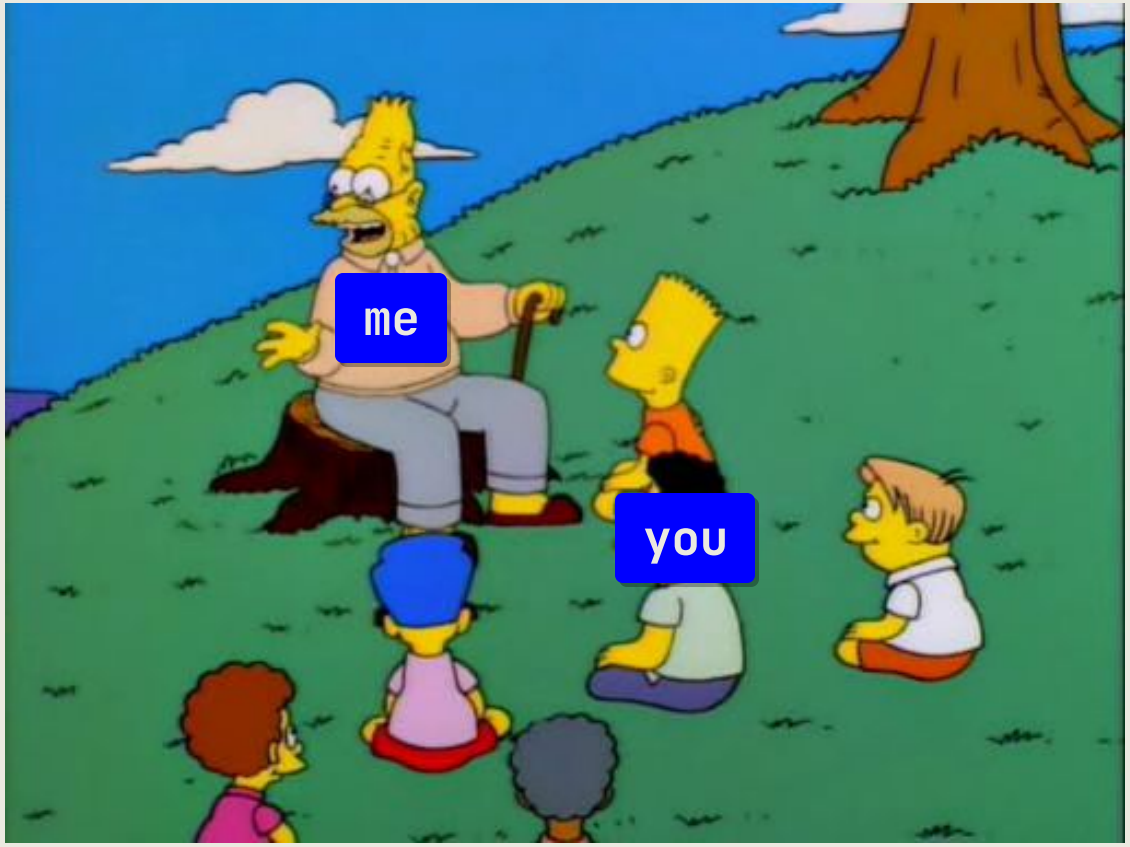
uv for ES

A fast Python toolchain for reproducible training runs.

Lluís Palma

BSC · Earth Sciences

> *"back in the days I had to spend two weeks to install pytorch in cte-pw9"*



What is uv?

An extremely fast Python package and project manager, written in Rust.

Install:

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

On hubs and mn5, it's already a module *(thanks to Albert Vila & Stamen)*:

```
module load uv
```

Start a project

New:

```
uv init my_project && cd my_project
uv python pin 3.10
```

Existing — `uv sync` is the whole onboarding:

```
git clone git@gitlab.example.org:user/my_project.git
cd my_project
uv sync
```

Add, remove dependencies

```
uv add xarray "torch>=2.8"       # edits pyproject + lock + installs
uv add --group dev pytest ruff   # dev-only
uv remove seaborn
```

pyproject.toml

```
[project]
name = "my_project"
version = "0.1.0"
requires-python = ">=3.10"
dependencies = ["torch>=2.8", "xarray"]
[dependency-groups]
dev = ["pytest", "ruff"]
```

Commit `pyproject.toml` + `uv.lock`. Never commit `.venv/`.

Locking and syncing

- **Lock** — resolve all dependencies into `uv.lock` (reproducible).
- **Sync** — install the subset of the lockfile into `.venv/`.

Opt out when you need to:

```
uv run --locked ...   # error if lock is stale (CI-friendly)
uv run --frozen ...   # use lock as-is, don't re-resolve
uv run --no-sync ...  # don't touch .venv/
```

Run things — `uv run`

```
uv run python script.py
```

Same for anything else:

```
uv run pytest
uv run jupyter-lab --no-browser --port 8888
uv tool run ruff check src/      # syntax checker
```

On SLURM (no internet) — lock in advance:

```
#SBATCH --gres=gpu:4
cd $PROJECT_DIR
uv run torchrun --nproc_per_node=4 script.py
```

Depend on your own code

Add `my_lib` to the baseline dependencies, then declare its source:

pyproject.toml

```
# in [project]:
dependencies = ["torch>=2.8", "xarray", "my_lib"]

[tool.uv.sources]
# GitLab branch:
my_lib = { git = "https://gitlab.example.org/user/my_lib.git", branch = "main" }

# pin a commit:
# my_lib = { git = "...", rev = "a3f2c9d" }

# local sibling checkout:
# my_lib = { path = "../my_lib", editable = true }
```

Same mechanism: GitLab, GitHub, local paths.

PyTorch + CUDA

Add these two sections to the baseline:

pyproject.toml

```
[tool.uv.sources]
torch = { index = "pytorch-cu128", marker = "sys_platform == 'linux' or sys_platform == 'win32'" }

[[tool.uv.index]]
name = "pytorch-cu128"
url = "https://download.pytorch.org/whl/cu128"
explicit = true
```

- `explicit = true` → index used only when referenced
- Marker keeps macOS / CI on plain PyPI
- Swap `cu128` → `cu121` / `cu124` per cluster

Feature map

Area	Commands
Python	`uv python install 3.12` · `list` · `find` · `uninstall`
Project	`uv init` · `add` · `remove` · `sync` · `lock` · `run` · `tree` · `build` · `publish`
Tools	`uvx <tool>` (ephemeral) · `uv tool install` · `list` · `uninstall` · `update-shell`
pip	`uv venv` · `uv pip install` / `show` / `freeze` / `list` / `check` / `uninstall` · `uv pip compile` · `uv pip sync`
Gotchas	commit `uv.lock` · PyTorch index needs `explicit = true` · use `uv run` everywhere

→ Full docs: **docs.astral.sh/uv**

Bonus: self-contained scripts

Declare deps in the script header:

plot.py

```
# /// script
# requires-python = ">=3.10"
# dependencies = ["xarray", "matplotlib"]
# ///
import xarray as xr
xr.open_dataset("sst.nc").sst.mean("time").plot()
```

```
uv run plot.py
```

No project. No venv.

Thanks!