

# LUMI

A white wolf is the central focus, standing in a futuristic, blue-toned digital environment. The background is filled with vertical light beams, floating particles, and a grid-like pattern, creating a high-tech, cybernetic atmosphere. The wolf is looking slightly to the right of the camera.

**LUMI Software Stacks**

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

April 2024

# What this talk is about...

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

LUMI

Lmod

EASYBUILD<sup>™</sup>  
building software with ease



# Design considerations

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

# The LUMI solution

L U M I

- 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



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

# Organisation: Software stacks

- **CrayEnv:** Cray environment with some additional tools pushed in through EasyBuild
- **LUMI** stacks, each one corresponding to a particular release of the PE
  - Work with the Cray PE modules, but accessed through a replacement for the PrgEnv-\* modules
  - Tuned versions for the 3 ~~4~~ types of hardware: zen2 (login, large memory nodes), zen3 (LUMI-C compute nodes), ~~zen2 + NVIDIA GPU (visualisation partition)~~, zen3 + MI250X (LUMI-G GPU partition)
- **spack:** Install software with Spack using compilers from the PE
  - Offered as-is for users who know Spack, but we do not do development in Spack
- Far future: Stack based on common EB foss 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 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

LUMI

## The LUMI software stack

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

# 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/5.4.8-cpeGNU-23.09` : 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/3.27.7` : Will tell you which module contains CMake and how to load it





## module spider (command) (2)

```
kulust@uan03.lumi.csc - ~
kulust@uan03.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-archive: Autoconf-archive/2021.02.19 (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-21.12, Blosc/1.21.1-cpeAOCC-22.08, ...
  Blosc is an extremely fast, multi-threaded, meta-compressor library

Boost: Boost/1.77.0-cpeAOCC-21.12, Boost/1.77.0-cpeCray-21.12, Boost/1.77.0-cpeGNU-21.12, ...
  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)

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

zlib: zlib/1.2.11-cpeA OCC-21.12, zlib/1.2.11-cpeCray-21.12, zlib/1.2.11-cpeGNU-21.12, ...
      Free lossless data-compression library, not covered by any patents.

zstd: zstd/1.5.0-cpeA OCC-21.12, zstd/1.5.0-cpeCray-21.12, zstd/1.5.0-cpeGNU-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][kulust@uan03-1002 ~]$
```

```
kulust@uan04.lumi.csc - ~
kulust@uan04.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-21.12
  gnuplot/5.4.3-cpeAOCC-22.08
  gnuplot/5.4.3-cpeCray-21.12
  gnuplot/5.4.3-cpeCray-22.06
  gnuplot/5.4.3-cpeCray-22.08
  gnuplot/5.4.3-cpeGNU-21.12
  gnuplot/5.4.3-cpeGNU-22.06
  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
```

lines 1-22

## module spider gnuplot (2)

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

gnuplot/5.4.3-cpeCray-22.06
gnuplot/5.4.3-cpeCray-22.08
gnuplot/5.4.3-cpeGNU-21.12
gnuplot/5.4.3-cpeGNU-22.06
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-23.09
gnuplot/5.4.8-cpeAOCC-23.09
gnuplot/5.4.8-cpeGNU-23.09

-----
For detailed information about a specific "gnuplot" package (including how to load the module
s) 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/5.4.8-cpeGNU-23.09

-----
[lumi][kulust@uan04-1002 ~]$
```



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

-----
CMake:
-----

Versions:
  CMake/3.22.2 (E)
  CMake/3.23.2 (E)
  CMake/3.24.0 (E)
  CMake/3.25.2 (E)
  CMake/3.27.7 (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/3.27.7

-----
lines 1-21
```

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

-----
gnuplot: gnuplot/5.4.8-cpeGNU-23.09
-----

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/5.4.8-cpeGNU-23.09" module is available to load.

LUMI/23.09  partition/C
LUMI/23.09  partition/G
LUMI/23.09  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
```

```
kulust@uan04.lumi.csc - ~
kulust@uan04.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@uan04-1003 ~]$
```

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

$ module spider CMake/3.27.7
-----
[lumi][kulust@uan03-1002 ~]$ module spider CMake/3.27.7
-----
CMake: CMake/3.27.7 (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 locatio
n in the software hierarchy.

    buildtools/23.09 (LUMI/23.09 partition/L)
    buildtools/23.09 (LUMI/23.09 partition/G)
    buildtools/23.09 (LUMI/23.09 partition/C)
    buildtools/23.09 (CrayEnv)

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

[lumi][kulust@uan03-1003 ~]$
```

# 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

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

-----

The following modules match your search criteria: "https"

-----

cURL: cURL/7.78.0-cpeAOCC-21.12, cURL/7.78.0-cpeCray-21.12, cURL/7.78.0-cpeGNU-21.12, ...
      Command line tool and library for transferring data with URLs.

wget: wget/1.21.2-cpeAOCC-21.12, wget/1.21.2-cpeCray-21.12, wget/1.21.2-cpeGNU-21.12, ...
      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
```

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

cURL: cURL/7.78.0-cpeAOCC-21.12, cURL/7.78.0-cpeCray-21.12, cURL/7.78.0-cpeGNU-21.12, ...
  Command line tool and library for transferring data with URLs.

wget: wget/1.21.2-cpeAOCC-21.12, wget/1.21.2-cpeCray-21.12, wget/1.21.2-cpeGNU-21.12, ...
  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][kulust@uan03-1010 ~]$
```



# 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, so any change to modules that are loaded by these modules will be wiped out.

kulust@uan02.lumi.csc - ~

⌘1

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

⌘1

## ----- EasyBuild managed systemwide software -----

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

## ----- HPE-Cray PE modules -----

PrgEnv-amd/8.3.3		cray-mpixlate/1.0.0.6	
PrgEnv-amd/8.4.0	(D)	cray-mpixlate/1.0.1.10	
PrgEnv-aocc/8.3.3		cray-mpixlate/1.0.2	(D)
PrgEnv-aocc/8.4.0	(D)	cray-mrnet/5.0.4	
PrgEnv-cray-amd/8.3.3		cray-mrnet/5.1.1	(D)
PrgEnv-cray-amd/8.4.0	(D)	cray-openshmemx/11.5.6	
PrgEnv-cray/8.3.3		cray-openshmemx/11.5.7	
PrgEnv-cray/8.4.0	(L,D)	cray-openshmemx/11.5.8	
PrgEnv-gnu-amd/8.3.3		cray-openshmemx/11.6.1	(D)
PrgEnv-gnu-amd/8.4.0	(D)	cray-pals/1.2.0	
PrgEnv-gnu/8.3.3		cray-pals/1.2.5	
PrgEnv-gnu/8.4.0	(D)	cray-pals/1.2.11	
amd-mixed/5.2.3		cray-pals/1.2.12	(D)
amd/5.2.3	(5.0.2:5.2.0)	cray-parallel-netcdf/1.12.2.5	
aocc-mixed/3.2.0		cray-parallel-netcdf/1.12.3.1	

## module av (2)

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

aocc/3.2.0
atp/3.14.13
atp/3.14.16
atp/3.14.18
atp/3.15.1 (D)
cce-mixed/14.0.2
cce-mixed/15.0.0
cce-mixed/15.0.1
cce-mixed/16.0.1 (D)
cce/14.0.2
cce/15.0.0
cce/15.0.1
cce/16.0.1 (L,D)
cray-R/4.1.3.1
cray-R/4.2.1.1
cray-R/4.2.1.2 (D)
cray-ccdb/4.12.13
cray-ccdb/5.0.1 (D)
cray-cti/2.15.13
cray-cti/2.15.14
cray-cti/2.16.0
cray-cti/2.17.1
cray-parallel-netcdf/1.12.3.3
cray-parallel-netcdf/1.12.3.7 (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 (D)
cray-python/3.9.12.1
cray-python/3.9.13.1
cray-python/3.10.10 (D)
cray-stat/4.11.12
cray-stat/4.11.13
cray-stat/4.12.1 (D)
craype/2.7.17
craype/2.7.19
craype/2.7.20
craype/2.7.23 (L,D)
craypkg-gen/1.3.25
craypkg-gen/1.3.28
craypkg-gen/1.3.30 (D)
gcc-mixed/11.2.0

lines 23-44
```

## module av (3)

kulust@uan02.lumi.csc - ~

⌘1

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

⌘1

```
cray-cti/2.17.2
cray-cti/2.18.1 (D)
cray-dsmml/0.2.2 (L)
cray-dyninst/12.1.1
cray-dyninst/12.3.0 (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 (D)
cray-hdf5-parallel/1.12.1.5
cray-hdf5-parallel/1.12.2.1
cray-hdf5-parallel/1.12.2.3
cray-hdf5-parallel/1.12.2.7 (D)
cray-hdf5/1.12.1.5
cray-hdf5/1.12.2.1
cray-hdf5/1.12.2.3
cray-hdf5/1.12.2.7 (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
gcc-mixed/12.2.0 (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 (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 (D)
perftools-base/22.06.0
perftools-base/22.12.0
perftools-base/23.03.0
perftools-base/23.09.0 (L,D)
perftools-lite-events
perftools-lite-gpu
perftools-lite-hbm
perftools-lite-loops
perftools-lite
```

# module av (4)

```

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

cray-libsci/22.08.1.1          perftools-preload
cray-libsci/22.12.1.1          perftools
cray-libsci/23.02.1.1          rocm/5.2.3 (D:5.0.2:5.2.0)
cray-libsci/23.09.1.1 (L,D)  sanitizers4hpc/1.0.1
cray-libsci_acc/22.08.1.1      sanitizers4hpc/1.0.4
cray-libsci_acc/22.12.1.1      sanitizers4hpc/1.1.1 (D)
cray-libsci_acc/23.09.1.1 (D)  valgrind4hpc/2.12.10
cray-mpich-abi/8.1.27          valgrind4hpc/2.12.11
cray-mpich/8.1.27 (L,8.1.18:8.1.23:8.1.25) valgrind4hpc/2.13.1 (D)

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

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

----- Software stacks -----
CrayEnv (S)  LUMI/22.12 (S)  LUMI/23.09 (S)  spack/22.08-2  spack/23.03-2

lines 67-88

```

kulust@uan02.lumi.csc - ~

~#1

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

#1 +

LUMI/22.08 (S,D)    LUMI/23.03 (S)    spack/22.08    spack/23.03    spack/23.09 (D)

----- Modify the module display style -----

ModuleColour/off	(S)	ModuleLabel/label	(S,L,D)	ModuleStyle/default
ModuleColour/on	(S,D)	ModuleLabel/PEhierarchy	(S)	ModuleStyle/reset (D)
ModuleExtensions/hide	(S)	ModuleLabel/system	(S)	
ModuleExtensions/show	(S,D)	ModulePowerUser/LUMI	(S)	

----- System initialisation -----

init-lumi/0.2 (S,L)

----- Non-PE HPE-Cray modules -----

chapel/1.28.0  
 cray-lustre-client-ofed/2.15.0.4\_rc2\_cray\_179\_gb09cfbe-2.4\_17.8\_\_gb09cfbe6c5.shasta  
 dvs/2.15\_4.4.234-2.4\_49.1\_\_gfd0d6e85  
 libfabric/1.15.2.0 (L)  
 rocm/5.2.3 (5.0.2:5.2.0)  
 xpmem/2.5.2-2.4\_3.50\_\_gd0f7936.shasta (L)

----- This is a list of module extensions "module --nx avail ..." to not show. -----

rclone (E)    restic (E)    s3cmd (E)

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

----- This is a list of module extensions "module --nx avail ..." to not show.
-----
  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 information on searching modules.
If then you still miss software, contact LUMI User Support via https://lumi-supercomputer.eu/user-support/need-help/.

[lumi][kulust@uan02-1004 ~]$
```



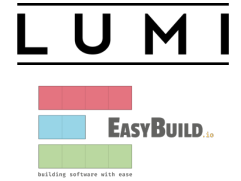
# Changing how the module list is displayed



- 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 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`
- Show some hidden modules with `ModulePowerUser/LUMI`
  - This will also show undocumented/unsupported modules!
- More customisation possible via LMOD environment variables

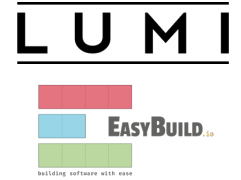


# Installing software on HPC systems



- 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



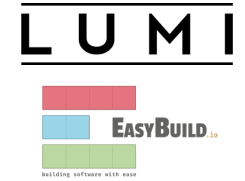
- Fully integrated in the LUMI software stack
  - Load the LUMI module and modules should appear in your module view
  - EasyBuild-user module to install packages in your user space
  - Will use existing modules for dependencies if those are already on the system or in your personal/project stack
- EasyBuild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

# EasyBuild recipes - easyconfigs



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

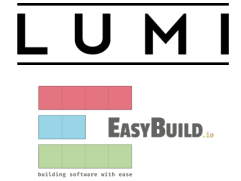
# The toolchain concept



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

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

# The toolchain concept (2)



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

# easyconfig names and module names

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



Name of the package



Version of the package



Toolchain name and version (missing for SYSTEM)



Additional information

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

# Installing

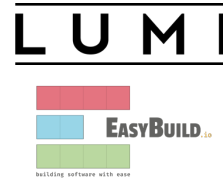
## Step 1: Where to install



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

# Installing

## Step 2: Configure the environment



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



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

**LUMI**

kulust@uan02.lumi.csc - ~

⌘2

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

#1

\*\*\*\*\*

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

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

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

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

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

EasyBuild configured to install software in the user tree at `/users/kulust/EasyBuild` for the LUMI/23.09 software stack for the LUMI/C partition.

- \* Software installation directory: `/users/kulust/EasyBuild/SW/LUMI-23.09/C`
  - \* Modules installation directory: `/users/kulust/EasyBuild/modules/LUMI/23.09/partition/C`
  - \* Repository: `/users/kulust/EasyBuild/ebrepo_files/LUMI-23.09/LUMI-C`
  - \* Work directory for builds and logs: `/run/user/327000143/easybuild`
- Clear work directory with `clear-eb`

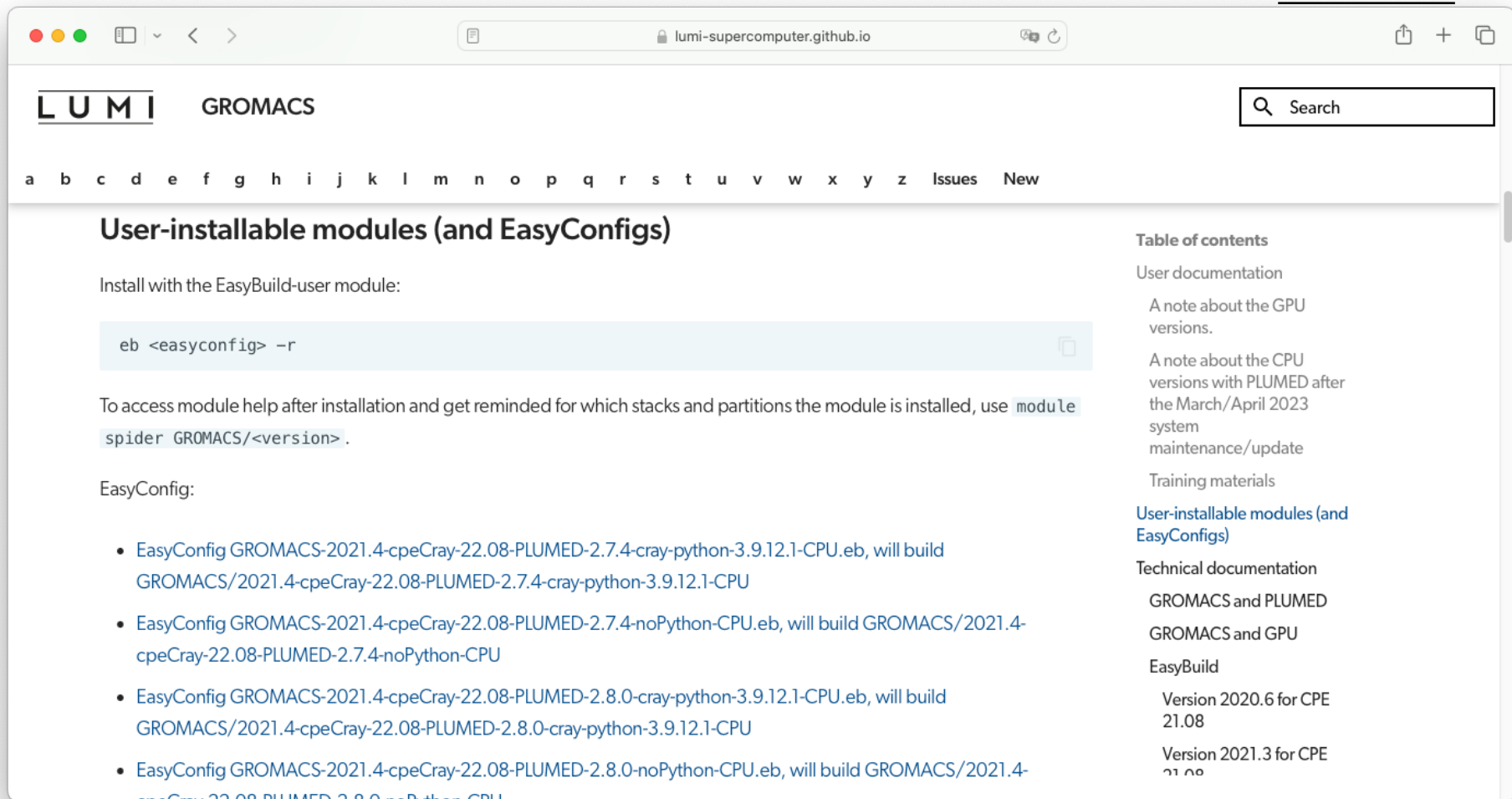
```
[lumi][kulust@uan02-1003 ~]$
```

# Installing

## Step 3: Install the software



- Let's, e.g., install GROMACS
  - Search if GROMACS build recipes are available:
    - Search the [LUMI Software Library](#) that lists all available software through EasyBuild.
    - Or on the command line:  
`eb --search GROMACS`  
`eb -S GROMACS`



LUMI GROMACS

Search

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

## User-installable modules (and EasyConfigs)

Install with the EasyBuild-user module:

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

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

EasyConfig:

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

### Table of contents

- User documentation
  - A note about the GPU versions.
  - A note about the CPU versions with PLUMED after the March/April 2023 system maintenance/update
  - Training materials
- User-installable modules (and EasyConfigs)
- Technical documentation
  - GROMACS and PLUMED
  - GROMACS and GPU
  - EasyBuild
    - Version 2020.6 for CPE 21.08
    - Version 2021.3 for CPE 21.08

eb --search GROMACS | less

LUMI

kulust@uan02.lumi.csc - -

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

⌂ #2

#1

```
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeCray-22.08-PLUMED-2.7.4-cray-python-3.9.12.1-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeCray-22.08-PLUMED-2.7.4-noPython-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-cray-python-3.9.12.1-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeCray-22.08-PLUMED-2.8.0-noPython-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeGNU-22.08-PLUMED-2.7.4-cray-python-3.9.12.1-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeGNU-22.08-PLUMED-2.7.4-noPython-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeGNU-22.08-PLUMED-2.8.0-cray-python-3.9.12.1-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.4-cpeGNU-22.08-PLUMED-2.8.0-noPython-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.6-cpeCray-22.08-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.6-cpeGNU-22.08-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.7-cpeCray-23.09-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.7-cpeGNU-23.09-CPU.eb
* /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/g/GROMACS/GROMACS-2021.7-cpeGNU-23.09-PLUMED-2.8.3-noPython-CPU.eb
```

Lines 1-13

kulust@uan02.lumi.csc - ~

⌘2

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

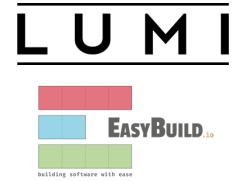
#1

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

Lines 1-22

# Installing

## Step 3: Install the software



- Let's, e.g., install GROMACS
  - Search if GROMACS build recipes are available:
    - Search the [LUMI Software Library](#) that lists all available software through EasyBuild.
    - Or on the command line:  
`eb --search GROMACS`  
`eb -S GROMACS`
  - Let's take GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb:  
`eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -D`



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

LUMI

kulust@uan02.lumi.csc --

⌘%2

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

#1

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

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

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

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



# Installing

## Step 3: Install the software



- Let's, e.g., install GROMACS
  - Search if GROMACS build recipes are available:
    - Search the [LUMI Software Library](#) that lists all available software through EasyBuild.
    - Or on the command line:  
`eb --search GROMACS`  
`eb -S GROMACS`
  - Let's take GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb:  
`eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -D`  
`eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -r`

kulust@uan02.lumi.csc - ~

⌵⌘2

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

⌘1

```
== Temporary log file in case of crash /run/user/327000143/easybuild/tmp/eb-_gplx801/easybuild-rk0zww73.l
og
== resolving dependencies ...
== processing EasyBuild easyconfig /appl/lumi/LUMI-EasyBuild-contrib/easybuild/easyconfigs/p/PLUMED/PLUME
D-2.9.0-cpeGNU-23.09-noPython.eb
== building and installing PLUMED/2.9.0-cpeGNU-23.09-noPython...
== fetching files...
== ... (took 4 secs)
== creating build dir, resetting environment...
== unpacking...
== ... (took 4 secs)
== patching...
== preparing...
== ... (took 8 secs)
== configuring...
== ... (took 1 min 17 secs)
== building...
== ... (took 3 mins 55 secs)
== testing...
== installing...
== ... (took 51 secs)
== taking care of extensions...
```

lines 1-20

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

kulust@uan02.lumi.csc ~

⌵%2

kulust@uan02.lumi.csc ~ (ssh)

%1

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

lines 21-40

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

L U M I

kulust@uan02.lumi.csc ~

⌵⌘2

kulust@uan02.lumi.csc ~ (ssh)

⌘1

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

lines 41-62

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

kulust@uan02.lumi.csc ~

⌘2

kulust@uan02.lumi.csc ~ (ssh)

#1

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

lines 63-84

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

kulust@uan02.lumi.csc ~

⌘2

kulust@uan02.lumi.csc ~ (ssh)

⌘1

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

lines 85-106

eb GROMACS-2022.5-cpeGNU-23.09-PLUMED-2.9.0-noPython-CPU.eb -r (6) L U M I

kulust@uan02.lumi.csc ~

⌘2

kulust@uan02.lumi.csc ~ (ssh)

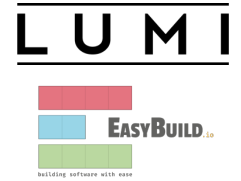
⌘1

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

lines 92-111/111 (END)

# Installing

## Step 3: Install the software



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

```
eb --search GROMACS
```

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

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

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

```
module avail GROMACS
```



# Installing

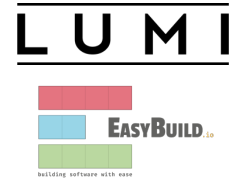
## Step 3: Install the software - Note

L U M I



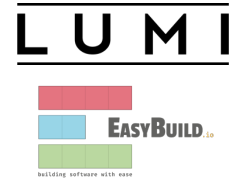
- Installing this way is 100% equivalent to an installation in the central software tree. The application is compiled in exactly the same way as we would do and served from the same file systems.
  - And you are in control of updates.
- Note: EasyBuild clears the Lmod user cache so in principle newly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

# More advanced work



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

# More advanced work (2): Repositories



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

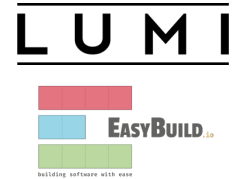
# More advanced work (3): Reproducibility



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

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

# EasyBuild tips&tricks



- **Updating version:** Often some trivial changes in the EasyConfig (.eb) file
  - Checksums may be annoying: Use `--ignore-checksums` with the `eb` command
- **Updating to a new toolchain:**
  - Be careful, it is more than changing one number
  - Versions of preinstalled dependencies should be changed and EasyConfig files of other dependencies also checked
- [LUMI Software Library](https://lumi-supercomputer.github.io/LUMI-EasyBuild-docs) at [lumi-supercomputer.github.io/LUMI-EasyBuild-docs](https://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

L U M I



- EasyBuild web site: [easybuild.io](https://easybuild.io)
- Generic EasyBuild training materials on [tutorial.easybuild.io](https://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](https://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!

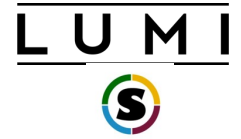
# What do containers not provide?



- **Full reproducibility of results** 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-down 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 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
- **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

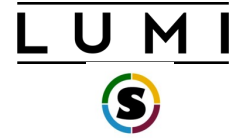
```
kulust@uan01.lumi.csc - ~/container-demo
kulust@uan01.lumi.csc - ~/container-demo (ssh)

[lumi][kulust@uan01-1012 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 34f65707cdc9 done
Copying blob 517972d95169 done
Copying blob 9e3ea8720c6d done
Copying blob bf4da5f2ad94 done
Copying config 4839902eb6 done
Writing manifest to image destination
Storing signatures
2023/05/12 17:18:31  info unpack layer: sha256:9e3ea8720c6de96cc9ad544dddc695a3ab73f5581c5d954e0504cc4f80fb5e5c
2023/05/12 17:18:31  warn xattr{/etc/gshadow} ignoring ENOTSUP on setxattr "user.rootlesscontainers"
2023/05/12 17:18:31  warn xattr{/tmp/build-temp-2626795503/rootfs/etc/gshadow} destination filesystem does not support xattrs, further warnings will be suppressed
2023/05/12 17:18:33  info unpack layer: sha256:bf4da5f2ad94273f80352cb6898e2347ef78a3570c60ee03d652a6123a571f70
2023/05/12 17:18:33  warn xattr{/var/cache/apt/archives/partial} ignoring ENOTSUP on setxattr "user.rootlesscontainers"
2023/05/12 17:18:33  warn xattr{/tmp/build-temp-2626795503/rootfs/var/cache/apt/archives/partial} destination filesystem does not support xattrs, further warnings will be suppressed
```

```
kulust@uan01.lumi.csc - ~/container-demo
kulust@uan01.lumi.csc - ~/container-demo (ssh)
PERM on setxattr "user.rootlesscontainers"
2023/05/12 17:18:36 warn rootless{usr/local/julia/lib/julia/libumfpack.so.5} ignoring (usually) harmless
EPERM on setxattr "user.rootlesscontainers"
2023/05/12 17:18:36 warn rootless{usr/local/julia/lib/julia/libunwind.so} ignoring (usually) harmless EP
ERM on setxattr "user.rootlesscontainers"
2023/05/12 17:18:36 warn rootless{usr/local/julia/lib/julia/libunwind.so.8} ignoring (usually) harmless
EPERM on setxattr "user.rootlesscontainers"
2023/05/12 17:18:36 warn rootless{usr/local/julia/lib/julia/libuv.so} ignoring (usually) harmless EPERM
on setxattr "user.rootlesscontainers"
2023/05/12 17:18:36 warn rootless{usr/local/julia/lib/julia/libuv.so.2} ignoring (usually) harmless EPER
M on setxattr "user.rootlesscontainers"
2023/05/12 17:18:36 warn rootless{usr/local/julia/lib/julia/libz.so} ignoring (usually) harmless EPERM o
n setxattr "user.rootlesscontainers"
2023/05/12 17:18:36 warn rootless{usr/local/julia/lib/julia/libz.so.1} ignoring (usually) harmless EPERM
on setxattr "user.rootlesscontainers"
2023/05/12 17:18:38 warn rootless{usr/local/julia/lib/libjulia.so} ignoring (usually) harmless EPERM on
setxattr "user.rootlesscontainers"
2023/05/12 17:18:38 warn rootless{usr/local/julia/lib/libjulia.so.1} ignoring (usually) harmless EPERM o
n setxattr "user.rootlesscontainers"
2023/05/12 17:18:39 info unpack layer: sha256:517972d951693e767dcac01bd8871495974d4bdab2446521630a3bb1c8
97d0fc
INFO: Creating SIF file...
[lumi][kulust@uan01-1013 container-demo]$
```

```
kulust@uan01.lumi.csc - ~/.singularity
kulust@uan01.lumi.csc - ~/.singularity (ssh)
2023/05/12 17:18:38 warn rootless{usr/local/julia/lib/libjulia.so.1} ignoring (usually) harmless EPERM o
n setattr "user.rootlesscontainers"
2023/05/12 17:18:39 info unpack layer: sha256:517972d951693e767dcac01bd8871495974d4bdab2446521630a3bb1c8
97d0fc
INFO: Creating SIF file...
[lumi][kulust@uan01-1013 container-demo]$ cd ~/.singularity
[lumi][kulust@uan01-1014 .singularity]$ ls -la
total 12
drwx----- 3 kulust pepr_kulust 4096 May 12 17:18 .
drwx----- 28 kulust pepr_kulust 4096 May 9 16:20 ..
drwx----- 8 kulust pepr_kulust 4096 May 12 17:18 cache
[lumi][kulust@uan01-1015 .singularity]$ du -h
4.0K    ./cache/shub
175M   ./cache/blob/blobs/sha256
175M   ./cache/blob/blobs
175M   ./cache/blob
4.0K   ./cache/net
4.0K   ./cache/oras
4.0K   ./cache/library
171M   ./cache/oci-tmp
346M   ./cache
346M   .
[lumi][kulust@uan01-1016 .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.
  - You should either pull or copy containers from outside
  - 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/23.09](#) module in CrayEnv and LUMI/23.09.
  - 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@uan02.lumi.csc - /scratch/project\_465000095/kulust/container-demo

⌘3

kulust@uan02.lumi.csc - /scratch/project\_465000095/kulust/container-demo (ssh)

#1

```
[lumi][kulust@uan02-1018 container-demo]$ ls /opt
admin-pe AMD cray esmi modulefiles rocm rocm-5.2.3 slingshot
[lumi][kulust@uan02-1019 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@uan02-1020 container-demo]$
```



singularity exec julia\_latest.sif uname -a

LUMI

kulust@uan02.lumi.csc - /scratch/project\_465000095/kulust/container-demo

⌘3

kulust@uan02.lumi.csc - /scratch/project\_465000095/kulust/container-demo (ssh)

#1 +

```
[lumi][kulust@uan02-1021 container-demo]$ uname -a
```

```
Linux uan02 5.14.21-150400.24.81_12.0.75-cray_shasta_c #1 SMP Thu Sep 7 00:12:59 UTC 2023 (1027017) x86_64  
x86_64 x86_64 GNU/Linux
```

```
[lumi][kulust@uan02-1022 container-demo]$ singularity exec julia_latest.sif uname -a
```

```
Linux uan02 5.14.21-150400.24.81_12.0.75-cray_shasta_c #1 SMP Thu Sep 7 00:12:59 UTC 2023 (1027017) x86_64  
GNU/Linux
```

```
[lumi][kulust@uan02-1023 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
```

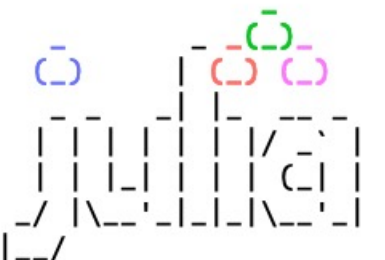
```
HOME_URL="https://www.debian.org/"
```

```
SUPPORT_URL="https://www.debian.org/support"
```

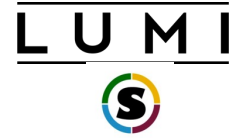
```
BUG_REPORT_URL="https://bugs.debian.org/"
```

```
[lumi][kulust@uan02-1024 container-demo]$
```

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

```
kulust@uan02.lumi.csc - /scratch/project_465000095/kulust/container-demo  3  
kulust@uan02.lumi.csc - /scratch/project_465000095/kulust/container-demo (ssh)  #1 +  
[lumi][kulust@uan02-1025 container-demo]$ singularity run julia_latest.sif  
 Documentation: https://docs.julialang.org  
Type "?" for help, "]?" for Pkg help.  
Version 1.10.2 (2024-03-01)  
Official https://julialang.org/ release  
  
julia>  
[lumi][kulust@uan02-1026 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 "$OCI_ENTRYPOINT" ]; then  
        set -- 'docker-entrypoint.sh' "$@"
```

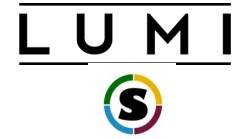
# Running containers on LUMI



- 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@uan04.lumi.csc - ~/Tykky-demo
kulust@uan04.lumi.csc - ~/Tykky-demo (ssh)

*          *          *          *          *          *  , / (      *
*          *          *          *          *          *  / |  |  /
*          *          *          *          *          *  \ |  |  \
*  *   *   The Supercomputer of the North *  | ` \ _ |  _ _ / _ |  |  |
*          **          *          *          *          *  \ _ _ _ \ _ _ _ ) \ _ _ )
.*****-----*****-----*****-----*****-----*****
| User guide and support   _____|
|          https://docs.lumi-supercomputer.eu
* |          https://lumi-supercomputer.eu/user-support
** `-----*****-----*****-----*****-----*****

[lumi][kulust@uan04-1001 ~]$ cd Tykky-demo/
[lumi][kulust@uan04-1002 Tykky-demo]$ ls
conda-cont-1  env.yml
[lumi][kulust@uan04-1003 Tykky-demo]$ cat env.yml
channels:
- conda-forge
dependencies:
- python=3.8.8
- scipy
- nglview

[lumi][kulust@uan04-1004 Tykky-demo]$ module load LUMI/22.12 lumi-container-wrapper
[lumi][kulust@uan04-1005 Tykky-demo]$
```

## lumi-container-wrapper (2)

L U M I

```
kulust@uan04.lumi.csc - ~/Tykky-demo
kulust@uan04.lumi.csc - ~/Tykky-demo (ssh)

[lumi][kulust@uan04-1004 Tykky-demo]$ module load LUMI/22.12 lumi-container-wrapper
[lumi][kulust@uan04-1005 Tykky-demo]$ conda-containerize new --prefix ./conda-cont-1 env.yml
[ INFO ] Constructing configuration
[ INFO ] Using /tmp/kulust/cw-YSVL4M as temporary directory
[ INFO ] Fetching container docker://opensuse/leap:15.4
[ INFO ] Running installation script
[ INFO ] Using miniconda version Miniconda3-latest-Linux-x86_64
[ INFO ] Installing miniconda
=====
PREFIX=/LUMI_TYKKY_4EJoer8/miniconda
REFIX=/LUMI_TYKKY_4EJoer8/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_TYKKY_4EJoer8/miniconda
=====
[ INFO ] Creating env, full log in /tmp/kulust/cw-YSVL4M/build.log
```



# lumi-container-wrapper (3)

```
kulust@uan04.lumi.csc - ~/Tykky-demo
kulust@uan04.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.
[=====\] 37447/37447 100%

Exportable Squashfs 4.0 filesystem, gzip compressed, data block size 131072
  compressed data, compressed metadata, compressed fragments,
  scipy-1.compressed xattrs, compressed ids3    | 63%
  duplicates are removed
Filesystem size 521302.16 Kbytes (509.08 Mbytes)
executin33.62% of uncompressed filesystem size (1550478.89 Kbytes)
Inode table size 408231 bytes (398.66 Kbytes) | 1%
  23.20% of uncompressed inode table size (1759978 bytes)
Directory table size 578457 bytes (564.90 Kbytes)
  41.52% of uncompressed directory table size (1393078 bytes)
Number of duplicate files found 5545
Number of inodes 37685
Number of files 27719
Number of fragments 1698
Number of symbolic links 4872
Number of device nodes 0
```



# lumi-container-wrapper (4)

```
kulust@uan04.lumi.csc - ~/Tykky-demo
kulust@uan04.lumi.csc - ~/Tykky-demo (ssh)

Directory table size 578457 bytes (564.90 Kbytes)
    41.52% of uncompressed directory table size (1393078 bytes)
Number of duplicate files found 5545
Number of inodes 37685
Number of files 27719
Number of fragments 1698
Number of symbolic links 4872
Number of device nodes 0
Number of fifo nodes 0
Number of socket nodes 0
Number of directories 5094
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: 263s
[ 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:$PATH"
[lumi][kulust@uan04-1006 Tykky-demo]$
```

# lumi-container-wrapper (5)

```
kulust@uan04.lumi.csc - ~/Tykky-demo
kulust@uan04.lumi.csc - ~/Tykky-demo (ssh)

[ 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@uan04-1006 Tykky-demo]$ ls conda-cont-1/
_bin bin common.sh container.sif img.sqfs share

[lumi][kulust@uan04-1007 Tykky-demo]$ ls conda-cont-1/bin
2to3          ipython3      lzmadec       python3.8     wish
2to3-3.8      jupyter       lzmainfo      python3.8-config wish8.6
captaininfo   jupyter-kernel lzmore         python3-config x86_64-conda_cos6-linux-gnu-ld
clear         jupyter-kernelspec ncurses6-config reset          x86_64-conda-linux-gnu-ld
c_rehash      jupyter-migrate ncursesw6-config sqlite3        xz
curve_keygen  jupyter-run   nglview       sqlite3_analyzer xzcat
_debug_exec   jupyter-troubleshoot normalizer     tabs          xzcmp
_debug_shell  list-packages openssl        tclsh         xzdec
f2py          lzcat         pip           tclsh8.6     xzdiff
f2py3         lzcmp         pip3          tic           xzegrep
f2py3.8       lzdiff        pydoc         toe           xzfgrep
idle3         lzgrep        pydoc3        tput         xzgrep
idle3.8       lzfgrep       pydoc3.8      tset         xzless
infocmp       lzgrep        pygmentize    unlzma       xzmore
infotocap     lzless        python        unxz
ipython       lzma          python3       wheel

[lumi][kulust@uan04-1008 Tykky-demo]$
```

# lumi-container-wrapper (6)

```
kulust@uan04.lumi.csc - ~/Tykky-demo/conda-cont-1/bin
kulust@uan04.lumi.csc - ~/Tykky-demo/conda-cont-1/bin (ssh)

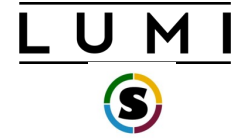
2to3          ipython3      lzmadec       python3.8     wish
2to3-3.8      jupyter       lzmainfo      python3.8-config wish8.6
captaininfo   jupyter-kernel lzmore        python3-config x86_64-conda_cos6-linux-gnu-ld
clear         jupyter-kernelspec ncurse6-config reset          x86_64-conda-linux-gnu-ld
c_rehash     jupyter-migrate ncursew6-config sqlite3        xz
curve_keygen jupyter-run    nglview       sqlite3_analyzer xzcat
_debug_exec  jupyter-troubleshoot normalizer     tabs          xzcmp
_debug_shell list-packages  openssl       tclsh         xzdec
f2py         lzcat         pip           tclsh8.6     xzdiff
f2py3        lzcmp        pip3          tic           xzegrep
f2py3.8      lzdiff       pydoc        toe          xzfgrep
idle3        lzgrep       pydoc3       tput         xzgrep
idle3.8      lzfgrep      pydoc3.8     tset         xzless
infocmp      lzgrep       pygmentize   unlzma       xzmore
infotocap    lzless
ipython      lzma
lzmadec     python3.8
lzmainfo    python3.8-config
lzmore      python3-config
ncurses6-config reset
ncursesw6-config sqlite3
nglview     sqlite3_analyzer
normalizer  tabs
openssl    tclsh
pip        tclsh8.6
pip3      tic
pydoc     toe
pydoc3    tput
pydoc3.8 tset
pygmentize unlzma
python    unxz
python3   wheel

[lumi][kulust@uan04-1008 Tykky-demo]$ cd conda-cont-1/bin
[lumi][kulust@uan04-1009 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)

- 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 next week's 2-day course in Amsterdam when they become available

# 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 very limited support to build containers on LUMI due to security concerns.