Cluster Cheat Sheet

1 Preliminaries

- The cluster runs on SLURM job management system.
- You access the cluster through ssh via a public-private key setup.
- We use apptainer to create environment containers which we can load as a single block of data, greatly alleviating the strain on the filesystem for everyone.
- You will be given a directory in which you can work
- The head node through which you 'enter' the cluster is purposefully under-resourced as it serves as a staging ground to specific high performance nodes.

1.1 Getting Access to Cluster

- Check for your public key at `cat ~/.ssh/id_rsa.pub`
- If not present, create a private-public key pair via this tutorial
- Send your **full name**, **desired username** and **public ssh key ending with .pub (Not the private!)** to our cluster admin the email of whom we will publish at an appropriate time.
- Upon confirmation you can access the cluster with `ssh username@hydra.ml.tu-berlin.de`

2 Setting Up Environment

- In your directory, create a `requirements.txt` file to which you can obviously add additional packages via their `pip install` name

```
--extra-index-url https://download.pytorch.org/whl/cu118
numpy
torch
torchvision
lightning
matplotlib
pandas
```

- In the same level of the directory (for relative paths) create a `pml.def` file while paying attention to the tab indentation

```
Bootstrap: docker
From: python:3.10

%files
        $PWD/requirements.txt requirements.txt

%post
        pip install --root-user-action=ignore -r requirements.txt
        pip install --root-user-action=ignore scikit-learn
```

- Obtain a node on the cpu only partition with enough run time `srun --partition=cpu-2h --pty bash`
- Build the container with `apptainer build pml.sif pml.def` in the directory where your container definition resides which takes **10-15 minutes**

3 Running Jobs

- We can run a python script through a container by first requesting a node on the GPU partition where it's important to request a GPU alongside the request of a job on a GPU partition

```
srun --partition=gpu-test --gpus=1 --pty bash
```

- Once on the node we can run a python script with the command

```
apptainer run --nv pml.sif python -c "import torch; print(torch.cuda.is_available())
```

```
apptainer run --nv pml.sif python my_little_script.py
```

where the flag `--nv` requests a GPU

- For running multiple in parallel, we require a shell script with the necessary commands

```
#!/bin/bash -l

#SBATCH --job-name=NiceNameWVU
#SBATCH --partition=gpu-2h # Run on the 2h GPU runtime partition, also 5h, 2d and 7d availabl
#SBATCH --gpus-per-node=40gb:1 # One A100 with 40GB
#SBATCH --ntasks-per-node=4 # 4 CPU threads
#SBATCH --output=/home/path_to_output_log_file_directoryABC/%A.%a-error.txt
#SBATCH --error=/home/path_to_error_log_file_directoryABC/%A.%a-output.txt
#SBATCH --chdir=/home/path_to_root_directoryXYZ
#SBATCH --array=1-10 # run ten times

container=../path_to_container_relative_to_--chdir_directory/pml.sif

apptainer run --nv $container \
        wandb agent wandb_UserA/wandb_projectB/sweepC
apptainer run --nv $container \
        python MyNiceLittlePythonScript.py --argflag1 3 --argflag2 -10 --argflag3 abc
```

- Naturally, you can also write a launch bash script, which launches multiple SLURM scripts in parallel, instead of running sequentially through the commands of a single script.
- To look your running jobs and inspect the job ids, use `squeue -u <username>`
- To kill a pending or running job, use `scancel <job-id>`
- There are different partitions available to choose from

Name	Type	Runtime
cpu-9m	CPU	9m
cpu-2h	CPU	2h
cpu-5h	CPU	5h
cpu-2d	CPU	2d
cpu-7d	CPU	7d
gpu-9m	GPU	9m
gpu-2h	GPU	2h
gpu-5h	GPU	5h
gpu-2d	GPU	2d
gpu-7d	GPU	7d
cpu-test	CPU	15m
gpu-test	GPU	15m

and naturally, the shorter the allocated time of the partition, the faster you get a job running on that partition.

- We recommend running your job on a short partition until a meaningful checkpoint, after which all code has been run at least once, before optimizing a model on a longer partition.