



LUMI

LUMI Software Stacks

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

What this talk is about...

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



Design considerations

- Very leading edge and inhomogeneous machine (new interconnect, new GPU architecture with an immature software ecosystem, some NVIDIA GPUs for visualisation, a mix of zen2 and zen3)
 - Need to remain agile
- Users that come to LUMI from 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 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

- 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
- 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, Python installations need to go in containers
 - We offer a container-based wrapper ([lumi-container-wrapper](#)) 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 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 Compiler 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

Accessing the Cray PE on LUMI

The LUMI software stack

- The LUMI software stack uses two levels of modules
 - LUMI/21.12, 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` can only be loaded if a compiler module and network target module are loaded
 - Many of the performance monitoring tools only become available after loading `perftools-base`
 - `cray-fftw` only becomes available when a processor target module is loaded
- Tools
 - `module avail` is for searching in the available modules
 - `module spider` and `module keyword` are for searching in the installed modules

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”
- `module spider gnuplot` : Shows all versions of gnuplot on the system
`module spider CMake`
- `module spider gnuplot/5.4.3-cpeGNU-22.08` : 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/3.24.0` : Will tell you which module contains CMake and how to load it

module spider (command) (1)

LUMI

module spider (command) (2)

LUMI

```
kurtlust@uan03.lumi.csc - ~
kurtlust@uan03.lumi.csc - ~ (ssh) ⌘2 ⌘1 +
```

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-archive: [Autoconf-archive/2021.02.19](#) (E), ...

Automake: [Automake/1.16.4](#) (E), [Automake/1.16.5](#) (E)

Bison: [Bison/3.8.1](#) (E), [Bison/3.8.2](#) (E)

Blosc: Blosc/1.21.0-cpeCray-21.08, Blosc/1.21.0-cpeGNU-21.08, ...
Blosc is an extremely fast, multi-threaded, meta-compressor library

Boost: Boost/1.77.0-cpeAOCC-21.12, Boost/1.77.0-cpeCray-21.08, ...
Boost provides free peer-reviewed portable C++ source libraries.

Lines 1-21

module spider (command) (3)

LUMI

```
kurtlust@uan03.lumi.csc - ~
kurtlust@uan03.lumi.csc - ~ (ssh)
zlib: zlib/1.2.11-cpeAOCC-21.12, zlib/1.2.11-cpeCray-21.08, ...
    Free lossless data-compression library, not covered by any patents.

zstd: zstd/1.5.0-cpeAOCC-21.12, zstd/1.5.0-cpeCray-21.08, zstd/1.5.0-cpeCray-21.12, ...

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@uan03-1001 ~]$
```

```
module spider gnuplot
```

LUMI

```
kurtlust@uan03.lumi.csc - ~
kurtlust@uan03.lumi.csc - ~ (ssh) ⌘1 +
```

gnuplot:

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

Versions:

- gnuplot/5.4.2-cpeCray-21.08
- gnuplot/5.4.2-cpeGNU-21.08
- gnuplot/5.4.3-cpeAOCC-21.12
- gnuplot/5.4.3-cpeAOCC-22.08
- gnuplot/5.4.3-cpeCray-21.12
- gnuplot/5.4.3-cpeCray-22.08
- gnuplot/5.4.3-cpeGNU-21.12
- gnuplot/5.4.3-cpeGNU-22.08

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.

lines 1-20

module spider gnuplot (2)

LUMI

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

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

Versions:

```
gnuplot/5.4.2-cpeCray-21.08
gnuplot/5.4.2-cpeGNU-21.08
gnuplot/5.4.3-cpeAOCC-21.12
gnuplot/5.4.3-cpeAOCC-22.08
gnuplot/5.4.3-cpeCray-21.12
gnuplot/5.4.3-cpeCray-22.08
gnuplot/5.4.3-cpeGNU-21.12
gnuplot/5.4.3-cpeGNU-22.08
```

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/5.4.3-cpeGNU-22.08
```

```
[lumi] [kurtlust@uan01-1006 ~]$
```

```
module spider cmake
```

LUMI

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

CMake:

Versions:

- [CMake/3.21.2](#) (E)
- [CMake/3.22.2](#) (E)
- [CMake/3.23.2](#) (E)
- [CMake/3.24.0](#) (E)

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

For detailed information about a specific "CMake" 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 CMake/3.24.0
```

```
[lumi] [kurtlust@uan01-1005 ~]$
```

```
kurtlust@uan03.lumi.csc - ~
kurtlust@uan03.lumi.csc - ~ (ssh)
-----
gnuplot: gnuplot/5.4.3-cpeGNU-22.08
-----
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
/5.4.3-cpeGNU-22.08" module is available to load.

LUMI/22.08 partition/C
LUMI/22.08 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
lines 1-20
```

A screenshot of a terminal window titled "kurtlust@uan03.lumi.csc - ~". The window shows the following text:

```
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 GNUpot 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] [kurtlust@uan03-1004 ~]\$

```
kurtlust@uan03.lumi.csc - ~
kurtlust@uan03.lumi.csc - ~ (ssh) ⌘2 ⌘1 +
```

CMake: [CMake/3.24.0 \(E\)](#)

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 hierarchy.

```
buildtools/22.08 (LUMI/22.08 partition/L)
buildtools/22.08 (LUMI/22.08 partition/G)
buildtools/22.08 (LUMI/22.08 partition/EAP)
buildtools/22.08 (LUMI/22.08 partition/D)
buildtools/22.08 (LUMI/22.08 partition/C)
buildtools/22.08 (CrayEnv)
buildtools/22.08-noPython (LUMI/22.08 partition/L)
buildtools/22.08-noPython (LUMI/22.08 partition/G)
buildtools/22.08-noPython (LUMI/22.08 partition/EAP)
buildtools/22.08-noPython (LUMI/22.08 partition/D)
buildtools/22.08-noPython (LUMI/22.08 partition/C)
buildtools/22.08-noPython (CrayEnv)
```

lines 1-19

```
kurtlust@uan03.lumi.csc - ~
kurtlust@uan03.lumi.csc - ~ (ssh) ⌘1 + ⌘2

buildtools/22.08 (LUMI/22.08 partition/EAP)
buildtools/22.08 (LUMI/22.08 partition/D)
buildtools/22.08 (LUMI/22.08 partition/C)
buildtools/22.08 (CrayEnv)
buildtools/22.08-noPython (LUMI/22.08 partition/L)
buildtools/22.08-noPython (LUMI/22.08 partition/G)
buildtools/22.08-noPython (LUMI/22.08 partition/EAP)
buildtools/22.08-noPython (LUMI/22.08 partition/D)
buildtools/22.08-noPython (LUMI/22.08 partition/C)
buildtools/22.08-noPython (CrayEnv)
buildtools/22.08-minimal (LUMI/22.08 partition/L)
buildtools/22.08-minimal (LUMI/22.08 partition/G)
buildtools/22.08-minimal (LUMI/22.08 partition/EAP)
buildtools/22.08-minimal (LUMI/22.08 partition/D)
buildtools/22.08-minimal (LUMI/22.08 partition/C)
buildtools/22.08-minimal (CrayEnv)

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

[lumi] [kurtlust@uan03-1005 ~]$
```

module keyword

- Currently not yet very useful due to a bug in Cray Lmod
- 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@uan03.lumi.csc - ~
kurtlust@uan03.lumi.csc - ~ (ssh) ⌘1 +
```

The following modules match your search criteria: "https"

Autoconf: [Autoconf/2.71](#) (E)

Autoconf-archive: [Autoconf-archive/2021.02.19](#) (E), ...

Automake: [Automake/1.16.4](#) (E), [Automake/1.16.5](#) (E)

Bison: [Bison/3.8.1](#) (E), [Bison/3.8.2](#) (E)

CMake: [CMake/3.21.2](#) (E), [CMake/3.22.2](#) (E), ...

CubeLib: [CubeLib/4.6](#) (E)

CubeWriter: [CubeWriter/4.6](#) (E)

Doxygen: [Doxygen/1.9.2](#) (E), [Doxygen/1.9.3](#) (E), ...

lines 1-21

module keyword https (2)

LUMI

A screenshot of a terminal window titled "kurtlust@uan03.lumi.csc - ~". The window shows the output of a command related to the "https" module. The output lists various tools and their versions, all in blue text, indicating they are links:

- GPP: [GPP/2.27](#) (E)
- M4: [M4/1.4.19](#) (E)
- Meson: [Meson/0.59.1](#) (E), [Meson/0.61.1](#) (E), ...
- NASM: [NASM/2.15.05](#) (E)
- Ninja: [Ninja/1.10.2](#) (E), [Ninja/1.11.0](#) (E)
- OPARI2: [OPARI2/2.0.6](#) (E)
- OTF2: [OTF2/2.3](#) (E)
- SCons: [SCons/4.2.0](#) (E), [SCons/4.3.0](#) (E)
- Yasm: [Yasm/1.3.0](#) (E)
- byacc: [byacc/20210808](#) (E), [byacc/20220128](#) (E)
- cURL: [cURL/7.78.0-cpeAOCC-21.12](#), [cURL/7.78.0-cpeCray-21.08](#), ...

The bottom left corner of the terminal window has a black bar with white text that reads "lines 22-42".

module keyword https (3)

LUMI

A screenshot of a terminal window titled "kurtlust@uan03.lumi.csc - ~". The window shows the output of the command "module keyword https (3)". The output lists several modules and their versions:

- byacc: [byacc/20210808](#) (E), [byacc/20220128](#) (E)
- cURL: cURL/7.78.0-cpeAOCC-21.12, cURL/7.78.0-cpeCray-21.08, ...
Command line tool and library for transferring data with URLs.
- flex: [flex/2.6.4](#) (E)
- gperf: [gperf/3.1](#) (E)
- help2man: [help2man/1.48.5](#) (E), [help2man/1.49.2](#) (E)
- htop: [htop/3.1.1](#) (E), [htop/3.1.2](#) (E), ...
- libtool: [libtool/2.4.6](#) (E)
- make: [make/4.3](#) (E)
- makeinfo: [makeinfo/6.8](#) (E)
- patchelf: [patchelf/0.13](#) (E), [patchelf/0.14.3](#) (E), ...

At the bottom left, there is a button labeled "lines 40-60".

module keyword https (4)

LUMI

```
kurtlust@uan03.lumi.csc - ~
kurtlust@uan03.lumi.csc - ~ (ssh)
re2c: re2c/2.2 (E), re2c/3.0 (E)

sec: sec/4.8 (E)

tree: tree/1.8.0 (E), tree/2.0.2 (E)

wget: wget/1.21.1-cpeCray-21.08, wget/1.21.1-cpeGNU-21.08, ...
      wget - GNU wget, a free software package for retrieving files using HTTP, HTTPS
      and FTP

xxd: xxd/8.2.4293 (E), xxd/8.2.5172 (E), ...

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.
lines 61-81
```

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`
- 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, which implies that the target modules and partition module will be switched (back) to those for the current node.

```
kurtlust@uan03.lumi.csc - ~
kurtlust@uan03.lumi.csc - ~ (ssh)
----- EasyBuild managed software for software stack unknown on LUMI-X -----
ARMForge/22.0.1      lumi-vnc/20220715      lumi-workspaces/0.1
Vampir/10.0.0        lumi-vnc/20221010 (D)

----- HPE-Cray PE modules -----
PrgEnv-amd/8.3.3          cray-mpich-abi/8.1.18      (D)
PrgEnv-aocc/8.2.0          cray-mpich/8.1.12       (8.1.8)
PrgEnv-aocc/8.3.3          (D)                         cray-mpich/8.1.17
PrgEnv-cray/8.2.0          (8.1.0)                      cray-mpich/8.1.18   (L,D)
PrgEnv-cray/8.3.3          (L,D)                        cray-mrnet/5.0.2
PrgEnv-gnu/8.2.0           (8.1.0)                      cray-mrnet/5.0.4    (D)
PrgEnv-gnu/8.3.3           (D)                         cray-openshmemx/11.5.0 (11.3.2)
amd-mixed/5.0.2            cray-openshmemx/11.5.5
amd/5.0.2                  cray-openshmemx/11.5.6      (D)
aocc-mixed/3.1.0           (D)                         cray-pals/1.1.3     (1.0.14)
aocc-mixed/3.2.0           cray-pals/1.1.8
aocc/3.1.0                 (D)                         cray-pals/1.2.0     (D)
aocc/3.2.0
atp/3.14.8                 (3.14.3)                     cray-parallel-netcdf/1.12.2.5
atp/3.14.11                cray-pmi-lib/6.0.16      (6.0.13)
                                         cray-pmi-lib/6.0.17      (D)

lines 1-21
```

```
kurtlust@uan03.lumi.csc - ~
kurtlust@uan03.lumi.csc - ~ (ssh)  #1 + ↻%2

cray-hdf5-parallel/1.12.1.5          perftools-lite-gpu
cray-hdf5/1.12.1.5                  perftools-lite-hbm
cray-libpals/1.1.3      (1.0.14)    perftools-lite-loops
cray-libpals/1.1.8                  perftools-preload
cray-libpals/1.2.0      (D)         rocm/5.0.2           (D)
cray-libsci/21.08.1.2              sanitizers4hpc/1.0.0
cray-libsci/22.06.1.3              sanitizers4hpc/1.0.1   (D)
cray-libsci/22.08.1.1      (L,D)     settarg
cray-libsci_acc/22.08.1.1          valgrind4hpc/2.12.6   (2.12.2)
cray-mpich-abi/8.1.12      (8.1.8)   valgrind4hpc/2.12.8
cray-mpich-abi/8.1.17          valgrind4hpc/2.12.10   (D)

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

lines 50-70
```

```
kurtlust@uan03.lumi.csc - ~
kurtlust@uan03.lumi.csc - ~ (ssh) ⌘1 + ⌘2

----- Software stacks -----
CrayEnv (S)    LUMI/21.12 (S,D)   LUMI/22.06 (S)   LUMI/22.08 (S)   spack/22.08

----- Modify the module display style -----
ModuleColour/off (S)      ModuleLabel/PEhierarchy (S)      ModuleStyle/default
ModuleColour/on (S,D)      ModuleLabel/system (S)      ModuleStyle/reset (D)
ModuleLabel/label (S,L,D)  ModulePowerUser/LUMI (S)

----- System initialisation -----
init-lumi/0.1 (S,L)

----- Non-PE HPE-Cray modules -----
chapel/1.26.0
cray-lustre-client-ofed/2.15.0.2_rc2_cray_113_g62287d0-2.3_6.7__g62287d05d3.shasta
dvs/2.15_4.3.59-2.3_22.6__g2786751f
libfabric/1.15.0.0
rocm/5.0.2
xpmem/2.4.4-2.3_9.1__gff0e1d9.shasta

----- This is a list of module extensions -----
lines 71-91
```

```
kurtlust@uan03.lumi.csc - ~
kurtlust@uan03.lumi.csc - ~ (ssh)
Autoconf      (E)    GPP      (E)    Yasm     (E)    makeinfo (E)
Autoconf-archive (E)  M4       (E)    byacc    (E)    patchelf (E)
Automake       (E)    Meson    (E)    flex     (E)    re2c     (E)
Bison          (E)    NASM    (E)    gperf    (E)    sec      (E)
CMake          (E)    Ninja    (E)    help2man (E)  tree     (E)
CubeLib         (E)    OPARI2   (E)    htop     (E)    xxd      (E)
CubeWriter      (E)    OTF2     (E)    libtool   (E)
Doxygen         (E)    SCons    (E)    make     (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 fo
o/1.2.3
  D:           Default Module
  E:           Extension that is provided by another module

Additional ways to search for software:
lines 92-110
```

```
kurtlust@uan03.lumi.csc - ~
kurtlust@uan03.lumi.csc - ~ (ssh) ⌘2 ⌘1 +
```

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 information on searching modules.

If then you still miss software, contact LUMI User Support via <https://lumi-supercomputer.eu/user-support/need-help/>.

```
[lumi] [kurtlust@uan03-1007 ~]$
```

Changing how the module list is displayed

- You may have noticed that you don't see directories in the module view but descriptive texts
- This can be changed by loading a module
 - `ModuleLabel/label` : The default view
 - `ModuleLabel/PEhierarchy` : Descriptive texts, but the PE hierarchy is unfolded
 - `ModuleLabel/system` : Module directories
- Turn colour on or off using `ModuleColour/on` or `ModuleColour/off`
- Show some hidden modules with `ModulePowerUser/LUMI`
 - This will also show undocumented/unsupported modules!

Installing software on HPC systems

- 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
 - 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 [lumi-supercomputer.github.io/LUMI-EasyBuild-docs](#)

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_46500000/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
- Then load the LUMI module, partition module and EasyBuild-user module
 - In many cases, cross-compilation is possible by loading a different partition module than the one auto-loaded by LUMI

```
module load LUMI/22.08 partition/C  
module load EasyBuild-user
```

LUMI

A screenshot of a terminal window titled "kulust@uan01.lumi.csc - ~". The window shows the following text:

```
kulust@uan01.lumi.csc - ~  
kulust@uan01.lumi.csc - ~ (ssh)  
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 ~]$
```

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/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
```

Step 3: Install the software

- Let's, e.g., install GROMACS
 - Search if GROMACS build recipes are available

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

This will be improved in the future.
 - 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

LUMI

```
kulust@uan01.lumi.csc - ~
kulust@uan01.lumi.csc - ~ (ssh)
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2020.4-cpeCray-21.08-PLUMED-2.6.4-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2020.4-cpeGNU-21.08-PLUMED-2.6.4-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2020.6-cpeCray-21.08-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2020.6-cpeCray-21.08-PLUMED-2.7.2-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2020.6-cpeGNU-21.08-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2020.6-cpeGNU-21.08-PLUMED-2.7.2-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021-cpeCray-21.08-PLUMED-2.7.2-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021-cpeGNU-21.08-PLUMED-2.7.2-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.3-cpeCray-21.08-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.3-cpeGNU-21.08-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeAO
lines 1-11
```

eb -S GROMACS | less

LUMI

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

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

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

```
* [x] $CFG5/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/gzip/gzip-1.12-cpeCray-22.08.eb (module: gzip/1.12-cpeCray-22.08)
* [x] $CFG5/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] $CFG5/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/ncurses/ncurses-6.2-cpeCray-22.08.eb (module: ncurses/6.2-cpeCray-22.08)
* [x] $CFG5/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/gettext/gettext-0.21-cpeCray-22.08-minimal.eb (module: gettext/0.21-cpeCray-22.08-minimal)
* [x] $CFG5/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] $CFG5/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] $CFG5/mgmt/ebrepo_files/LUMI-22.08/LUMI-C/Boost/Boost-1.79.0-cpeCray-22.08.eb (module: Boost/1.79.0-cpeCray-22.08)
* [ ] $CFG5/LUMI-EasyBuild-contrib/easybuild/easyconfigs/p/PLUMED/PLUMED-2.8.0-cpeCray-22.08.eb (module: PLUMED/2.8.0-cpeCray-22.08)
* [ ] $CFG5/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeCray-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)
```

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

```
== 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
```

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

== 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
```

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

```
== preparing...
== ... (took 5 secs)
== configuring...
== ... (took 35 secs)
== building...
== ... (took 2 mins 37 secs)
== testing [skipped]
== installing...
== ... (took 5 secs)
== taking care of extensions...
== creating build dir, resetting environment...
== starting iteration #1 ...
== unpacking...
== ... (took 1 secs)
== patching...
== preparing...
== ... (took 3 secs)
== configuring...
== ... (took 33 secs)
== building...
== ... (took 2 mins 36 secs)
lines 39-59
```

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

```
== testing [skipped]
== installing...
== ... (took 3 secs)
== taking care of extensions...
== creating build dir, resetting environment...
== starting iteration #2 ...
== unpacking...
== ... (took 1 secs)
== patching...
== preparing...
== ... (took 3 secs)
== configuring...
== ... (took 32 secs)
== building...
== ... (took 2 mins 35 secs)
== testing [skipped]
== installing...
== ... (took 3 secs)
== taking care of extensions...
== creating build dir, resetting environment...
== starting iteration #3 ...
lines 60-80
```

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

```
== 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
```

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

```
== 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)
```

Step 3: Install the software - Note

- Note: Sometimes the module does not show up immediately. This is because Lmod keeps a cache and fails to detect that the cache is outdated.
 - Remove `$HOME/.lmod.d/.cache`
`rm -rf $HOME/.lmod.d/.cache`
 - We've seen rare cases where internal Lmod data structures were corrupt and logging out and in again was needed
- 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.

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/easybuild/easyconfigs

More advanced work (3): Reproducibility

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

This is meant to have all information about how EasyBuild installed the application and to help in reproducing

EasyBuild training for advanced users and developers



- 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 klust.github.io/easybuild-tutorial.

Containers

This is about containers on LUMI-C!

- 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** is a myth
- **Full portability:** Not every container prepared on your Ubuntu or CentOS cluster or workstation will work on LUMI.
 - Containers that rely on certain hardware, kernel modules and/or kernel versions may fail.
 - Problem cases: High-performance networking (MPI) and GPU (driver version)
- **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-back to slower protocol that works

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 quota.
- **Productivity:** When not hit by the portability constraints, still useful to reproduce sophisticated user environments, e.g., Python.
- **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.
- You're the system administrator of your container, not LUST!

Managing containers

- Supported runtimes
 - Docker is **NOT** directly available from user environment
 - Singularity is natively available (as a system command) on the login and compute nodes
- Pulling containers
 - DockerHub and other registries (example: Julia container)
`singularity pull docker://julia`
 - Singularity uses flat (single) sif file for storing container and pull command makes the conversion
 - Be carefull: cache in `.singularity` dir or `$XDG_RUNTIME_DIR` can easily exhaust your storage quota for larger images

singularity pull docker://julia

LUMI

```
kulust@uan02.lumi.csc - ~/container-demo
kulust@uan02.lumi.csc - ~/container-demo (ssh)
[lumi][kulust@uan02-949 container-demo]$ singularity pull docker://julia
INFO: Converting OCI blobs to SIF format
WARNING: 'nodev' mount option set on /tmp, it could be a source of failure during build process
INFO: Starting build...
Getting image source signatures
Copying blob a603fa5e3b41 done
Copying blob d73274d00911 done
Copying blob a62e801f4e68 done
Copying blob e94234493940 done
Copying config 69901c71a8 done
Writing manifest to image destination
Storing signatures
2022/11/18 10:30:45 info unpack layer: sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2
2022/11/18 10:30:45 warn xattr{/etc/gshadow} ignoring ENOTSUP on setattr "user.rootlesscontainers"
2022/11/18 10:30:45 warn xattr{/tmp/build-temp-466722546/rootfs/etc/gshadow} destination filesystem does not support xattrs, further warnings will be suppressed
2022/11/18 10:30:47 info unpack layer: sha256:d73274d00911696ee92745d24351073840637e85f3e788218163281a28997d0e
2022/11/18 10:30:47 warn xattr{/var/cache/apt/archives/partial} ignoring ENOTSUP on setattr "user.rootlesscontainers"
2022/11/18 10:30:47 warn xattr{/tmp/build-temp-466722546/rootfs/var/cache/apt/archives/partial} destination filesystem does not support xattrs, further warnings will be suppressed
2022/11/18 10:30:47 info unpack layer: sha256:a62e801f4e686d7fb8b582e13cd504fb0055540c787fe253076fc61caa4f857b
2022/11/18 10:30:47 warn xattr{/usr/local/julia/LICENSE.md} ignoring ENOTSUP on setattr "user.rootlesscontainers"
2022/11/18 10:30:47 warn xattr{/tmp/build-temp-466722546/rootfs/usr/local/julia/LICENSE.md} destination filesystem does not support xattrs, further warnings will be suppressed
```

singularity pull docker://julia

LUMI

```
kulust@uan02.lumi.csc - ~/container-demo
kulust@uan02.lumi.csc - ~/container-demo (ssh) #1 + 

2022/11/18 10:30:51 warn rootless{/usr/local/julia/lib/julia/liblibsitesparseconfig.so} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:51 warn rootless{/usr/local/julia/lib/julia/liblibsitesparseconfig.so.5} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:51 warn rootless{/usr/local/julia/lib/julia/libumfpack.so} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:51 warn rootless{/usr/local/julia/lib/julia/libumfpack.so.5} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:51 warn rootless{/usr/local/julia/lib/julia/libunwind.so} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:51 warn rootless{/usr/local/julia/lib/julia/libunwind.so.8} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:51 warn rootless{/usr/local/julia/lib/julia/libuv.so} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:51 warn rootless{/usr/local/julia/lib/julia/libuv.so.2} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:51 warn rootless{/usr/local/julia/lib/julia/libz.so} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:51 warn rootless{/usr/local/julia/lib/julia/libz.so.1} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:53 warn rootless{/usr/local/julia/lib/libjulia.so} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:53 warn rootless{/usr/local/julia/lib/libjulia.so.1} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:54 info unpack layer: sha256:e94234493940ca90da25f22e3046ed2294bc30e45e5a669d91bb770dc3432ef2
INFO: Creating SIF file...
[kulust@uan02-950 container-demo]$
```

singularity pull docker://julia

LUMI

```
kulust@uan02.lumi.csc - ~/.singularity
kulust@uan02.lumi.csc - ~/.singularity (ssh)

attr "user.rootlesscontainers"
2022/11/18 10:30:51 warn rootless{usr/local/julia/lib/julia/libz.so.1} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:53 warn rootless{usr/local/julia/lib/libjulia.so} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:53 warn rootless{usr/local/julia/lib/libjulia.so.1} ignoring (usually) harmless EPERM on setxattr "user.rootlesscontainers"
2022/11/18 10:30:54 info unpack layer: sha256:e94234493940ca90da25f22e3046ed2294bc30e45e5a669d91bb770dc3432ef2
INFO: Creating SIF file...
[lumi][kulust@uan02-950 container-demo]$ cd ~/.singularity
[lumi][kulust@uan02-951 .singularity]$ ls -la
total 12
drwx----- 3 kulust pepr_kulust 4096 Nov 18 10:30 .
drwx----- 17 kulust pepr_kulust 4096 Nov 18 10:30 ..
drwx----- 8 kulust pepr_kulust 4096 Nov 18 10:30 cache
[lumi][kulust@uan02-952 .singularity]$ du -h
4.0K ./cache/shub
172M ./cache/blob/blobs/sha256
172M ./cache/blob/blobs
172M ./cache/blob
4.0K ./cache/net
4.0K ./cache/oras
4.0K ./cache/library
162M ./cache/oci-tmp
333M ./cache
333M .
[lumi][kulust@uan02-953 .singularity]$
```

Managing containers (2)

- Building containers
 - Containers cannot be built on LUMI
 - You should either pull or copy containers from outside
 - Singularity can build from existing (base) container
 - We plan to provide a set of base LUMI images



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_BINDPATH` environment variable to mount other host directories (like `/projappl` or `/appl`)

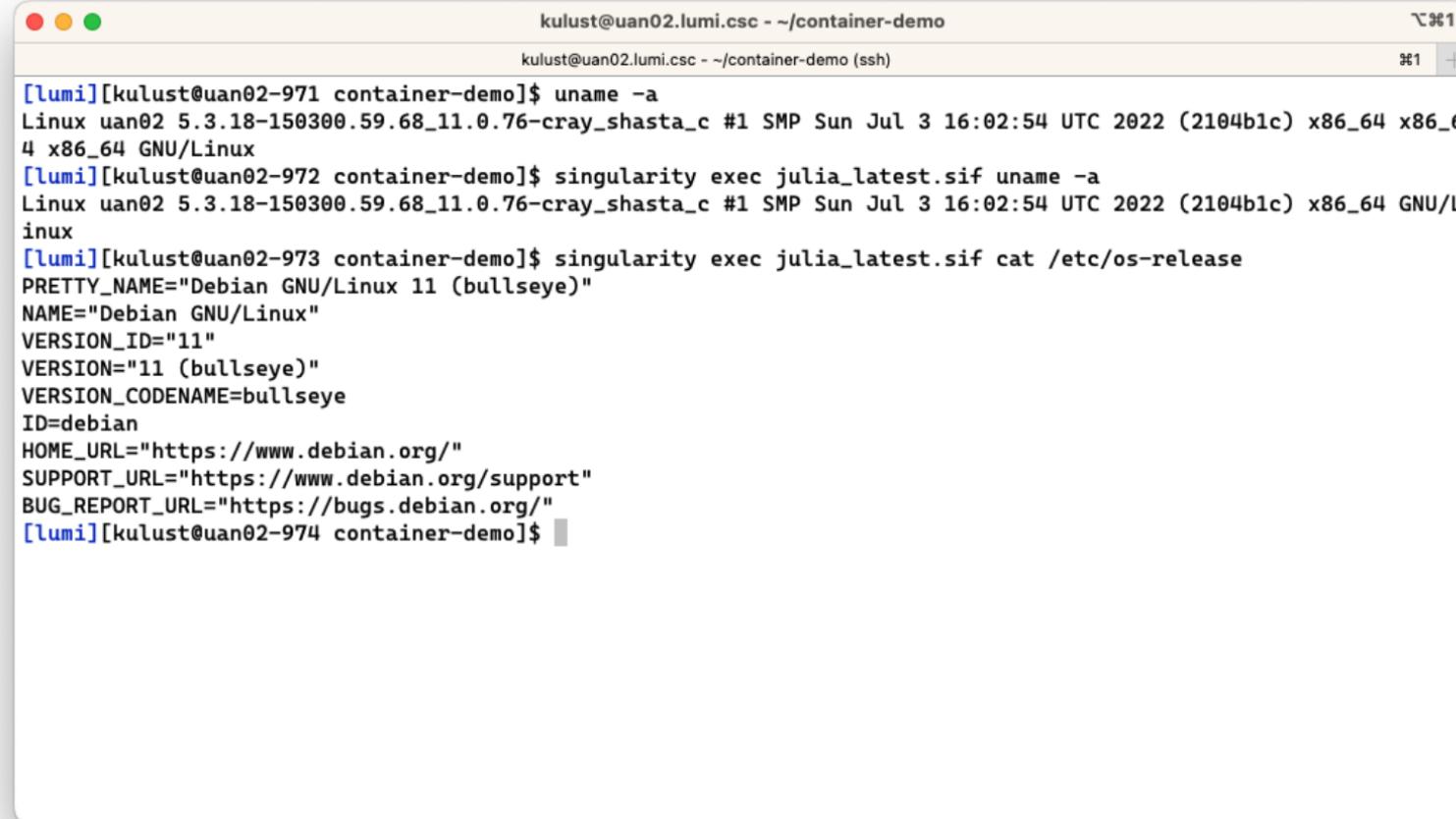
singularity shell julia_latest.sif

LUMI

```
kulust@uan02.lumi.csc - ~/container-demo
kulust@uan02.lumi.csc - ~/container-demo (ssh)
[lumi][kulust@uan02-965 container-demo]$ ls /opt
admin-pe AMD cray esmi modulefiles rocm rocm-5.0.2 slingshot
[lumi][kulust@uan02-966 container-demo]$ singularity shell julia_latest.sif
Singularity> ls /opt
Singularity> cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 11 (bullseye)"
NAME="Debian GNU/Linux"
VERSION_ID="11"
VERSION="11 (bullseye)"
VERSION_CODENAME=bullseye
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@uan02-967 container-demo]$
```

```
singularity exec julia_latest.sif uname -a
```

LUMI



A terminal window titled "kulust@uan02.lumi.csc - ~/container-demo" with a tab labeled "kulust@uan02.lumi.csc - ~/container-demo (ssh)". The window contains the following text:

```
[lumi] [kulust@uan02-971 container-demo]$ uname -a
Linux uan02 5.3.18-150300.59.68_11.0.76-cray_shasta_c #1 SMP Sun Jul 3 16:02:54 UTC 2022 (2104b1c) x86_64 x86_6
4 x86_64 GNU/Linux
[lumi] [kulust@uan02-972 container-demo]$ singularity exec julia_latest.sif uname -a
Linux uan02 5.3.18-150300.59.68_11.0.76-cray_shasta_c #1 SMP Sun Jul 3 16:02:54 UTC 2022 (2104b1c) x86_64 GNU/L
inux
[lumi] [kulust@uan02-973 container-demo]$ singularity exec julia_latest.sif cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 11 (bullseye)"
NAME="Debian GNU/Linux"
VERSION_ID="11"
VERSION="11 (bullseye)"
VERSION_CODENAME=bullseye
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
[lumi] [kulust@uan02-974 container-demo]$
```

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

LUMI

The screenshot shows a terminal window titled "kulust@uan02.lumi.csc - ~/container-demo". The window contains the command "singularity run julia_latest.sif" followed by the Julia 1.8.2 startup banner. Below this, the "julia>" prompt is shown, followed by the contents of the "docker-entrypoint.sh" script.

```
kulust@uan02.lumi.csc - ~/container-demo  
kulust@uan02.lumi.csc - ~/container-demo (ssh)  
[lumi][kulust@uan02-974 container-demo]$ singularity run julia_latest.sif  
Documentation: https://docs.julialang.org  
Type "?" for help, "]?" for Pkg help.  
Version 1.8.2 (2022-09-29)  
Official https://julialang.org/ release  
|__/  
  
julia>  
[lumi][kulust@uan02-975 container-demo]$ singularity inspect --runscript julia_latest.sif  
#!/bin/sh  
OCI_ENTRYPOINT='docker-entrypoint.sh'  
OCI_CMD='julia'  
CMDLINE_ARGS=""  
# prepare command line arguments for evaluation  
for arg in "$@"; do  
    CMDLINE_ARGS="${CMDLINE_ARGS} \"$arg\""  
done  
  
# ENTRYPOINT only - run entrypoint plus args  
if [ -z "$OCI_CMD" ] && [ -n "$OCI_ENTRYPOINT" ]; then  
    if [ $# -gt 0 ]; then  
        SINGULARITY OCI_RUN="${OCI_ENTRYPOINT} ${CMDLINE_ARGS}"  
    else  
        SINGULARITY OCI_RUN="${OCI_ENTRYPOINT}"
```

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

- 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 EasyBuild repository documentation when available
 - Provides basic mount points for using host MPI in the container setting
`SINGULARITY_BIND` and `SINGULARITY_LD_LIBRARY_PATH`

Environment enhancements (2)

- **lumi-vnc** (LUMI and CrayEnv software stacks)
 - Provides basic VNC virtual desktop for interacting with graphical interfaces via a web browser or VNC client
- **lumi-container-wrapper** (LUMI and CrayEnv software stacks)
 - Provides wrappers to encapsulate your custom environment in the container
 - Supports conda and pip environments
 - Helps with quota on the number of files in your project and I/O performance
 - Python provided by the cray-python module (so there is an optimised NumPy etc.)

lumi-container-wrapper (1)

LUMI

lumi-container-wrapper (2)

LUMI

```
kulust@uan02.lumi.csc - ~/Tykky-demo
kulust@uan02.lumi.csc - ~/Tykky-demo (ssh)
[lumi][kulust@uan02-906 Tykky-demo]$ module load LUMI/22.08 lumi-container-wrapper
[lumi][kulust@uan02-907 Tykky-demo]$ conda-containerize new --prefix ./conda-cont-1 env.yaml
[ INFO ] Constructing configuration
[ INFO ] Using /tmp/kulust/cw-NB1DJB as temporary directory
[ INFO ] Fetching container docker://opensuse/leap:15.2
[ INFO ] Running installation script
[ INFO ] Using miniconda version Miniconda3-latest-Linux-x86_64
[ INFO ] Installing miniconda
=====
PREFIX=/LUMI_CONTAINER/miniconda

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
  in Miniconda3: /LUMI_CONTAINER/miniconda
=====
```

lumi-container-wrapper (3)

LUMI

```
kulust@uan02.lumi.csc - ~/Tykky-demo
kulust@uan02.lumi.csc - ~/Tykky-demo (ssh) 2 1 +
```

directories of packages that are compatible with the Python interpreter
in Miniconda3: /LUMI_CONTAINER/miniconda

```
=====
[ INFO ] Creating env, full log in /tmp/kulust/cw-NB1DJB/build.log
=====
```

Executing transaction: ...working... done

```
#  
# To activate this environment, use  
#  
#     $ conda activate env1  
#  
# To deactivate an active environment, use  
#  
#     $ conda deactivate
=====
```

```
[ 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.
[=====] 38060/38060 100%
```

Exportable Squashfs 4.0 filesystem, gzip compressed, data block size 131072

lumi-container-wrapper (4)

LUMI

```
kulust@uan02.lumi.csc - ~/Tykky-demo
kulust@uan02.lumi.csc - ~/Tykky-demo (ssh) ⌘2
⌘1 +
```

```
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 529524.46 Kbytes (517.11 Mbytes)
    34.01% of uncompressed filesystem size (1556743.92 Kbytes)
Inode table size 417949 bytes (408.15 Kbytes)
    23.37% of uncompressed inode table size (1788760 bytes)
Directory table size 585179 bytes (571.46 Kbytes)
    41.08% of uncompressed directory table size (1424409 bytes)
Number of duplicate files found 5649
Number of inodes 38642
Number of files 28432
Number of fragments 1752
Number of symbolic links 4804
Number of device nodes 0
Number of fifo nodes 0
Number of socket nodes 0
Number of directories 5406
Number of ids (unique uids + gids) 1
Number of uids 1
```

lumi-container-wrapper (5)

LUMI

```
kulust@uan02.lumi.csc - ~/Tykky-demo
kulust@uan02.lumi.csc - ~/Tykky-demo (ssh)

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 529524.46 Kbytes (517.11 Mbytes)
    34.01% of uncompressed filesystem size (1556743.92 Kbytes)
Inode table size 417949 bytes (408.15 Kbytes)
    23.37% of uncompressed inode table size (1788760 bytes)
Directory table size 585179 bytes (571.46 Kbytes)
    41.08% of uncompressed directory table size (1424409 bytes)
Number of duplicate files found 5649
Number of inodes 38642
Number of files 28432
Number of fragments 1752
Number of symbolic links 4804
Number of device nodes 0
Number of fifo nodes 0
Number of socket nodes 0
Number of directories 5406
Number of ids (unique uids + gids) 1
Number of uids 1
```

lumi-container-wrapper (6)

LUMI

```
kulust@uan02.lumi.csc - ~/Tykky-demo
kulust@uan02.lumi.csc - ~/Tykky-demo (ssh) ⌘1 + ⌘2

[lumi][kulust@uan02-908 Tykky-demo]$ ls conda-cont-1/
_bin bin common.sh container.sif img.sqfs share
[lumi][kulust@uan02-909 Tykky-demo]$ ls conda-cont-1/bin
2to3      jupyter-kernelspec  openssl        tput
2to3-3.8   jupyter-migrate    pip           tset
captoinfo  jupyter-run       pip3          unlzma
clear      jupyter-troubleshoot pydoc         unxz
c_rehash   jupyter-trust     pydoc3         wheel
curve_keygen list-packages   pydoc3.8      wish
_debug_exec lzcat            pygmentize   wish8.6
_debug_shell lzcmp           python        x86_64-conda_cos6-linux-gnu-ld
f2py       lzdiff           python3       x86_64-conda-linux-gnu-ld
f2py3     lzgrep            python3.8    xz
f2py3.8   lzfgrep          python3.8-config xzcat
idle3     lzgrep            python3-config  xzcmp
idle3.8   lzless           reset         xzdec
infocmp   lzma             sqlite3      xzdiff
infotocap lzmadec          sqlite3_analyzer xzegrep
ipython   lzmainfo         tabs         xzfgrep
ipython3  lzmore           tclsh        xzgrep
jsonschema ncurses6-config  tclsh8.6    xzless
jupyter   ncursesw6-config   tic          xzmore
```

lumi-container-wrapper (7)

LUMI

```
kulust@uan02.lumi.csc - ~/Tykky-demo/conda-cont-1/bin
kulust@uan02.lumi.csc - ~/Tykky-demo/conda-cont-1/bin (ssh) ⌘2 ⌘1 +
```

_debug_exec	lzcat	pygmentize	wish8.6
_debug_shell	lzcmp	python	x86_64-conda_cos6-linux-gnu-ld
f2py	lzdiff	python3	x86_64-conda-linux-gnu-ld
f2py3	lzegrep	python3.8	xz
f2py3.8	lzfgrep	python3.8-config	xzcat
idle3	lzgrep	python3-config	xzcmp
idle3.8	lzless	reset	xzdec
infocmp	lzma	sqlite3	xzdiff
infotocap	lzmadec	sqlite3_analyzer	xzegrep
ipython	lzmainfo	tabs	xzfgrep
ipython3	lzmore	tclsh	xzgrep
jsonschema	ncurses6-config	tclsh8.6	xzless
jupyter	ncursesw6-config	tic	xzmore
jupyter-kernel	noglob	toe	

```
[lumi] [kulust@uan02-910 Tykky-demo]$  
[lumi] [kulust@uan02-910 Tykky-demo]$ cd conda-cont-1/bin  
[lumi] [kulust@uan02-911 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  
>>>
```

Container limitations on LUMI-C

- Containers use the host's operating system kernel which may be different from your system.
- 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.
- Building containers is not supported on LUMI due to security concerns.