Building containers from conda/pip environments

Christian Schou Oxvig – LUMI User Support Team
Danish e-infrastructure Consortium - DeiC, Denmark

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

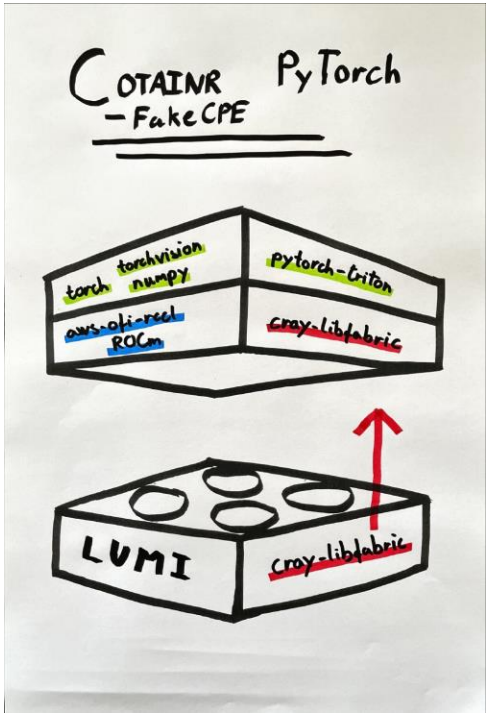
  `conda env export -n <name_of_conda_env> -f conda_env.yml`
- Installing the conda enviornment is easy

  `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
    - ...
```

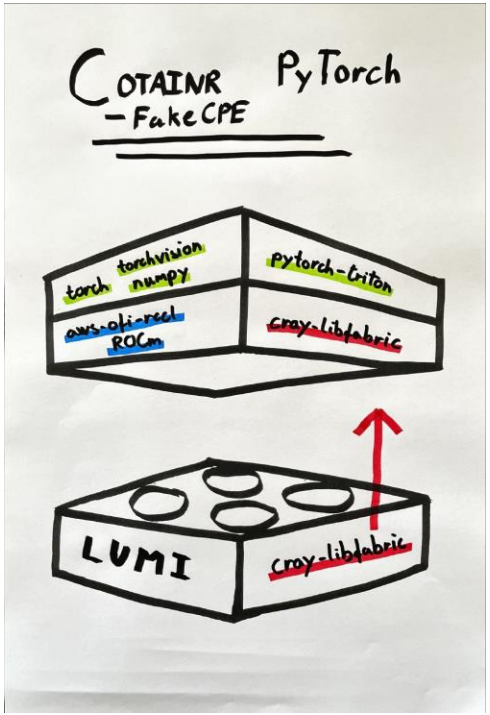
Building containers for LUMI

- Separation of concerns:
  - Host libraries (OS, drivers, container runtime, …)
    - Provided by the LUMI system administrators
  - System specific libraries (ROCm, aws-ofi-rccl, …)
    - Provided by the LUMI User Support Team 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



Cotainr
– the shortcut to building containers based on conda environments

- Instead of
  ```
  conda env create -f my_AI_env.yml
  ```
  you run*
  ```
  cotainr build my_AI_container.sif --system=lumi-g
   --conda-env=my_AI_env.yml
  ```
- The `--system=lumi-g` is a shortcut to getting a proper base image. Alternatively, specify `--base-image=...`
- Cotainr is installed in the central LUMI software stack
  ```
  module use /appl/local/training/modules/AI-20240529
  module load LUMI cotainr
  ```
- To avoid putting stress on the login-nodes, you may consider running cotainr non-interactively on a compute node
  ```
  srun --output=cotainr.out --error=cotainr.err --
  account=<project_ID> --time=00:30:00 --mem=64G --cpus-per-task=32
  --partition=debug cotainr build minimal_pytorch.sif --system=lumi-
  g --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/...

A note on ROCm compatibility for LUMI-G

LUMI

- 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
- Unfortunately, AMD only provides a +/- 2 ROCm release "tested compatibility" claim
  - As of May 2024, the AMDGPU/KFD driver on LUMI, aligned with ROCm 5.2, has "tested compatibility" with ROCm versions 5.0, 5.1, 5.2, 5.3, and 5.4. However, ROCm 5.5 and 5.6 also works for most use cases.
  - **WARNING**: To use ROCm >= 5.7, we need a newer AMDGPU/KFD driver on LUMI. Such an upgrade is currently scheduled for September 2024. Using anything built for ROCm >= 5.7 before the next system upgrade is very likely going to break
  - 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

LUMI

- 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 cotainr, 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 repices: 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 LUMI cotainr
cotainr build minimal_pytorch.sif
  --system=lumi-g
  --conda-env=minimal_pytorch.yml
```

minimal_pytorch.yml

```
name: minimal_pytorch
channels:
  - conda-forge
dependencies:
  - filelock=3.13.4
  - fsspec=2024.3.1
  - jinja2=3.1.3
  - markupsafe=2.1.5
  - mpmath=1.3.0
  - networkx=3.3
  - numpy=1.26.4
  - pillow=10.3.0
  - pip=24.0
  - python=3.11.9
  - sympy=1.12
  - typing-extensions=4.11.0
  - pip:
    - --extra-index-url
https://download.pytorch.org/whl/rocm5.6/
    - pytorch-triton-rocm==2.2.0
    - torch==2.2.2+rocm5.6
    - torchaudio==2.2.2+rocm5.6
    - torchvision==0.17.2+rocm5.6
```

Cotainr repices: TensorFlow for LUMI-G

- No official conda package or pip wheel for ROCm Tensorflow exists, but AMD provides pip wheels on PyPI
  - Install `tensorflow-rocm` instead of `tensorflow`
  - Browse available versions at https://github.com/ROCm/tensorflow-upstream/blob/develop-upstream/rocm_docs/tensorflow-rocm-release.md
  - Note the X.Y.Z.ROCm versioning scheme, e.g. version 2.12.0.560 is Tensorflow 2.12.0 built for ROCm 5.6
- Building the container on LUMI:

```
module load LUMI cotainr
cotainr build minimal_tensorflow.sif
  --system=lumi-g
  --conda-env=minimal_tensorflow.yml
```

minimal_tensorflow.yml

```
name: minimal_tensorflow
channels:
  - conda-forge
dependencies:
  - astunparse=1.6.3
  - cachetools=5.3.3
  - gast=0.4.0
  - google-pasta=0.2.0
  - h5py=3.10.0
  - jax=0.4.26
  - keras=2.12.0
  - numpy=1.23.5
  - opt_einsum=3.3.0
  - pip=24.0
  - python=3.11.9
  - scipy=1.13.0
  - tensorboard=2.12.3
  - termcolor=2.4.0
  - typing_extensions=4.11.0
  - wrapt=1.14.1
  - pip:
    - flatbuffers==24.3.25
    - libclang==18.1.1
    - tensorflow-estimator==2.12.0
    - tensorflow-io-gcs-filesystem==0.37.0
    - tensorflow-rocm==2.12.0.560
```

Cotainr repices: JAX/Flax for LUMI-G

- No official conda package or pip wheel for ROCm jaxlib exists, but AMD provides pip wheels on GitHub
  - Browse releases at https://github.com/ROCm/jax/releases
  - Select a version compatible with your Python and ROCm versions
  - Add the selected wheel URL directly to your conda environment pip specification
- Building the container on LUMI:

```
module load LUMI cotainr
cotainr build minimal_jaxflax.sif
  --system=lumi-g
  --conda-env=minimal_jaxflax.yml
```

minimal_jaxflax.yml

```
name: minimal_jaxflax
channels:
  - conda-forge
dependencies:
  - absl-py=2.1.0
  - etils=1.8.0
  - fsspec=2024.3.1
  - importlib_resources=6.4.0
  - msgpack-python=1.0.7
  - nest-asyncio=1.6.0
  - opt_einsum=3.3.0
  - pip=24.0
  - protobuf=4.25.3
  - pyyaml=6.0.1
  - python=3.11.9
  - scipy=1.13.0
  - rich=13.7.1
  - tensorstore=0.1.57
  - toolz=0.12.1
  - zipp=3.17.0
  - pip:
    -
https://github.com/ROCmSoftwarePlatform/jax/releases/download/jaxlib-v0.4.21/jaxlib-0.4.21+rocm560-cp311-cp311-manylinux2014_x86_64.whl
    - chex==0.1.86
    - flax==0.8.3
    - jax==0.4.21
    - optax==0.2.2
    - orbax-checkpoint==0.5.10
```

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