“How can I use GPUs with Jupyter Notebooks on Puhti?”

It’s complicated...

GPUs are an expensive resource - bought by taxpayers - and it’s CSC’s responsibility to use them efficiently

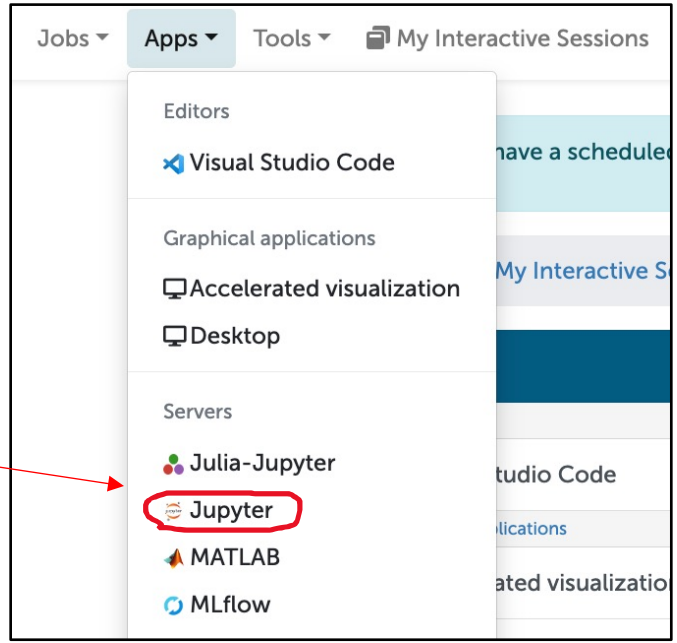
Interactive use is **very inefficient**, as most of the time you’re editing or looking at the code and not running any computation, but GPU is still reserved and cannot be used by anyone else

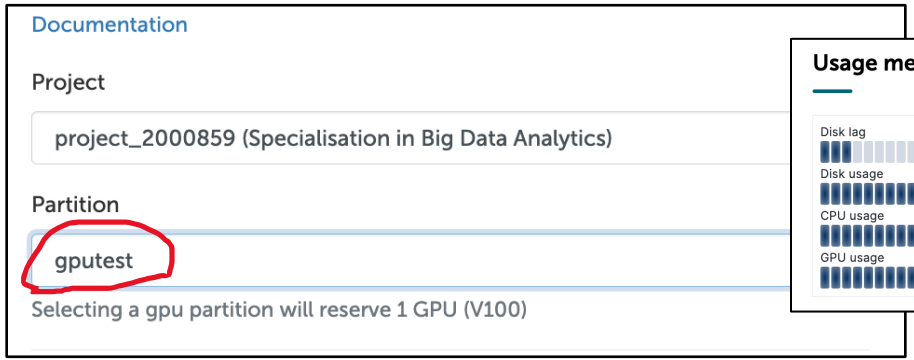
https://docs.csc.fi/support/tutorials/rstudio-or-jupyter-notebooks

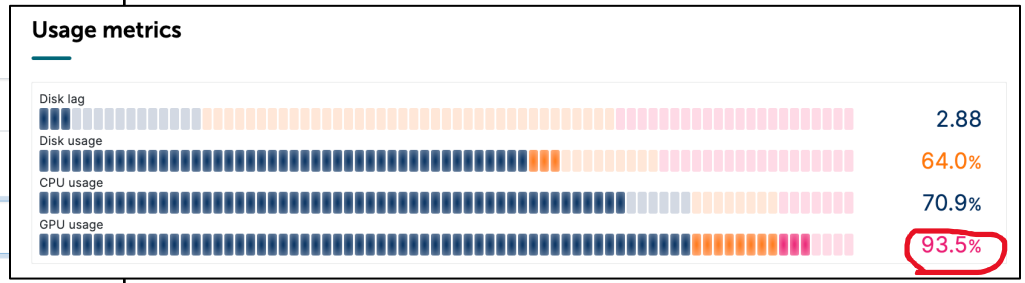
It's possible

select the `gputest` or `gpu` partition
when launching an interactive Jupyter session,
with some restrictions:

- Only 1 GPU can be used
- You might have queue for a long time...







Alternative workflow A

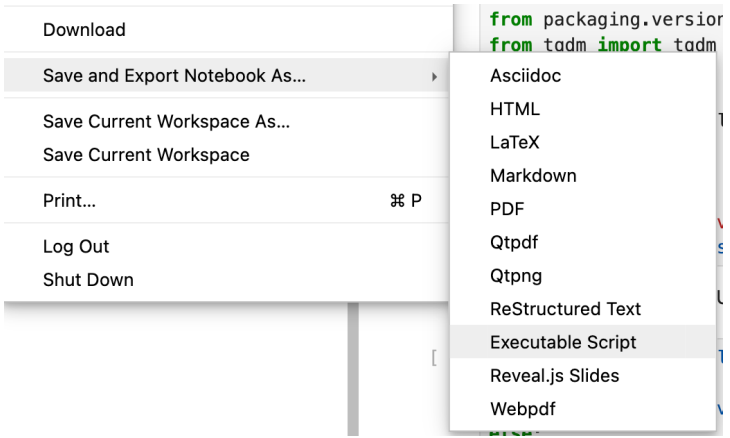
Work interactively using the **CPU interactive** partition

1. Less queueing
2. With enough CPU cores you can easily debug and test things like data loading and even small-scale training
3. Tip: with interactive you can also test “Local disk” (NVMe) for data-intensive workloads

When you’re ready for the real run:

1. Save your notebook as a **Python script**
2. Run as usual with `slurm`
   `https://docs.csc.fi/computing/running/submitting-jobs/`

Drawback: You need to go back and forth between script and Notebook



Alternative workflow B

Use the **CPU interactive** partition (as with workflow A)

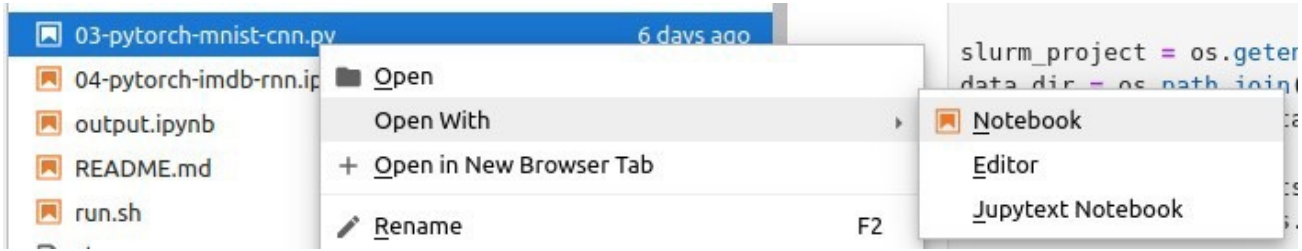
Work interactively directly with the **Python script**:

Open the script “with Notebook”

Run cells interactively, but it’s still a Python script

Submit directly with `slurm`

No going back and forth between Notebook and script



```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=10
#SBATCH --partition=gputest
#SBATCH --gres=gpu:v100:1
#SBATCH --time=0:15:00
#SBATCH --mem=32G
#SBATCH --account=project_2000859

module purge
module load ## pick your modules https://docs.csc.fi/computing/modules/
module list
export PROJ="2000859"
export DATADIR=/scratch/project_$PROJ/BDA2024/$USER

set -xv # a comprehensive view of what's happening in the script
        # can be incredibly helpful for debugging complex scripts
srun python3 $*
```