Cineca HPC Quick Start Guide (Leonardo Cluster)

Livia Oddi

Contents

Accessing Cineca

Step 1: Provision via STEP

Every 12 hours, you must re-authenticate through Cineca's STEP system:

```
step ssh login your.username@studenti.uniroma1.it --provisioner cineca-hpc
```

Step 2: Connect to Leonardo

Once provisioned, you can log in to Leonardo using:

```
# Generic login
ssh <username>@login.leonardo.cineca.it

# Or use specific login nodes:
ssh <username>@login01-ext.leonardo.cineca.it
ssh <username>@login02-ext.leonardo.cineca.it
ssh <username>@login05-ext.leonardo.cineca.it
ssh <username>@login07-ext.leonardo.cineca.it
```

File transfer

You can upload files to Cineca either from your local computer or from Google Drive, depending on what's more convenient.

From your PC to Cineca (via `scp`)

This method is run from your local terminal (PowerShell, Git Bash, or Linux shell).

Upload of a full folder

```
# Optional: clear old ssh key
ssh-keygen -R login.leonardo.cineca.it

# Upload a local folder to Cineca
scp -r "path\to\local\project\folder" <username>@login.leonardo.cineca.it:/
    leonardo/home/userexternal/<username>
```

Update a specific file

```
# Update just the test.sh file inside your project folder on Cineca
scp "path\to\local\file" <username>@login.leonardo.cineca.it:/leonardo/home/
    userexternal/<username>/project file name/test.sh
```

From Google Drive to Cineca (via `gdown`)

You can also upload files by downloading them from a public Google Drive link.

Install gdown

```
pip install gdown
```

Download a folder

```
gdown --folder --fuzzy https://drive.google.com/drive/folders/<your-folder-id>
```

Download a single file

```
gdown --fuzzy https://drive.google.com/file/d/<file-id>/view?usp=sharing
```

From Cineca to your PC (download results)

This command is run from your local machine (not inside Cineca). It downloads your results folder from Cineca to your computer:

```
scp -r <username>@login01-ext.leonardo.cineca.it:/leonardo/home/userexternal/<
    username>/project name/results "/your/local/folder/path/results"
```

Conda Environment Setup

1. Load conda module

```
module load anaconda3
# if it fails, e.g. tells that module conda is not present, try:

module purge
module load anaconda3
conda --version
```

2. Create and activate virtual environment

```
conda create -n myenv -c conda-forge python=3.11
source activate myenv
```

3. Install from requirements.txt

```
cd <your project file name>
conda create -n myenv --file requirements.txt -y
```

4. Check Installed Python Libraries and Parameters

After setting up your environment, it's often useful to check the versions of key Python libraries you've installed. You can do this using the Python command line.

Example (general approach):

```
python -c "import LIBRARY_NAME; print(LIBRARY_NAME.__version__)"
```

Replace `LIBRARY_NAME` with the actual name of the library you want to check.

For multiple libraries, you can chain them like this:

```
python -c "import library1, library2; print(library1.__version__, library2.
    __version__)"
```

To inspect available parameters of a class (e.g. HuggingFace's `TrainingArguments`):

```
python -c "from transformers import TrainingArguments; print(TrainingArguments.
    __init__.__code__.co_varnames)"
```

1 SLURM Job Management

Submit a Job

Make sure your script uses Unix-style line endings (LF) instead of Windows (CRLF).
If you're using VSCode, you can change this by clicking on the line ending selector in the bottom-right corner of the window and choosing "LF".

```
sbatch test.sh
```

Monitor Job Queue

```
squeue -u $USER
watch squeue -u $USER #to exit from the view you can simply do: CTLR+C
```

Check completed or failed jobs

```
#check the status of completed or failed jobs
sacct -u $USER

#get information on a completed job
sacct -j <job_id> --format=JobID,State,Elapsed,MaxRSS,Start,End
```

Running job

```
#how many job are before yours
squeue -p boost_usr_prod

#running job info
scontrol show job <job_id>
```

Check Queue Priority

```
sprio -u $USER
```

Check GPU

```
#check GPU availability
sinfo -p boost_usr_prod -o "%P %D %C %m %G"

#check if I am on a GPU node
head -n 10 logs/haiku_tuning.out #(-n -> on test.sh where I have "#SBATCH --
    output=logs/project name.out)
```

List all available partitions

You can display all available partitions with:

```
sinfo -o "%P %a %l %F"
```

Log Monitoring and Output

Live log monitor

```
#to monitor the % of the job execution
tail -f logs/haiku_tuning.out #to be executed in another ssh authenticated
    PowerShell window
```

View Script Output

```
cat logs/haiku_tuning.out
```

Filesystem Commands

```
#list all files with details
ls -la

#create new folder
mkdir myfolder

#delete folder and contents
cd ~
rm -rf myfolder

#delete folder content
find /path/to/folder -mindepth 1 -delete
```

Quota and Project Information

```
saldo -b    # Check remaining compute hours / budget
```

2 Recommended Project Structure

```
haiku_project/

          project_name.py             # Main training script
          test.sh                     # SLURM job script
          requirements.txt            # Conda requirements
          train_df.jsonl              # Training dataset
          tokenizer_model/            # Tokenizer files
          logs/                       # Training logs
          results/                    # Output models/results
```

3 Tips and Best Practices

- Always run `module purge` before loading new environments.
- Use `LF` line endings for SLURM scripts (convert in VSCode if needed).
- Prefer GPU queues only if your job requires it.
- Clean your folders periodically to save space.
- Use `sacct` and `scontrol` to debug failed jobs.