CCIN2P3

Centre de Calcul
de l’Institut National de Physique Nucléaire
et de Physique des Particules

cnrs

The Jupyter notebooks platform at CC-IN2P3

Bernard CHAMBON - GPU training session for HumaNum - November 27-28, 2025

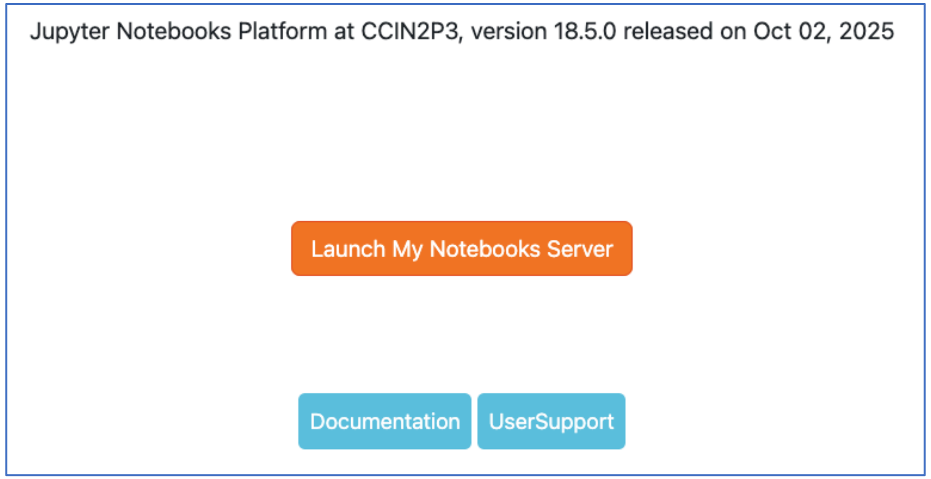
Outline

- Introduction
- Architecture
- Running CPU or GPU notebooks servers
- About the 2 features ‘Dask+SLURM’ and ‘MLFlow server’
- Annex

Introduction

- Objective
  - Provide Jupyter notebooks with access to the common storage systems at CC-IN2P3
    - HOME (/pbs/…), THRONG (/pbs/throng/…), GROUPS (/sps/…), SOFTWARE (/pbs/software/…), CVMFS (/cvmfs/…)

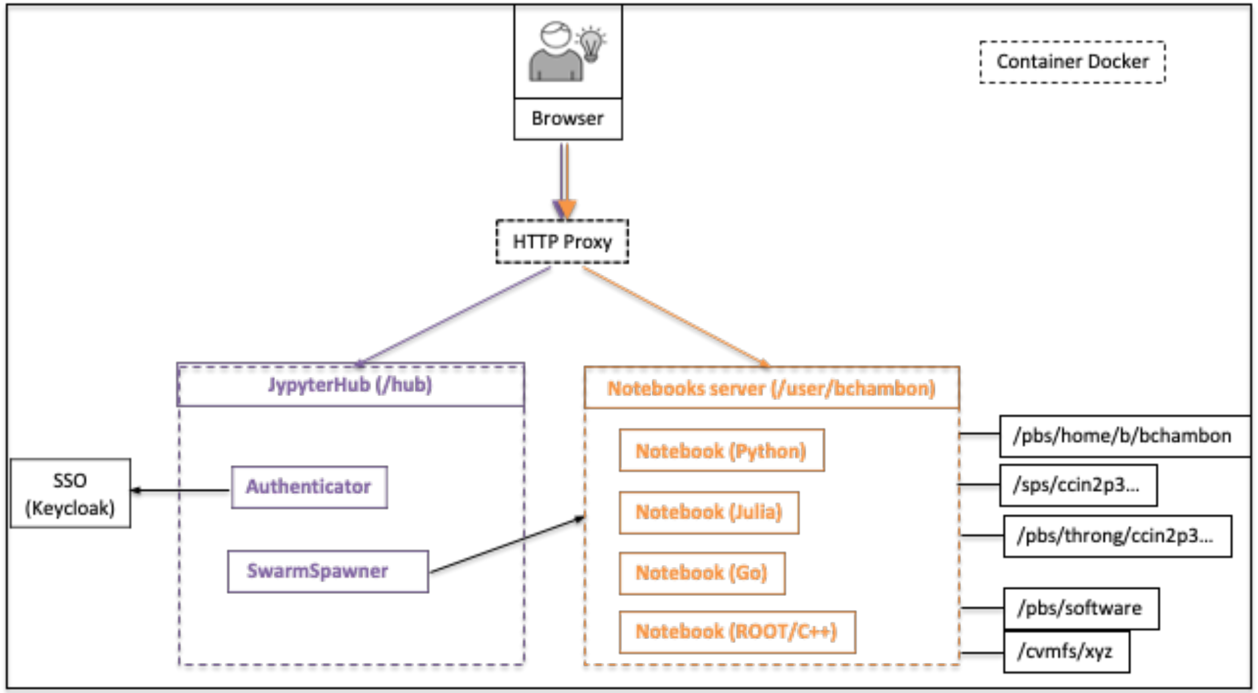
- How-to
  - Authentication via login/password or via certificate (eduGAIN)
    - Access allowed for users having a 'computing' account at CC-IN2P3
  - Login page https://notebook.cc.in2p3.fr/hub/login



Jupyter Notebooks Platform at CCIN2P3, version 18.5.0 released on Oct 02, 2025

Architecture

- Built around **JupyterHub**
  - Component allowing to plug an authenticator (OAuth), to provide Options form, to plug spawner, to handle services
  - Python config file allowing advanced configuration



- Running on a docker cluster with Swarm as orchestrator via SwarmSpawner to run the notebooks servers from docker image prepared at CC-IN2P3 and based on RHEL9

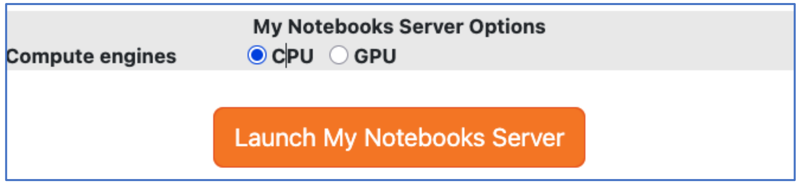
Running CPU or GPU notebooks servers 1/2

- CPU
    - Access not restricted (= available for everyone using the Jupyter notebooks platform)
    - Memory limit : Default is 8 GB, but higher value possible per group or per user
      (32 GB for group ‘humanum’, also 32 GB for group ‘training’ for this training session !)
    - No #CPUs limit, per default

- GPU
    - Granted access upon request, per group or per user, per GPU models (e.g. L40S for group ‘humanum’, but also ‘’ztf, ‘lsst’, etc.)
    - Providing an options form, to select the model of GPU, the number of GPU (on a same host)
    - Memory limit can be adjusted by end-user inside a range based on total RAM of hosts / #GPUs ([180, 260, 20])
    - No #CPUs limit, per default
    - User will obtain
        - A running notebooks server with dedicated GPU(s) (= not shared with other users)
        - With some ready-to-use machine learning (ML) frameworks, already installed in the docker image (Cf. memo displayed via Options form)

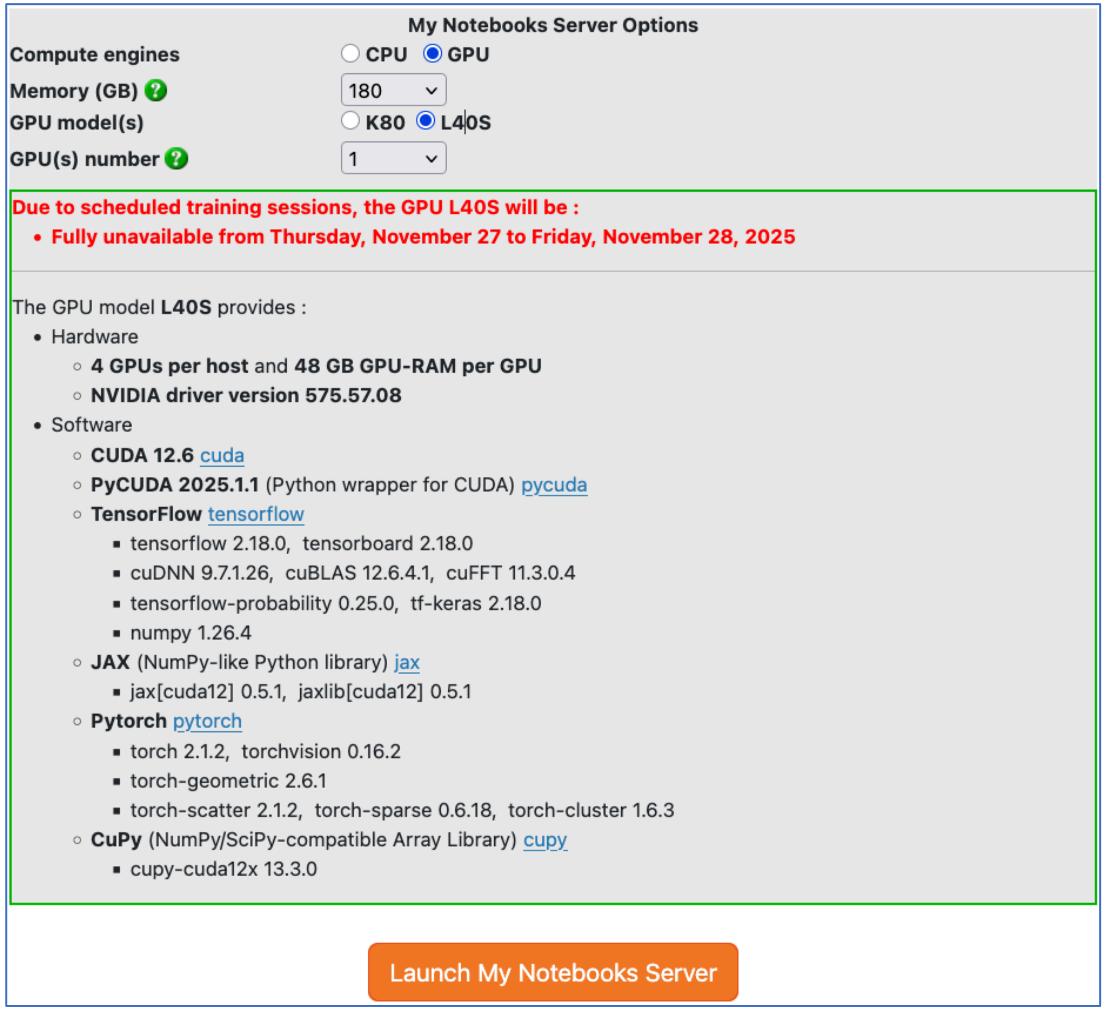
RAM, CPU and I/O consumptions are monitored for internal usage only (using cAdvisor, Prometheus and Grafana tools)

Running CPU or GPU notebooks servers 2/2



My Notebooks Server Options

Compute engines ● CPU ○ GPU



My Notebooks Server Options

Compute engines ○ CPU ● GPU

Memory (GB) 180

GPU model(s) ○ K80 ● L40S

GPU(s) number 1

Due to scheduled training sessions, the GPU L40S will be :

- **Fully unavailable from Thursday, November 27 to Friday, November 28, 2025**

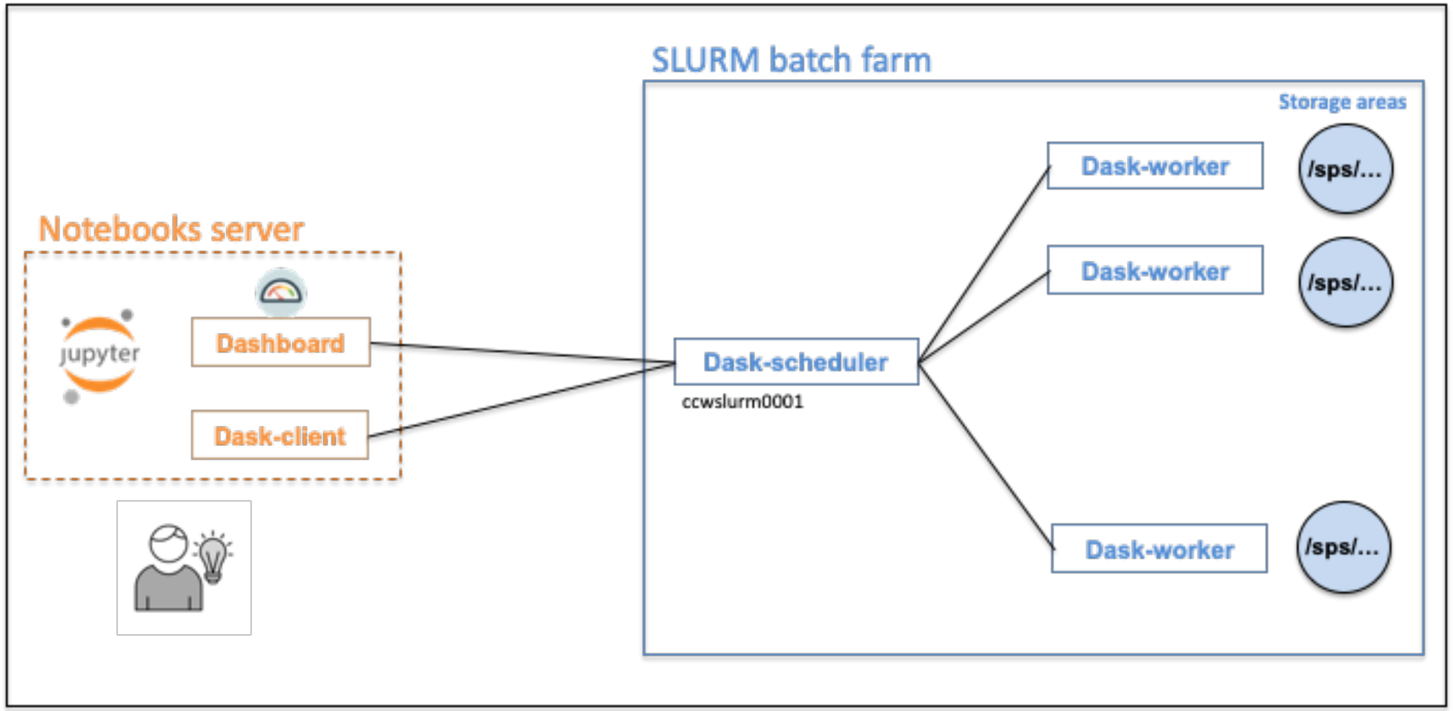
The GPU model **L40S** provides :

- Hardware
  - **4 GPUs per host** and **48 GB GPU-RAM per GPU**
  - **NVIDIA driver version 575.57.08**
- Software
  - **CUDA 12.6** cuda
  - **PyCUDA 2025.1.1** (Python wrapper for CUDA) pycuda
  - **TensorFlow** tensorflow
    - tensorflow 2.18.0, tensorboard 2.18.0
    - cuDNN 9.7.1.26, cuBLAS 12.6.4.1, cuFFT 11.3.0.4
    - tensorflow-probability 0.25.0, tf-keras 2.18.0
    - numpy 1.26.4
  - **JAX** (NumPy-like Python library) jax
    - jax[cuda12] 0.5.1, jaxlib[cuda12] 0.5.1
  - **Pytorch** pytorch
    - torch 2.1.2, torchvision 0.16.2
    - torch-geometric 2.6.1
    - torch-scatter 2.1.2, torch-sparse 0.6.18, torch-cluster 1.6.3
  - **CuPy** (NumPy/SciPy-compatible Array Library) cupy
    - cupy-cuda12x 13.3.0

The ‘Dask+SLURM’ feature 1/2

- Objectives
  - Allow interactive analysis for huge amount of data via parallel processing
  - From notebooks server (for interactivity) and by using resources from SLURM batch farm (for performance)
  - By spreading computing tasks, with Dask, over potentially several thousands of SLURM jobs

- Architecture



The ‘Dask+SLURM’ feature 2/2

- How-to
  - This feature is not restricted (= available for everyone using the Jupyter notebooks platform)
  - By writing Dask code (= in Python) and using the ‘dask4in2p3’ package (already installed in the docker image)

- User will be able to specify
  - The number of jobs, the RAM and elapse time of the jobs (same values for all jobs)
  - A virtual environment (where the package ‘dask4in2p3’ is installed, since it’s also required on ‘SLURM batch farm’ side)
  - Other parameters (timeouts, etc.) See docstring of methods (*via Shift-Tab after selecting the method*)

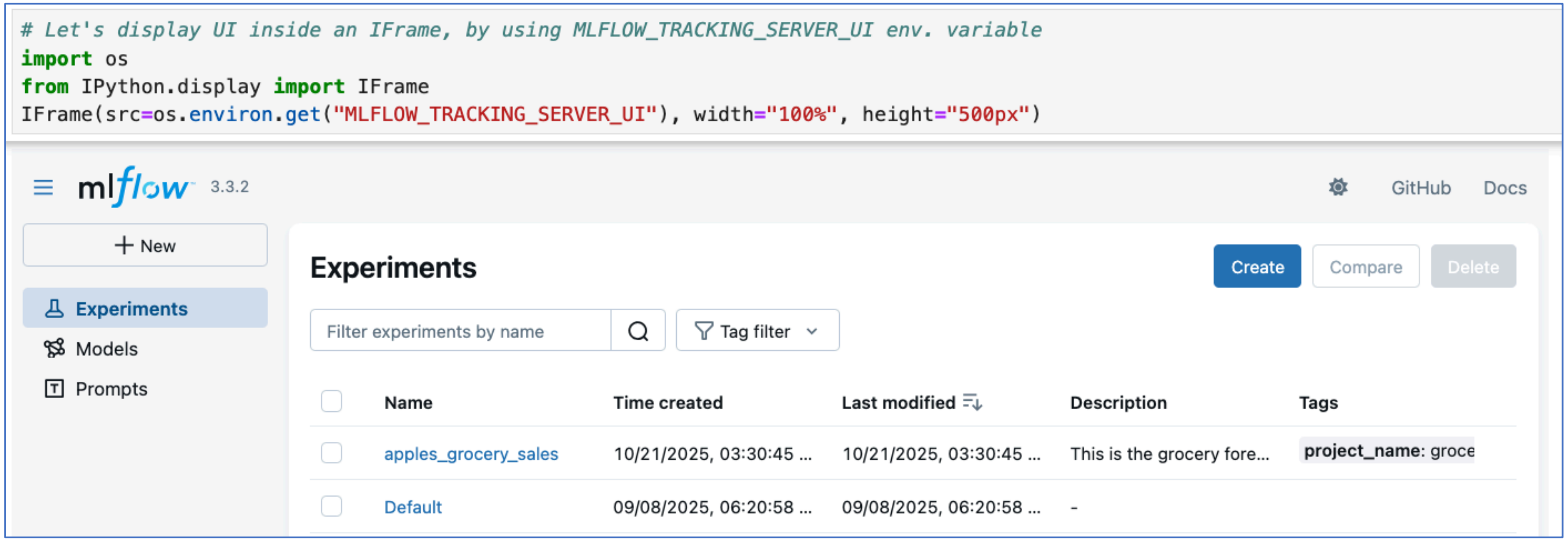
All parameters having a default value, it’s usable without any specification

- User will obtain
  - A dask-client connected to the dask-scheduler
  - A dashboard showing live metrics related to dask-workers (via the extension ‘dask-labextension’)
  - The possibility to connect other dask-clients on the running dask-cluster (only one dask-cluster per user)

- Demo via a mute video of 2’30 DemoDask+SLURM(ProcessingImages).m4v

The MLFlow feature (experimental, implementation to refine) 1/2

- Usage
  - Granted access upon request, per user
- User will obtain
  - A MLFlow tracking server running inside his container
    - With MySQL as backend store and HOME directory as artifact store (into $HOME/MLFLOW_ARTIFACT_STORE_DIR)
    - UI accessible via the env. variable $MLFLOW_TRACKING_SERVER_UI, possibly inside an IFrame
  - A ready to use environment, since some packages (mlflow[extras], scikit-learn,) are already installed in the docker image
- As example



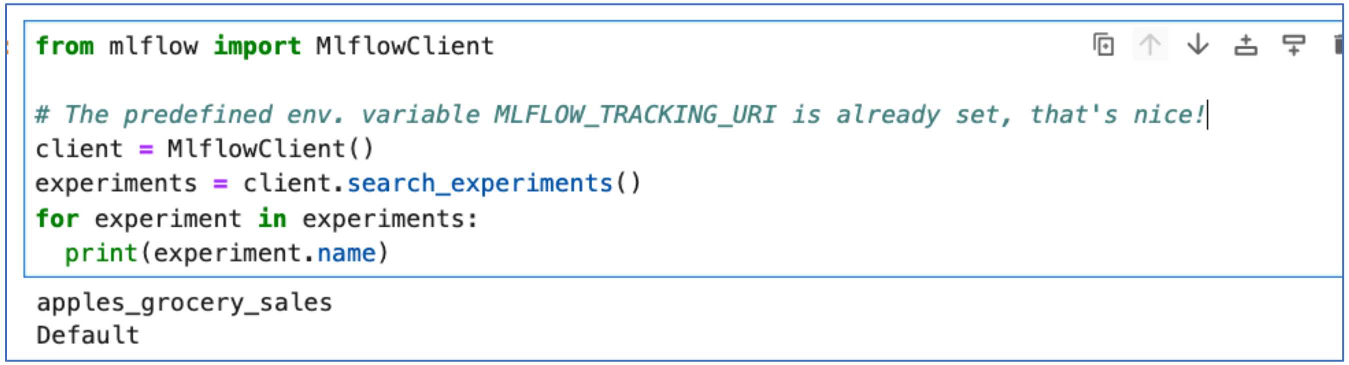
```
# Let's display UI inside an IFrame, by using MLFLOW_TRACKING_SERVER_UI env. variable
import os
from IPython.display import IFrame
IFrame(src=os.environ.get("MLFLOW_TRACKING_SERVER_UI"), width="100%", height="500px")
```

Experiments

☐	Name	Time created	Last modified	Description	Tags
☐	apples_grocery_sales	10/21/2025, 03:30:45 ...	10/21/2025, 03:30:45 ...	This is the grocery fore...	project_name: groce
☐	Default	09/08/2025, 06:20:58 ...	09/08/2025, 06:20:58 ...	-	

The MLFlow feature (experimental, implementation to refine) 2/2

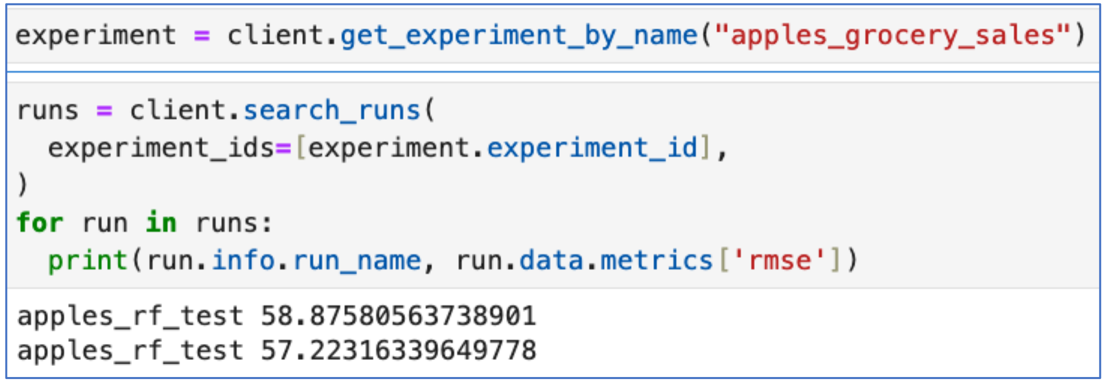
- Playing with MlflowClient



```
from mlflow import MlflowClient

# The predefined env. variable MLFLOW_TRACKING_URI is already set, that's nice!
client = MlflowClient()
experiments = client.search_experiments()
for experiment in experiments:
  print(experiment.name)
```

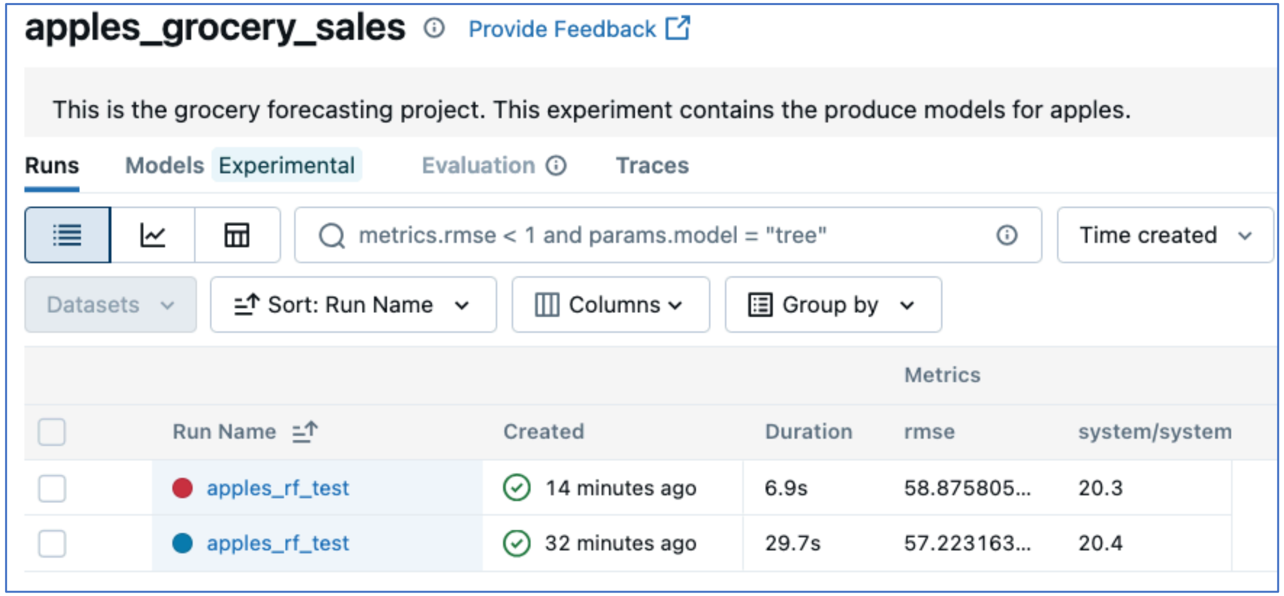
```
apples_grocery_sales
Default
```



```
experiment = client.get_experiment_by_name("apples_grocery_sales")
```

```
runs = client.search_runs(
  experiment_ids=[experiment.experiment_id],
)
for run in runs:
  print(run.info.run_name, run.data.metrics['rmse'])
```

```
apples_rf_test 58.87580563738901
apples_rf_test 57.22316339649778
```



apples_grocery_sales

This is the grocery forecasting project. This experiment contains the produce models for apples.

metrics.rmse < 1 and params.model = "tree"

				Metrics	
□	Run Name	Created	Duration	rmse	system/system
□	● apples_rf_test	✓ 14 minutes ago	6.9s	58.875805...	20.3
□	● apples_rf_test	✓ 32 minutes ago	29.7s	57.223163...	20.4

Outcome

- A service setup since July 2020
  - Available for all users having a ‘computing’ account
  - Configured to serve various needs
    - For data analysis, for training sessions
    - Providing both CPU or GPU resources (GPU K80 since May 2021)
    - Providing CPU resources of the SLURM batch farm, by using the Dask framework (since April 2023)
    - Providing 30 GPUs model L40S, highly configured in terms of RAM and CPU (since March 2025)

- URLs
  - Read the documentation Jupyter Notebooks Platform
  - Access to the service https://notebook.cc.in2p3.fr/
  - Ask for support https://support.cc.in2p3.fr/

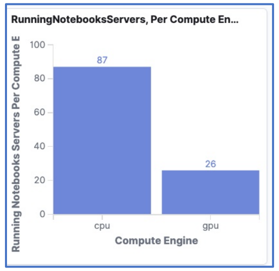
Thank you for your attention

Annex

- Infrastructure figures
- Usage figures for L40S (as example during ‘AstroInfo’ training session on October 2025)

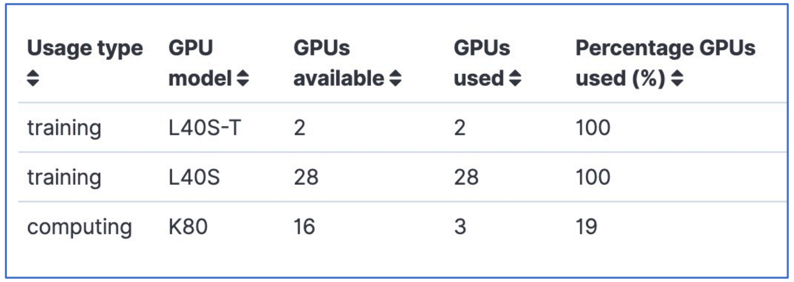
Infrastructure figures

- 1 server: VM with 8 CPUs, 16 GB RAM where JupyterHub runs
- 28 worker-nodes: where notebooks servers run
  - 11 VMs run CPU notebooks servers (8 CPUs, 64 or 96 GB RAM)
  - 9 bare metal hosts to run K80 GPU or CPU notebooks servers Decommissioned by the end of November 2025
    - 4 GPUs /host, CUDA 11.4 ; 16 CPUs, 130 GB RAM, 1 Gbps I/O
  - 7 + *1* = 8 bare metal hosts to run L40S GPU notebooks servers or CPU notebooks servers
    - 4 GPUs /host but restricted to 2 GPUs /user, 48 GB GPU-RAM per GPU, CUDA 12.x + recent versions of ML frameworks
    - 48 CPUs, 810 GB RAM, 10 Gbps I/O
    - Total amount **30 GPUs** (7 hosts with 4 GPUs, 1 host with 2 GPUs)
    - One host with only 2 GPUs, 405 GB RAM, most of the time integrated into the prod. Instance (<5 % of time into dev instance)
- This infrastructure can serve 100+ of users
  - So far, the maximum observed has been 110+ users (during a training session on April 2025) ➔
  - **Possibly up to 30 users running GPU L40S notebooks servers, thanks to the contribution of HumaNum**



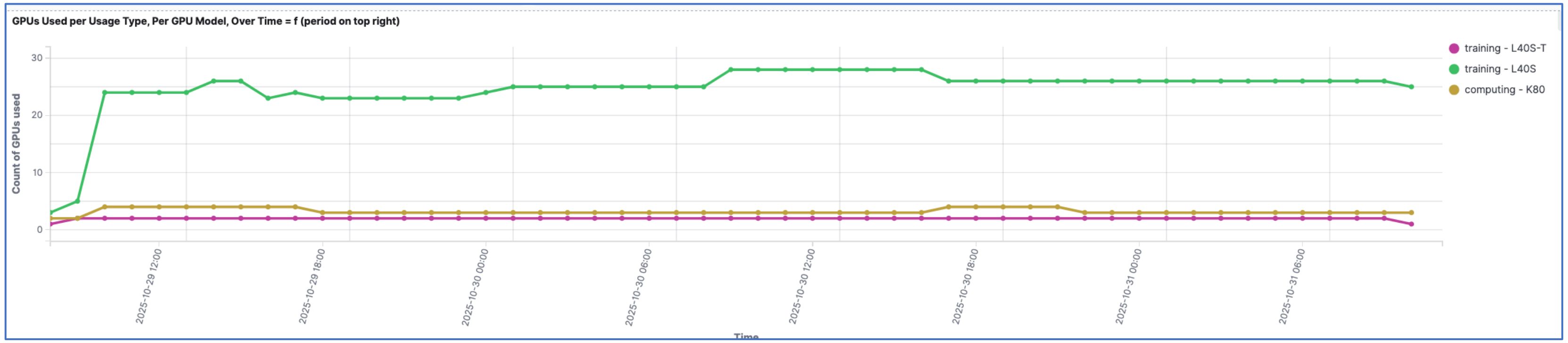
Usage figures for L40S (AstroInfo training session on October 2025)

100% of the GPUs L40S|L40S-T were being used.

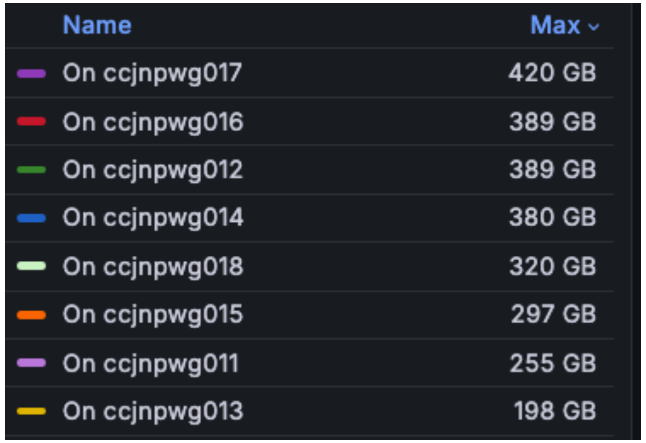


Usage type	GPU model	GPUs available	GPUs used	Percentage GPUs used (%)
training	L40S-T	2	2	100
training	L40S	28	28	100
computing	K80	16	3	19

Almost all the GPUs L40S|L40S-T were used for 3 days.

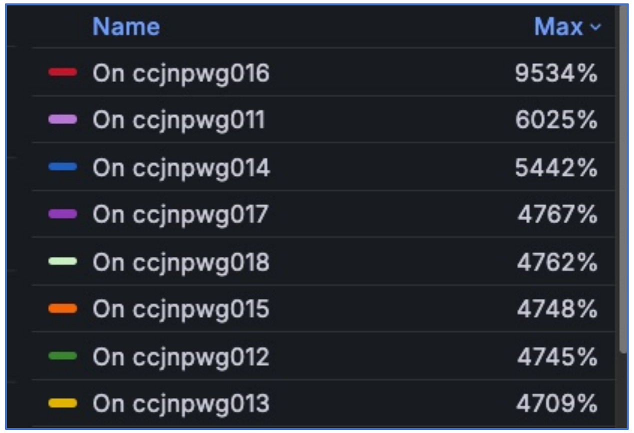


Max memory used per hostname (hosts *ccjnpwg011-018 are L40S)*
Max installed RAM is 810 GB (=754 GiB), 405 GB (=377 GiB) for host ccjnpwg011



	Name	Max
🟪	On ccjnpwg017	420 GB
🟥	On ccjnpwg016	389 GB
🟩	On ccjnpwg012	389 GB
🟦	On ccjnpwg014	380 GB
🟩	On ccjnpwg018	320 GB
🟧	On ccjnpwg015	297 GB
🟪	On ccjnpwg011	255 GB
🟨	On ccjnpwg013	198 GB

Max number of CPUs used, per hostname (hosts *ccjnpwg011-018 are L40S)*
Max installed #CPUs is 48, 96 CPUs for hosts ccjnpwg011 and 16 !



	Name	Max
🟥	On ccjnpwg016	9534%
🟪	On ccjnpwg011	6025%
🟦	On ccjnpwg014	5442%
🟪	On ccjnpwg017	4767%
🟩	On ccjnpwg018	4762%
🟧	On ccjnpwg015	4748%
🟩	On ccjnpwg012	4745%
🟨	On ccjnpwg013	4709%