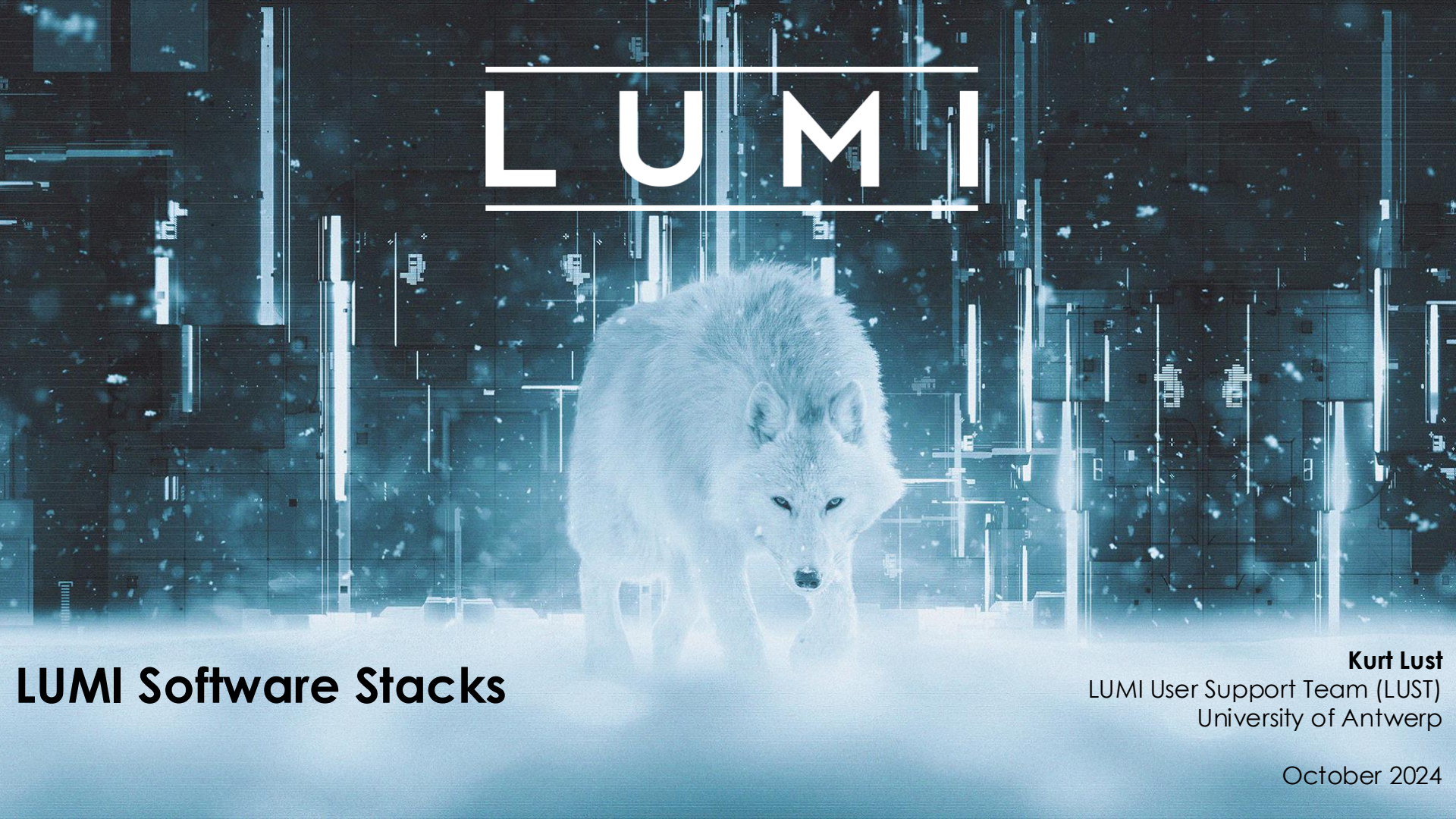

LUMI Software Stacks

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

October 2024

What this talk is about…

- Software stacks on LUMI
- Some remarks about Lmod
- Creating your customised environment with EasyBuild
- Containers

Design considerations

LUMI

- Very leading edge and inhomogeneous machine (new interconnect, new GPU architecture with a still maturing software ecosystem, 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

LUMI

- 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 which is also supported by HPE Cray
  - 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

LUMI

- 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
  - Tools: lumi-container-wrapper , cotainr and SingularityCE unprivileged proot build

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
- Some local organisations also provide software pre-installed on LUMI
  - Look for **`Local-*` modules**
- 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

LUMI

- 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 (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

LUMI

- The LUMI software stack uses two levels of modules
  - LUMI/24.03, LUMI/23.12, LUMI/23.09, LUMI/23.03, LUMI/22.08: 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!

Exploring modules with Lmod

- Contrary to some other module systems, not all modules are immediately available for loading
  - Installed modules: All modules on the system that can be loaded one way or another
  - Available modules: Can be loaded without first loading another module
- Examples in the HPE Cray PE:
  - `cray-mpich` requires a compiler module and network target module first
  - Many of the performance monitoring tools require `perftools-base` first
  - `cray-fftw` only becomes available when a processor target module is loaded
- Tools
  - `module avail` searches in the available modules
  - `module spider` and `module keyword` search in the installed modules
    - But with some restrictions on LUMI

module spider

- `module spider` : Long list of all installed software with short description
  - Will also look into modules for "extensions" and show those also, marked with an "E"
  - By default only in the main software stacks on LUMI
- `module spider gnuplot` : Shows all versions of gnuplot on the system
  `module spider CMake`
- `module spider gnuplot/5.4.10-cpeGNU-24.03` : Shows help information for the specific module, including what should be done to make the module available
  - But this does not completely work with the Cray PE modules
- `module spider CMake/3.29.3` : Will tell you which module contains CMake and how to load it

module spider (command) (1)

LUMI

kulust@uan02.lumi.csc - ~

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

```
  *|   bug fixing in Spack but do offer a configuration compatible  |
** |   with the Cray PE.                                            |
** `--*****-------------------***-----------------------*****-------'

Did you know?
*************
The behaviour of many modules of the Cray Programming Environment depends
on the target modules that are loaded. All compilers on LUMI except for the
gcc implementation provided by the operating system support proper
optimizations for the zen2 and zen3 processors. The zen3 architectures is
the core architecture of the AMD Epyc Milan CPUs in the LUMI-C compute nodes
and the AMD EPYC Trento CPUs in the LUMI-G compute nodes. (Except for
gcc 10.3.0 that still lacks full zen3 support.) You may get get a little
extra performance by optimising for zen3 specifically. The CrayEnv environment
will automatically load the best target CPU, network and accelerator target
modules for each node. You can always restore them by reloading the
CrayEnv module (and there is no need to unload first).

In a job script, the environment is copied from the login nodes, so you'll
still have the Rome target modules. Here also reloading the CrayEnv module
will set the ones for the compute node you're running on.

[lumi][kulust@uan02-1000 ~]$ module spider
```

module spider (command) (2)

LUMI

kurtlust@uan06.lumi.csc - ~

kurtlust@uan06.lumi.csc - ~ (ssh)

```
-----------------------------------------------------------------------------------------------------
The following is a list of the modules and extensions currently available:
-----------------------------------------------------------------------------------------------------
  ARMForge: ARMForge/22.0.1
    Arm Forge debugging and profiling tools

  Autoconf: Autoconf/2.71 (E), Autoconf/2.72 (E)

  Autoconf-archive: Autoconf-archive/2022.02.11 (E), ...

  Automake: Automake/1.16.5 (E)

  Bison: Bison/3.8.2 (E)

  Blosc: Blosc/1.21.1-cpeAMD-22.08, Blosc/1.21.1-cpeAOCC-22.08, Blosc/1.21.1-cpeCray-22.08, ...
    Blosc is an extremely fast, multi-threaded, meta-compressor library

  Boost: Boost/1.79.0-cpeAMD-22.08, Boost/1.79.0-cpeAOCC-22.08, Boost/1.79.0-cpeCray-22.08, ...
    Boost provides free peer-reviewed portable C++ source libraries.

  Brotli: Brotli/1.0.9-cpeAMD-22.08, Brotli/1.0.9-cpeAMD-22.12, Brotli/1.0.9-cpeAMD-23.09, ...
lines 1-22
```

module spider (command) (3)

LUMI

kurtlust@uan06.lumi.csc - ~

kurtlust@uan06.lumi.csc - ~ (ssh)

```
  zlib: zlib/1.2.12-cpeAMD-22.08, zlib/1.2.12-cpeAMD-22.12, zlib/1.2.12-cpeAOCC-22.08, ...
    Free lossless data-compression library, not covered by any patents.

  zstd: zstd/1.5.2-cpeAMD-22.08, zstd/1.5.2-cpeAMD-22.12, zstd/1.5.2-cpeAOCC-22.08, ...

Names marked by a trailing (E) are extensions provided by another module.

-----------------------------------------------------------------------------------------------------

To learn more about a package execute:

   $ module spider Foo

where "Foo" is the name of a module.

To find detailed information about a particular package you
must specify the version if there is more than one version:

   $ module spider Foo/11.1

-----------------------------------------------------------------------------------------------------
[lumi][kurtlust@uan06-1002 ~]$
```

module spider gnuplot

LUMI

kurtlust@uan06.lumi.csc - ~

kurtlust@uan06.lumi.csc - ~ (ssh)

```
-----------------------------------------------------------------------------------------------------
  gnuplot:
-----------------------------------------------------------------------------------------------------
    Description:
      Gnuplot is a portable command-line driven graphing utility

     Versions:
        gnuplot/5.4.3-cpeAMD-22.08
        gnuplot/5.4.3-cpeAOCC-22.08
        gnuplot/5.4.3-cpeCray-22.08
        gnuplot/5.4.3-cpeGNU-22.08
        gnuplot/5.4.6-cpeAMD-22.12
        gnuplot/5.4.6-cpeAOCC-22.12
        gnuplot/5.4.6-cpeCray-22.12
        gnuplot/5.4.6-cpeCray-23.03
        gnuplot/5.4.6-cpeGNU-22.12
        gnuplot/5.4.8-cpeAMD-23.12
        gnuplot/5.4.8-cpeAOCC-23.09
        gnuplot/5.4.8-cpeAOCC-23.12
        gnuplot/5.4.8-cpeCray-23.12
        gnuplot/5.4.8-cpeGNU-23.09
lines 1-22
```

module spider gnuplot (2)

LUMI

kurtlust@uan06.lumi.csc - ~

kurtlust@uan06.lumi.csc - ~ (ssh)

```
        gnuplot/5.4.6-cpeCray-22.12
        gnuplot/5.4.6-cpeCray-23.03
        gnuplot/5.4.6-cpeGNU-22.12
        gnuplot/5.4.8-cpeAMD-23.12
        gnuplot/5.4.8-cpeAOCC-23.09
        gnuplot/5.4.8-cpeAOCC-23.12
        gnuplot/5.4.8-cpeCray-23.12
        gnuplot/5.4.8-cpeGNU-23.09
        gnuplot/5.4.8-cpeGNU-23.12
        gnuplot/5.4.10-cpeAMD-24.03
        gnuplot/5.4.10-cpeAOCC-24.03
        gnuplot/5.4.10-cpeCray-24.03
        gnuplot/5.4.10-cpeGNU-24.03

-------------------------------------------------------------------------------------------------
  For detailed information about a specific "gnuplot" package (including how to load the modules) use the
module's full name.
  Note that names that have a trailing (E) are extensions provided by other modules.
  For example:

     $ module spider gnuplot/5.4.10-cpeGNU-24.03
-------------------------------------------------------------------------------------------------
[lumi][kurtlust@uan06-1006 ~]$
```

module spider CMake

LUMI

kulust@uan02.lumi.csc - ~

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

```
-----------------------------------------------------------------------------------------------------
  CMake:
-----------------------------------------------------------------------------------------------------
     Versions:
        CMake/3.24.0 (E)
        CMake/3.25.2 (E)
        CMake/3.27.7 (E)
        CMake/3.29.3 (E)
     Other possible modules matches:
        cmake  cmake-3.26.3-gcc-7.5.0-3og5f6c  cmake-3.26.3-gcc-7.5.0-cl73x27  rocm-cmake

Names marked by a trailing (E) are extensions provided by another module.


-----------------------------------------------------------------------------------------------------
  To find other possible module matches execute:

      $ module -r spider '.*CMake.*'

-----------------------------------------------------------------------------------------------------
  For detailed information about a specific "CMake" package (including how to load the modules) use the mo
lines 1-22
```

module spider gnuplot/5.4.10-cpeGNU-24.03

LUMI

kulust@uan02.lumi.csc - ~

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

```
-----------------------------------------------------------------------------------------------------
  gnuplot: gnuplot/5.4.10-cpeGNU-24.03
-----------------------------------------------------------------------------------------------------
    Description:
      Gnuplot is a portable command-line driven graphing utility


    You will need to load all module(s) on any one of the lines below before the "gnuplot/5.4.10-cpeGNU-24
.03" module is available to load.

      LUMI/24.03  partition/C
      LUMI/24.03  partition/G
      LUMI/24.03  partition/L

    Help:
      Description
      ===========
      Gnuplot is a portable command-line driven graphing utility available for many
      platforms. The source code is copyrighted but freely distributed (i.e., you
      don't have to pay for it). It was originally created to allow scientists and
      students to visualize mathematical functions and data interactively, but has
lines 1-21
```

module spider gnuplot/5.4.10-cpeGNU-24.03 (2)

LUMI

kulust@uan02.lumi.csc - ~

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

```
    ===========
    Gnuplot is a portable command-line driven graphing utility available for many
    platforms. The source code is copyrighted but freely distributed (i.e., you
    don't have to pay for it). It was originally created to allow scientists and
    students to visualize mathematical functions and data interactively, but has
    grown to support many non-interactive uses such as web scripting. It is also
    used as a plotting engine by third-party applications like Octave. Gnuplot has
    been supported and under active development since 1986.

    This version of GNUplot does not use Qt5 for its GUI, so the GUI is rather
    primitive.


    More information
    ================
     - Homepage: http://gnuplot.sourceforge.net/
     - Documentation:
        - Web-based documentation: http://gnuplot.sourceforge.net/documentation.html
        - Manual page for gnuplot
     - Site contact: LUMI User Support @ https://lumi-supercomputer.eu/user-support/need-help/


[lumi][kulust@uan02-1008 ~]$
```

module spider CMake/3.29.3

LUMI

kulust@uan02.lumi.csc - ~

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

```
-----------------------------------------------------------------------------------------------------
  CMake: CMake/3.29.3 (E)
-----------------------------------------------------------------------------------------------------
    This extension is provided by the following modules. To access the extension you must load one of the
following modules. Note that any module names in parentheses show the module location in the software hier
archy.


       buildtools/24.03 (LUMI/24.03 partition/L)
       buildtools/24.03 (LUMI/24.03 partition/G)
       buildtools/24.03 (LUMI/24.03 partition/C)
       buildtools/24.03 (CrayEnv)
       buildtools/23.12 (LUMI/23.12 partition/L)
       buildtools/23.12 (LUMI/23.12 partition/G)
       buildtools/23.12 (LUMI/23.12 partition/C)
       buildtools/23.12 (CrayEnv)


Names marked by a trailing (E) are extensions provided by another module.


lines 1-20
```

module keyword

- It searches in the module short description and help for the keyword.
  - E.g., try
    `module keyword https`
- We do try to put enough information in the modules to make this a suitable additional way to discover software that is already installed on the system

module keyword https

LUMI

kurtlust@uan06.lumi.csc - ~

kurtlust@uan06.lumi.csc - ~ (ssh)

```
-------------------------------------------------------------------------------------------------

The following modules match your search criteria: "https"
-------------------------------------------------------------------------------------------------

  cURL: cURL/7.83.1-cpeAMD-22.08, cURL/7.83.1-cpeAOCC-22.08, cURL/7.83.1-cpeCray-22.08, ...
    Command line tool and library for transferring data with URLs.

  wget: wget/1.21.3-cpeAMD-22.08, wget/1.21.3-cpeAMD-22.12, wget/1.21.3-cpeAMD-23.09, ...
    wget - GNU wget, a free software package for retrieving files using HTTP, HTTPS and FTP

-------------------------------------------------------------------------------------------------

To learn more about a package execute:

   $ module spider Foo

where "Foo" is the name of a module.

To find detailed information about a particular package you
must specify the version if there is more than one version:

lines 1-22
```

module keyword https (2)

LUMI

kurtlust@uan06.lumi.csc - ~

kurtlust@uan06.lumi.csc - ~ (ssh)

```
-------------------------------------------------------------------------------------------------

  cURL: cURL/7.83.1-cpeAMD-22.08, cURL/7.83.1-cpeAOCC-22.08, cURL/7.83.1-cpeCray-22.08, ...
    Command line tool and library for transferring data with URLs.

  wget: wget/1.21.3-cpeAMD-22.08, wget/1.21.3-cpeAMD-22.12, wget/1.21.3-cpeAMD-23.09, ...
    wget - GNU wget, a free software package for retrieving files using HTTP, HTTPS and FTP

-------------------------------------------------------------------------------------------------

To learn more about a package execute:

   $ module spider Foo

where "Foo" is the name of a module.

To find detailed information about a particular package you
must specify the version if there is more than one version:

   $ module spider Foo/11.1

-------------------------------------------------------------------------------------------------
[lumi][kurtlust@uan06-1007 ~]$
```

Sticky modules and module purge

- On some systems, you will be taught to avoid `module purge` (which unloads all modules)
- Sticky modules are modules that are not unloaded by `module purge`, but reloaded.
  - They can be force-unloaded with `module --force purge` and `module --force unload` but do so at your own risk
- Used on LUMI for the software stacks and modules that set the display style of the modules
  - But keep in mind that the modules are reloaded, so any change to modules that are loaded by these modules will be wiped out.

module av

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh)

```
--------------------------------- EasyBuild managed systemwide software ----------------------------------
   ARMForge/22.0.1        lumi-tools/23.03 (S)    lumi-tools/24.05    (S,L,D)    lumio-ext-tools/1.0.0
   Vampir/10.0.0          lumi-tools/23.04 (S)    lumi-vnc/20230110              lumio/1.0.0
   Vampir/10.2.1   (D)    lumi-tools/23.11 (S)    lumi-workspaces/0.1

----------------------------------------- HPE-Cray PE modules ------------------------------------------
   PrgEnv-amd/8.3.3
   PrgEnv-amd/8.4.0
   PrgEnv-amd/8.5.0               (D)
   PrgEnv-aocc/8.3.3
   PrgEnv-aocc/8.4.0
   PrgEnv-aocc/8.5.0              (D)
   PrgEnv-cray-amd/8.3.3
   PrgEnv-cray-amd/8.4.0
   PrgEnv-cray-amd/8.5.0          (D)
   PrgEnv-cray/8.3.3
   PrgEnv-cray/8.4.0
   PrgEnv-cray/8.5.0              (L,D)
   PrgEnv-gnu-amd/8.3.3
   PrgEnv-gnu-amd/8.4.0
   PrgEnv-gnu-amd/8.5.0           (D)
lines 1-22
```

module av (2)

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh)

```
   PrgEnv-gnu/8.3.3
   PrgEnv-gnu/8.4.0
   PrgEnv-gnu/8.5.0                (D)
   PrgEnv-nvhpc/8.5.0
   PrgEnv-nvidia/8.5.0
   amd-mixed/6.0.3
   amd/6.0.3                       (5.0.2:5.1.0:5.2.0:5.2.3:5.5.1:5.7.0:6.0.0)
   aocc-mixed/3.2.0                (D)
   aocc-mixed/4.1.0
   aocc/3.2.0                      (D)
   aocc/4.1.0
   atp/3.14.13
   atp/3.14.16
   atp/3.14.18
   atp/3.15.1
   atp/3.15.2
   atp/3.15.3                      (D)
   cce-mixed/14.0.2
   cce-mixed/15.0.0
   cce-mixed/15.0.1
   cce-mixed/16.0.1
   cce-mixed/17.0.0
lines 23-44
```

module av (3)

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh)

```
   cce-mixed/17.0.1                (D)
   cce/14.0.2
   cce/15.0.0
   cce/15.0.1
   cce/16.0.1
   cce/17.0.0
   cce/17.0.1                      (L,D)
   cpe/22.08
   cpe/22.12
   cpe/23.03
   cpe/23.09
   cpe/23.12
   cpe/24.03                       (D)
   cray-R/4.1.3.1
   cray-R/4.2.1.1
   cray-R/4.2.1.2
   cray-R/4.3.1
   cray-R/4.3.2                    (D)
   cray-ccdb/4.12.13
   cray-ccdb/5.0.1
   cray-ccdb/5.0.2
   cray-ccdb/5.0.3                 (D)
lines 45-66
```

module av (4)

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh)

```
   cray-cti/2.15.13
   cray-cti/2.15.14
   cray-cti/2.16.0
   cray-cti/2.17.1
   cray-cti/2.17.2
   cray-cti/2.18.1
   cray-cti/2.18.2
   cray-cti/2.18.3                (D)
   cray-dsmml/0.2.2
   cray-dsmml/0.3.0               (L,D)
   cray-dyninst/12.1.1
   cray-dyninst/12.3.0
   cray-dyninst/12.3.1            (D)
   cray-fftw/3.3.8.13
   cray-fftw/3.3.10.1
   cray-fftw/3.3.10.3
   cray-fftw/3.3.10.5
   cray-fftw/3.3.10.6
   cray-fftw/3.3.10.7             (D)
   cray-hdf5-parallel/1.12.2.9
   cray-hdf5-parallel/1.12.2.11   (D)
   cray-hdf5/1.12.2.9
lines 67-88
```

module av (5)

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh)

```
   cray-hdf5/1.12.2.11            (D)
   cray-libpals/1.2.0
   cray-libpals/1.2.5
   cray-libpals/1.2.11
   cray-libpals/1.2.12            (D)
   cray-libsci/21.08.1.2
   cray-libsci/22.08.1.1
   cray-libsci/22.12.1.1
   cray-libsci/23.02.1.1
   cray-libsci/23.09.1.1
   cray-libsci/23.12.5
   cray-libsci/24.03.0            (L,D)
   cray-libsci_acc/22.08.1.1
   cray-libsci_acc/22.12.1.1
   cray-libsci_acc/23.09.1.1
   cray-libsci_acc/23.12.0
   cray-libsci_acc/24.03.1        (D)
   cray-mpich-abi/8.1.28
   cray-mpich-abi/8.1.29          (D)
   cray-mpich/8.1.28
   cray-mpich/8.1.29              (L,D)
   cray-mpixlate/1.0.3
lines 89-110
```

module av (6)

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh)

```
   cray-mpixlate/1.0.4            (D)
   cray-mrnet/5.0.4
   cray-mrnet/5.1.1
   cray-mrnet/5.1.2               (D)
   cray-openshmemx/11.5.6
   cray-openshmemx/11.5.7
   cray-openshmemx/11.5.8
   cray-openshmemx/11.6.1
   cray-openshmemx/11.7.0
   cray-openshmemx/11.7.1         (D)
   cray-pals/1.2.0
   cray-pals/1.2.5
   cray-pals/1.2.11
   cray-pals/1.2.12               (D)
   cray-parallel-netcdf/1.12.3.9
   cray-parallel-netcdf/1.12.3.11 (D)
   cray-pmi-lib/6.0.17
   cray-pmi/6.0.17
   cray-pmi/6.1.3
   cray-pmi/6.1.8
   cray-pmi/6.1.10
   cray-pmi/6.1.12
lines 111-132
```

module av (7)

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh)

```
   cray-pmi/6.1.13
   cray-pmi/6.1.14                (D)
   cray-python/3.9.12.1
   cray-python/3.9.13.1
   cray-python/3.10.10
   cray-python/3.11.5
   cray-python/3.11.7             (D)
   cray-stat/4.11.12
   cray-stat/4.11.13
   cray-stat/4.12.1
   cray-stat/4.12.2               (D)
   craype/2.7.17
   craype/2.7.19
   craype/2.7.20
   craype/2.7.23
   craype/2.7.30
   craype/2.7.31.11               (L,D)
   craypkg-gen/1.3.25
   craypkg-gen/1.3.28
   craypkg-gen/1.3.30
   craypkg-gen/1.3.31
   craypkg-gen/1.3.32             (D)
lines 133-154
```

module av (8)

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh)

```
   gcc-mixed/11.2.0
   gcc-mixed/12.2.0                (D)
   gcc-native-mixed/12.3
   gcc-native-mixed/13.2           (D)
   gcc-native/12.3
   gcc-native/13.2                 (D)
   gcc/10.3.0
   gcc/11.2.0
   gcc/12.2.0                      (D)
   gdb4hpc/4.14.2
   gdb4hpc/4.14.6
   gdb4hpc/4.14.7
   gdb4hpc/4.15.1
   gdb4hpc/4.16.0
   gdb4hpc/4.16.1                  (D)
   iobuf/2.0.10
   papi/6.0.0.15
   papi/6.0.0.17
   papi/7.0.0.1
   papi/7.0.1.1
   papi/7.0.1.2
   papi/7.1.0.1                    (D)
lines 155-176
```

module av (9)

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh)

```
   perftools-base/22.06.0
   perftools-base/22.12.0
   perftools-base/23.03.0
   perftools-base/23.09.0
   perftools-base/23.12.0
   perftools-base/24.03.0          (L,D)
   perftools-lite-events
   perftools-lite-gpu
   perftools-lite-hbm
   perftools-lite-loops
   perftools-lite
   perftools-preload
   perftools
   rocm/6.0.3                      (5.0.2:5.1.0:5.2.0:5.2.3:5.5.1:5.7.0:6.0.0)
   sanitizers4hpc/1.0.1
   sanitizers4hpc/1.0.4
   sanitizers4hpc/1.1.1
   sanitizers4hpc/1.1.2            (D)
   valgrind4hpc/2.12.10
   valgrind4hpc/2.12.11
   valgrind4hpc/2.13.1
   valgrind4hpc/2.13.2             (D)
lines 177-198
```

module av (10)

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh)

```
------------------------------- /opt/cray/pe/lmod/lmod/modulefiles/Core --------------------------------
   lmod    settarg

-------------------------------------- HPE-Cray PE target modules --------------------------------------
   craype-accel-amd-gfx908    craype-arm-grace        craype-hugepages4M          craype-x86-milan-x
   craype-accel-amd-gfx90a    craype-hugepages128M    craype-hugepages512M        craype-x86-milan
   craype-accel-amd-gfx940    craype-hugepages16M     craype-hugepages64M         craype-x86-rome    (L)
   craype-accel-amd-gfx942    craype-hugepages1G      craype-hugepages8M          craype-x86-spr-hbm
   craype-accel-host          craype-hugepages256M    craype-network-none         craype-x86-spr
   craype-accel-nvidia70      craype-hugepages2G      craype-network-ofi    (L)   craype-x86-trento
   craype-accel-nvidia80      craype-hugepages2M      craype-network-ucx
   craype-accel-nvidia90      craype-hugepages32M     craype-x86-genoa

---------------------------------------------- Software stacks ----------------------------------------------
   CrayEnv     (S)    LUMI/23.03 (S)    LUMI/24.03            (S,D)    spack/22.08      spack/23.03-2
   LUMI/22.08 (S)    LUMI/23.09 (S)    Local-CSC/default     (S)      spack/22.08-2    spack/23.09    (D)
   LUMI/22.12 (S)    LUMI/23.12 (S)    Local-quantum/default (S)      spack/23.03

------------------------------------- Modify the module display style -------------------------------------
   ModuleColour/off       (S)      ModuleFullSpider/off    (S)        ModuleLabel/system    (S)
   ModuleColour/on        (S,D)    ModuleFullSpider/on     (S,D)      ModulePowerUser/LUMI (S)
lines 199-220
```

module av (11)

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh)

```
   ModuleExtensions/hide (S)      ModuleLabel/label       (S,L,D)    ModuleStyle/default
   ModuleExtensions/show (S,D)    ModuleLabel/PEhierarchy (S)        ModuleStyle/reset    (D)

------------------------------------------ System initialisation ------------------------------------------
   init-lumi/0.2 (S,L)

------------------------------------------ Non-PE HPE-Cray modules ----------------------------------------
   dvs/2.15_4.7.82-1.0_23.1__gfba2684e        xpmem/2.8.2-1.0_5.1__g84a27a5.shasta (L)
   libfabric/1.15.2.0                  (L)

-------- This is a list of module extensions. Use "module --nx avail ..." to not show extensions. --------
    rclone (E)     restic (E)     s3cmd (E)

These extensions cannot be loaded directly, use "module spider extension_name" for more information.

  Where:
   L:        Module is loaded
   S:        Module is Sticky, requires --force to unload or purge
   Aliases:  Aliases exist: foo/1.2.3 (1.2) means that "module load foo/1.2" will load foo/1.2.3
   D:        Default Module
   E:        Extension that is provided by another module

lines 221-242
```

module av (12)

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh)

```
   libfabric/1.15.2.0                  (L)

-------- This is a list of module extensions. Use "module --nx avail ..." to not show extensions. --------
    rclone (E)     restic (E)     s3cmd (E)

These extensions cannot be loaded directly, use "module spider extension_name" for more information.

  Where:
   L:        Module is loaded
   S:        Module is Sticky, requires --force to unload or purge
   Aliases:  Aliases exist: foo/1.2.3 (1.2) means that "module load foo/1.2" will load foo/1.2.3
   D:        Default Module
   E:        Extension that is provided by another module

Additional ways to search for software:
* Use "module spider" to find all possible modules and extensions.
* Use "module keyword key1 key2 ..." to search for all possible modules matching any of the "keys".
See the LUMI documentation at https://docs.lumi-supercomputer.eu/runjobs/lumi_env/Lmod_modules/ for more i
nformation on searching modules.
If then you still miss software, contact LUMI User Support via https://lumi-supercomputer.eu/user-support/
need-help/.

[lumi][kulust@uan04-1003 ~]$
```

Changing how the module list is displayed

- You may have noticed that you see descriptive texts in the module, not directories
- This can be changed by loading a module
  - `ModuleLabel/label` : The default view
  - `ModuleLabel/PEhierarchy` : Descriptive texts and unfolded PE hierarchy
  - `ModuleLabel/system` : Module directories
- Turn colour on or off using `ModuleColour/on` or `ModuleColour/off`
- Show or hide the module extensions with `ModuleExtensions/show` or `ModuleExtensions/hide`
- Index all modules with the spider command using `ModuleFullSpider/on`
- Show some hidden modules with `ModulePowerUser/LUMI`
  - This will also show undocumented/unsupported modules!
  - Can use `module --show_hidden avail` instead
- More customisation possible via LMOD environment variables

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 on LUMI, not even on LUMI-C
  - GNU-based toolchains: Would give problems with MPI (Open MPI)
  - Intel-based toolchains: Intel tools 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-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb

Additional information

Toolchain name and version (missing for SYSTEM)

Version of the package

Name of the package

Module: GROMACS/2024.3-cpeGNU-24.03-PLUMED-2.9.2-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

- Load the modules for the LUMI software stack and partition that you want to use. E.g.,
  `module load LUMI/24.03 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/24.03 partition/C
module load EasyBuild-user
```

kulust@uan03.lumi.csc - ~/EasyBuild

kulust@uan03.lumi.csc - ~/EasyBuild (ssh)

```
Did you know?
*************
On LUMI, it is not possible to use ssh to reach the compute nodes. All
connections to the compute nodes have to be made through the resource
manager, so you have to use srun instead. See also
https://docs.lumi-supercomputer.eu/runjobs/scheduled-jobs/interactive/

[lumi][kulust@uan03-1000 ~]$ module load LUMI/24.03 partition/C

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

[lumi][kulust@uan03-1001 ~]$ module load EasyBuild-user

EasyBuild configured to install software in the user tree at /users/kulust/EasyBuild for the LUMI/24.03
software stack for the LUMI/C partition.
  * Software installation directory: /users/kulust/EasyBuild/SW/LUMI-24.03/C
  * Modules installation directory: /users/kulust/EasyBuild/modules/LUMI/24.03/partition/C
  * Repository: /users/kulust/EasyBuild/ebfiles_repo/LUMI-24.03/LUMI-C
  * Work directory for builds and logs: /run/user/327000143/easybuild
    Clear work directory with clear-eb

[lumi][kulust@uan03-1002 EasyBuild]$
```

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

LUMI 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-2021.7-cpeGNU-23.09-CPU.eb, will build GROMACS/2021.7-cpeGNU-23.09-CPU
- EasyConfig GROMACS-2021.7-cpeGNU-23.09-PLUMED-2.8.3-noPython-CPU.eb, will build GROMACS/2021.7-cpeGNU-23.09-PLUMED-2.8.3-noPython-CPU
- EasyConfig GROMACS-2021.7-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb, will build GROMACS/2021.7-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU
- EasyConfig GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.8.3-noPython-CPU.eb, will build GROMACS/2022.5-cpeGNU-23.09-PLUMED-2.8.3-noPython-CPU

Table of contents

eb --search GROMACS | less

LUMI

kulust@uan03.lumi.csc - ~/EasyBuild

kulust@uan03.lumi.csc - ~/EasyBuild (ssh)

```
 * /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
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.7-cpeGNU-23.09-PLUMED-2.
9.0-noPython-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.
8.3-noPython-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.
9.0-noPython-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2022.6-cpeCray-23.09-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2022.6-cpeGNU-23.09-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2023.3-cpeCray-23.09-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2023.3-cpeGNU-23.09-CPU.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2024.1-cpeAMD-23.09-HeFFTe-ro
cm.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2024.1-cpeAMD-23.09-VkFFT-roc
m.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2024.3-cpeAMD-24.03-HeFFTe-ro
cm.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2024.3-cpeAMD-24.03-PLUMED-2.
9.2-noPython-rocm.eb
 * /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2024.3-cpeAMD-24.03-rocm.eb
lines 1-14
```

eb -S GROMACS | less

LUMI

kulust@uan03.lumi.csc - ~/EasyBuild

kulust@uan03.lumi.csc - ~/EasyBuild (ssh)

```
CFGS1=/appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs
 * $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-2023.3-cpeCray-23.09-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2023.3-cpeGNU-23.09-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2024.1-cpeAMD-23.09-HeFFTe-rocm.eb
 * $CFGS1/g/GROMACS/GROMACS-2024.1-cpeAMD-23.09-VkFFT-rocm.eb
 * $CFGS1/g/GROMACS/GROMACS-2024.3-cpeAMD-24.03-HeFFTe-rocm.eb
 * $CFGS1/g/GROMACS/GROMACS-2024.3-cpeAMD-24.03-PLUMED-2.9.2-noPython-rocm.eb
 * $CFGS1/g/GROMACS/GROMACS-2024.3-cpeAMD-24.03-rocm.eb
 * $CFGS1/g/GROMACS/GROMACS-2024.3-cpeCray-24.03-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2024.3-cpeCray-24.03-PLUMED-2.9.2-noPython-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2024.3-cpeGNU-24.03-CPU.eb
 * $CFGS1/g/GROMACS/GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb
 * $CFGS1/g/GROMACS/gromacs_cmake_manage_sycl.patch

Note: 46 matching archived easyconfig(s) found, use --consider-archived-easyconfigs to see them
```

http://CFGS1/g/GROMACS/GROMACS-2024.3-cpeAMD-24.03-HeFFTe-rocm.eb

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-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb:

    ```
    eb GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb -D
    ```

eb GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb -D

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh)

```
[lumi][kulust@uan04-1029 ~]$ eb GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb -D
== Temporary log file in case of crash /run/user/327000143/easybuild/tmp/eb-oxkbiye4/easybuild-e28t06p6.lo
g
Dry run: printing build status of easyconfigs and dependencies
CFGS=/appl/lumi
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-common/buildtools/buildtools-24.03-bootstrap.eb (module: bu
ildtools/24.03-bootstrap)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-C/cpeGNU/cpeGNU-24.03.eb (module: cpeGNU/24.03)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-common/syslibs/syslibs-24.03-static.eb (module: syslibs/24.
03-static)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-common/buildtools/buildtools-24.03.eb (module: buildtools/2
4.03)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-C/zlib/zlib-1.3.1-cpeGNU-24.03.eb (module: zlib/1.3.1-cpeGN
U-24.03)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-C/bzip2/bzip2-1.0.8-cpeGNU-24.03.eb (module: bzip2/1.0.8-cp
eGNU-24.03)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-C/GSL/GSL-2.7.1-cpeGNU-24.03-OpenMP.eb (module: GSL/2.7.1-c
peGNU-24.03-OpenMP)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-C/ICU/ICU-74.1-cpeGNU-24.03.eb (module: ICU/74.1-cpeGNU-24.
03)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-C/gzip/gzip-1.13-cpeGNU-24.03.eb (module: gzip/1.13-cpeGNU-
24.03)
http://CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-common/syslibs/syslibs-24.03-static.eb
/lz4-1.9.4-cpeGNU-24.03.eb (module: lz4/1.9.4-cpeGNU-2
```

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

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh)

```
03)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-C/gzip/gzip-1.13-cpeGNU-24.03.eb (module: gzip/1.13-cpeGNU-
24.03)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-C/lz4/lz4-1.9.4-cpeGNU-24.03.eb (module: lz4/1.9.4-cpeGNU-2
4.03)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-C/ncurses/ncurses-6.4-cpeGNU-24.03.eb (module: ncurses/6.4-
cpeGNU-24.03)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-C/gettext/gettext-0.22-cpeGNU-24.03-minimal.eb (module: get
text/0.22-cpeGNU-24.03-minimal)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-C/XZ/XZ-5.4.4-cpeGNU-24.03.eb (module: XZ/5.4.4-cpeGNU-24.0
3)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-C/zstd/zstd-1.5.5-cpeGNU-24.03.eb (module: zstd/1.5.5-cpeGN
U-24.03)
 * [x] $CFGS/mgmt/ebfiles_repo/LUMI-24.03/LUMI-C/Boost/Boost-1.83.0-cpeGNU-24.03.eb (module: Boost/1.83.0-
cpeGNU-24.03)
 * [ ] $CFGS/LUMI-EasyBuild-contrib/easybuild/easyconfigs/p/PLUMED/PLUMED-2.9.2-cpeGNU-24.03-noPython.eb (
module: PLUMED/2.9.2-cpeGNU-24.03-noPython)
 * [ ] $CFGS/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9
.2-noPython-CPU.eb (module: GROMACS/2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU)
== Temporary log file(s) /run/user/327000143/easybuild/tmp/eb-oxkbiye4/easybuild-e28t06p6.log* have been r
emoved.
== Temporary directory /run/user/327000143/easybuild/tmp/eb-oxkbiye4 has been removed.
[lumi][kulust@uan04-1030 ~]$
```

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-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb:

    ```
    eb GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb -D
    eb GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb -r
    ```

eb GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb -r

LUMI

kulust@uan03.lumi.csc - ~

kulust@uan03.lumi.csc - ~ (ssh)

```
== Temporary log file in case of crash /run/user/327000143/easybuild/tmp/eb-e8ec6ox3/easybuild-mggnwsu8.lo
g
== resolving dependencies ...
== processing EasyBuild easyconfig /users/kulust/EasyBuild/ebfiles_repo/LUMI-24.03/LUMI-C/PLUMED/PLUMED-2.
9.2-cpeGNU-24.03-noPython.eb
== building and installing PLUMED/2.9.2-cpeGNU-24.03-noPython...
== fetching files...
== creating build dir, resetting environment...
== unpacking...
== ... (took 4 secs)
== patching...
== preparing...
== ... (took 8 secs)
== configuring...
== ... (took 45 secs)
== building...
== ... (took 3 mins 54 secs)
== testing...
== installing...
== ... (took 44 secs)
== taking care of extensions...
== restore after iterating...
lines 1-20
```

eb GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb -r (2)

LUMI

kulust@uan03.lumi.csc - ~

kulust@uan03.lumi.csc - ~ (ssh)

```
== postprocessing...
== sanity checking...
== ... (took 7 secs)
== cleaning up...
== creating module...
== ... (took 3 secs)
== permissions...
== ... (took 1 secs)
== packaging...
== COMPLETED: Installation ended successfully (took 5 mins 52 secs)
== Results of the build can be found in the log file(s) /users/kulust/EasyBuild/SW/LUMI-24.03/C/PLUMED/2.9
.2-cpeGNU-24.03-noPython/easybuild/easybuild-PLUMED-2.9.2-20241007.123721.log
== processing EasyBuild easyconfig /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMA
CS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb
== building and installing GROMACS/2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU...
== fetching files...
== creating build dir, resetting environment...
== starting iteration #0 ...
== unpacking...
== ... (took 1 secs)
== patching...
== preparing...
lines 21-40
```

eb GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb -r(3)

LUMI

kulust@uan03.lumi.csc - ~

kulust@uan03.lumi.csc - ~ (ssh)

```
== ... (took 7 secs)
== configuring...
== ... (took 56 secs)
== building...
== ... (took 2 mins 57 secs)
== testing [skipped]
== installing...
== ... (took 13 secs)
== taking care of extensions...
== creating build dir, resetting environment...
== starting iteration #1 ...
== unpacking...
== ... (took 4 secs)
== patching...
== preparing...
== ... (took 5 secs)
== configuring...
== ... (took 58 secs)
== building...
== ... (took 2 mins 56 secs)
== testing [skipped]
== installing...
lines 41-62
```

eb GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb -r (4)

LUMI

kulust@uan03.lumi.csc - ~

kulust@uan03.lumi.csc - ~ (ssh)

```
== ... (took 5 secs)
== taking care of extensions...
== creating build dir, resetting environment...
== starting iteration #2 ...
== unpacking...
== ... (took 1 secs)
== patching...
== preparing...
== ... (took 5 secs)
== configuring...
== ... (took 53 secs)
== building...
== ... (took 2 mins 43 secs)
== testing [skipped]
== installing...
== ... (took 5 secs)
== taking care of extensions...
== creating build dir, resetting environment...
== starting iteration #3 ...
== unpacking...
== ... (took 4 secs)
== patching...
lines 63-84
```

eb GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb -r (5)

LUMI

kulust@uan03.lumi.csc - ~

kulust@uan03.lumi.csc - ~ (ssh)

```
== preparing...
== ... (took 5 secs)
== configuring...
== ... (took 58 secs)
== building...
== ... (took 2 mins 50 secs)
== testing [skipped]
== installing...
== ... (took 4 secs)
== taking care of extensions...
== restore after iterating...
== postprocessing...
== sanity checking...
== ... (took 6 secs)
== cleaning up...
== creating module...
== ... (took 3 secs)
== permissions...
== packaging...
== COMPLETED: Installation ended successfully (took 16 mins 33 secs)
== Results of the build can be found in the log file(s) /users/kulust/EasyBuild/SW/LUMI-24.03/C/GROMACS/20
24.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU/easybuild/easybuild-GROMACS-2024.3-20241007.125355.log
lines 85-105
```

eb GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb -r (6)

LUMI

kulust@uan03.lumi.csc - ~

kulust@uan03.lumi.csc - ~ (ssh)

```
== ... (took 2 mins 50 secs)
== testing [skipped]
== installing...
== ... (took 4 secs)
== taking care of extensions...
== restore after iterating...
== postprocessing...
== sanity checking...
== ... (took 6 secs)
== cleaning up...
== creating module...
== ... (took 3 secs)
== permissions...
== packaging...
== COMPLETED: Installation ended successfully (took 16 mins 33 secs)
== Results of the build can be found in the log file(s) /users/kulust/EasyBuild/SW/LUMI-24.03/C/GROMACS/20
24.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU/easybuild/easybuild-GROMACS-2024.3-20241007.125355.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-e8ec6ox3/easybuild-mggnwsu8.log* have been r
emoved.
== Temporary directory /run/user/327000143/easybuild/tmp/eb-e8ec6ox3 has been removed.
lines 90-109/109 (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-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb:

    ```
    eb GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-noPython-CPU.eb -D
    eb GROMACS-2024.3-cpeGNU-24.03-PLUMED-2.9.2-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-6.4.2-cpeGNU-23.09-build02.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/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/ebfiles_repo`
    - 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.

Containers

This is about containers on LUMI-C and LUMI-G!

- What can they do and what can't they do?
- Getting containers onto LUMI
- Running containers on LUMI
- Enhancements to the LUMI environment to help you

- But remember: LUMI is an HPC infrastructure, not a container cloud!

What do containers not provide?

- **Full reproducibility of your science** is a myth
  - Only reproducibility of the software stack, not of the results
- **Performance portability**:
  - A container built from sources on one CPU will not be optimal for another one.
  - Containers built from downloaded binaries may not exploit all architectural features of the CPU.
  - No support for the LUMI interconnect may lead to fall-down to slower protocol that works
- **Full portability**: Not every container prepared on your Ubuntu or CentOS cluster or workstation will work on LUMI.
  - Containers that rely on certain hardware, drivers/kernel modules and/or kernel versions may fail.
  - Problem cases: High-performance networking (MPI) and GPU (driver version)

But what can they then do on LUMI?

- **Storage manageability:** Lower pressure on the filesystems (for software frameworks that access hundreds of thousands of small files) for better I/O performance and management of your disk file quota.
  - E.g., conda installations are not appreciated straight on the Lustre file system
- **Software installation:** Can be a way to install software with an installation process that is not aware of multi-user HPC systems and is too complicated to recompile.
  - E.g., GUI applications that need a fat library stack
  - E.g., experiment with software that needs a newer version or ROCm, though with limitations
- **Isolation:** More important for services; often a pain instead
- **But note:** You're the system administrator of your container, not LUST!

Managing containers

- Supported runtimes
  - Docker is **NOT** directly available from user environment (and will never be)
  - Singularity Community Edition is natively available (as a system command) on the login and compute nodes
- But you can convert docker containers to singularity: Pulling containers
  - DockerHub and other registries (example: Julia container)
    `singularity pull docker://julia`
  - Singularity uses flat (single) sif file for storing container and the pull command makes the conversion
  - Be carefull: cache in `.singularity` dir can easily exhaust your storage quota for larger images
    - May want to set `SINGULARITY_CACHEDIR`

singularity pull docker://julia

LUMI

kulust@uan03.lumi.csc - ~/container-demo

kulust@uan03.lumi.csc - ~/container-demo (ssh)

```
[lumi][kulust@uan03-1004 container-demo]$ singularity pull docker://julia
INFO:    Converting OCI blobs to SIF format
INFO:    Starting build...
INFO:    Fetching OCI image...
5.4MiB / 5.4MiB [=====================================================================] 100 % 8.2 MiB/s 0s
27.8MiB / 27.8MiB [===================================================================] 100 % 8.2 MiB/s 0s
168.2MiB / 168.2MiB [=================================================================] 100 % 8.2 MiB/s 0s
INFO:    Extracting OCI image...
2024/10/07 17:05:53  warn rootless{usr/local/julia/lib/julia/libLLVM.so} ignoring (usually) harmless EPERM
 on setxattr "user.rootlesscontainers"
2024/10/07 17:05:53  warn rootless{usr/local/julia/lib/julia/libamd.so} ignoring (usually) harmless EPERM
on setxattr "user.rootlesscontainers"
2024/10/07 17:05:53  warn rootless{usr/local/julia/lib/julia/libamd.so.3} ignoring (usually) harmless EPER
M on setxattr "user.rootlesscontainers"
2024/10/07 17:05:53  warn rootless{usr/local/julia/lib/julia/libatomic.so} ignoring (usually) harmless EPE
RM on setxattr "user.rootlesscontainers"
2024/10/07 17:05:53  warn rootless{usr/local/julia/lib/julia/libatomic.so.1} ignoring (usually) harmless E
PERM on setxattr "user.rootlesscontainers"
2024/10/07 17:05:53  warn rootless{usr/local/julia/lib/julia/libblastrampoline.so} ignoring (usually) harm
less EPERM on setxattr "user.rootlesscontainers"
2024/10/07 17:05:53  warn rootless{usr/local/julia/lib/julia/libblastrampoline.so.5.11.0} ignoring (usuall
y) harmless EPERM on setxattr "user.rootlesscontainers"
2024/10/07 17:05:53  warn rootless{usr/local/julia/lib/julia/libbtf.so} ignoring (usually) harmless EPERM
```

singularity pull docker://julia

LUMI

kulust@uan03.lumi.csc - ~/container-demo

kulust@uan03.lumi.csc - ~/container-demo (ssh)

```
2024/10/07 17:05:55  warn rootless{usr/local/julia/lib/julia/libumfpack.so} ignoring (usually) harmless EP
ERM on setxattr "user.rootlesscontainers"
2024/10/07 17:05:55  warn rootless{usr/local/julia/lib/julia/libumfpack.so.6} ignoring (usually) harmless
EPERM on setxattr "user.rootlesscontainers"
2024/10/07 17:05:55  warn rootless{usr/local/julia/lib/julia/libunwind.so} ignoring (usually) harmless EPE
RM on setxattr "user.rootlesscontainers"
2024/10/07 17:05:55  warn rootless{usr/local/julia/lib/julia/libunwind.so.8} ignoring (usually) harmless E
PERM on setxattr "user.rootlesscontainers"
2024/10/07 17:05:55  warn rootless{usr/local/julia/lib/julia/libuv.so} ignoring (usually) harmless EPERM o
n setxattr "user.rootlesscontainers"
2024/10/07 17:05:55  warn rootless{usr/local/julia/lib/julia/libuv.so.2} ignoring (usually) harmless EPERM
 on setxattr "user.rootlesscontainers"
2024/10/07 17:05:55  warn rootless{usr/local/julia/lib/julia/libz.so} ignoring (usually) harmless EPERM on
 setxattr "user.rootlesscontainers"
2024/10/07 17:05:55  warn rootless{usr/local/julia/lib/julia/libz.so.1} ignoring (usually) harmless EPERM
on setxattr "user.rootlesscontainers"
2024/10/07 17:05:58  warn rootless{usr/local/julia/lib/libjulia.so} ignoring (usually) harmless EPERM on s
etxattr "user.rootlesscontainers"
2024/10/07 17:05:58  warn rootless{usr/local/julia/lib/libjulia.so.1.10} ignoring (usually) harmless EPERM
 on setxattr "user.rootlesscontainers"
INFO:    Inserting Singularity configuration...
INFO:    Creating SIF file...
[lumi][kulust@uan03-1005 container-demo]$
```

singularity pull docker://julia

LUMI

kulust@uan03.lumi.csc - ~/.singularity

kulust@uan03.lumi.csc - ~/.singularity (ssh)

```
2024/10/07 17:09:40  warn rootless{usr/local/julia/lib/libjulia.so.1.10} ignoring (usually) harmless EPERM
 on setxattr "user.rootlesscontainers"
INFO:    Inserting Singularity configuration...
INFO:    Creating SIF file...
[lumi][kulust@uan03-1016 container-demo]$ cd ~/.singularity/
[lumi][kulust@uan03-1017 .singularity]$ ls -la
total 12
drwx------  3 kulust pepr_kulust 4096 Oct  7 17:09 .
drwx------ 40 kulust pepr_kulust 4096 Oct  7 17:04 ..
drwx------  9 kulust pepr_kulust 4096 Oct  7 17:09 cache
[lumi][kulust@uan03-1018 .singularity]$ du -h
4.0K    ./cache/shub
202M    ./cache/blob/blobs/sha256
202M    ./cache/blob/blobs
202M    ./cache/blob
4.0K    ./cache/net
4.0K    ./cache/oras
4.0K    ./cache/oci-sif
4.0K    ./cache/library
197M    ./cache/oci-tmp
398M    ./cache
398M    .
[lumi][kulust@uan03-1019 .singularity]$
```

Managing containers (2)

- Building containers
  - Support for building containers is very limited on LUMI: No elevated privileges but also no fakeroot and no user namespaces.
    We can support `proot` though.
  - One optiont is to pull or copy containers from outside
  - But singularity can build from existing (base) container in some cases (but need to load a recent systools module for `proot`)
    - Build type called "Unprivileged proot builds" in the Singularity CE manual
    - Needs `proot` from the `systools/24.03` module in CrayEnv and LUMI/24.03.
  - We provide some base images adapted for LUMI

Interacting with containers

- Accessing a container with the shell command
  `singularity shell container.sif`
- Executing a command in the container with exec
  `singularity exec container.sif uname -a`
- "Running" a container
  `singularity run container.sif`
- Inspecting run definition script
  `singularity inspect --runscript container.sif`
- Accessing host filesystem with bind mounts
  - Singularity will mount `$HOME`, `/tmp`, `/proc`, `/sys`, `/dev` into container by default
  - Use `--bind src1:dest1,src2:dest2` or the `SINGULARITY_BIND(PATH)` environment variable to mount other host directories (like `/project` or `/appl`)

singularity shell julia_latest.sif

LUMI

kulust@uan03.lumi.csc - ~/container-demo

kulust@uan03.lumi.csc - ~/container-demo (ssh)

```
[lumi][kulust@uan03-1023 container-demo]$ ls /opt
admin-pe  AMD  cray  esmi  modulefiles  rocm  rocm-6.0.3  slingshot
[lumi][kulust@uan03-1024 container-demo]$ singularity shell julia_latest.sif
Singularity> ls /opt
Singularity> cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION_ID="12"
VERSION="12 (bookworm)"
VERSION_CODENAME=bookworm
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
Singularity> exit
exit
[lumi][kulust@uan03-1025 container-demo]$
```

singularity exec julia_latest.sif uname -a

LUMI

kulust@uan03.lumi.csc - ~/container-demo

kulust@uan03.lumi.csc - ~/container-demo (ssh)

```
[lumi][kulust@uan03-1026 container-demo]$ uname -a
Linux uan03 5.14.21-150500.55.49_13.0.56-cray_shasta_c #1 SMP Mon Mar 4 14:19:49 UTC 2024 (9d8355b) x86_64
 x86_64 x86_64 GNU/Linux
[lumi][kulust@uan03-1027 container-demo]$ singularity exec julia_latest.sif uname -a
Linux uan03 5.14.21-150500.55.49_13.0.56-cray_shasta_c #1 SMP Mon Mar 4 14:19:49 UTC 2024 (9d8355b) x86_64
 GNU/Linux
[lumi][kulust@uan03-1028 container-demo]$ singularity exec julia_latest.sif cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION_ID="12"
VERSION="12 (bookworm)"
VERSION_CODENAME=bookworm
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
[lumi][kulust@uan03-1029 container-demo]$
```

```
singularity run julia_latest.sif
singularity inspect -runscript julia_latest.sif
```

LUMI

kulust@uan03.lumi.csc - ~/container-demo

kulust@uan03.lumi.csc - ~/container-demo (ssh)

```
[lumi][kulust@uan03-1030 container-demo]$ singularity run julia_latest.sif

               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.10.5 (2024-08-27)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia>
[lumi][kulust@uan03-1031 container-demo]$ singularity inspect --runscript julia_latest.sif
#!/bin/sh
OCI_ENTRYPOINT='"docker-entrypoint.sh"'
OCI_CMD='"julia"'

# When SINGULARITY_NO_EVAL set, use OCI compatible behavior that does
# not evaluate resolved CMD / ENTRYPOINT / ARGS through the shell, and
# does not modify expected quoting behavior of args.
if [ -n "$SINGULARITY_NO_EVAL" ]; then
        # ENTRYPOINT only - run entrypoint plus args
        if [ -z "$OCI_CMD" ] && [ -n "$OCI_ENTRYPOINT" ]; then
                set -- 'docker-entrypoint.sh' "$@"
```

Interacting with containers

- Accessing a container with the shell command
  `singularity shell container.sif`
- Executing a command in the container with exec
  `singularity exec container.sif uname -a`
- "Running" a container
  `singularity run container.sif`
- Inspecting run definition script
  `singularity inspect --runscript container.sif`
- Accessing host filesystem with bind mounts
  - Singularity will mount `$HOME`, `/tmp`, `/proc`, `/sys`, `/dev` into container by default
  - Use `--bind src1:dest1,src2:dest2` or the `SINGULARITY_BIND(PATH)` environment variable to mount other host directories (like `/project` or `/appl`)

Running containers on LUMI

- Use SLURM to run containers on compute nodes
- Use srun to execute MPI containers

```
srun singularity exec --bind ${BIND_ARGS} \
${CONTAINER_PATH} my_mpi_binary ${APP_PARAMS}
```

- **Be aware your container must be compatible with Cray MPI** (MPICH ABI compatible)
  - Configure suggestion: see next slide
- Open MPI based containers need workarounds and are not well supported on LUMI at the moment (and even more problematic for the GPU)

Environment enhancements (1)

- LUMI specific tools for container interaction provided as modules
- **singularity-bindings/system** (available via easyconfig)
  - Sets the environment to use Cray MPICH provided outside the container
  - Requires a LUMI software stack
  - Use EasyBuild-user module and eb --search singularity-bindings to find the easyconfig or copy from our LUMI Software Library web site
  - Provides basic bind mounts for using the host MPI in the container setting `SINGULARITY_BIND` and `SINGULARITY_LD_LIBRARY_PATH`
- **lumi-vnc** (LUMI and CrayEnv software stacks)
  - Provides basic VNC virtual desktop for interacting with graphical interfaces via a web browser or VNC client
  - Open OnDemand a better alternative for many

Environment enhancements (2)
Containerising tools

- **cotainr** (LUMI and CrayEnv software stacks)
  - A tool to pack conda installations in a singularity container
  - Use the singularity commands as shown on earlier slides to run
- **lumi-container-wrapper** (LUMI and CrayEnv software stacks)
  - Supports conda and pip environments
  - With pip: Python provided by the cray-python module (so there is an optimised NumPy etc.)
  - Software installation in two parts: a base container and a SquashFS file which is mounted in that container with the conda/pip environment
  - Provides wrappers to encapsulate your custom environment in a container (so you don't use singularity commands directly)
  - Still helps with quota on the number of files in your project and I/O performance

lumi-container-wrapper (1)

LUMI

kulust@uan03.lumi.csc - ~/Tykky-demo ⌥⌘2

kulust@uan03.lumi.csc - ~/Tykky-demo (ssh) ⌘1

```
[lumi][kulust@uan03-1033 container-demo]$ cd
[lumi][kulust@uan03-1034 ~]$ cd Tykky-demo/
[lumi][kulust@uan03-1035 Tykky-demo]$ ls
conda-cont-1  env.yml
[lumi][kulust@uan03-1036 Tykky-demo]$ cat env.yml
channels:
  - conda-forge
dependencies:
  - python=3.8.8
  - scipy
  - nglview
[lumi][kulust@uan03-1037 Tykky-demo]$ module load LUMI/24.03 lumi-container-wrapper
[lumi][kulust@uan03-1038 Tykky-demo]$
```

lumi-container-wrapper (2)

LUMI

kulust@uan03.lumi.csc - ~/Tykky-demo ⌥⌘2

kulust@uan03.lumi.csc - ~/Tykky-demo (ssh) ⌘1

```
[lumi][kulust@uan03-1051 Tykky-demo]$ module load LUMI/24.03 lumi-container-wrapper
[lumi][kulust@uan03-1052 Tykky-demo]$ conda-containerize new --prefix ./conda-cont-1 env.yml
[ INFO ] Constructing configuration
[ INFO ] Using /tmp/kulust/cw-5G4U3S as temporary directory
[ INFO ] Installation dir ./conda-cont-1 does not exist, creating it for you
[ INFO ] Fetching container docker://opensuse/leap:15.5
[ INFO ] Running installation script
[ INFO ] Using miniconda version Miniconda3-latest-Linux-x86_64
[ INFO ] Installing miniconda
==========================
PREFIX=/LUMI_TYKKY_ngNvc4X/miniconda
Unpacking payload ...

Installing base environment...

Preparing transaction: ...working... done
Executing transaction: ...working... done
installation finished.
WARNING:
    You currently have a PYTHONPATH environment variable set. This may cause
    unexpected behavior when running the Python interpreter in Miniconda3.
    For best results, please verify that your PYTHONPATH only points to
    directories of packages that are compatible with the Python interpreter
```

lumi-container-wrapper (3)

LUMI

kulust@uan03.lumi.csc - ~/Tykky-demo

kulust@uan03.lumi.csc - ~/Tykky-demo (ssh)

```
=========================
[ INFO ] Running user supplied commands
[ INFO ] Creating sqfs image
Parallel mksquashfs: Using 8 processors
Creating 4.0 filesystem on _deploy/img.sqfs, block size 131072.
[=================================================================================|] 49574/49574 100%

Exportable Squashfs 4.0 filesystem, gzip compressed, data block size 131072
        compressed data, compressed metadata, compressed fragments,
        compressed xattrs, compressed ids
        duplicates are removed
Filesystem size 728765.46 Kbytes (711.69 Mbytes)
        38.40% of uncompressed filesystem size (1897964.71 Kbytes)
Inode table size 548501 bytes (535.65 Kbytes)
        23.36% of uncompressed inode table size (2347783 bytes)
Directory table size 782658 bytes (764.31 Kbytes)
        41.93% of uncompressed directory table size (1866647 bytes)
Number of duplicate files found 7700
Number of inodes 50922
Number of files 38183
Number of fragments 2292
Number of symbolic links 5296
Number of device nodes 0
```

lumi-container-wrapper (4)

LUMI

kulust@uan03.lumi.csc - ~/Tykky-demo

kulust@uan03.lumi.csc - ~/Tykky-demo (ssh)

```
        41.93% of uncompressed directory table size (1866647 bytes)
Number of duplicate files found 7700
Number of inodes 50922
Number of files 38183
Number of fragments 2292
Number of symbolic links 5296
Number of device nodes 0
Number of fifo nodes 0
Number of socket nodes 0
Number of directories 7443
Number of hard-links 27284
Number of ids (unique uids + gids) 1
Number of uids 1
        kulust (327000143)
Number of gids 1
        pepr_kulust (327000143)
[ INFO ] Creating wrappers
[ INFO ] Installing to ./conda-cont-1
[ INFO ] Done, duration: 125s
[ INFO ] Program has been installed to ./conda-cont-1
                To use add the bin folder to your path e.g:
                export PATH="/users/kulust/Tykky-demo/conda-cont-1/bin:$PATH"
[lumi][kulust@uan03-1053 Tykky-demo]$
```

lumi-container-wrapper (5)

LUMI

kulust@uan03.lumi.csc - ~/Tykky-demo

kulust@uan03.lumi.csc - ~/Tykky-demo (ssh)

```
[lumi][kulust@uan03-1053 Tykky-demo]$ ls conda-cont-1/
_bin  bin  common.sh  container.sif  img.sqfs  share
[lumi][kulust@uan03-1054 Tykky-demo]$ ls conda-cont-1/bin
2to3            jsonschema              lzegrep             python3             wsdump
2to3-3.8        jupyter                 lzfgrep             python3.8           x86_64-conda_cos6-linux-gnu-ld
captoinfo       jupyter-dejavu          lzgrep              python3.8-config    x86_64-conda-linux-gnu-ld
clear           jupyter-events          lzless              python3-config      xz
c_rehash        jupyter-execute         lzma                reset               xzcat
curve_keygen    jupyter-kernel          lzmadec             send2trash          xzcmp
_debug_exec     jupyter-kernelspec      lzmainfo            sqlite3             xzdec
debugpy         jupyter-lab             lzmore              sqlite3_analyzer    xzdiff
_debug_shell    jupyter-labextension    ncurses6-config     tabs                xzegrep
f2py            jupyter-labhub          ncursesw6-config    tclsh               xzfgrep
f2py3           jupyter-migrate         normalizer          tclsh8.6            xzgrep
f2py3.8         jupyter-nbconvert       openssl             tic                 xzless
httpx           jupyter-notebook        pip                 toe                 xzmore
idle3           jupyter-run             pip3                tput                zstd
idle3.8         jupyter-server          pybabel             tset                zstdcat
infocmp         jupyter-troubleshoot    pydoc               unlzma              zstdgrep
infotocap       jupyter-trust           pydoc3              unxz                zstdless
ipython         list-packages           pydoc3.8            unzstd              zstdmt
ipython3        lzcat                   pygmentize          wheel
jlpm            lzcmp                   pyjson5             wish
```

lumi-container-wrapper (6)

LUMI

kulust@uan03.lumi.csc - ~/Tykky-demo/conda-cont-1/bin ⌥⌘2

kulust@uan03.lumi.csc - ~/Tykky-demo/conda-cont-1/bin (ssh) ⌘1

```
curve_keygen   jupyter-kernel         lzmadec            send2trash         xzcmp
_debug_exec    jupyter-kernelspec     lzmainfo           sqlite3            xzdec
debugpy        jupyter-lab            lzmore             sqlite3_analyzer   xzdiff
_debug_shell   jupyter-labextension   ncurses6-config    tabs               xzegrep
f2py           jupyter-labhub         ncursesw6-config   tclsh              xzfgrep
f2py3          jupyter-migrate        normalizer         tclsh8.6           xzgrep
f2py3.8        jupyter-nbconvert      openssl            tic                xzless
httpx          jupyter-notebook       pip                toe                xzmore
idle3          jupyter-run            pip3               tput               zstd
idle3.8        jupyter-server         pybabel            tset               zstdcat
infocmp        jupyter-troubleshoot   pydoc              unlzma             zstdgrep
infotocap      jupyter-trust          pydoc3             unxz               zstdless
ipython        list-packages          pydoc3.8           unzstd             zstdmt
ipython3       lzcat                  pygmentize         wheel
jlpm           lzcmp                  pyjson5            wish
jsonpointer    lzdiff                 python             wish8.6
[lumi][kulust@uan03-1055 Tykky-demo]$ cd conda-cont-1/bin
[lumi][kulust@uan03-1056 bin]$ ./python3
Python 3.8.8 | packaged by conda-forge | (default, Feb 20 2021, 16:22:27)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>>
```

Environment enhancements (3): Prebuilt containers for AI (and some others)

LUMI

- Currently available
  - PyTorch: Best tested
  - TensorFlow
  - JAX
  - AlphaFold
  - ROCm and mpi4py
- See the LUMI documentation and LUMI Software Library for more information
  - Or check out the materials from the past 2-day course in Amsterdam in May 2024 for more on this (though the examples are for the containers that were on the system at that time)

Container limitations on LUMI

- Containers use the host's operating system kernel which may be different from your system. Containers do not abstract hardware.
- A generic container may not offer sufficiently good support for the Slingshot 11 interconnect on LUMI and fall back to TCP sockets resulting in poor performance, or not work at all.
  - Solution by injecting Cray MPICH, but only for containers with ABI compatibility with MPICH.
  - Distributed AI: Need to inject the proper RCCL plugin.
- AMD driver version may pose problems also.
- Only limited support to build containers on LUMI due to security concerns.