HPC.NRW

NATIONALES HOCHLEISTUNGS RECHNEN

EDIH Rheinland

WEST AI KI-Servicezentrum

PPCES: Machine and Deep Learning

Execution Options for ML / DL software

DAS KOMPETENZNETZWERK FÜR HOCHLEISTUNGSRECHNEN.

Running ML / DL Software (Mostly Python based)

HPC.NRW

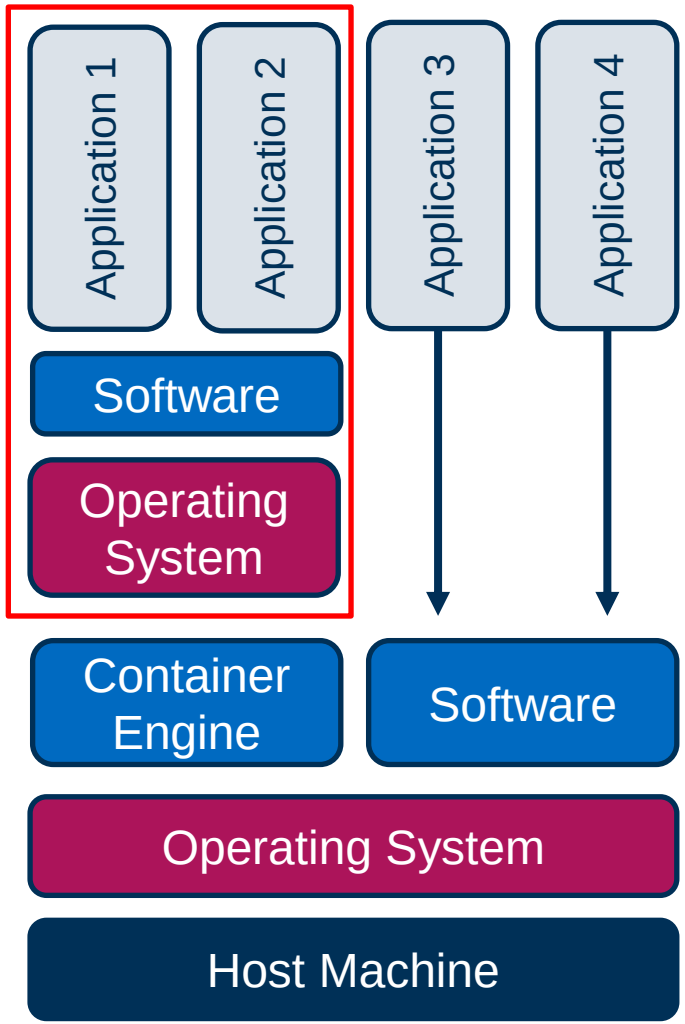
Approach 1: Containers (recommended)

- Virtualized OS encapsulating software with all dependencies to make it available on other machines
- Predefined containers available
- Can be used by all users on the cluster
  - Reproducibility (development + production)
  - Less pressure on file system (only 1 copy)
- Once build they can't be modified
  - How to handle missing packages?

Approach 2: Virtual Environments

- Different solutions available
  - Python virtualenv (pip)
  - Conda / Anaconda
- Can also be used locally on your desktop
- Highly flexible
  - You can install and manage your package according to your needs
- Every user has its own installations
  - GBs and ~35k files per environment. High pressure on cluster file system / slow
  - Less reproducibility when working in teams

More Information about Containers



- The container engine (e.g., Docker or Apptainer), virtualizes parts of the operating system (process space, user namespace, network namespace)
- Containerized and native applications share the same host OS kernel. You can build it on one machine and use it somewhere else
- Negligible performance overhead when using containers

Available Containers on CLAIX

HPC.NRW

- **Question:** What containers are available on CLAIX?

```
# show available software
module avail

------------------------------------------ Container Image Modules -------------------------------------------
datascience-notebook/6.4.8        PyTorch/nvcr-23.08-py3                    TensorFlow/nvcr-23.08-tf2-py3
datascience-notebook/7.0.3        PyTorch/nvcr-24.01-py3             (D)    TensorFlow/nvcr-24.01-tf2-py3 (D)
datascience-notebook/7.0.6 (D)    rapids/nvcr-22.02-cuda11.0-runtime
```

- Standard Apptainer containers for most common frameworks
  - Regular updates of container versions
  - Based on Docker images from DockerHub or NVIDIA
- Support for user containers
  - Build and run your own containers
  - Possibility to convert Docker images to Apptainer images
  - Requests for standard containers → IT service desk (mailto:servicedesk@itc.rwth-aachen.de)

APPTAINER

Documentation

Containers – Handling Missing Packages

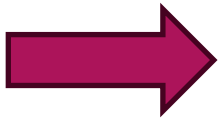
HPC.NRW

- **Reminder:** Containers are read-only after they are built
- **Question:** What happens if packages are missing in container?

- Sample Code:

```
#!/usr/bin/env python3
import matplotlib.pyplot as plt
import numpy as np
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)

# maybe some pytorch related code
fig, ax = plt.subplots()
ax.plot(t, s)
ax.grid()
fig.savefig("testplot.png")
print("I'm done!")
```



```
# loading the tensorflow container
module load TensorFlow/nvcr-24.01-tf2-py3

# run a shell in the container
apptainer shell -e ${TENSORFLOW_IMAGE}

# execute the sample in the container
Apptainer> python3 sample.py

Traceback (most recent call last):
  File "sample.py", line 3, in <module>
    import matplotlib.pyplot as plt
ModuleNotFoundError: No module named 'matplotlib'
```

Containers – Handling Missing Packages

HPC.NRW

- **Solution 1:** Clear separation of concerns. Different containers for different tasks
- **Solution 2:** Workaround to make additional packages available to container

- How does that work?
  - Option 1: Install additional required packages in user space

```
pip install --user matplotlib
```

  - Option 2: Append `PYTHONPATH` with a venv holding those packages

    Note: Can lead to inconsistencies e.g., if Python versions differ

```
# loading the tensorflow container
module load TensorFlow/nvcr-24.01-tf2-py3

# run a shell in the container
apptainer shell -e ${TENSORFLOW_IMAGE}

# install missing package in users home directory
pip install --user matplotlib

# execute the sample in the container
Apptainer> python3 sample.py
I'm done!
```

Backup Slides

Containers in HPC

HPC.NRW

- Docker is usually the container engine out there. But it is not well suited for HPC systems
  - Containers need to integrate neatly with the host environment
    - MPI and special hardware needs to be accessible in the container
    - Docker does not meet those requirements
  - Security risks in multi-user environments

- Other container engines: Shifter and Singularity
  - Especially developed for HPC
  - Supports using GPUs, interconnects and multi-node setups
  - Avoid root-privileged processes
  - Conversion for Docker images available

➢ **We use Singularity on CLAIX since 2019**

Examples: Virtual Environments (venv)

HPC.NRW

```
# load default python module
module load Python

# make sure virtualenv is installed
python3 -m pip install --user virtualenv

# create a new environment named myenv1 (will be created in current working directory)
python3 -m virtualenv myenv1

# activate that environment. Note: after activation environment name is usually shown
source myenv1/bin/activate

# install packages in that environment
python3 -m pip install python-dev-tools matplotlib
pip3 install python-dev-tools matplotlib

# list installed packages in environment
python3 -m pip list -v
pip list -v

# deactivate environment again
deactivate
```

Examples: Virtual Environments (conda)

HPC.NRW

- Download and install Miniconda
  - Download: https://docs.conda.io/en/latest/miniconda.html
  - Installation: https://docs.conda.io/projects/conda/en/latest/user-guide/install/linux.html

```
# create a new environment named myenv1 (conda saves envs at a central location)
conda env create -n myenv1
conda config --set pip_interop_enabled True # allow installing packages with pip
conda env list                              # list existing environments

# activate that environment. Note: after activation environment name is usually shown
conda activate myenv1

# install packages in that environment
conda install python-dev-tools matplotlib
pip install python-dev-tools matplotlib

# list installed packages in environment
conda list

# deactivate again
conda deactivate
```