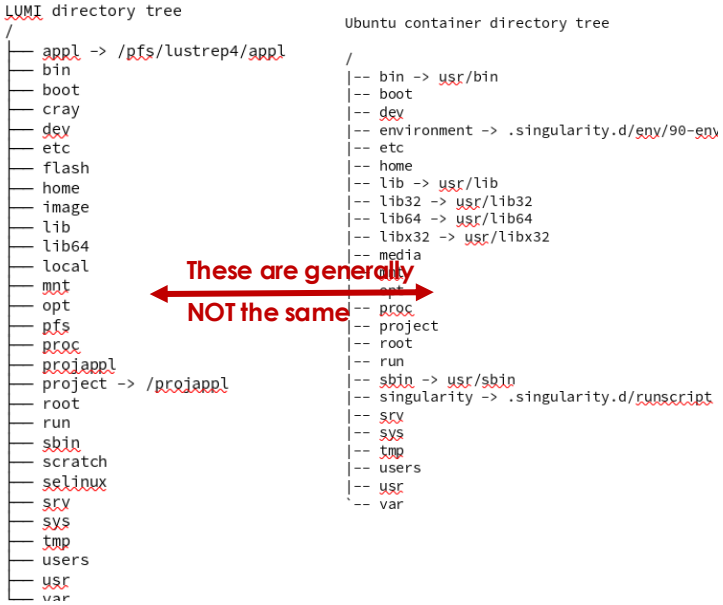
Running containers on LUMI

Julius Roeder – Danish e-infrastructure Consortium (DeiC)

Containers – bring your own user space

- When running containers, the kernel, drivers and hardware is still provided by the host (LUMI) - but the user space (directory tree) changes
- Benefits of using containers:
  - **Enhanced reproducibility**: A fixed (read-only) user space for each computational experiment
    - Have a fully self-contained software environment
    - Pin all versions of the software packages used
    - Version control software environments
  - **Limited portability**: Makes it easier to share your software environment with others – just share the container
    - Same environment work on all compute platforms – maybe in the future
  - **Easily test and trash**: Try a new container – if it doesn't work just trash the container and start over again
    - Building weird libraries & packages can be easier.
    - Test new bleeding edge versions of software.
  - **Bonus**: Your software environment is a single file (the container) on the Lustre shared filesystems, which is much less stressful to Lustre and more performant, making for a much nicer experience for everyone on LUMI.



Running containers on LUMI

- On LUMI, you can run Singularity/Apptainer containers
  - Singularity/Apptainer are HPC container runtimes that allow you to run unprivileged containers, i.e. no need for root or sudo
  - Singularity/Apptainer is not Docker, but if you have an existing Docker container, you can run it using Singularity/Apptainer
- Main singularity commands:
  - Getting (pulling) a container from a container registry
    ```
    singularity pull my_container.sif docker://ubuntu:22.04
    ```
  - Opening a shell inside the container
    ```
    singularity shell my_container.sif
    ```
  - Executing a command inside the container
    ```
    singularity exec my_container.sif python3 my_script.py
    ```
- Running containers on compute nodes
  - Launch computation using srun
    ```
    srun <options> singularity exec my_container.sif python3 my_script.py
    ```

Bind mounting parts of the host file system (1)

- When running a container on LUMI, where is /project, /scratch, etc.?
- You may "inject" parts of the host (LUMI) file system into the container by bind mounting it

```
singularity exec --bind /project/<project_ID>
my_containr.sif ls -lh /
```

```
your_lumi_username@uan03:~> singularity exec --bind /project/project_465001363 ubuntu_tree.sif tree -L 1 /
/
|-- bin -> usr/bin
|-- boot
|-- dev
|-- environment -> .singularity.d/env/90-environment.sh
|-- etc
|-- home
|-- lib -> usr/lib
|-- lib32 -> usr/lib32
|-- lib64 -> usr/lib64
|-- libx32 -> usr/libx32
|-- media
|-- mnt
|-- opt
|-- proc
|-- project
|-- root
|-- run
|-- sbin -> usr/sbin
|-- singularity -> .singularity.d/runscript
|-- srv
|-- sys
|-- tmp
|-- users
|-- usr
`-- var
```

```
LUMI directory tree
/
├── appl -> /pfs/lustrep4/appl
├── bin
├── boot
├── cray
├── dev
├── etc
├── flash
├── home
├── image
├── lib
├── lib64
├── local
├── mnt
├── opt
├── pfs
├── proc
├── projappl
├── project -> /projappl
├── root
├── run
├── sbin
├── scratch
├── selinux
├── srv
├── sys
├── tmp
├── users
├── usr
└── var
```

```
ubuntu container directory tree
/
|-- bin -> usr/bin
|-- boot
|-- dev
|-- environment -> .singularity.d/env/90-environment
|-- etc
|-- home
|-- lib -> usr/lib
|-- lib32 -> usr/lib32
|-- lib64 -> usr/lib64
|-- libx32 -> usr/libx32
|-- media
|-- mnt
|-- opt
|-- proc
|-- root
|-- run
|-- sbin -> usr/sbin
|-- singularity -> .singularity.d/runscript
|-- srv
|-- sys
|-- tmp
|-- users
|-- usr
`-- var
```

Bind mounting parts of the host file system (2)

You typically want to bind mount your project folders (/project, /scratch, /flash). A shortcut is:

```
module load Local-LAIF lumi-aif-singularity-bindings
```

Note: The current LAIF containers do not have the "tree" command. You can instead inspect if the correct folders are mounted with e.g. "ls –lh /"

```
4.0K Apr 26  2022 appl
   7 Apr 22  2024 bin -> usr/bin
  60 Feb 23 18:40 boot
 14K Jun  5 17:40 dev
  36 Apr 15 17:04 environment -> .singu
2.2K Apr 15 16:33 etc
 34K Jun 10 03:43 flash
  29 Mar 24 09:24 home
   7 Apr 22  2024 lib -> usr/lib
   9 Apr 22  2024 lib64 -> usr/lib64
   3 Nov 14  2024 lib.usr-is-merged
   3 Mar 24 09:15 media
 278 Jun  9 14:26 minimal_pytorch.yml
   3 Mar 24 09:15 mnt
 110 Jun  9 14:26 opt
 140 May 25 19:40 pfs
   0 May 25 19:07 proc
 34K Jun 10 03:42 projappl
 34K Jun 10 03:42 project
  72 Apr 15 16:11 root
 155 Apr 15 16:56 run
   8 Apr 22  2024 sbin -> usr/sbin
 34K Jun 10 03:42 scratch
  24 Apr 15 17:04 singularity -> .singu
   3 Mar 24 09:15 srv
   0 Jun 10 10:31 sys
4.0K Jun 10 10:31 tmp
  31 Jun  9 14:26 users
 225 Apr 15 16:33 usr
  60 Jun 10 10:42 var
```

Official LAIF containers

Containers available on LUMI under /appl/local/laifs/containers/ :

- Application images:
  - ROCm
  - Libfabric
  - **mpich**
  - **PyTorch**
  - **Full**
- Documentation:
  - https://docs.lumi-supercomputer.eu/laif/software/ai-environment/#examples-for-using-the-container-images
  - https://github.com/lumi-ai-factory/laifs-container-recipes
- Probably a good idea to copy these to your project folder

```
ls -l /appl/local/laifs/containers/lumi-multitorch-u24r70f21m50t210-20260415_130625/*.sif
06:18 /appl/local/laifs/containers/lumi-multitorch-u24r70f21m50t210-20260415_130625/lumi-multitorch-full-u24r70f21m50t210-20260415_130625.sif
16:44 /appl/local/laifs/containers/lumi-multitorch-u24r70f21m50t210-20260415_130625/lumi-multitorch-libfabric-u24r70f21m50t210-20260415_130625.sif
17:07 /appl/local/laifs/containers/lumi-multitorch-u24r70f21m50t210-20260415_130625/lumi-multitorch-mpich-u24r70f21m50t210-20260415_130625.sif
07:38 /appl/local/laifs/containers/lumi-multitorch-u24r70f21m50t210-20260415_130625/lumi-multitorch-plus-u24r70f21m50t210-20260415_130625.sif
16:30 /appl/local/laifs/containers/lumi-multitorch-u24r70f21m50t210-20260415_130625/lumi-multitorch-rocm-u24r70f21m50t210-20260415_130625.sif
18:49 /appl/local/laifs/containers/lumi-multitorch-u24r70f21m50t210-20260415_130625/lumi-multitorch-torch-u24r70f21m50t210-20260415_130625.sif
```

Further reading

- LUMI Docs running containers page:
  https://docs.lumi-supercomputer.eu/laif/software/ai-environment/#examples-for-using-the-container-images
- LAIF Container recipes
  https://github.com/lumi-ai-factory/laifs-container-recipes