ECMWF JupyterHub on Atos HPC and cloud

Introduction to ECMWF computing services

Valters Zeizis

Cloud and software support @ User Service Section

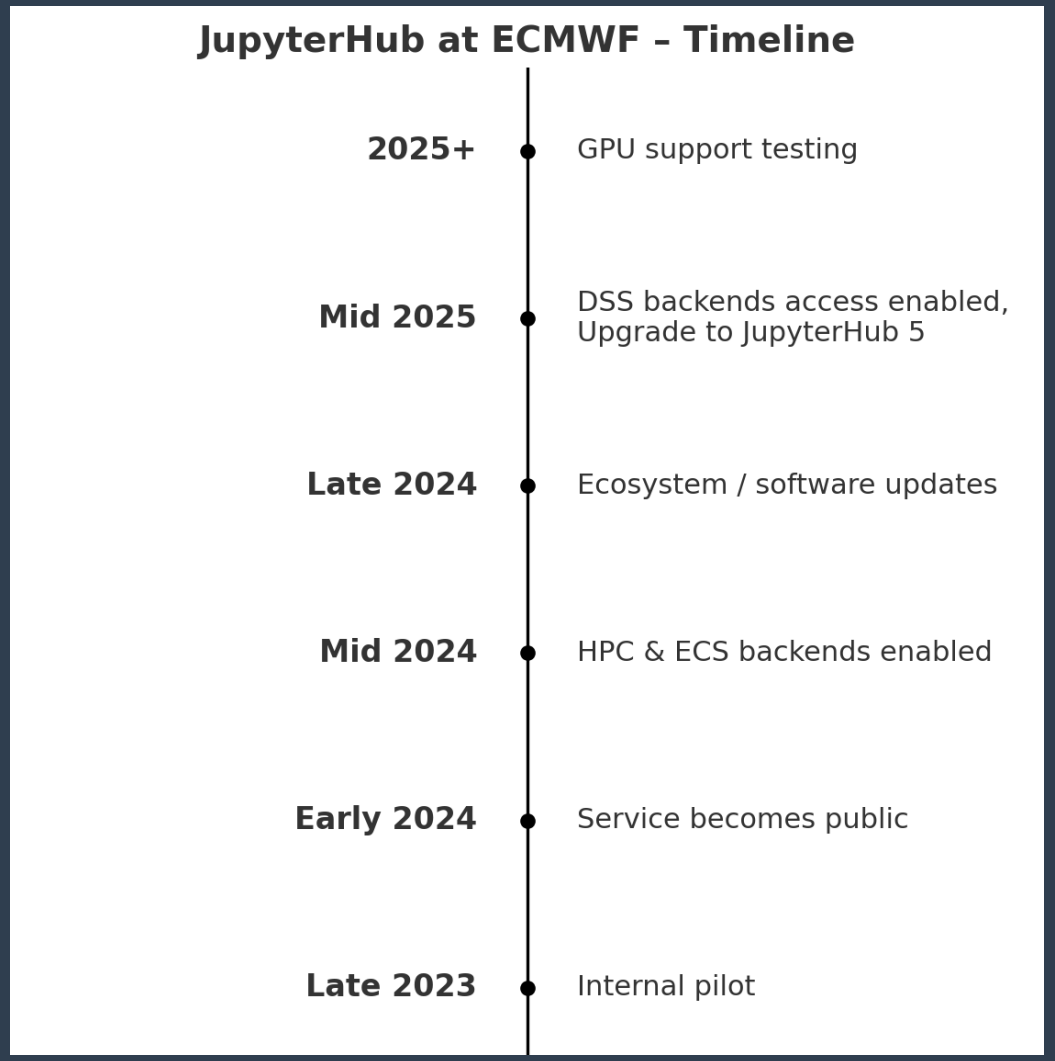
Valters.Zeizis@ecmwf.int

ECMWF

What is JupyterHub

- Browser-based Jupyter Lab environment
- Users can use the service to launch an isolated server on any of the available backends
- Perform on-the-fly data processing close to data using the readily available software and tools (just like working on HPC*) on a shared resource
- It is not a replacement for all batch/HPC workflows, but provides interactive means to interact with data and software for building prototypes

*if you are using the HPC backend

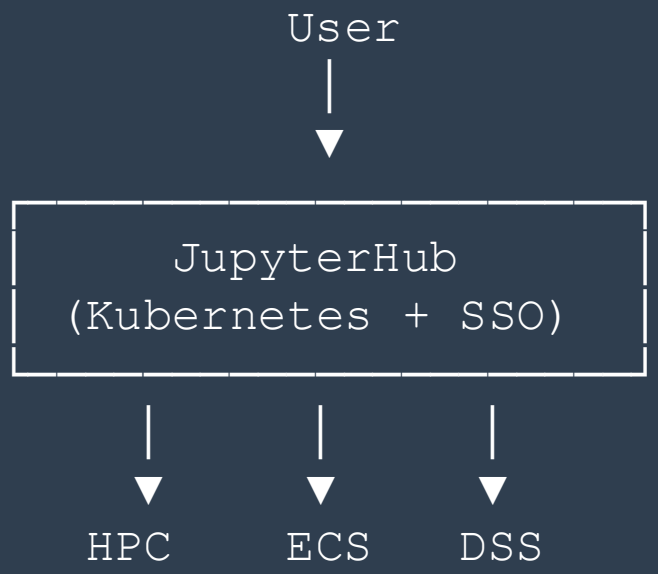


Architecture - one front door, many engines

JupyterHub orchestrates the service, while compute happens on the backend you choose:

- HPCF (Slurm) for larger/comprehensive workloads
- ECS (ecgate-class) for testing and smaller tasks
- DSS (near data) for lightweight, data-proximate tasks

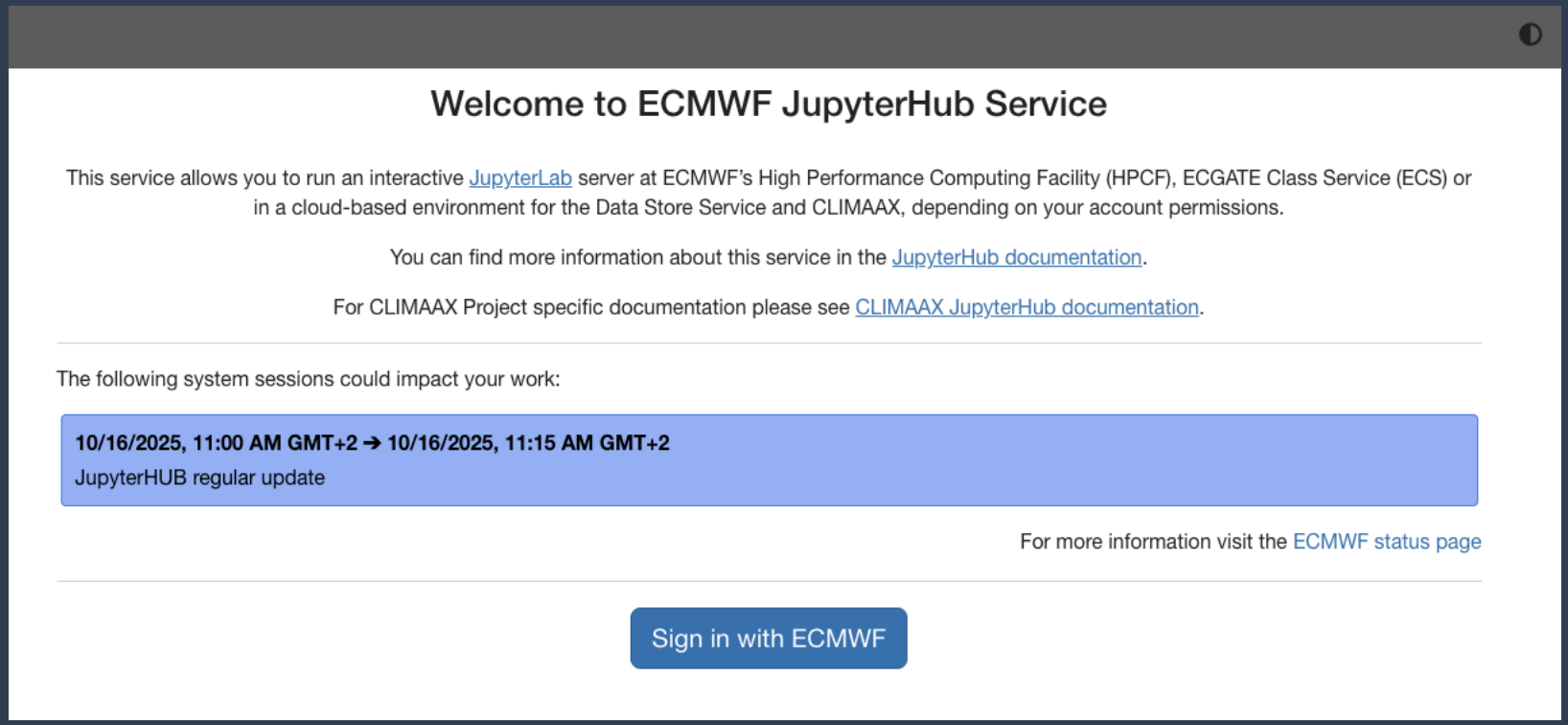
Backend decoupling brings flexibility, easy integration of new backends



Access & first-use

Prerequisites:

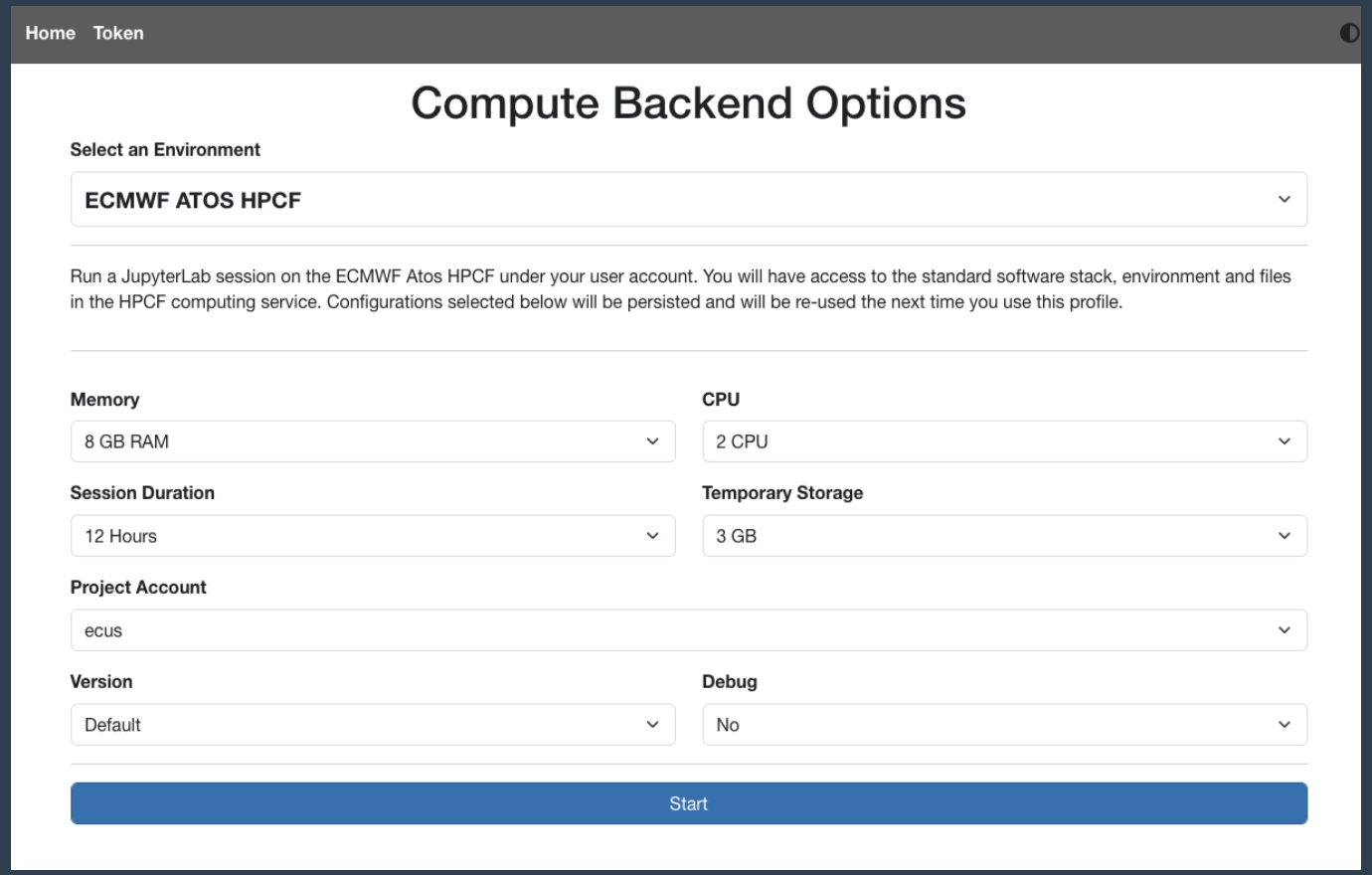
- SSO + MFA with ECMWF account
- add JupyterHub's **SSH public key** to `~/.ssh/authorized_keys`
- each user can have **one active** JupyterHub **session** at a time
- sessions are time limited and in case of inactivity they expire



User documentation covering the initial setup can be found here:
https://confluence.ecmwf.int/display/UDOC/First+use%3A+enabling+access+to+your+HPCF+or+ECS+account+from+JupyterHub

Starting a session (Backend Options)

- Pick engine/environment
- Set **CPUs** and **Memory**
- Choose **Duration** (auto-stop)
- Size of **TMPDIR** for temporary files
- Set **Project** for accounting
- Choose a **Version** (base stack)



If startup misbehaves enable debug option and inspect the log.

This is helpful evidence and very useful for cases when you need further support, to help us replicate and investigate issue.

JupyterLab tour

- **Launcher →**

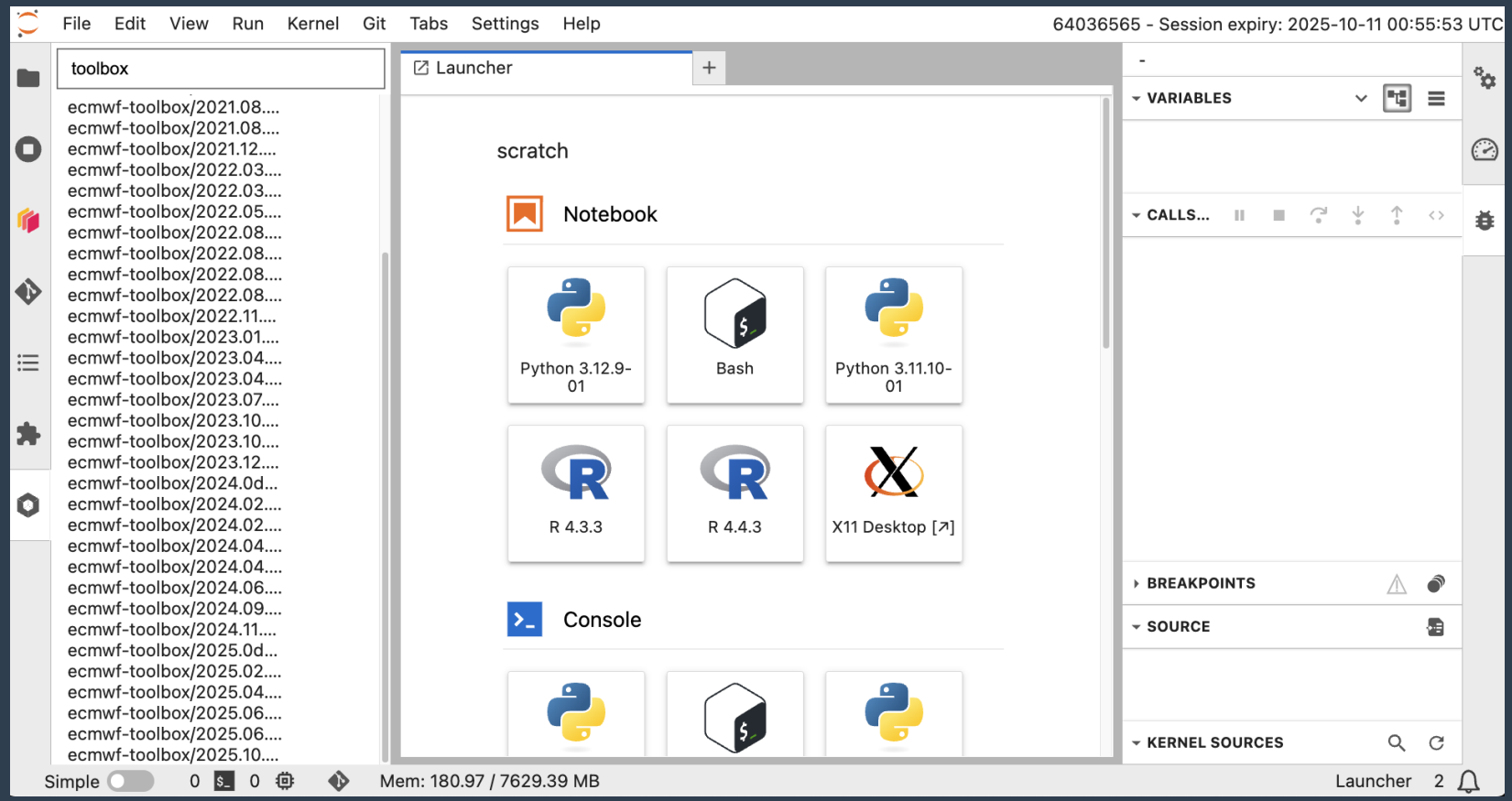
Notebooks, Consoles, Terminal

- **Left sidebar →**

File browser, Git (if enabled), Running, Commands
Kernel menu → restart/change kernel (environment captured at start)

- **Extensions →**

modules panel, Git, PyPI UI



Software stacks: modules, kernels and conda

- ECMWF software arrives via modules (eccodes, magics, …)
- **Golden rule:** load modules **before** starting a kernel (or restart it).
- To use your own conda environment as a kernel for Jupyter notebook you will need to have *ipykernel* installed in the conda environment before starting your Jupyter instance.

ipykernel can be installed with:

```
conda activate myCondaEnv
conda install ipykernel
python3 -m ipykernel install --user --name=myCondaEnv --env PATH "$CONDA_PREFIX/bin:\$PATH"
```

- If you want to make your own Python *virtual env* visible in Jupyterlab, the steps are similar:

```
source /path/to/myVenv/bin/activate
pip3 install ipykernel
python3 -m ipykernel install --user --name=myVenv --env PATH "/path/to/myVenv/bin:\$PATH"
```

You can remove unnecessary personal kernels simply:

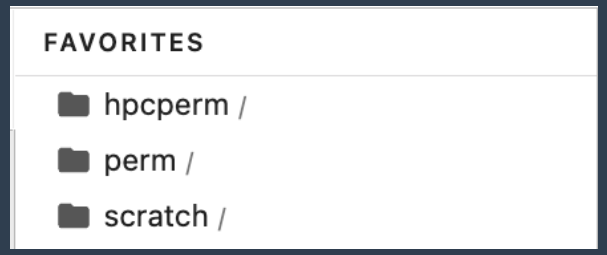
```
jupyter kernelspec uninstall myKernelName
```

Managing storage & best practices

- **HOME**: small, backed-up; default start location
- **PERM/HPCPERM**: used for long-term storage
- **SCRATCH**: high-capacity, temporary storage for big temporary files

You can add handy shortcuts in HOME:

```
for s in perm hpcperm scratch;
    do ln -snf /$s/$USER ~/$s; done
```



Limits, etiquette & troubleshooting

- Sessions auto-end at the time you choose - save before that.
- One active Hub session by design; avoid parallel *ecinteractive* Jupyter.
- ‘No space’ usually means **TMPDIR** is too small—start with a bigger one.
- Startup failed? **Debug = Yes**, read the log, seek support.

Resources

- Login page: https://jupyterhub.ecmwf.int/hub/login
- User docs: https://confluence.ecmwf.int/display/UDOC/JupyterHub
- Support portal: https://www.ecmwf.int/en/support

Thank you!

Q&A