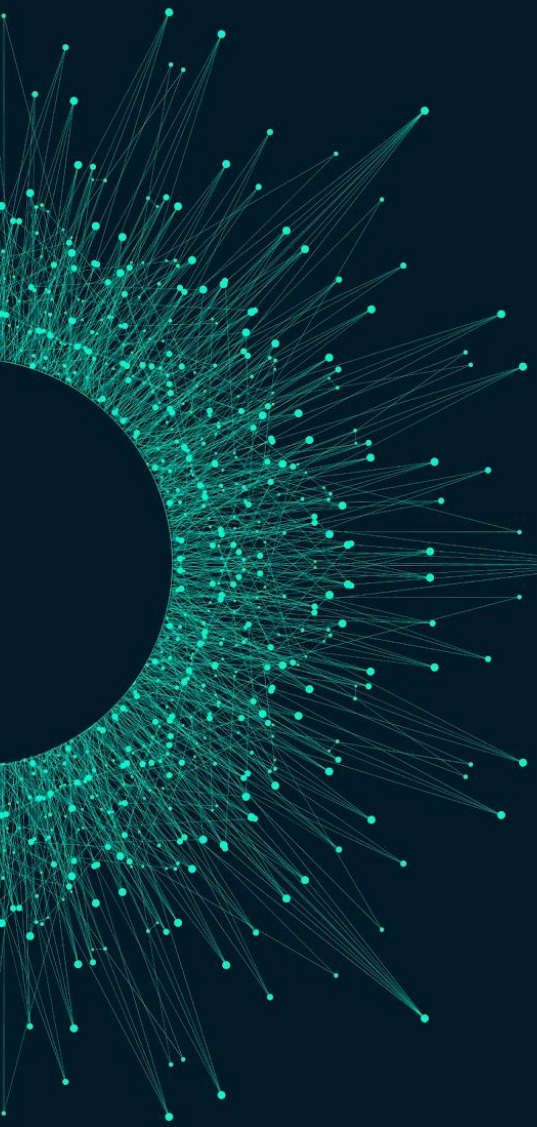


Containers as a Self-Contained Environment for HPC

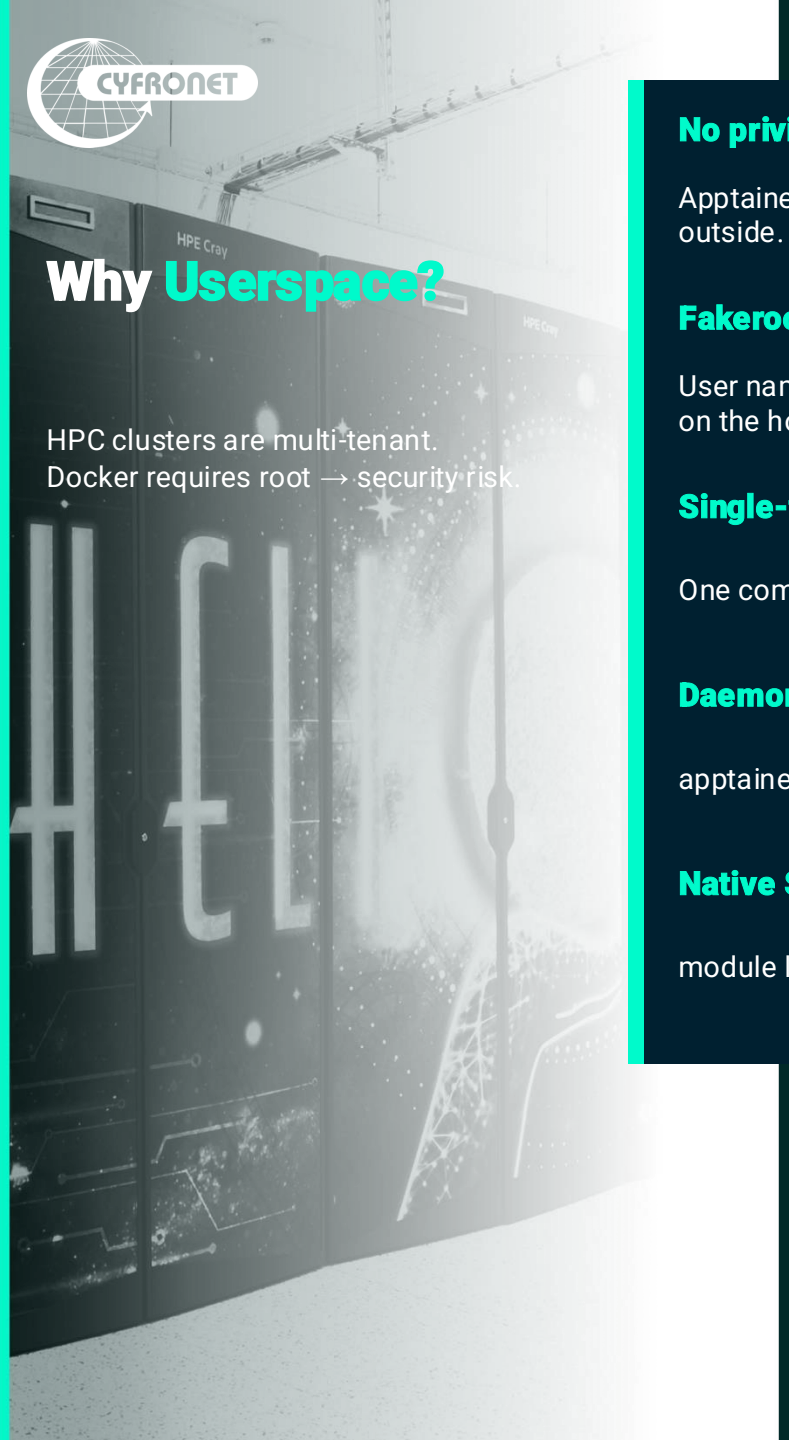
KUKDM 2026 · Zakopane, 13–15 April 2026

What Is a Container?

Why userspace in HPC? — Containers vs. VMs — Apptainer architecture

Containers vs. Virtual Machines

Container	Virtual Machine
Shares the host kernel — no hypervisor	Own kernel via hypervisor (KVM, VMware)
Starts in milliseconds, near-zero overhead	Boot time: seconds to minutes
Single portable SIF file (Apptainer format)	Large image files (VMDK/QCOW2, GB range)
Runs as your own UID — no privilege escalation	Full root access inside the VM
Native SLURM integration — no daemon needed	Requires hypervisor on compute nodes
Supports GPU passthrough (--nv flag)	GPU pass-through complex to configure
Standard in HPC: Apptainer / Singularity	Rarely used on shared HPC clusters



Why Userspace?

HPC clusters are multi-tenant.
Docker requires root → security risk.

No privilege escalation

Apptainer runs containers as your own UID. The process inside the container has exactly the same permissions as outside.

Fakeroot mode

User namespaces simulate UID 0 inside the container for build-time package installation — without granting real root on the host.

Single-file SIF format

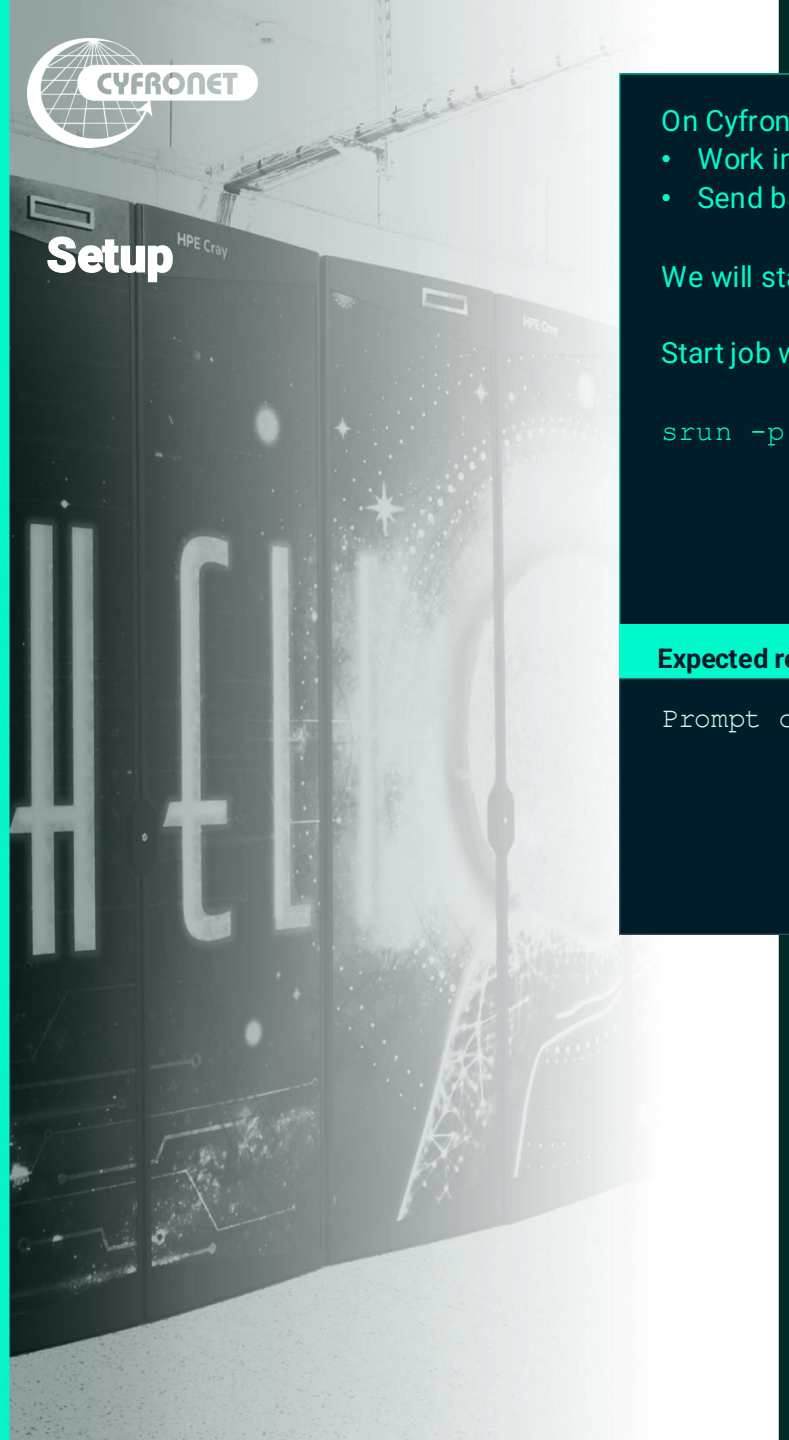
One compressed SquashFS image file. Easy to copy, checksum, and cache on scratch storage.

Daemon-free execution

apptainer exec runs like any other binary. No background service required, no socket to attack.

Native SLURM integration

module load apptainer is all you need. SIF images work inside any batch script without special configuration.



Setup

On Cyfronet clusters Apptainer is available only on compute nodes

- Work interactively via interactive job
- Send batch job to the SLURM queueing system

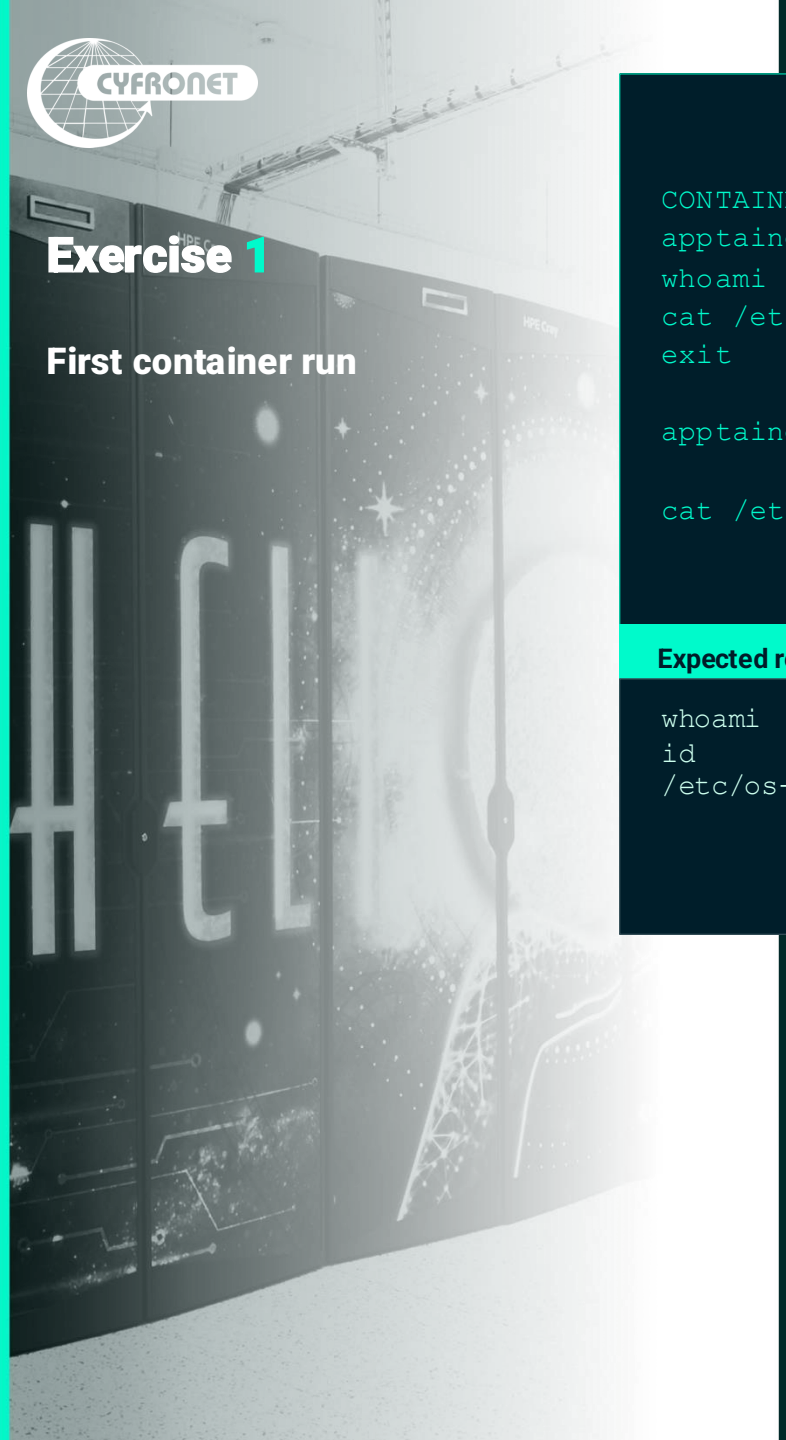
We will start hands-on in interactive job

Start job with srun

```
srun -p cpu -N 1 -n 4 -A tutorial -C memfs --time=0-1:00 --pty /bin/bash -l
```

Expected result

```
Prompt on compute node [ares][tutorialXXX@ac0YYY]$
```



Exercise 1

First container run

```
CONTAINER=/net/pr2/projects/tutorial/2026-04-13-kukdm2026-containers/ubuntu.sif
apptainer shell $CONTAINER
whoami && id
cat /etc/os-release
exit

apptainer exec $CONTAINER cat /etc/os-release

cat /etc/os-release
```

Expected result

```
whoami  → your cluster login (not root)
id      → uid matches your host UID
/etc/os-release shows Ubuntu 22.04 regardless of host OS
```

Filesystems & SLURM

Binding host paths — Running containers on a single node through SLURM

Binding Filesystems

Default auto-mounts: $HOME, $PWD, /tmp, /proc, /sys, /dev

HPC paths (scratch, project storage) need explicit --bind

--bind flag

```
# Single bind
apptainer exec \
  --bind $SCRATCH:/data \
  $CONTAINER ls /data


# Persistent (session-wide)
export APPTAINER_BINDPATH=\
  $SCRATCH:/data
```

SLURM pattern

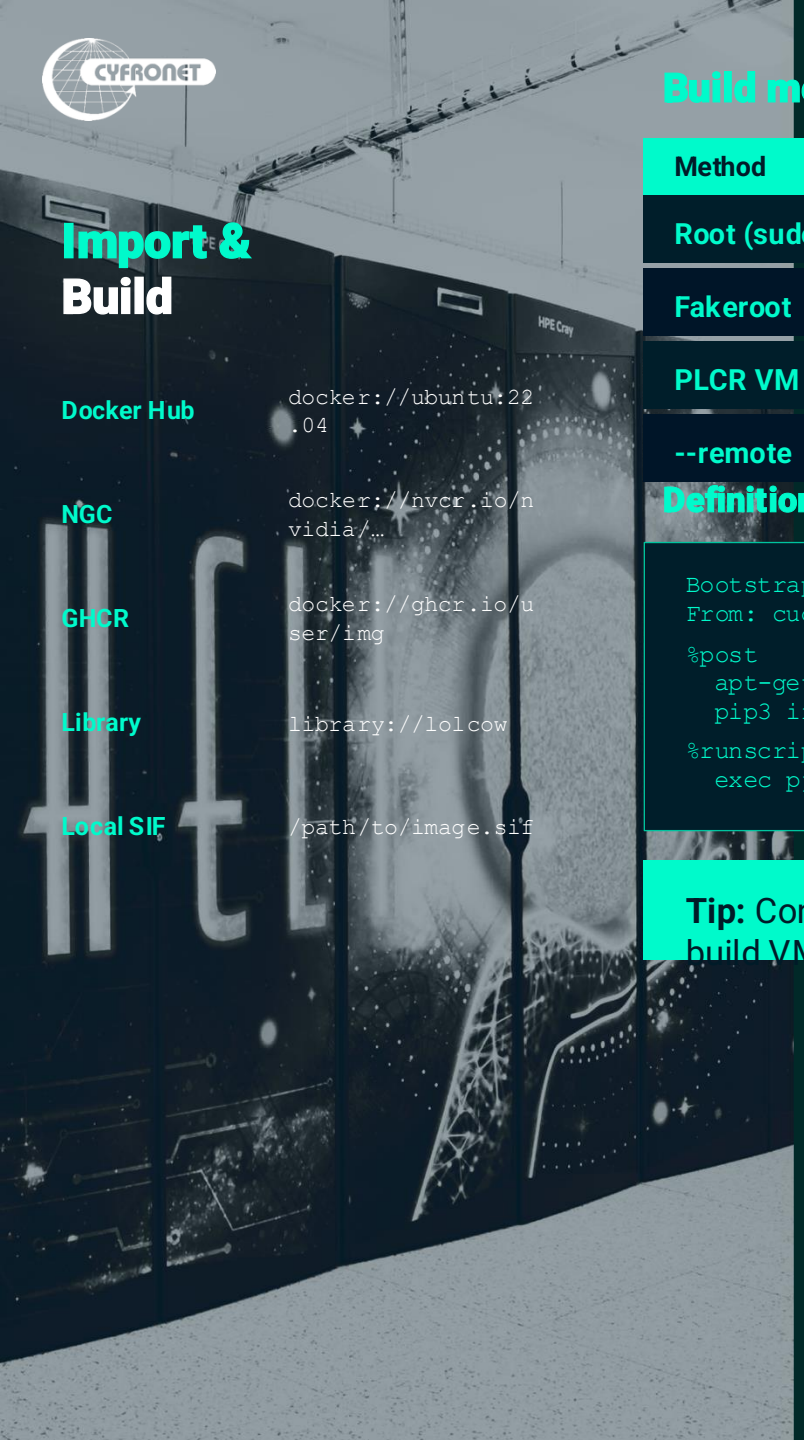
```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --cpus-per-task=4
#SBATCH --partition=\
         plgrid-testing

apptainer exec \
  --bind $SCRATCH:/data \
  image.sif python3 script.py
```

Best practices:

- Store SIF files on $SCRATCH (Lustre, fast I/O)
- Set APPTAINER_TMPDIR=$SCRATCH to avoid home quota issues.
- Verify before submit: apptainer inspect image.sif

Import & Build

Docker Hub — NGC — Apptainer Library — .def files — fakeroot — PLCR — VM



Import & Build

Docker Hub	`docker://ubuntu:22.04`
NGC	`docker://nvcr.io/nvidia/…`
GHCR	`docker://ghcr.io/user/img`
Library	`library://lolcow`
Local SIF	`/path/to/image.sif`

Build methods

Method	On Ares	Limitations
Root (sudo)	No (HPC)	Requires VM/dev server
Fakeroot	Yes*	No kernel mounts, limited net
PLCR VM	On request	Scheduling needed
--remote	Yes (Sylabs)	Size limits, privacy concerns

Definition file (.def)

```
Bootstrap: docker
From: cuda:12.4.1-devel-ubuntu22.04

%post
  apt-get install -y python3-pip
  pip3 install numpy scipy

%runscript
  exec python3 "$@"
```

```
# build:
apptainer build --fakeroot \
  myimage.sif myimage.def

# pull:
apptainer pull
  myimage.sif repository.url
```

Tip: Contact helpdesk@cyfronet.pl or the PLGrid portal to request PLCR build VM (root access)

Block 4 GPU Containers

NVIDIA GPU Cloud (NGC) registry — --nv driver injection — CUDA compatibility

NVIDIA GPU Cloud (NGC) — nvcr.io

Container	nvcr.io URI (example tag)	HPC Use-case
PyTorch	nvcr.io/nvidia/pytorch:24.05-py3	DL training / inference
TensorFlow	nvcr.io/nvidia/tensorflow:24.05-tf2-py3	ML training / inference
HPC SDK	nvcr.io/nvidia/nvhpc:24.5-devel-cuda12.4-…	OpenACC, MPI, Fortran/C++
LAMMPS	nvcr.io/nvidia/lammps:29Sep2021	Molecular dynamics
GROMACS	nvcr.io/hpc/gromacs:2023.2	Biomolecular simulation
RAPIDS	nvcr.io/nvidia/rapidsai/base:24.06-…	GPU data analytics

Note: The CUDA Toolkit version inside the container must be ≤ the maximum CUDA version supported by the host driver.
Check:
`nvidia-smi | head -3` before pulling an NGC image.

--nv GPU passthrough

Injects host NVIDIA driver libs at runtime
Container needs CUDA Toolkit only (no driver)
libcuda.so and libnvidia-*.so auto-bound
GPU visible natively — no overhead

```
# SLURM: request GPU
#SBATCH --gpus=1
#SBATCH --partition=plgrid-gpu

# Run with GPU passthrough
apptainer exec --nv \
  $SCRATCH/pytorch.sif \
  nvidia-smi
```

Exercise 4

NGC PyTorch GPU benchmark

Objective: Pull NGC PyTorch container, conf
GPU visibility, run CUDA matrix multiply via
SLURM.

```
# 1. Set cache directory
export APPTAINER_CACHEDIR=$SCRATCH/.apptainer_cache

# 2. Pull NGC PyTorch (~8 GB)
apptainer pull $SCRATCH/pytorch.sif \
    docker://nvcr.io/nvidia/pytorch:24.05-py3

# 3. Quick version check (no GPU)
apptainer exec $SCRATCH/pytorch.sif \
    python3 -c "import torch; print(torch.__version__)"

# 4. SLURM script (save as run_gpu.sh)
#SBATCH --gres=gpu:1 --partition=plgrid-gpu
apptainer exec --nv \
    --bind $SCRATCH/training:/work \
    $SCRATCH/pytorch.sif \
    python3 /work/gpu_bench.py
```

Expected: PyTorch version, CUDA available: True, GPU name (e.g. NVIDIA A100 80GB), matrix multiply OK

MPI & NCCL Across Nodes

Hybrid-MPI pattern — NGC HPC SDK container — InfiniBand & NCCL multi-node

MPI strategies for containers

Hybrid-MPI (recommended)

srun / mpirun on the HOST, container per rank

MPI in image must be ABI-compatible with host

InfiniBand/UCX provided by host MPI stack

Low complexity — standard SLURM pattern

Use this by default on Ares

```
module load openmpi/4.1.x
srun apptainer exec \
  --bind $SCRATCH:/work \
  $SCRATCH/nvhpc.sif \
  /work/hello_mpi
```

SLURM: --nodes=2 --ntasks-per-node=4

Bind-mount MPI

mpirun launched INSIDE the container

MPI in image may be a different build

Host IB libs must be bind-mounted manually

High complexity — avoid unless portability is critical

Use when image must be self-contained

NCCL multi-GPU tip:

NGC pytorch:24.05-py3 includes NCCL 2.x.
Set NCCL_DEBUG=INFO for diagnostics.
Set NCCL_IB_DISABLE=0 for InfiniBand.

NGC HPC SDK container:

nvcr.io/nvidia/nvhpc:24.5-devel-cuda12.4-ubuntu22.04
Includes OpenMPI + UCX for Ares InfiniBand fabric.

CYFRONET Command Cheat Sheet

Running containers

```
apptainer shell image.sif
```
Interactive shell inside container

```
apptainer exec image.sif cmd
```
Run a single command

```
apptainer run image.sif
```
Run default runscript

```
apptainer exec --nv image.sif cmd
```
Run with GPU passthrough

```
apptainer exec --bind /h:/c image.sif
```
Bind host path /h to /c

```
apptainer exec --cleanenv image.sif
```
Clear host env vars

```
apptainer pull img.sif docker://…
```
Pull from Docker Hub

```
apptainer pull img.sif docker://nvcr.io/…
```
Pull NGC container

```
apptainer inspect --labels image.sif
```
Show image metadata

```
apptainer build --fakeroot a.sif a.def
```
Build from .def file

Env vars, SLURM & NGC

APPTAINER_CACHEDIR=$SCRATCH/apptainer-cache → cache dir (avoid $HOME)

APPTAINER_TMPDIR=$SCRATCH/apptainer-tmp → tmp dir for build

APPTAINER_BINDPATH=/a:/b → persistent binds

NCCL_DEBUG=INFO → NCCL communication diagnostics

NCCL_IB_DISABLE=0 → enable InfiniBand for NCCL

SLURM GPU directives:

```
#SBATCH --gres=gpu:1
#SBATCH --partition=plgrid-gpu
#SBATCH --nodes=2  --ntasks-per-node=4
```

Thank you

Questions & discussion

Apptainer docs: https://apptainer.org/docs/user/latest/
NGC Catalog: https://catalog.ngc.nvidia.com
Cyfronet docs: https://docs.hpc.cyfronet.pl/
PLGrid Guide: https://guide.plgrid.pl/
PLGrid Portal: https://portal.plgrid.pl/
Training page: https://events.plgrid.pl/