

A white wolf is walking towards the viewer in a snowy, futuristic cityscape. The scene is illuminated with a strong blue light, creating a cold, digital atmosphere. In the background, there are tall, thin structures resembling skyscrapers or data centers, some with glowing windows. The ground is covered in snow, and there are faint digital patterns and lines scattered throughout the scene.

LUMI

Building containers from
conda/pip environments

Julius Roeder – Danish e-infrastructure Consortium (DeiC)

Conda environment.yml files

- A conda environment can be described in a YAML file
- You may write this file yourself
 - Prefer conda packages over pip
 - Pin version numbers (and ideally also build numbers)
Format: <package_name>=<version>=<build_number>
- You may export an already existing conda environment
`conda env export -n <name_of_conda_env> -f conda_env.yml`
- Installing the conda environment is easy
`conda env create -f conda_env.yml`
- We ask you to **NOT** install such conda environments directly on the Lustre files systems (/project, /scratch, /flash or your home folder) - use a container instead
 - But how do I easily build a container from a conda_env.yml file?

name: PandasAI

channels:

- conda-forge

dependencies:

- pandas=1.5.3

- pip=24.0

- python=3.12.3

- ...

- **pip:**

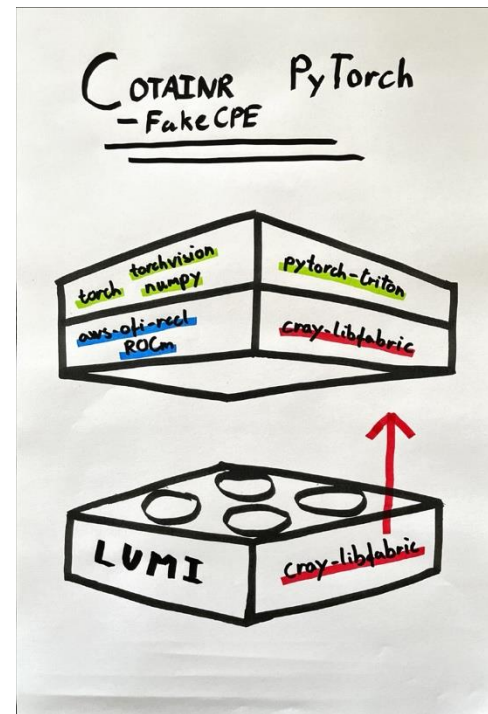
- pandasai==2.0.35

- ...

Building containers for LUMI

LUMI

- Separation of concerns:
 - Host libraries (OS, drivers, container runtime, ...)
 - Provided by the LUMI system administrators
 - System specific libraries (ROCM, libfabric, ...)
 - Provided by LAIF as container base images
 - Application libraries (PyTorch, NumPy, ...)
 - Provided by **you**
- **For application libraries specified in a conda_env.yml file, you may use cotainr to easily build a container on LUMI**
- For the more general case:
 - Singularity + proot may be used to build containers on LUMI
 - Singularity/Aptainer may be used to build containers on your laptop



Cotair

– the shortcut to building containers based on conda environments

- Instead of

```
conda env create -f my_AI_env.yml
```

you run*

```
cotair build my_AI_container.sif --base-image=$SIF_IMAGE
--conda-env=my_AI_env.yml
```

- The `--base-image=...` specifies an existing container to use.

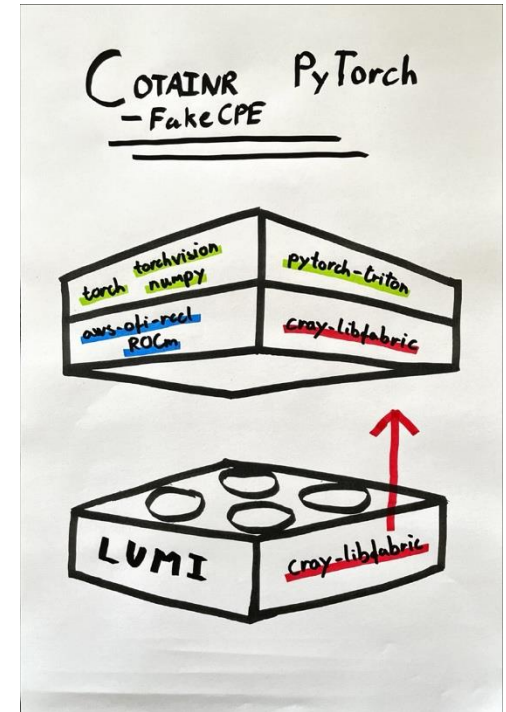
- Cotair is installed in the central LUMI software stack

```
module purge
```

```
module load CrayEnv cotair
```

- To avoid putting stress on the login-nodes, you may consider running cotair non-interactively on a compute node

```
srun --output=cotair.out --error=cotair.err --account=<project_ID>
--time=00:30:00 --mem=60G --cpus-per-task=32 --partition=debug
cotair build minimal_pytorch.sif --base-image=$SIF_IMAGE --conda-
env=minimal_pytorch.yml --accept-licenses
```



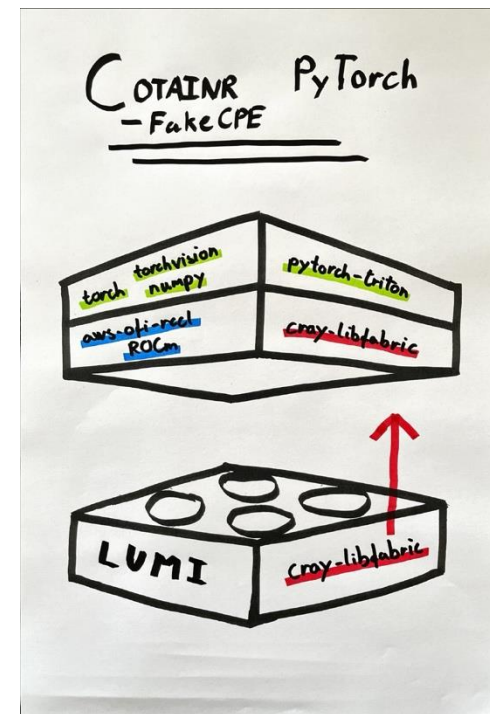
* The my_AI_env.yml Conda environment file must specify packages that are compatible with the GPUs on LUMI, i.e. ROCm builds of PyTorch/Tensorflow/Jax/...

Cotair

– Pros & Cons

- Alternatives
 - No container – pip/conda directly on file sytem
Problem: Many files will cause you issues with your quota and affects the file system performance
 - Self-built containers
Not straightforward to built conda environment inside container.
 - Prebuilt LAIF containers
Many packages included → expanding can be challenging
- Pros of cotair
 - Very easy to use
 - Forces good practices on how to use conda
- Cons of cotair
 - Building containers takes time
 - (Internode communication doesn't work at the moment on LUMI-C)
 - Only works for packages that can be installed with conda/pip

LUMI



A note on ROCm compatibility for LUMI-G



- To have AMD GPU support, you need ROCm compatible versions of:
 - The AMDGPU/KFD Linux kernel driver – installed by the LUMI system administrators
 - The ROCm user space components – included in the base image
 - The application built with ROCm support – the conda/pip package (or source) selected by you
- AMD only provides a +/- 2 ROCm release "tested compatibility" claim
 - As of January 2026, the AMDgpu driver on LUMI, aligned with ROCm 6.3.4, has "tested compatibility" with ROCm versions 6.1, 6.2, 6.3, 6.4 and 7.0
 - There are LAIF images for ROCm 6.4 and 7.0.
 - Historically, we have seen 4-5 ROCm releases a year, roughly corresponding to a "tested compatibility" window of about +/- ½ year. Thus, older versions of ROCm quickly become unsupported on LUMI following system upgrades
 - Applications are typically built for a range of ROCm releases, providing a larger compatibility window, but you might not be able to run very old or the most recent versions of PyTorch/TensorFlow/Jax/... on LUMI.

A note on pip installable software

- Anything you can specify in a pip requirements.txt file, you can specify in the same way in the pip section of a conda_env.yml file.
- Thus, ANY binary/source that installs using a pip requirements.txt file can be installed using cotair, e.g.
 - Standard PyPI packages (wheels or from source)
 - Local wheels or source archives
 - Wheels or source archives hosted on blob storage, e.g. LUMI-O
 - Git(Hub/Lab) repos that contain a proper setup.py (or the more modern pyproject.toml)
- If you need to install something from source that needs to compile C and/or ROCm extensions, you must make sure everything needed is available for pip to build and install the software, i.e. you must
 - Install all necessary libraries and compilers if they are not part of the base image, e.g. as conda dependencies in your conda_env.yml
 - Set/export any required environment variables for the build
 - Make sure that any CUDA extensions have been ported to HIP/ROCm
- Consider building wheels for source or local projects separately from building containers to maximize reproducibility.

Cotainr recipes: PyTorch for LUMI-G

- No official conda package for ROCm PyTorch exists, but official pip wheels do exist
 - Browse available versions at:
<https://download.pytorch.org/whl/>
 - Add the --extra-index-url <https://download.pytorch.org/whl/<rocm-version>> to your conda environment pip specification
 - Add the relevant "+rocmX.Y" package to your conda environment pip specification
- Building the container on LUMI:
- ```
module load CrayEnv cotainr
export SIF_IMAGE=/appl/local/laifs/containers/lumi-
multitorch-u24r70f21m50t210-20260415_130625/lumi-
multitorch-mpich-u24r70f21m50t210-20260415_130625.sif
cotainr build minimal_pytorch.sif
--base-image=$SIF_IMAGE --conda-
env=minimal_pytorch.yml
```

[minimal\\_pytorch.yml](#)

**name:** minimal\_pytorch

**channels:**

- conda-forge

**dependencies:**

- pip=24.0

- python=3.12.3

- **pip:**

- --extra-index-url

<https://download.pytorch.org/whl/rocm6.0/>

- pytorch-triton-rocm==3.0.0

- torch==2.4.1+rocm6.0

- torchaudio==2.4.1+rocm6.0

- torchvision==0.19.1+rocm6.0

**LUMI**

# Cotainr repices:

## TensorFlow for LUMI-G

- No official conda package or pip wheel for ROCm TensorFlow exists, but AMD provides pip wheels on their own website
  - Check the documentation:  
<https://rocm.docs.amd.com/projects/install-on-linux/en/latest/install/3rd-party/tensorflow-install.html>
  - Browse available versions at  
<https://repo.radeon.com/rocm/manylinux/>
- Building the container on LUMI:
- `module load CrayEnv cotainr`  
`export SIF_IMAGE=/appl/local/laifs/containers/lumi-multitorch-u24r70f21m50t210-20260415_130625/lumi-multitorch-mpich-u24r70f21m50t210-20260415_130625.sif`  
`cotainr build minimal_tensorflow.sif`  
`--base-image=$SIF_IMAGE --conda-env=minimal_tensorflow.yml`

[minimal\\_tensorflow.yml](#)

name: minimal\_tensorflow

channels:

- conda-forge

dependencies:

- pip=24.0

- python=3.12.3

- pip:

- [https://repo.radeon.com/rocm/manylinux/rocm-rel-7.0.2/tensorflow\\_rocm-2.19.1-cp312-cp312-manylinux\\_2\\_28\\_x86\\_64.whl](https://repo.radeon.com/rocm/manylinux/rocm-rel-7.0.2/tensorflow_rocm-2.19.1-cp312-cp312-manylinux_2_28_x86_64.whl)

**LUMI**

## Cotainr repices: JAXfor LUMI-G

- Wheel releases exist for newer ROCm and Jax versions

<https://github.com/jax-ml/jax/blob/main/build/rocm/README.md>

- Building the container on LUMI:
- ```
module load CrayEnv cotainr
export SIF_IMAGE=/appl/local/laifs/containers/lumi-multitorch-
u24r70f21m50t210-20260415_130625/lumi-multitorch-mpich-
u24r70f21m50t210-20260415_130625.sif
cotainr build minimal_jaxflax.sif
--base-image=$SIF_IMAGE
--conda-env=minimal_jaxflax.yml
```

Jax.yml

```
name: jax
channels:
  - conda-forge
dependencies:
  - pip=24.0
  - python=3.12.3
  - pip:
    - jax[rocm7-local]
```

LUMI

Further reading

- LUMI Docs containers page
<https://docs.lumi-supercomputer.eu/software/containers/singularity/>
- LUMI Docs installing Python packages page
<https://docs.lumi-supercomputer.eu/software/installing/python/>
- Cotainr conda env documentation
https://cotainr.readthedocs.io/en/latest/user_guide/conda_env.html
- Conda environment documentation
<https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html>
- Pip requirements.txt file specification
<https://pip.pypa.io/en/stable/reference/requirements-file-format/>
- ROCm compatibility kernel / user space compatibility
<https://rocm.docs.amd.com/projects/install-on-linux/en/latest/reference/user-kernel-space-compat-matrix.html>