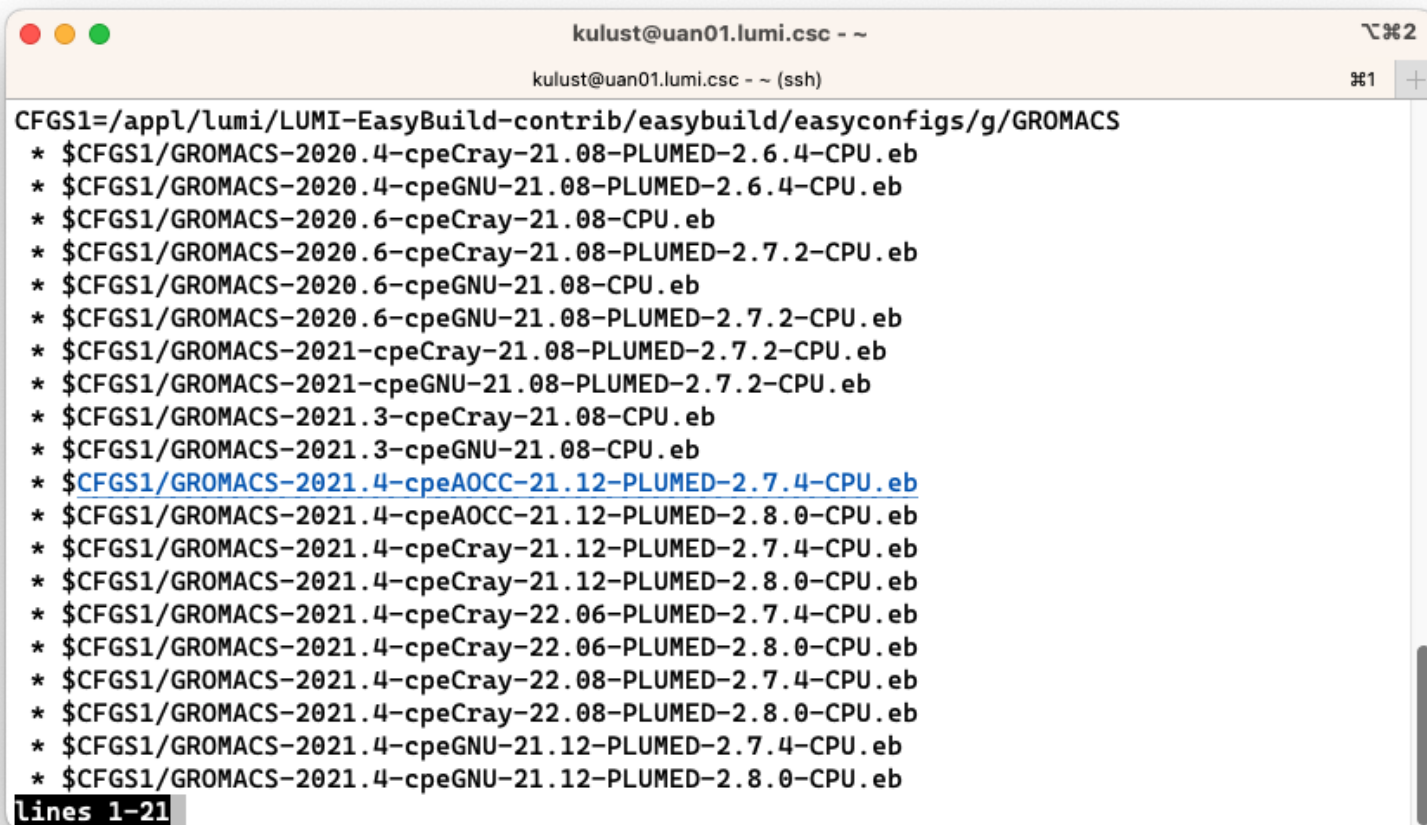
Step 3: Install the software

- 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-2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU.eb:
    ```
    eb GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU.eb -D
    eb GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU.eb -r
    ```

- Now the module should be available
  ```
  module avail GROMACS
  ```

`eb --search GROMACS | less`

```
eb -S GROMACS | less
```


LUMI

```
                        kulust@uan01.lumi.csc - ~                        ⌥⌘2

                      kulust@uan01.lumi.csc - ~ (ssh)                     ⌘1   +

CFGS1=/appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS
 * $CFGS1/GROMACS-2020.4-cpeCray-21.08-PLUMED-2.6.4-CPU.eb
 * $CFGS1/GROMACS-2020.4-cpeGNU-21.08-PLUMED-2.6.4-CPU.eb
 * $CFGS1/GROMACS-2020.6-cpeCray-21.08-CPU.eb
 * $CFGS1/GROMACS-2020.6-cpeCray-21.08-PLUMED-2.7.2-CPU.eb
 * $CFGS1/GROMACS-2020.6-cpeGNU-21.08-CPU.eb
 * $CFGS1/GROMACS-2020.6-cpeGNU-21.08-PLUMED-2.7.2-CPU.eb
 * $CFGS1/GROMACS-2021-cpeCray-21.08-PLUMED-2.7.2-CPU.eb
 * $CFGS1/GROMACS-2021-cpeGNU-21.08-PLUMED-2.7.2-CPU.eb
 * $CFGS1/GROMACS-2021.3-cpeCray-21.08-CPU.eb
 * $CFGS1/GROMACS-2021.3-cpeGNU-21.08-CPU.eb
 * $CFGS1/GROMACS-2021.4-cpeAOCC-21.12-PLUMED-2.7.4-CPU.eb
 * $CFGS1/GROMACS-2021.4-cpeAOCC-21.12-PLUMED-2.8.0-CPU.eb
 * $CFGS1/GROMACS-2021.4-cpeCray-21.12-PLUMED-2.7.4-CPU.eb
 * $CFGS1/GROMACS-2021.4-cpeCray-21.12-PLUMED-2.8.0-CPU.eb
 * $CFGS1/GROMACS-2021.4-cpeCray-22.06-PLUMED-2.7.4-CPU.eb
 * $CFGS1/GROMACS-2021.4-cpeCray-22.06-PLUMED-2.8.0-CPU.eb
 * $CFGS1/GROMACS-2021.4-cpeCray-22.08-PLUMED-2.7.4-CPU.eb
 * $CFGS1/GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU.eb
 * $CFGS1/GROMACS-2021.4-cpeGNU-21.12-PLUMED-2.7.4-CPU.eb
 * $CFGS1/GROMACS-2021.4-cpeGNU-21.12-PLUMED-2.8.0-CPU.eb
lines 1-21
```
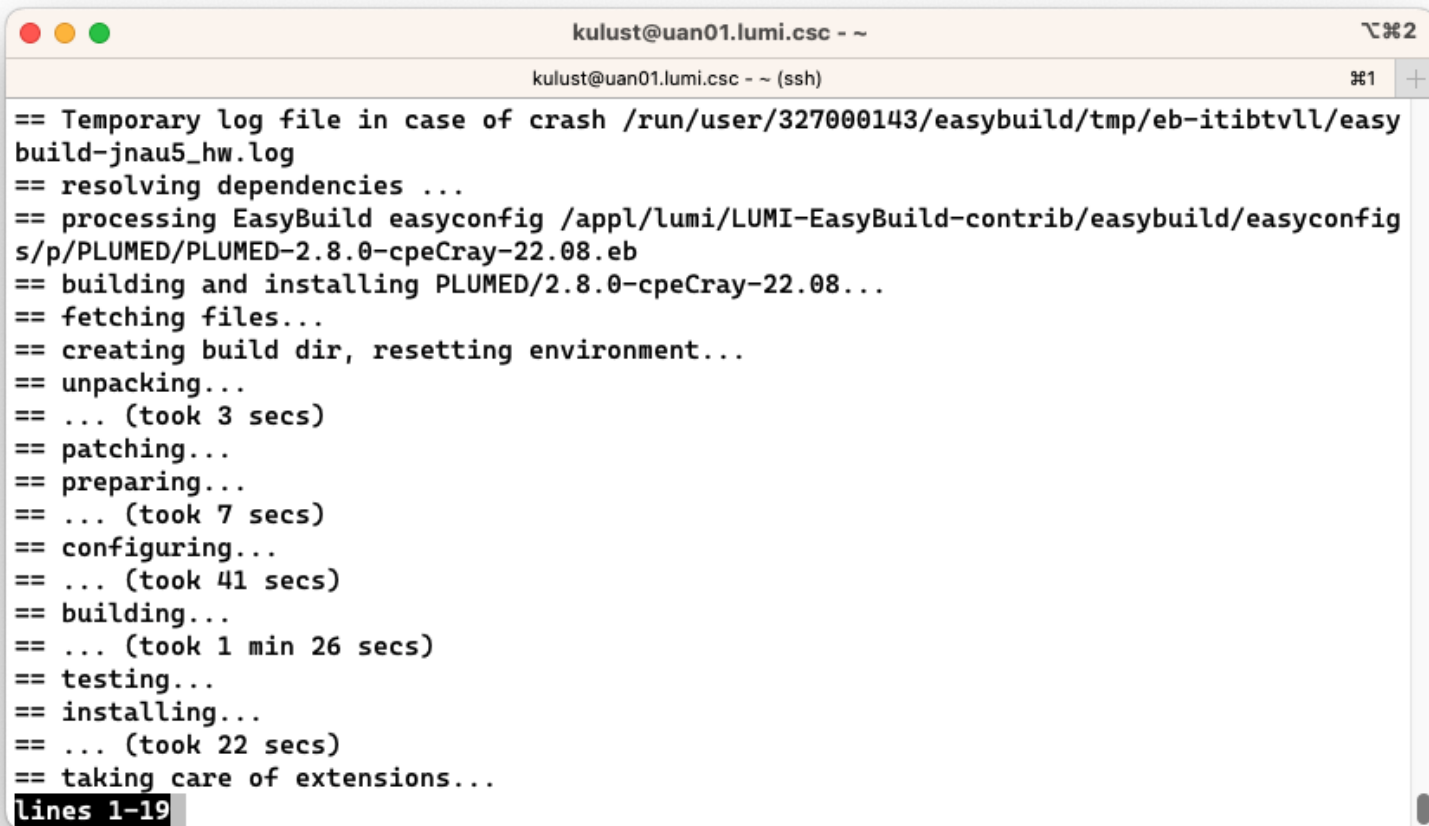
```
eb GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU.eb -D
```


LUMI

kulust@uan01.lumi.csc - ~

kulust@uan01.lumi.csc - ~ (ssh)

```
== Temporary log file in case of crash /run/user/327000143/easybuild/tmp/eb-1ckgk1_7/easy
build-tmiw_ajq.log
Dry run: printing build status of easyconfigs and dependencies
CFGS=/appl/lumi
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-common/buildtools/buildtools-22.08-minimal
.eb (module: buildtools/22.08-minimal)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/cpeCray/cpeCray-22.08.eb (module: cpeCra
y/22.08)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-common/syslibs/syslibs-22.08-static.eb (mo
dule: syslibs/22.08-static)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-common/buildtools/buildtools-22.08.eb (mod
ule: buildtools/22.08)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/zlib/zlib-1.2.12-cpeCray-22.08.eb (modul
e: zlib/1.2.12-cpeCray-22.08)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/bzip2/bzip2-1.0.8-cpeCray-22.08.eb (modu
le: bzip2/1.0.8-cpeCray-22.08)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/GSL/GSL-2.7.1-cpeCray-22.08-OpenMP.eb (m
odule: GSL/2.7.1-cpeCray-22.08-OpenMP)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/ICU/ICU-71.1-cpeCray-22.08.eb (module: I
CU/71.1-cpeCray-22.08)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/gzip/gzip-1.12-cpeCray-22.08.eb (module:
lines 1-12
```
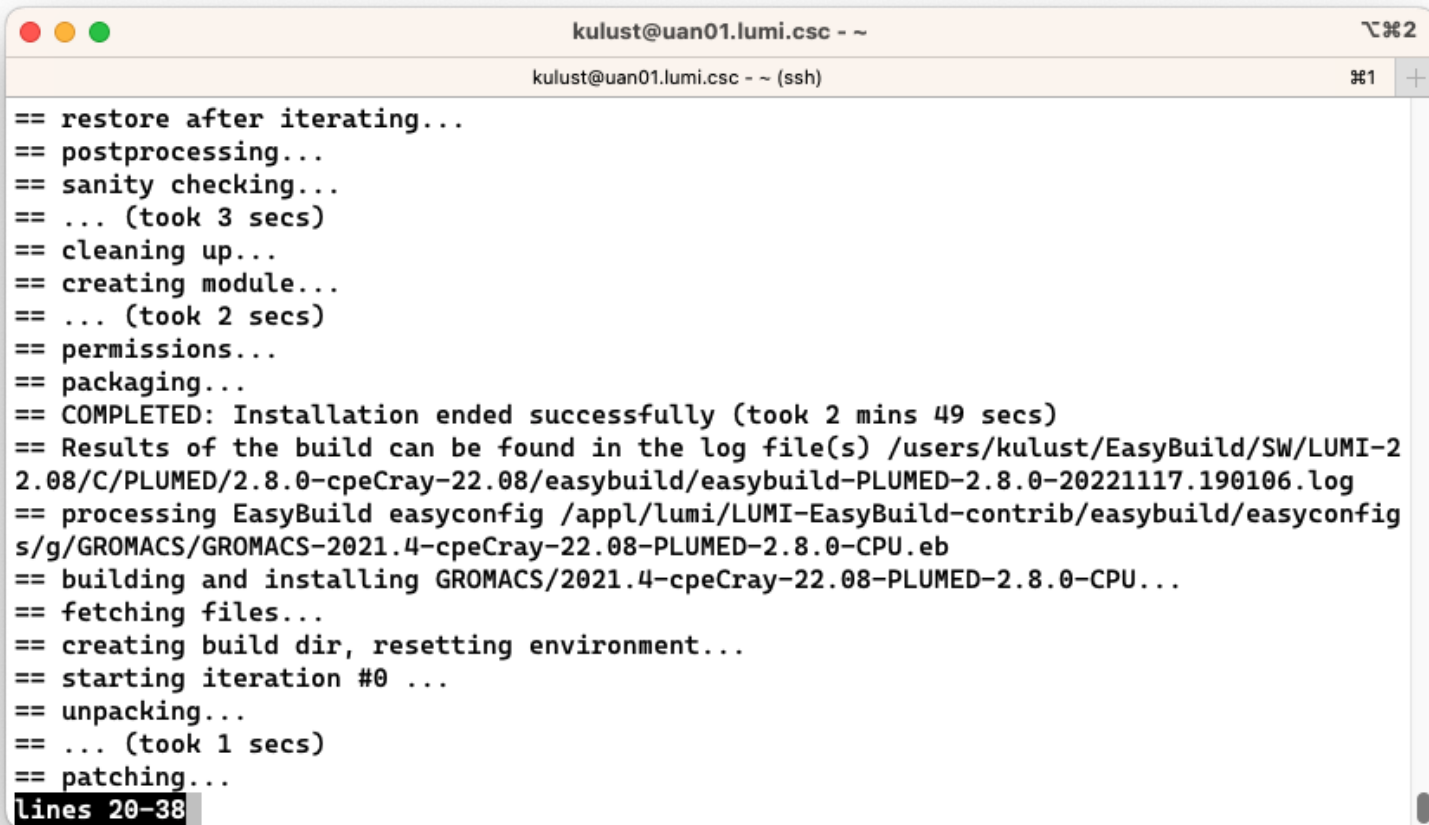
eb GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU.eb -D (2)


```
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/gzip/gzip-1.12-cpeCray-22.08.eb (module:
gzip/1.12-cpeCray-22.08)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/lz4/lz4-1.9.3-cpeCray-22.08.eb (module:
lz4/1.9.3-cpeCray-22.08)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/ncurses/ncurses-6.2-cpeCray-22.08.eb (mo
dule: ncurses/6.2-cpeCray-22.08)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/gettext/gettext-0.21-cpeCray-22.08-minim
al.eb (module: gettext/0.21-cpeCray-22.08-minimal)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/XZ/XZ-5.2.5-cpeCray-22.08.eb (module: XZ
/5.2.5-cpeCray-22.08)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/zstd/zstd-1.5.2-cpeCray-22.08.eb (module
: zstd/1.5.2-cpeCray-22.08)
 * [x] $CFGS/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/Boost/Boost-1.79.0-cpeCray-22.08.eb (mod
ule: Boost/1.79.0-cpeCray-22.08)
 * [ ] $CFGS/LUMI-EasyBuild-contrib/easybuild/easyconfigs/p/PLUMED/PLUMED-2.8.0-cpeCray-2
2.08.eb (module: PLUMED/2.8.0-cpeCray-22.08)
 * [ ] $CFGS/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeCra
y-22.08-PLUMED-2.8.0-CPU.eb (module: GROMACS/2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU)
== Temporary log file(s) /run/user/327000143/easybuild/tmp/eb-1ckgk1_7/easybuild-tmiw_ajq
.log* have been removed.
== Temporary directory /run/user/327000143/easybuild/tmp/eb-1ckgk1_7 has been removed.
lines 12-22/22 (END)
```

```
eb GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU.eb –r
```
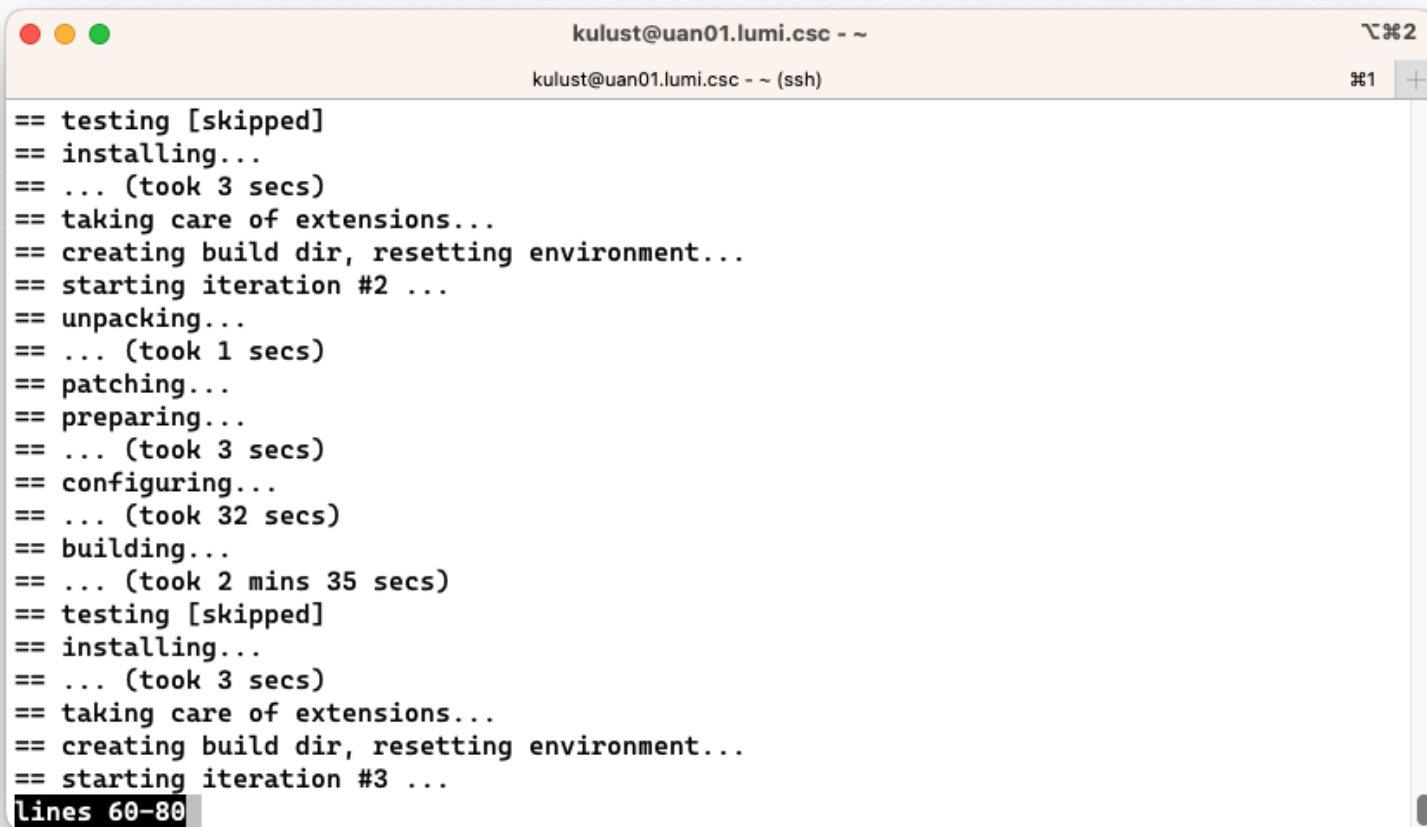


```
== Temporary log file in case of crash /run/user/327000143/easybuild/tmp/eb-itibtvll/easy
build-jnau5_hw.log
== resolving dependencies ...
== processing EasyBuild easyconfig /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfig
s/p/PLUMED/PLUMED-2.8.0-cpeCray-22.08.eb
== building and installing PLUMED/2.8.0-cpeCray-22.08...
== fetching files...
== creating build dir, resetting environment...
== unpacking...
== ... (took 3 secs)
== patching...
== preparing...
== ... (took 7 secs)
== configuring...
== ... (took 41 secs)
== building...
== ... (took 1 min 26 secs)
== testing...
== installing...
== ... (took 22 secs)
== taking care of extensions...
lines 1-19
```

L U M I

```
== restore after iterating...
== postprocessing...
== sanity checking...
== ... (took 3 secs)
== cleaning up...
== creating module...
== ... (took 2 secs)
== permissions...
== packaging...
== COMPLETED: Installation ended successfully (took 2 mins 49 secs)
== Results of the build can be found in the log file(s) /users/kulust/EasyBuild/SW/LUMI-2
2.08/C/PLUMED/2.8.0-cpeCray-22.08/easybuild/easybuild-PLUMED-2.8.0-20221117.190106.log
== processing EasyBuild easyconfig /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfig
s/g/GROMACS/GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU.eb
== building and installing GROMACS/2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU...
== fetching files...
== creating build dir, resetting environment...
== starting iteration #0 ...
== unpacking...
== ... (took 1 secs)
== patching...
lines 20-38
```

eb GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU.eb -r (3)

`eb GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU.eb -r (4)`
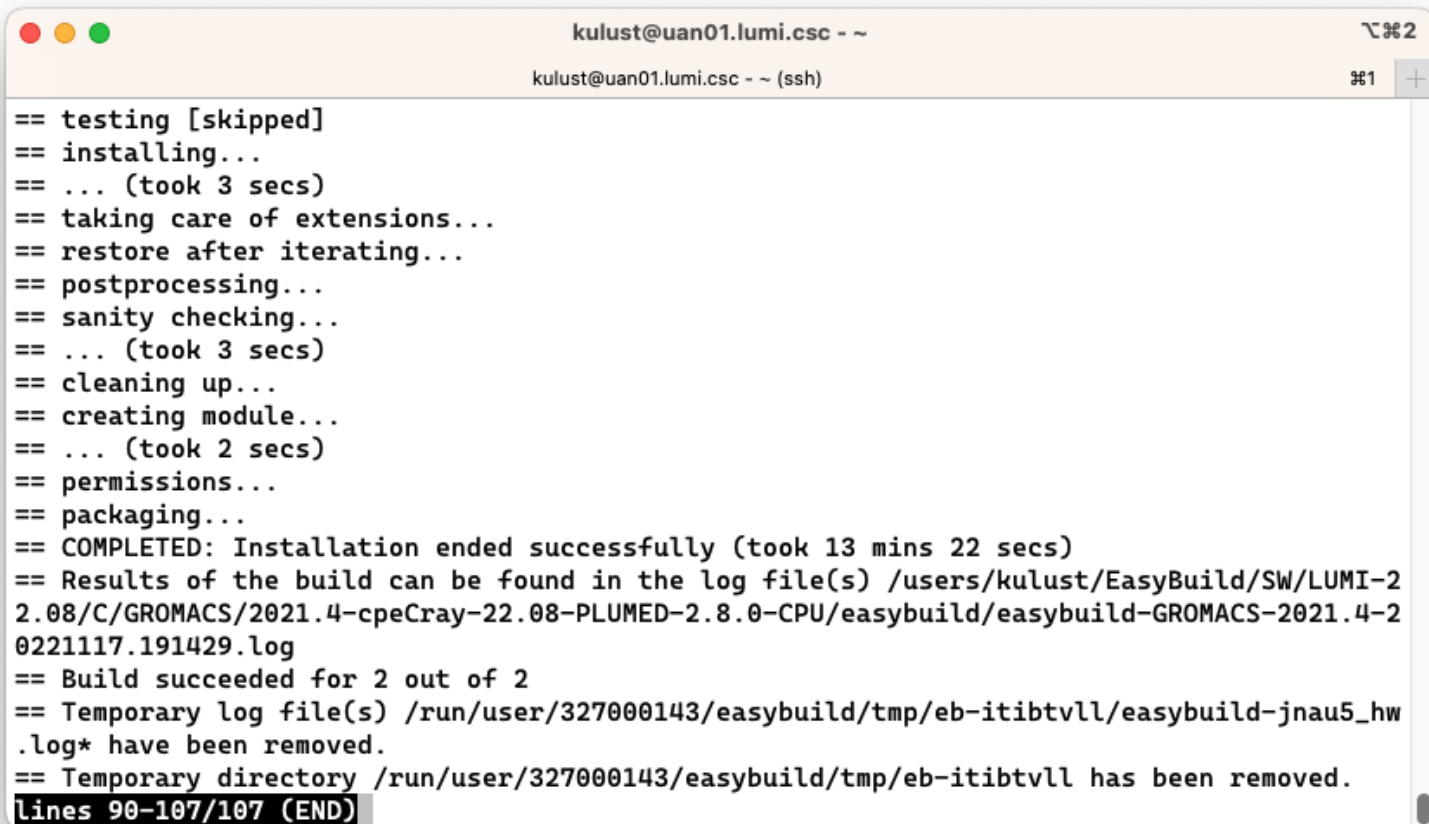
LUMI

```
== unpacking...
== ... (took 1 secs)
== patching...
== preparing...
== ... (took 3 secs)
== configuring...
== ... (took 33 secs)
== building...
== ... (took 2 mins 33 secs)
== testing [skipped]
== installing...
== ... (took 3 secs)
== taking care of extensions...
== restore after iterating...
== postprocessing...
== sanity checking...
== ... (took 3 secs)
== cleaning up...
== creating module...
== ... (took 2 secs)
== permissions...
lines 81-101
```

`eb GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU.eb -r (6)`


```
== testing [skipped]
== installing...
== ... (took 3 secs)
== taking care of extensions...
== restore after iterating...
== postprocessing...
== sanity checking...
== ... (took 3 secs)
== cleaning up...
== creating module...
== ... (took 2 secs)
== permissions...
== packaging...
== COMPLETED: Installation ended successfully (took 13 mins 22 secs)
== Results of the build can be found in the log file(s) /users/kulust/EasyBuild/SW/LUMI-2
2.08/C/GROMACS/2021.4-cpeCray-22.08-PLUMED-2.8.0-CPU/easybuild/easybuild-GROMACS-2021.4-2
0221117.191429.log
== Build succeeded for 2 out of 2
== Temporary log file(s) /run/user/327000143/easybuild/tmp/eb-itibtvll/easybuild-jnau5_hw
.log* have been removed.
== Temporary directory /run/user/327000143/easybuild/tmp/eb-itibtvll has been removed.
lines 90-107/107 (END)
```

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.

- 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 where corrupt and logging out and in again was needed.

- To manually remove the cache: Remove `$HOME/.lmod.d/.cache`
  `rm -rf $HOME/.lmod.d/.cache`

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-6.3.2-cpeGNU-22.08.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

L U M I

- 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

LUMI

- EasyBuild web site: easybuild.io

- Generic EasyBuild training materials on easybuilders.github.io/easybuild-tutorial.

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