# Onboarding

## Julia on HLRS Clusters/Laptops

## Most of the course can be completed on the laptops.

- Equipped with NVIDIA GPU
- Jupyter + VS Code
    - `jupyter lab`
    - `code`
- Course materials
    - `$HOME/JuliaHLRS24`

## HLRS Training Cluster

# The cluster training.hlrs.de has two types of nodes.

CPU nodes

- “skl”
- 2x Intel Skylake
- 40 cores total

GPU nodes

- “clx-ai”
- 2x Intel Cascade Lake
- 36 cores total
- 8x NVIDIA V100

## Jobs are scheduled with PBS Pro.

- Submit a job:
  - `qsub job_script.sh`
- Check on your queued/running jobs:
  - `qstat -nw`

## VS Code $\rightarrow$ HLRS Cluster

## Run VS Code on a cluster node via SSH.

### Login node

- Works fine, just connect to
  - `accountname@training.hlrs.de`

### Compute node

- At HLRS, possible but inconvenient
  - `SetEnv PBS_JOBID=…`
  - `SSH ProxyJump`

**We will stay on Login nodes for the course.**

## Julia on the Cluster

## Use a system module or standard Julia binaries.

- Modules on the HLRS training cluster
  - `module use julia`
  - `module use nvidia/nvhpc     # MPI+CUDA`
  - `module use compiler/nvidia # MPI+CUDA`

- If there is no (working) system module, use standard binaries provided by `juliaup`.

## Put the Julia depot on the parallel file system.

Already taken care of on HLRS training cluster!

- Julia depot = where Julia stores stuff
  - packages
  - binary dependencies
  - …
- Environment variable: `JULIA_DEPOT_PATH`
- Why not `$HOME`?
  - Quotas
  - Read-only from compute jobs (sometimes)

## Need a Julia wrapper for the Julia VS Code extension.

- Julia: Executable Path should point to a wrapper script, like this one:

```bash
#!/bin/bash
[...]

# Load modules
module load julia
module load nvidia/nvhpc
Module load compiler/nvidia

# Act like Julia (i.e. pass on all arguments)
exec julia "${@}"
```

## Let's try it!

→ exercises/Day1/1_cluster_onboarding/