Apptainer as Jupyter Kernel

L. Anderlini (INFN Firenze)

AI_INFN
Artificial Intelligence technologies for INFN research

Motivation

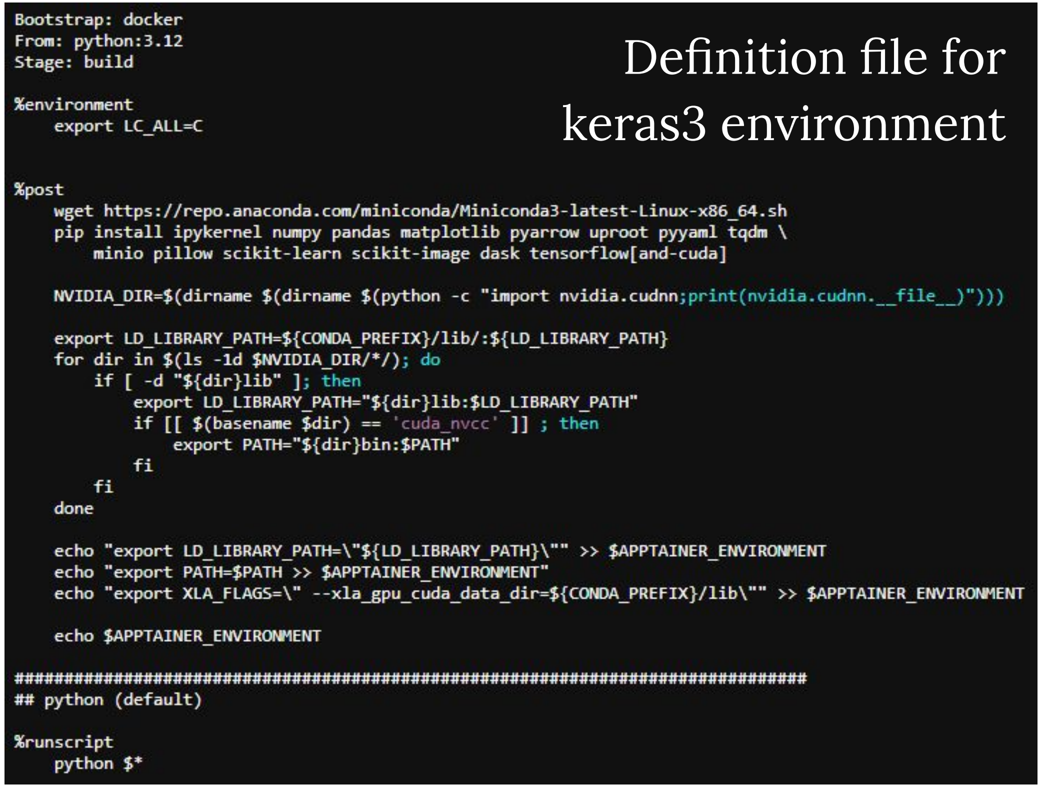
Providing “supported” environments with **conda is simple and effective**, but:

- propagating the **environment variables** to the ipykernel requires customizations;

- creates **large directories with hundreds of thousands of small files**, which are painful for NFS, and unmanageable for any “offloading”.

Apptainer image definition and kernel configuration

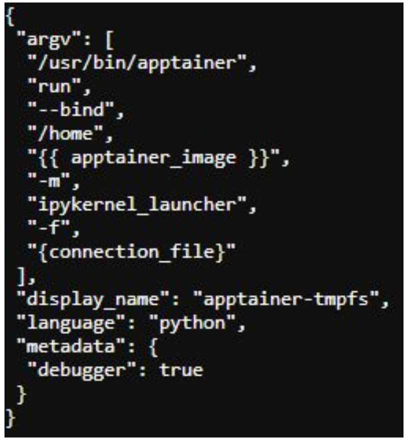
Definition file for keras3 environment



```
Bootstrap: docker
From: python:3.12
Stage: build

%environment
    export LC_ALL=C


%post
    wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
    pip install ipykernel numpy pandas matplotlib pyarrow uproot pyyaml tqdm \
        minio pillow scikit-learn scikit-image dask tensorflow[and-cuda]

    NVIDIA_DIR=$(dirname $(dirname $(python -c "import nvidia.cudnn;print(nvidia.cudnn.__file__)")))

    export LD_LIBRARY_PATH=${CONDA_PREFIX}/lib/:${LD_LIBRARY_PATH}
    for dir in $(ls -1d $NVIDIA_DIR/*/); do
        if [ -d "${dir}lib" ]; then
            export LD_LIBRARY_PATH="${dir}lib:$LD_LIBRARY_PATH"
            if [[ $(basename $dir) == 'cuda_nvcc' ]] ; then
                export PATH="${dir}bin:$PATH"
            fi
        fi
    done

    echo "export LD_LIBRARY_PATH=\"${LD_LIBRARY_PATH}\"" >> $APPTAINER_ENVIRONMENT
    echo "export PATH=$PATH >> $APPTAINER_ENVIRONMENT"
    echo "export XLA_FLAGS=\" --xla_gpu_cuda_data_dir=${CONDA_PREFIX}/lib\"" >> $APPTAINER_ENVIRONMENT

    echo $APPTAINER_ENVIRONMENT

################################################################################
## python (default)

%runscript
    python $*
```

Kernel template

{{ apptainer_image }} must be replaced with path to the apptainer image.



```
{
 "argv": [
  "/usr/bin/apptainer",
  "run",
  "--bind",
  "/home",
  "{{ apptainer_image }}",
  "-m",
  "ipykernel_launcher",
  "-f",
  "{connection_file}"
 ],
 "display_name": "apptainer-tmpfs",
 "language": "python",
 "metadata": {
  "debugger": true
 }
}
```

What works and what does not

Apptainer image distributed via S3 successfully (impossible with conda)

Verified it works with GPUs in the JupyterLab interface (env vars are propagated properly)

Can also be used for scripts, possibly in batch (not tested).

User-level customization require modifying the definition file (interactive `pip install` does not work).

Attempt using apptainer overlays failed (cannot write on overlays from multiple processes). Sandbox mode works, but is messy.