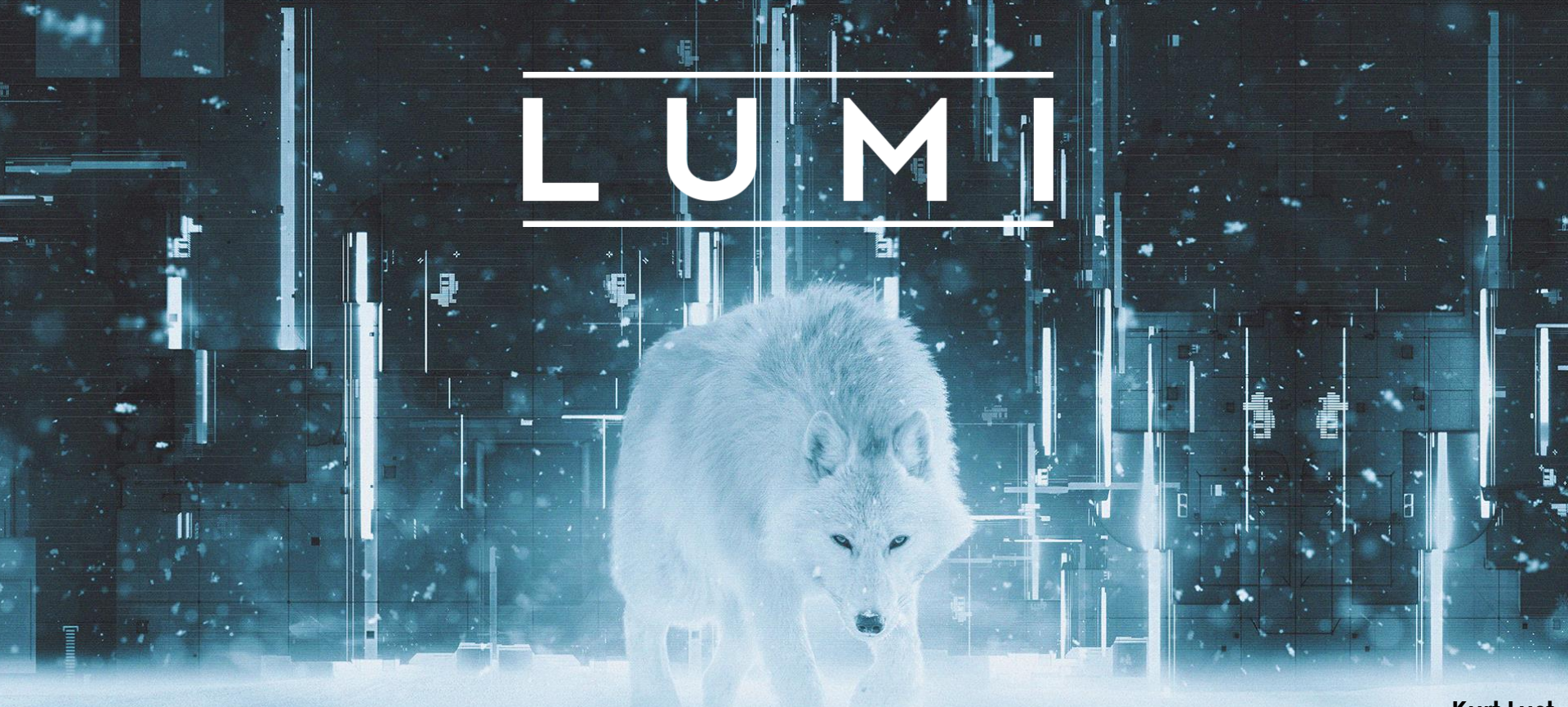

Containers on LUMI-C and LUMI-G

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

June 2025

Containers

LUMI

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
- Using some of our pre-built AI containers

- But remember: LUMI is an HPC infrastructure, not a container cloud!
  - HPC has is own container runtimes specifically for an HPC environment

What do containers not provide?

LUMI

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

But what can they then do on LUMI?

LUMI

- **Containers are a sofware management instrument**
- **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 of ROCm, though with limitations
- **Isolation:** More important for services; often a pain instead
- **But note:** You're the system administrator of your container, not LUST!

Managing containers

LUMI

- Supported runtimes
  - Not all container runtimes are a good match with HPC systems
  - Docker is **NOT** directly available in the 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 a flat (single) sif file for storing the 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` to move the cache

singularity pull docker://julia

LUMI

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

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

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

singularity pull docker://julia

LUMI

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

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

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

singularity pull docker://julia

LUMI

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

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

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

Managing containers (2)

LUMI

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

Interacting with containers

LUMI

- Accessing a container with the shell command

```
singularity shell container.sif
```

singularity shell julia_latest.sif

LUMI

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

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

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

Interacting with containers

LUMI

- 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`

singularity exec julia_latest.sif uname -a

LUMI

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

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

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

Interacting with containers

LUMI

- 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`

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

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

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

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

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

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

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

Interacting with containers

LUMI

- 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`). On LUMI you need `--bind /pfs,/scratch,/projappl,/project,/flash`

Running containers on LUMI

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) for good performance
  - 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

- 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`
- **singularity-AI-bindings** (easyconfig or `/appl/local/containers/ai-modules`)
  - Bindings for some of the AI containers that LUST provides
  - But not a generic binding that will work for all containers!

Environment enhancements (2) Containerising tools

LUMI

- **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)
    - Can even create wrappers for commands in an existing container
  - 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 ⌥⌘2

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

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

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

lumi-container-wrapper (2)

LUMI

kulust@uan04.lumi.csc - ~/Tykky-demo ⌥⌘2

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

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

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

Installing base environment...

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

lumi-container-wrapper (3)

LUMI

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

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

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

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

lumi-container-wrapper (4)

LUMI

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

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

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

lumi-container-wrapper (5)

LUMI

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

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

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

lumi-container-wrapper (6)

LUMI

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

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

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

Environment enhancements (3): Non-AI containers

LUMI

- **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
- **ccpe**: Containerised Cray Programming Environment
  - For advanced users only
  - User-installable as often customisations are needed
  - Experiment with newer versions of the Cray PE
  - Functionality may be limited due to ROCm driver compatibility

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

LUMI

- Currently available
  - PyTorch: Best tested
  - TensorFlow
  - JAX
  - AlphaFold
  - ROCm and mpi4py
- Where to find?
  - `/appl/local/containers/sif-images`: Links to the latest version of each container
  - `/appl/local/containers/easybuild-sif-images`: Images for EasyBuild
    - Recommended for inexperienced users, but work-in-progress
  - `/appl/local/containers/tested-containers`: Images linked to and docker tarballs
- Recommend to keep your own copy of the image you depend upon!

Running the AI containers (Complicated way)

LUMI

- The containers have everything they need to use RCCL and/or MPI on LUMI
- Need to take care of bindings:
  - Need `-B /var/spool/slurmd,/opt/cray,/usr/lib64/libcxi.so.1` at the minimum (and this list may change after a system update or changes in the container builds)
  - And add access to your space in `/project`, `/scratch` and/or `/flash` (default is only the home directory):
    `-B /pfs,/scratch,/projappl,/project,/flash`
- Components that need further initialisation:
  - MIOpen (the AMD cuDNN equivalent)
  - RCCL needs to be told the right network interfaces to use if you run across nodes
  - GPU-aware MPI may need to be set up (see earlier in the course)
  - Your AI package may need some too (e.g., `MASTER_ADDR` and `MASTER_PORT` for distributed learning with PyTorch)
- Containers with Python packages are built using Conda
  - Need to initialise the Conda environment via `$WITH_CONDA` in the container

Running the AI containers EasyBuild (1)

LUMI

- We provide EasyBuild recipes to "install" some of the containers and provide a module.
  - For those packages for which we know generic usage patterns, we provide some scripts that do most settings, and new PyTorch containers have scripts equivalent to the CSC ones
  - Define a number of environment variables to make life easier, e.g., popular bindings and a variable referring to the container
  - Newer versions (will) come with a Python virtual environment pre-initialised to add your own packages
    - No more `$WITH_CONDA` needed as the module takes care of injecting environment variables in the container that have the same effect as the Conda and Python virtual environment activate scripts
    - Management of the Python virtual environment: Create a SquashFS file from the installation
- Someone with some EasyBuild experience may further extend the recipe to, e.g., already install extra packages

Running the AI containers EasyBuild (2)

LUMI

- Install:
  - Set up your user environment for EasyBuild (`EBU_USER_PREFIX`)
  - Run
    ```
    module load LUMI partition/container EasyBuild-user
    eb PyTorch-2.6.0-rocm-6.2.4-python-3.12-singularity-20250404.eb
    ```
  - After that the container module is available in all LUMI stacks and in CrayEnv
- Best to clean up afterwards before running (or take a new shell)
- Will copy the .sif-file to the software installation directory.
  - To delete:
    ```
    module load PyTorch/2.6.0-rocm-6.2.4-python-3.12-singularity-20250404
    rm -f $SIF
    module load PyTorch/2.6.0-rocm-6.2.4-python-3.12-singularity-20250404
    ```
  - At your own risk as we may remove the image in `/appl/local/containers` without notice

Running: Example: Distributed learning Without EasyBuild (1)

LUMI

- Create file `get-master.py`:

```
import argparse
def get_parser():
    parser = argparse.ArgumentParser(description="Extract master node name from Slurm node list",
            formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument("nodelist", help="Slurm nodelist")
    return parser

if __name__ == '__main__’:
    parser = get_parser()
    args = parser.parse_args()

    first_nodelist = args.nodelist.split(',')[0]

    if '[' in first_nodelist:
        a = first_nodelist.split('[‘)
        first_node = a[0] + a[1].split('-')[0]

    else:
        first_node = first_nodelist

    print(first_node)
```

Running: Example: Distributed learning Without EasyBuild (2)

LUMI

- Create file `run-pytorch.sh`:

```
#!/bin/bash -e

# Make sure GPUs are up
if [ $SLURM_LOCALID -eq 0 ] ; then
    rocm-smi
fi
sleep 2

$WITH_CONDA

# Set MIOpen cache to a temporary folder.
export MIOPEN_USER_DB_PATH="/tmp/$(whoami)-miopen-cache-$SLURM_NODEID”
export MIOPEN_CUSTOM_CACHE_DIR=$MIOPEN_USER_DB_PATH

if [ $SLURM_LOCALID -eq 0 ] ; then
    rm -rf $MIOPEN_USER_DB_PATH
    mkdir -p $MIOPEN_USER_DB_PATH
fi
sleep 2

# Set ROCR_VISIBLE_DEVICES so that each task uses the proper GPU
export ROCR_VISIBLE_DEVICES=$SLURM_LOCALID

# Report affinity
echo "Rank $SLURM_PROCID --> $(taskset -p $$)”

# Set interfaces to be used by RCCL.
export NCCL_SOCKET_IFNAME=hsn0,hsn1,hsn2,hsn3
export NCCL_NET_GDR_LEVEL=3

# Set environment for the app
export MASTER_ADDR=$(python get-master.py "$SLURM_NODELIST")
export MASTER_PORT=29500
export WORLD_SIZE=$SLURM_NPROCS
export RANK=$SLURM_PROCID

# Run app
python -u mnist_DDP.py --gpu --modelpath model
```

Check for the GPUs

Initialise Conda

MIOpen configuration

GPU binding

RCCL configuration

Who’s the master?

Ready to run…

Running: Example: Distributed learning Without EasyBuild (3)

LUMI

- Create job script `my-job.sh`:

```
#!/bin/bash -e
#SBATCH --nodes=4
#SBATCH --gpus-per-node=8
#SBATCH --tasks-per-node=8
#SBATCH --output="output_%x_%j.txt”
#SBATCH --partition=standard-g
#SBATCH --mem=480G
#SBATCH --time=00:10:00
#SBATCH --account=project_<your_project_id>

PROJECT_DIR=/project/your_project/your_directory
SIF=/appl/local/containers/easybuild-sif-images/lumi-pytorch-rocm-6.2.4-python-3.12-pytorch-v2.6.0-dockerhash-36e16fb5b67b.sif

c=fe
MYMASKS="0x${c}000000000000,0x${c}00000000000000,0x${c}0000,0x${c}000000,0x${c},0x${c}00,0x${c}00000000,0x${c}0000000000”

srun --cpu-bind=mask_cpu:$MYMASKS \
  singularity exec \
    -B /var/spool/slurmd \
    -B /opt/cray \
    -B /usr/lib64/libcxi.so.1 \
    -B $PROJECT_DIR:/workdir \
    $SIF /workdir/run-pytorch.sh
```

Reorder CPU slots for easy GPU binding

Run the script from the previous slide

Running: Example: Distributed learning With EasyBuild-installed module

LUMI

- Create job script `my-job.sh`:

```
#!/bin/bash –e
#SBATCH --nodes=4
#SBATCH --gpus-per-node=8
#SBATCH --tasks-per-node=8
#SBATCH --output="output_%x_%j.txt”
#SBATCH --partition=standard-g
#SBATCH --mem=480G
#SBATCH --time=00:10:00
#SBATCH --account=project_<your_project_id>

module load CrayEnv PyTorch/2.6.0-rocm-6.2.4-python-3.12-singularity-20250404

c=fe
MYMASKS="0x${c}000000000000,0x${c}00000000000000,0x${c}0000,0x${c}000000,0x${c},
0x${c}00,0x${c}00000000,0x${c}0000000000”

srun --cpu-bind=mask_cpu:$MYMASKS \
  singularity exec $SIF \
    conda-python-distributed -u mnist_DDP.py --gpu --modelpath model
```

Extending container 1: cotainr

LUMI

- It is possible to use the ROCm containers in `/appl/local/containers/sif-images` as a base image for cotainr and build your own AI container
  - Be careful which version of the AI software you use as wheels are likely for a specific ROCm version (and you don't want to pick up wheels for NVIDIA)
  - MPI may be a problem as mpi4py has to come from Conda
- Process:
  - Create a yaml file with the setup for Conda (see notes)
  - Run cotainr:

```
module load LUMI/24.03 cotainr
cotainr build my-new-image.sif \
    --base-image=/appl/local/containers/sif-images/lumi-rocm-rocm-6.0.3.sif \
    --conda-env=py312_rocm603_pytorch.yml
```

- Run as a regular container
  - Or find someone who want to make an EasyConfig to create a module and point EasyBuild to the container .sif file with `--sourcepath`

Extending container 2: singularity build

LUMI

- Build a singularity-compatible container definition file, e.g.,

```
Bootstrap: localimage

From: /appl/local/containers/easybuild-sif-images/lumi-pytorch-
rocm-6.0.3-python-3.12-pytorch-v2.3.1-dockerhash-2c1c14cafd28.sif

%post

zypper -n install -y Mesa libglvnd libgthread-2_0-0 hostname
```

- And run:
  ```
  module load LUMI/24.03 systools
  singularity build my-new-container.sif my-container-definition.def
  ```
- Good way to add SUSE packages that may be needed to install extra software
- Tip: Start from a container with an EasyBuild module and the module might still work…

Extending container 3: Python virtual environment (1)

LUMI

- Some newer containers installed with EasyBuild have a pre-initialised virtual environment
  - In the container available as `/user-software/venv/<MyVEnv>`
  - Outside the container: `$CONTAINERROOT/user-software/venv/<MyVEnv>`
  - And `/user-software` can also be used to install other software if needed…
- How?

```
$> module load LUMI
$> module load PyTorch/2.6.0-rocm-6.2.4-python-3.12-singularity-20250404
$> singularity shell $SIF
Singularity> pip install pytorch-lightning
```

Extending container 3: Python virtual environment (2)

LUMI

- But what about the many small files?
  - Convert `$CONTAINERROOT/user-software` to a SquashFS file
    `make-squashfs`
    And reload the module…
  - You can then delete the `$CONTAINERROOT/user-software` subdirectory if you need the space (or file quota) and reconstruct it if needed with `unmake-squashfs`
  - To add additional packages afterwards:
    - Make sure the `$CONTAINERROOT/user-software` exists (outside the container)
    - Delete `$CONTAINERROOT/user-software.squashfs`
    - Reload the module
    - And start a shell in the container…
- You can of course do this with any container with Python, also when not using EasyBuild-built modules but the manual procedure takes a few more steps.

Container limitations on LUMI

LUMI

- "Bring your own userland and run on a system-optimised kernel" and you'll be fine is a myth
- Containers use the host's operating system kernel which may be different from what the container expects. Containers also do not abstract hardware.
- Much of system-specific optimisation is done in userland:
  - CPU and GPU instruction set specific
  - 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: inject Cray MPICH, but only for containers with ABI compatibility with MPICH.
    - Distributed AI: Need to inject the proper RCCL plugin.
  - AMD driver version may pose problems also.

- Only limited support for building containers on LUMI due to security concerns.

LUMI

Questions?