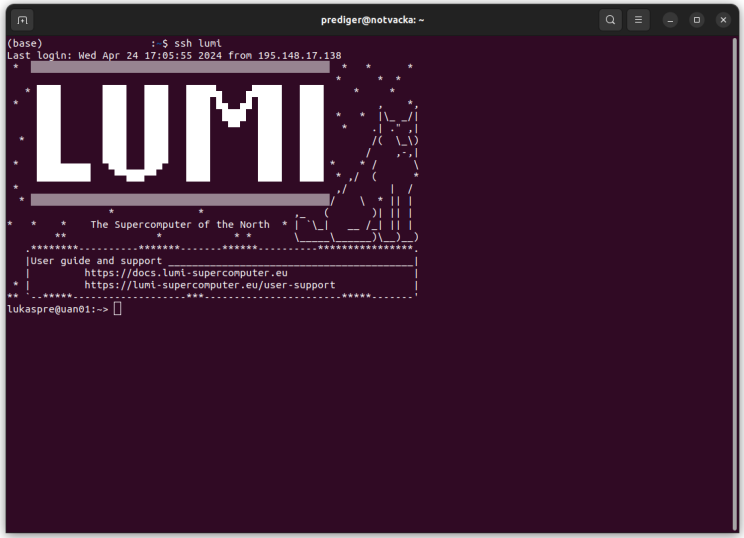
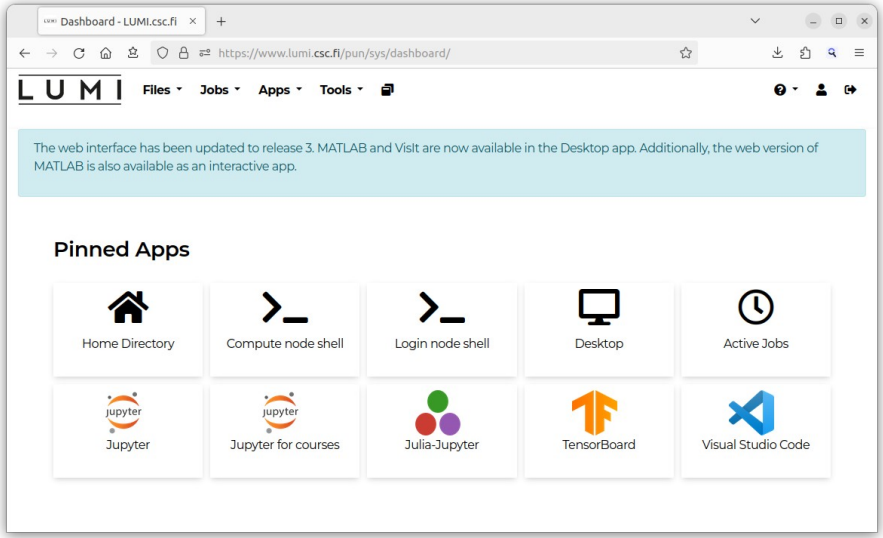
Using the LUMI web-interface

Mats Sjöberg, Lukas Prediger – CSC – IT Center for Science, Finland

Two ways of accessing LUMI

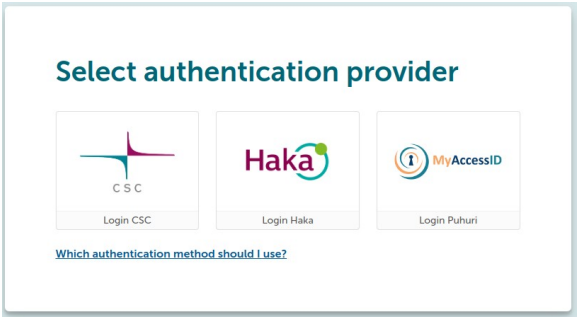
- via SSH connection
  - terminal access only
- via the LUMI web-interface
  - browser based
  - terminal + various apps

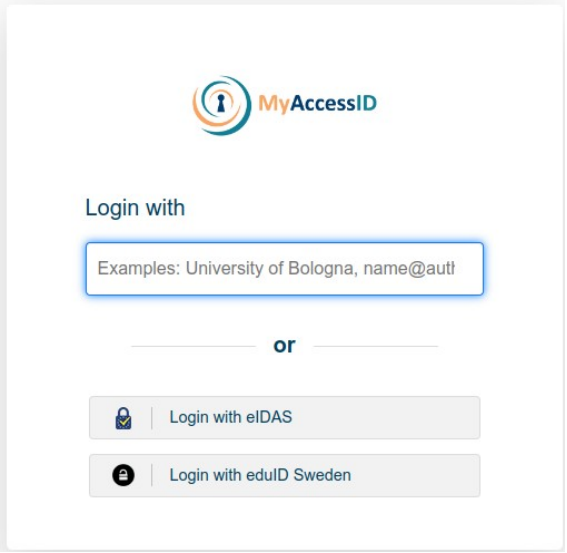


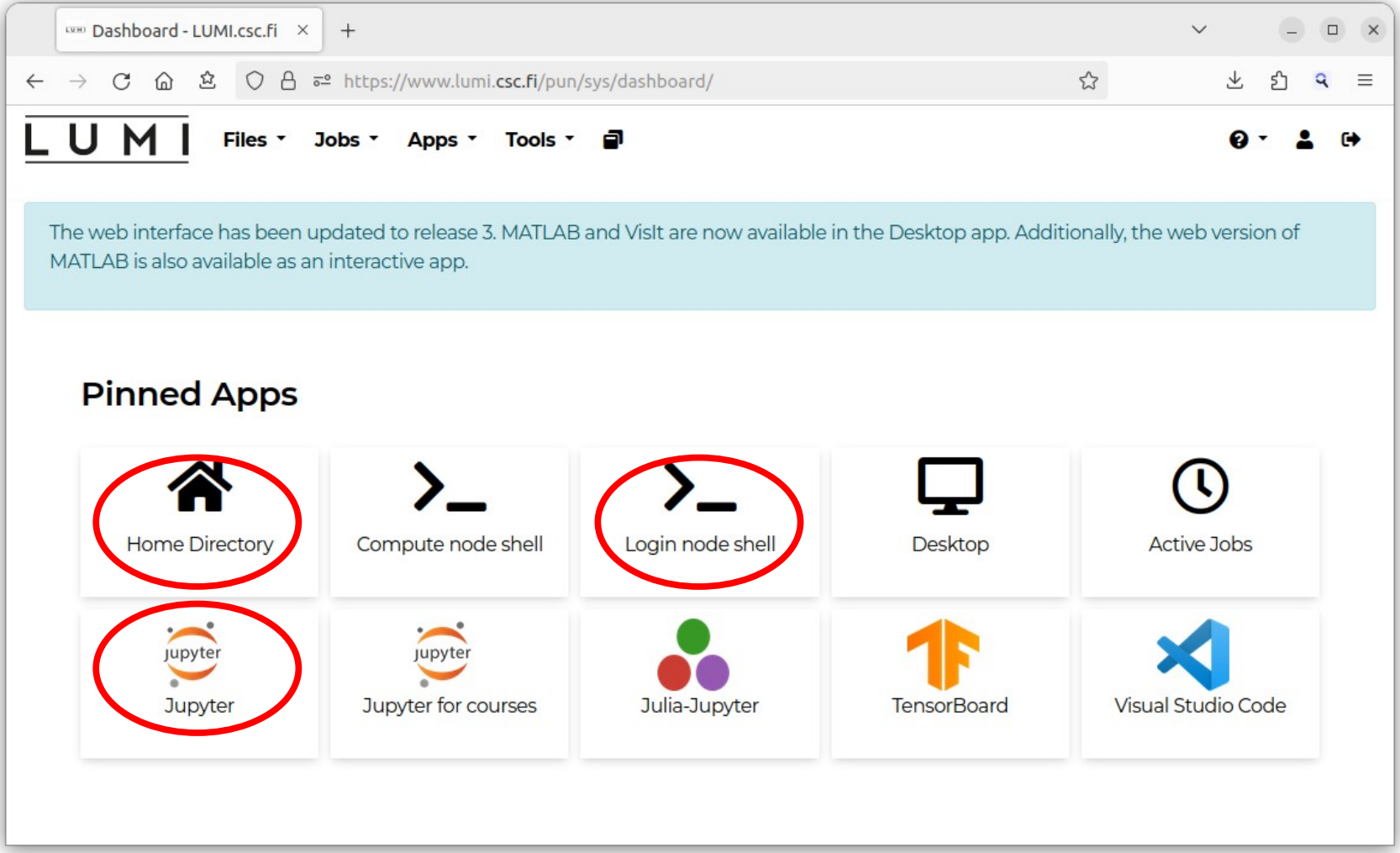


Accessing the LUMI web-interface

- Go to https://www.lumi.csc.fi
- Click “Go to login”
- Select login method
  - In most cases you should select MyAccessID / Puhuri
- MyAccessID: type the name of your institution / university
- You should be directed to your institution’s login page



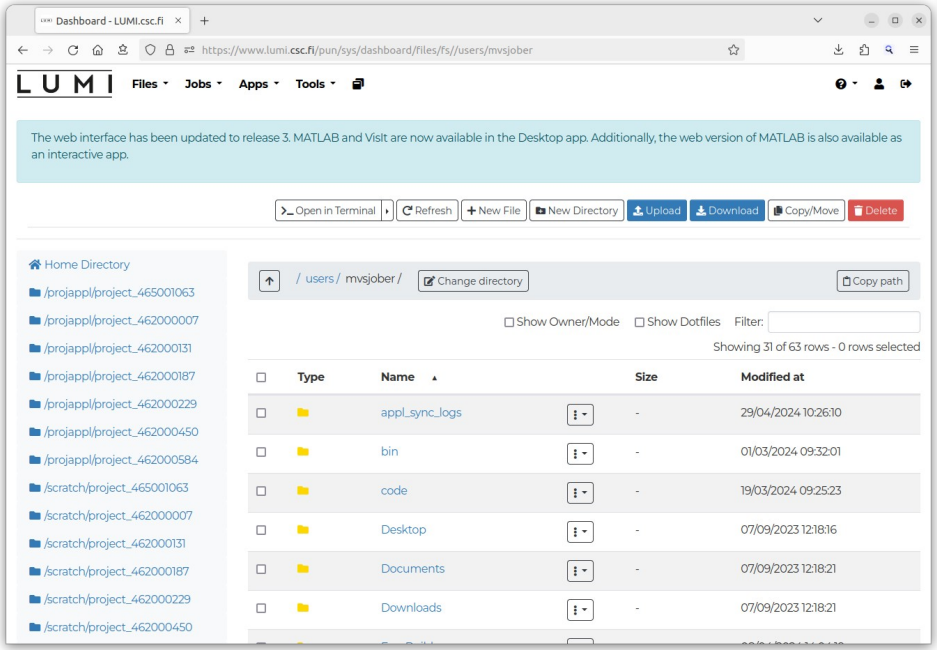


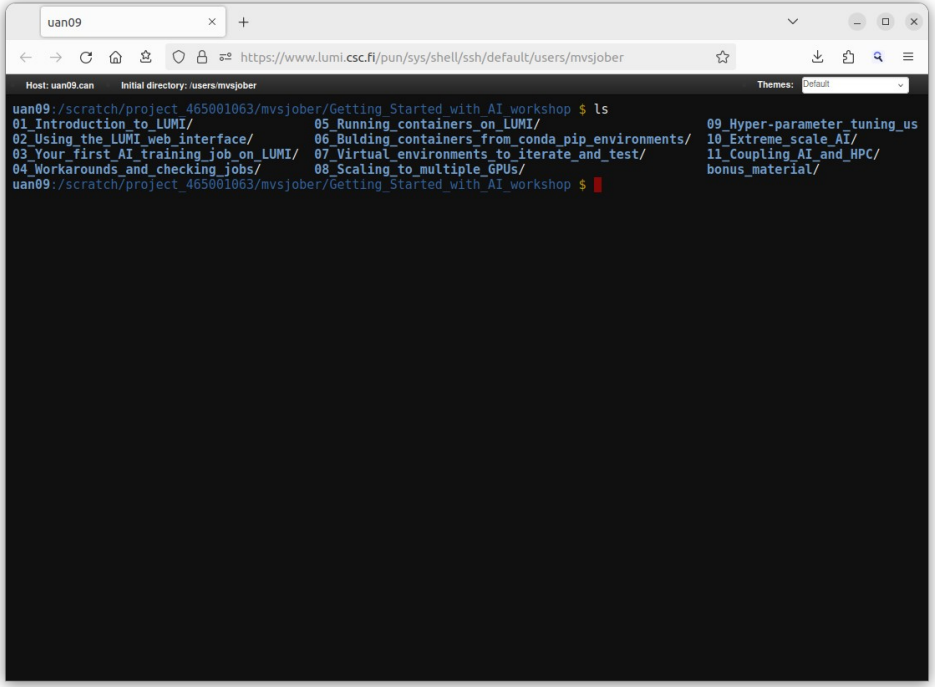

Dashboard - LUMI.csc.fi
https://www.lumi.csc.fi/pun/sys/dashboard/
LUMI
Files
Jobs
Apps
Tools
The web interface has been updated to release 3. MATLAB and VisIt are now available in the Desktop app. Additionally, the web version of MATLAB is also available as an interactive app.
Pinned Apps
Home Directory
Compute node shell
Login node shell
Desktop
Active Jobs
Jupyter
Jupyter for courses
Julia-Jupyter
TensorBoard
Visual Studio Code

Files browser and shell (login node)

Home Directory

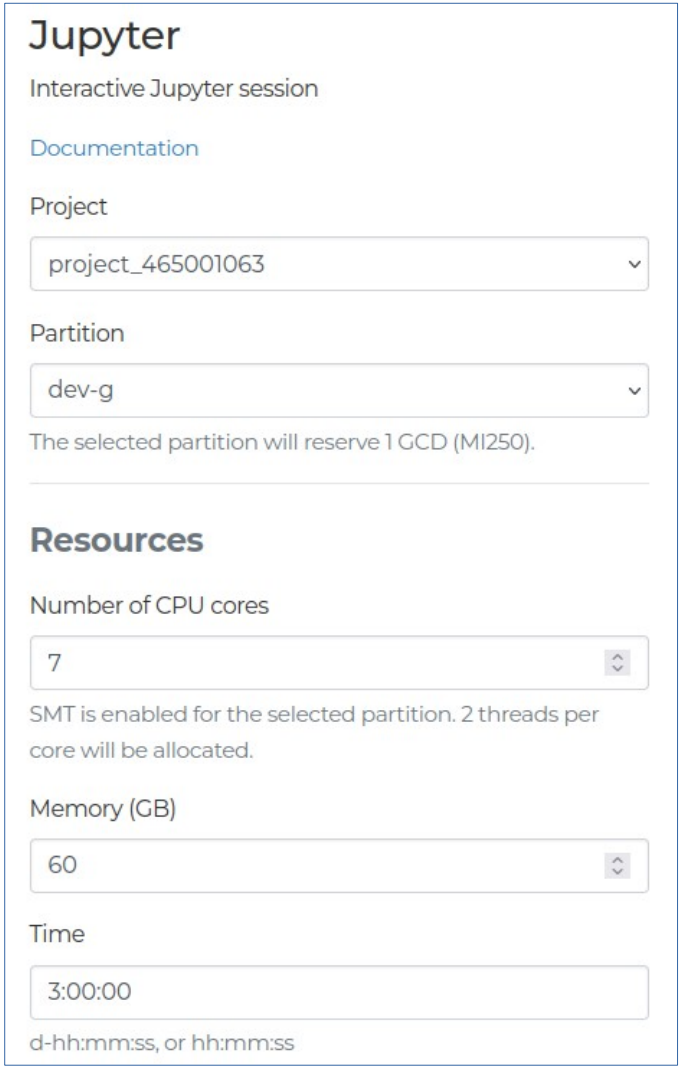
Login node shell





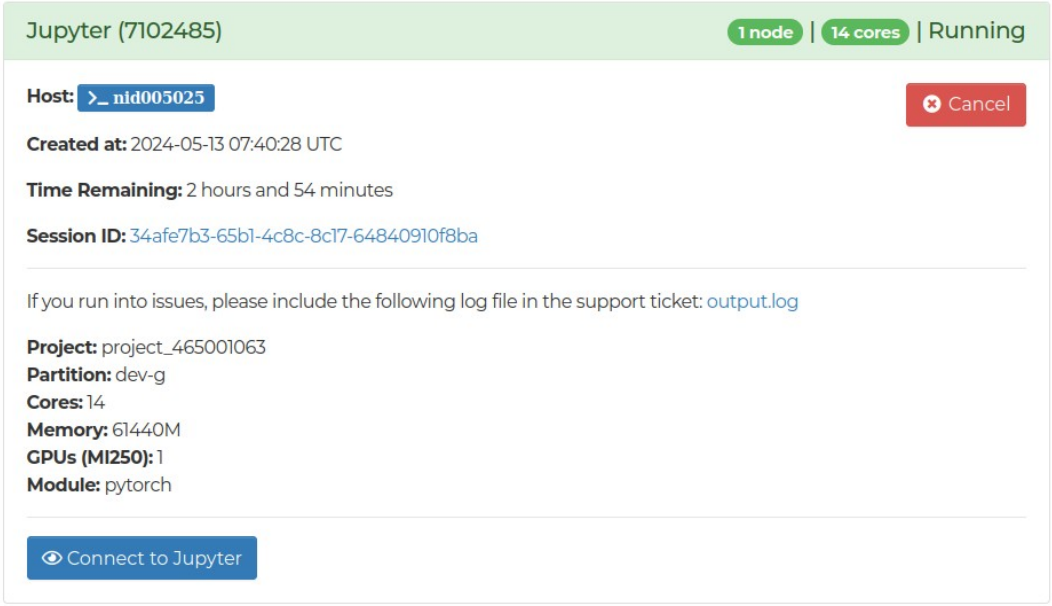
Jupyter notebook

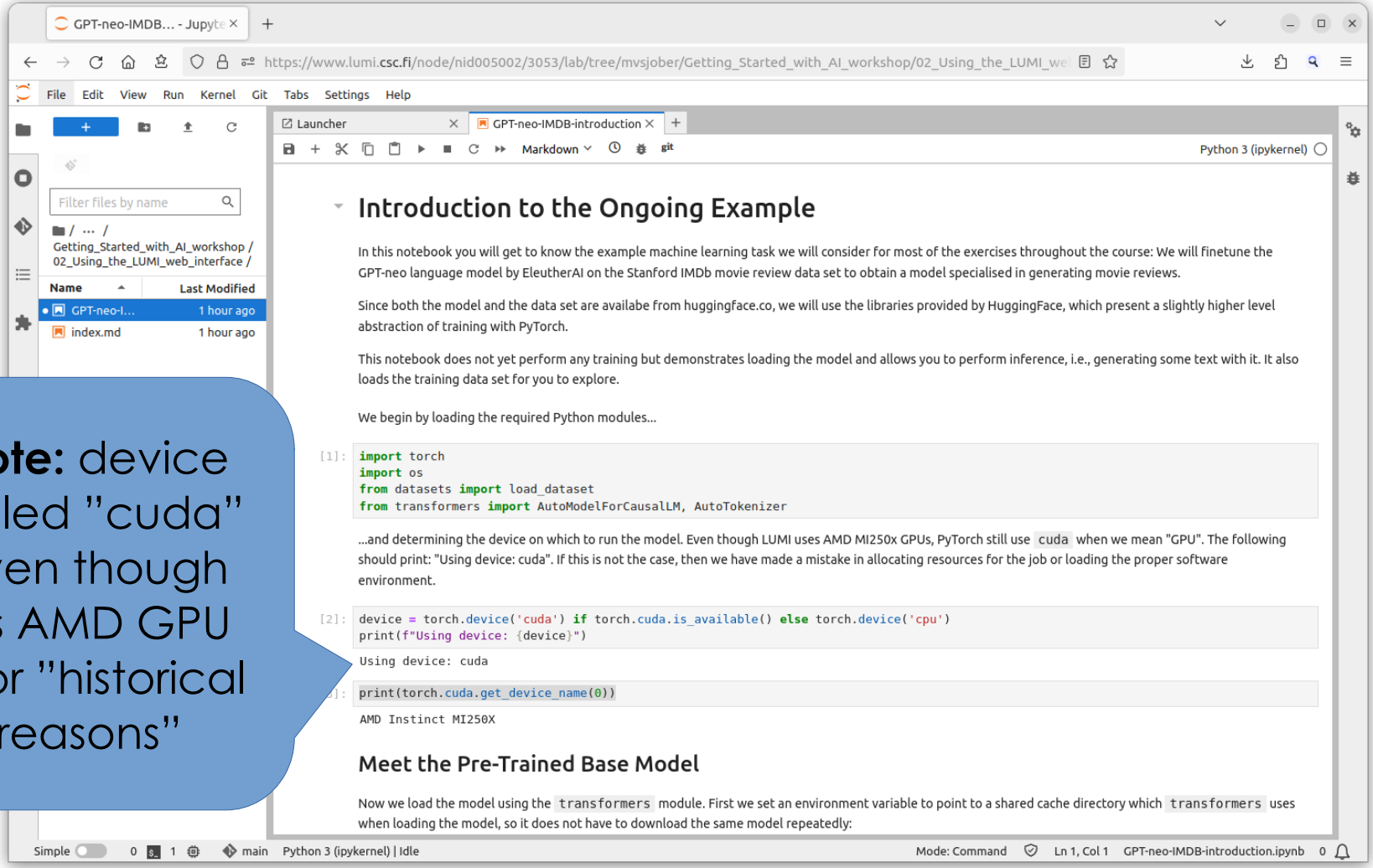
- Launching Jupyter actually launches a job in the cluster (more on that in the next lecture)
- You need to fill in some things:
  - Project
  - Partition
  - Resources
  - What Python installation to use
- Reasonable resources for a single GPU job – roughly 1/8 of a node:
  - Number of CPU cores: 7
  - Memory: 60 GB



Jupyter notebook

- The job is submitted to the normal Slurm queue
- Once it has started "Connect to Jupyter" button will appear





Note: device called "cuda" even though it's AMD GPU – for "historical reasons"

Limitations of interactive Jupyter runs

LUMI

- Using GPUs interactively is *inefficient usage of resources*
  - Most of the time, when you are editing code or grabbing a coffee, the GPU is idle (but nobody else can use it!)
  - Because of this interactive use is limited to a single GPU
- Running multiple copies of your job (e.g., hyperparameter search) is not possible
- Solution:
  - Use Jupyter for development and experimentation
  - Use terminal interface for real runs

more on this in the next session

Our running example for this course

LUMI

- Finetuning GPT-Neo LLM for generating movie reviews on the IMDb data set
- Using Hugging Face's `datasets` and `transformers` on top of PyTorch as training library

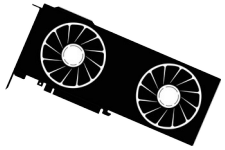
GPT-Neo 1.3B
- 1370 M parameters
- BF16
- ~2.67 GB

Stanford IMDb data set
- 100 000 movie reviews
- Varying lengths (low hundreds of words)
- 25 000 reserved for testing

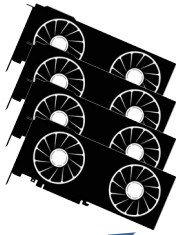
PyTorch

Our running example for this course

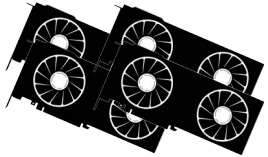
02: Get familiar with LUMI web interface and the example



03: Using Slurm scheduler to train on a single GPU
04: Checking on training jobs and some common problems



08: Training on multiple GPUs (on a single GPU node)
10: Training across multiple nodes



09: Parallel hyperparameter tuning

Icons: vecteezy.com

Course practicalities: Reservations

LUMI

- A portion of the cluster is reserved for the course
- When starting jobs you need to give the reservation names, otherwise you apply for resources in the general queues
  - in the web-interface, select from the drop down menu
- Day 1: `AI_workshop`
- Day 2: `AI_workshop_2`
- However, if you use a reservation that is not active, your job will not run until the reservation becomes active.