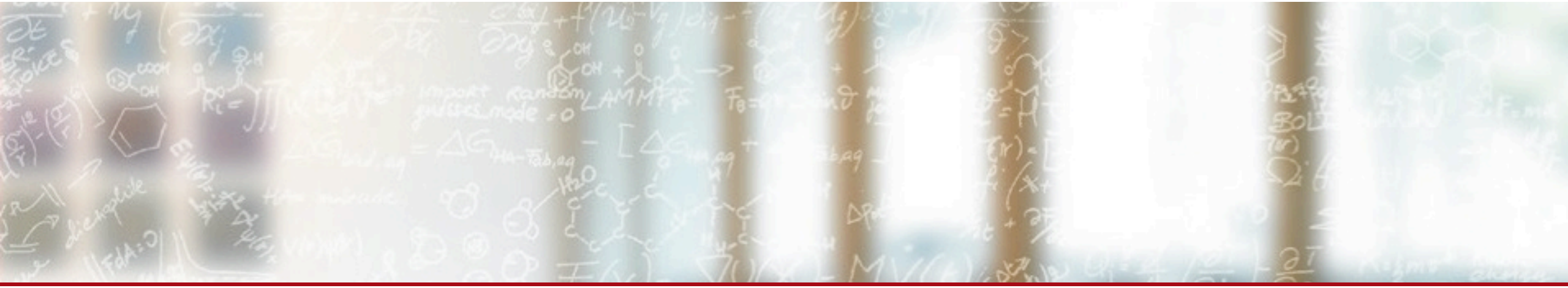
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Interactive HPC on Piz Daint with JupyterLab

HPC-CH

Maxime Martinasso, CSCS

Oct 22, 2020

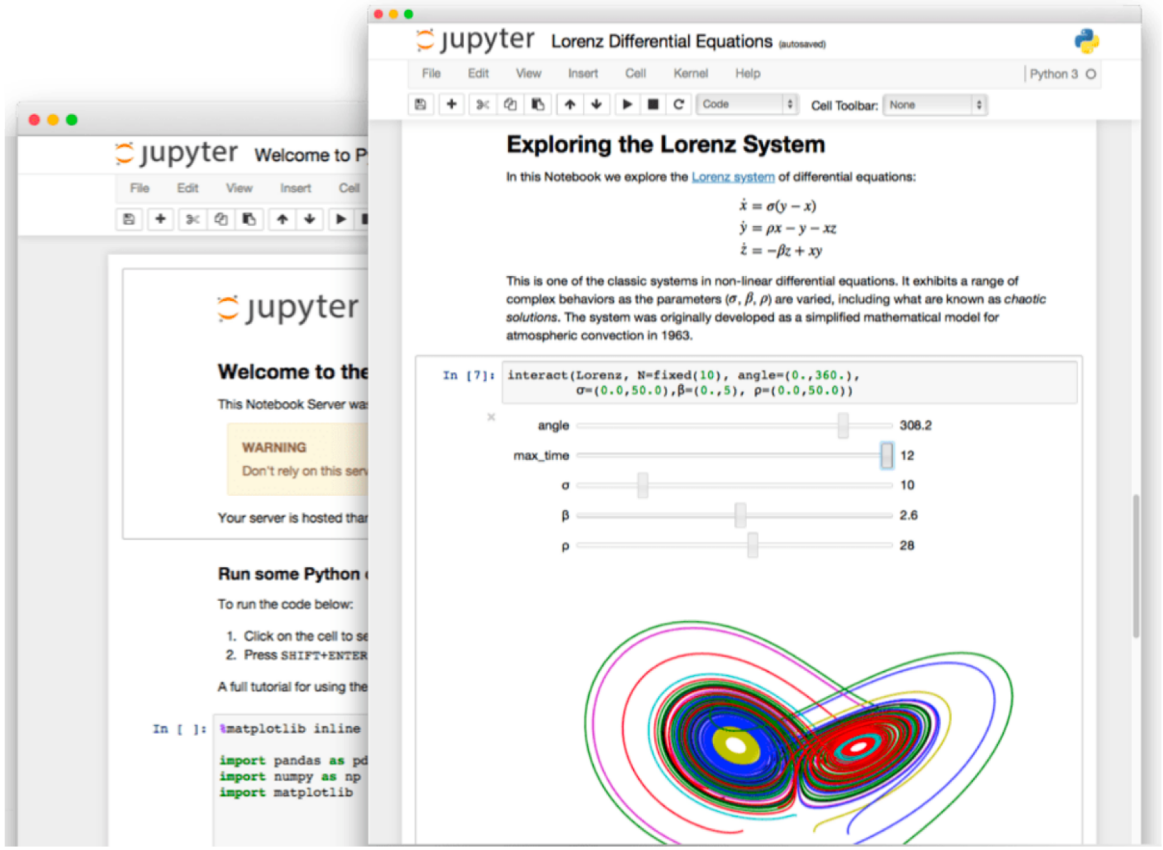
Outline

1. Interactive supercomputing
   - Jupyter notebooks, JupyterHub, JupyterLab
2. Challenges in offering on-demand resources
   - Interactive vs batch scheduling
   - Technical solution
3. Jupyter features enabled on Piz Daint
   - Parallel computing notebooks
   - Custom environments
   - Integration improvements
4. Future plans

Interactive supercomputing

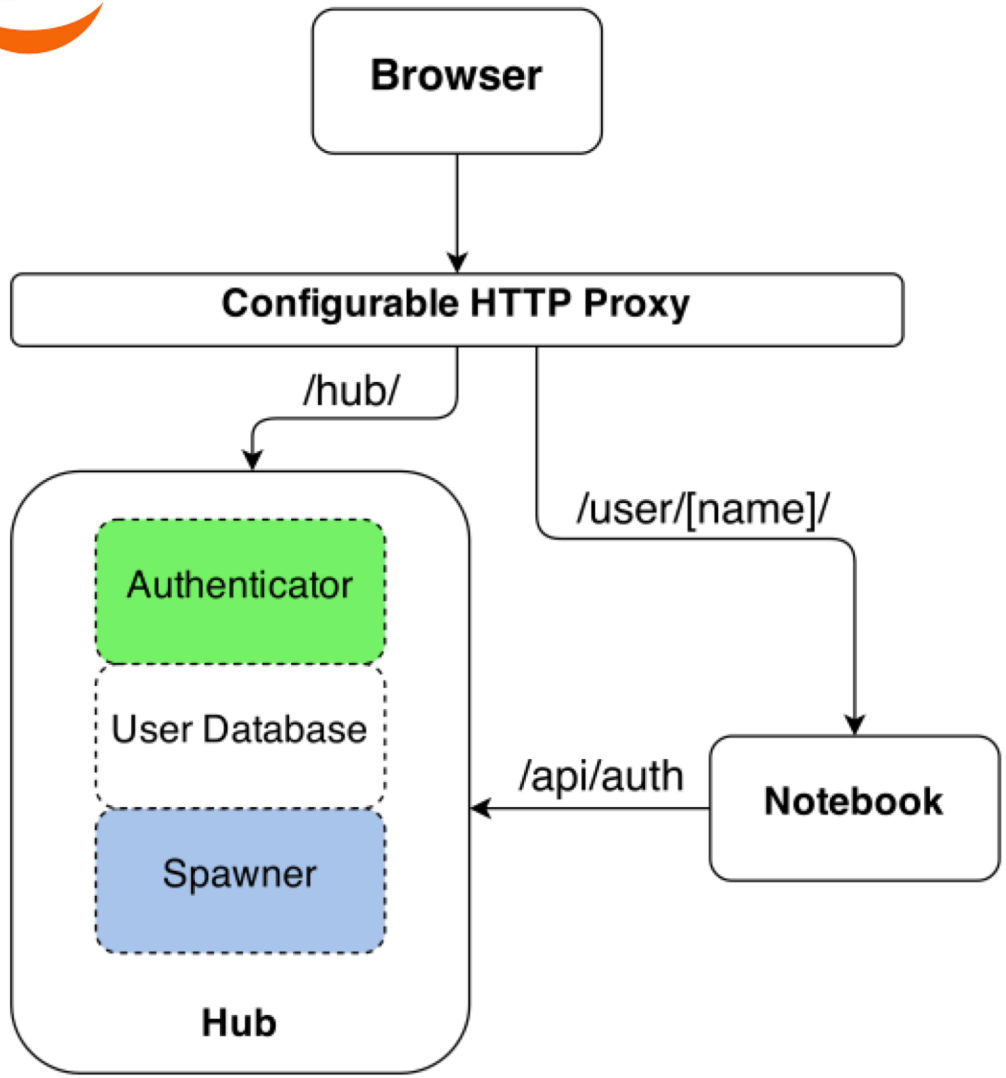
Browser-enabled working environments

- Project Jupyter – enabling interactive computational environments in a web browser
- Jupyter is an open-source web application for creating **reproducible computational narratives**
- Create documents that contain live code, equations, narrative text, visualizations, rich media
- The all-in-one document is also “Jupyter Notebook”
  - easily shared with others
  - in-browser terminal
  - file browsing
  - Python, R, Julia, C++, …
  - extensible design
  - many server/client plugins



JupyterHub

- Multi-user server for Jupyter Notebooks (designed for classrooms, research labs, Universities...)
- Spawns, manages and proxies multiple instances of the single-user Jupyter Notebook server
- Three main subsystems
  - a **multi-user Hub** (tornado process)
  - a configurable **http proxy** (node-http-proxy)
  - multiple **single-user Jupyter notebook servers** (Python/IPython/tornado)
- The key pluggable components are the authenticator and spawner



JupyterLab

File Edit View Run Kernel Tabs Settings Help

Files | Running | Commands | Cell Tools | Tabs

> notebooks

Name	Last Modified
Data.ipynb	an hour ago
Fasta.ipynb	a day ago
Julia.ipynb	a day ago
Lorenz.ipynb	seconds ago
R.ipynb	a day ago
iris.csv	a day ago
lightning.json	9 days ago
lorenz.py	3 minutes ago

Lorenz.ipynb | Terminal 1 | Console 1 | Data.ipynb | README.md

Code — Python 3

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

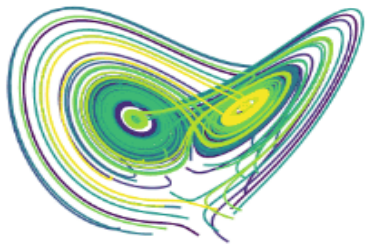
In [4]:
```
from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```

Output View

sigma 10.00

beta 2.67

rho 28.00



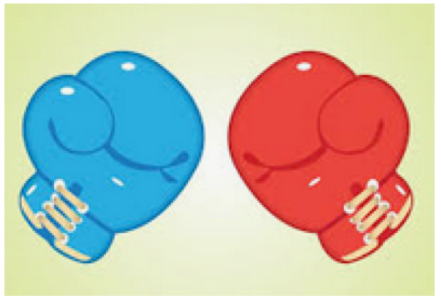
lorenz.py

```
def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))
```

Challenges in offering interactivity on HPC

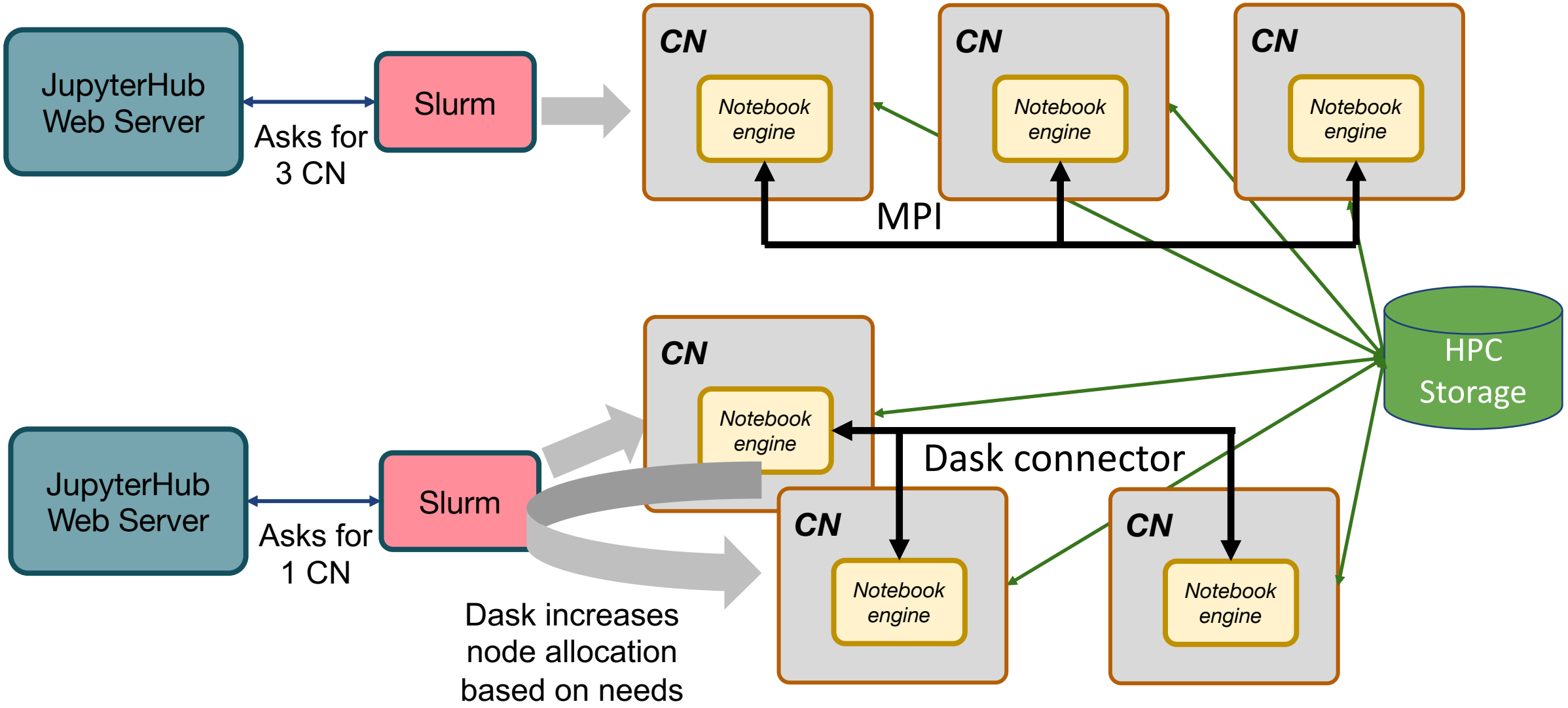
Challenge: Batch vs interactive computing

- How can we reconcile the apparent contradiction between **batch computing** and **interactive computing**?
- Batch is used on HPC to maximize utilization, large jobs
- Interactivity = small waiting time of jobs in queue
- Job pre-emption? Suspend/Resume? Reservation?



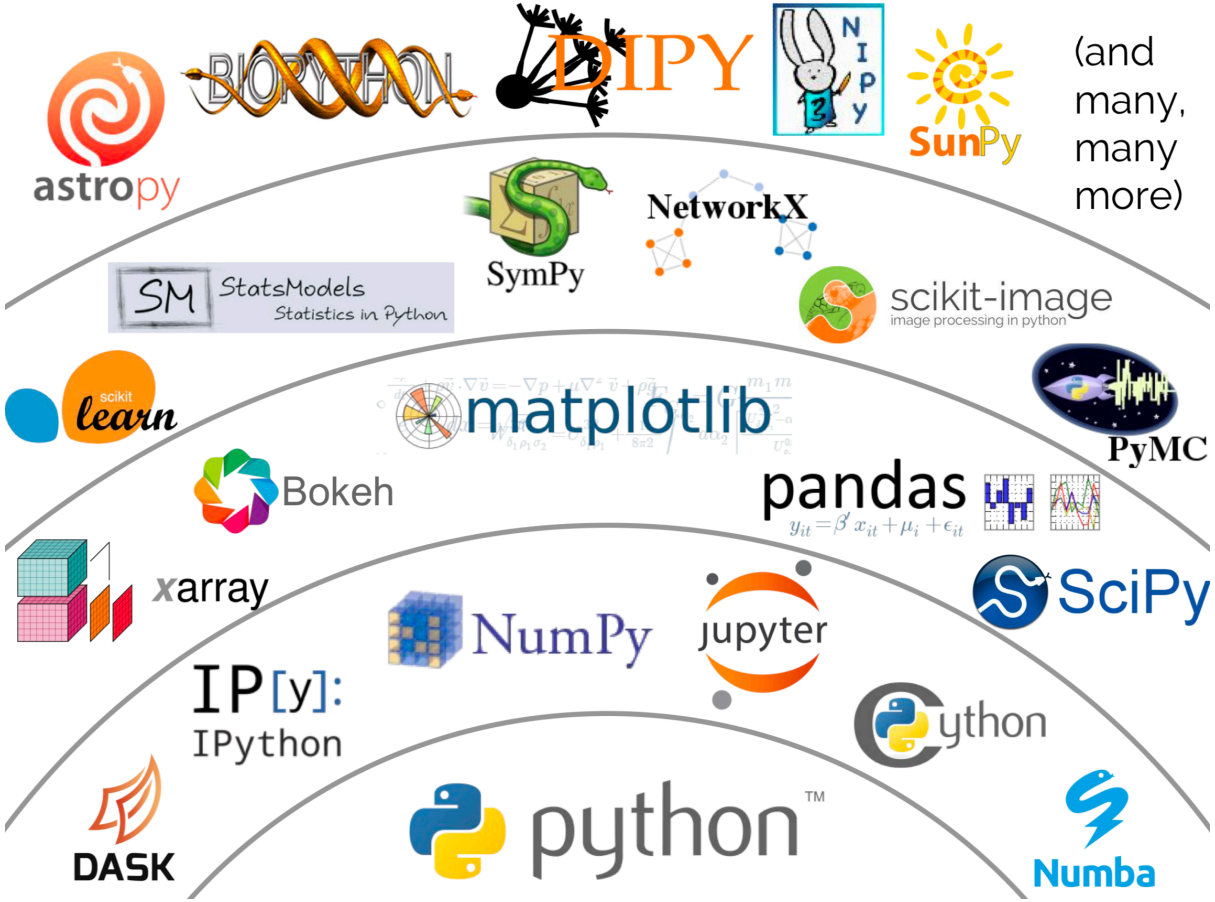
- Our solution: Elastic reservation
  - Increase/decrease number of nodes depending on the demand
  - Minimum fix number of nodes
  - Slurm flag REPLACE

Static and dynamic resource allocations

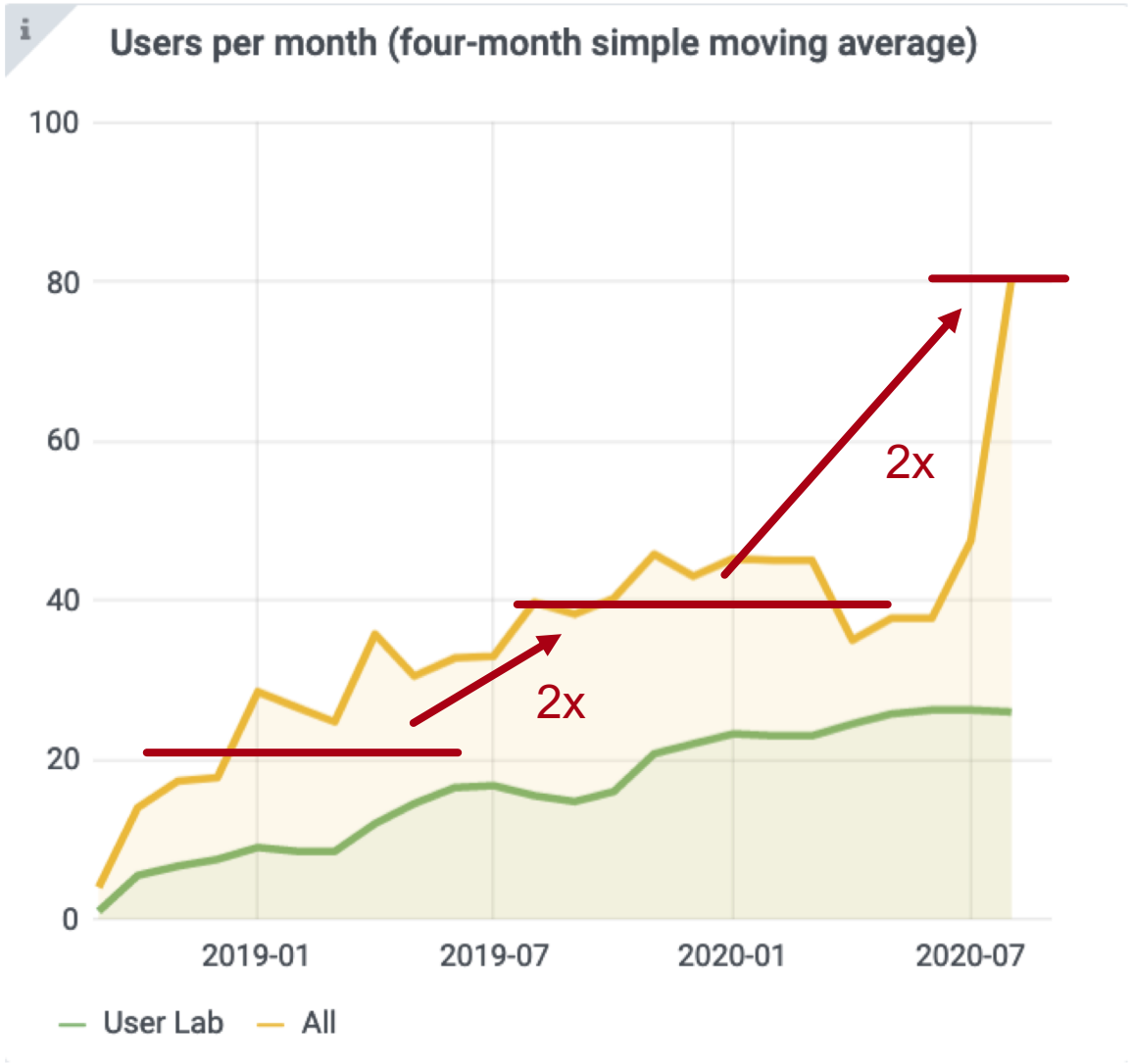


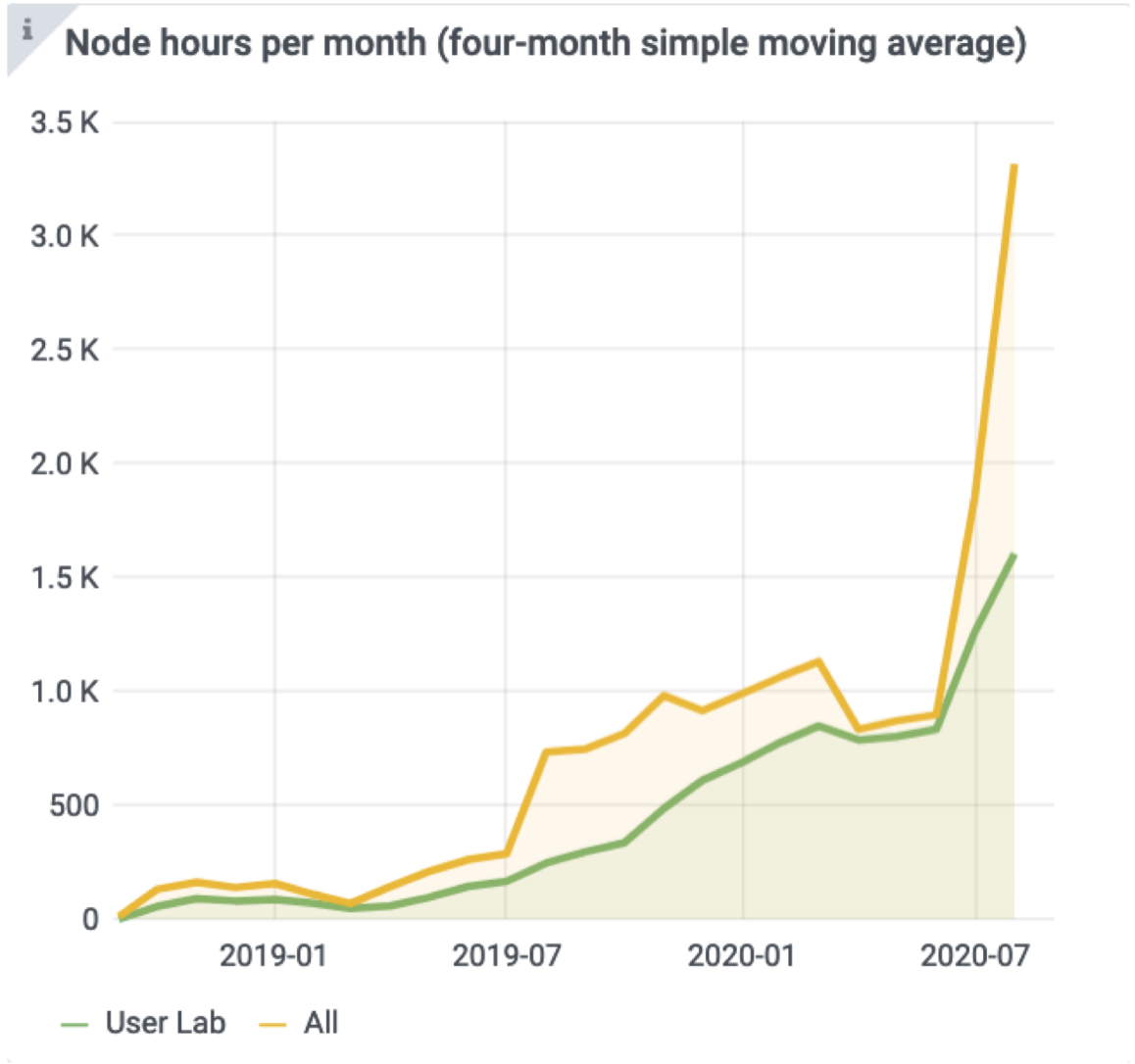
Jupyter software stack at CSCS

- Carefully manage package versions
  - Installed with EasyBuild
- Based on `cray-python/3.X`
  - provides numpy and scipy that call cray-libsci routines
- Parallel computing available in the notebook
  - ipyparallel (MPI with mpi4py)
  - distributed dask



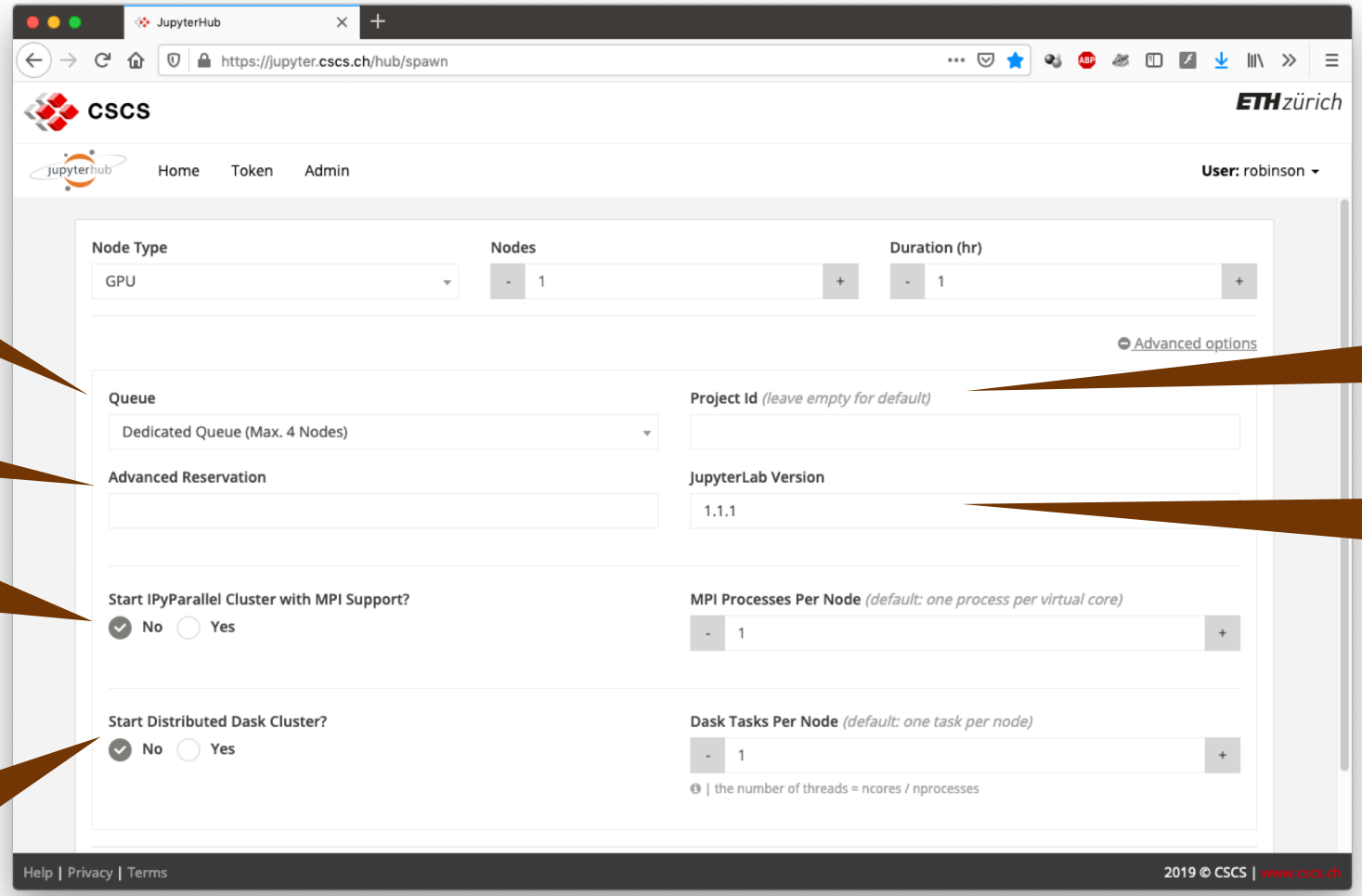
JupyterLab usage over time





Jupyter features on Piz Daint

Submission options



Queue

Reservation

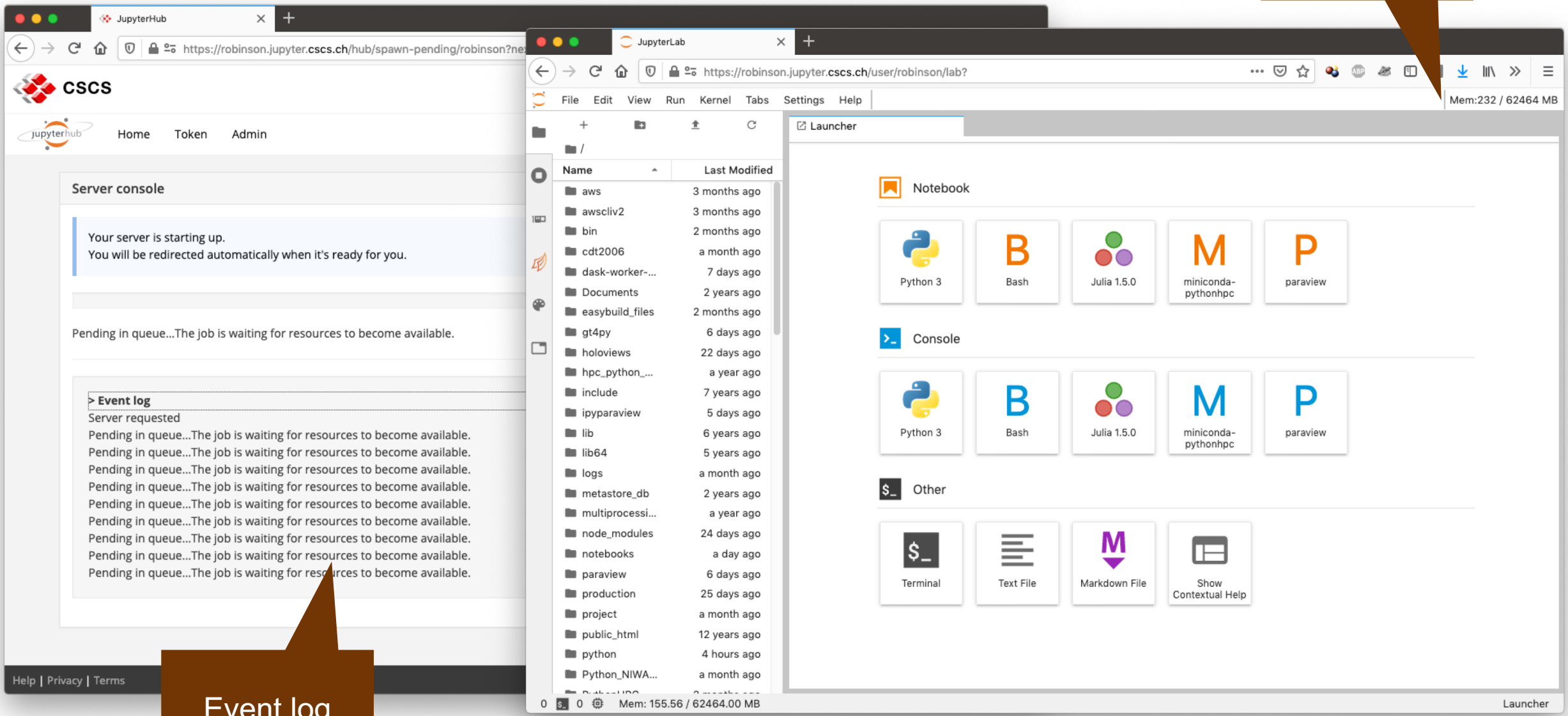
IPyParallel (multi-node notebooks)

Distributed Dask cluster (multi-node notebooks)

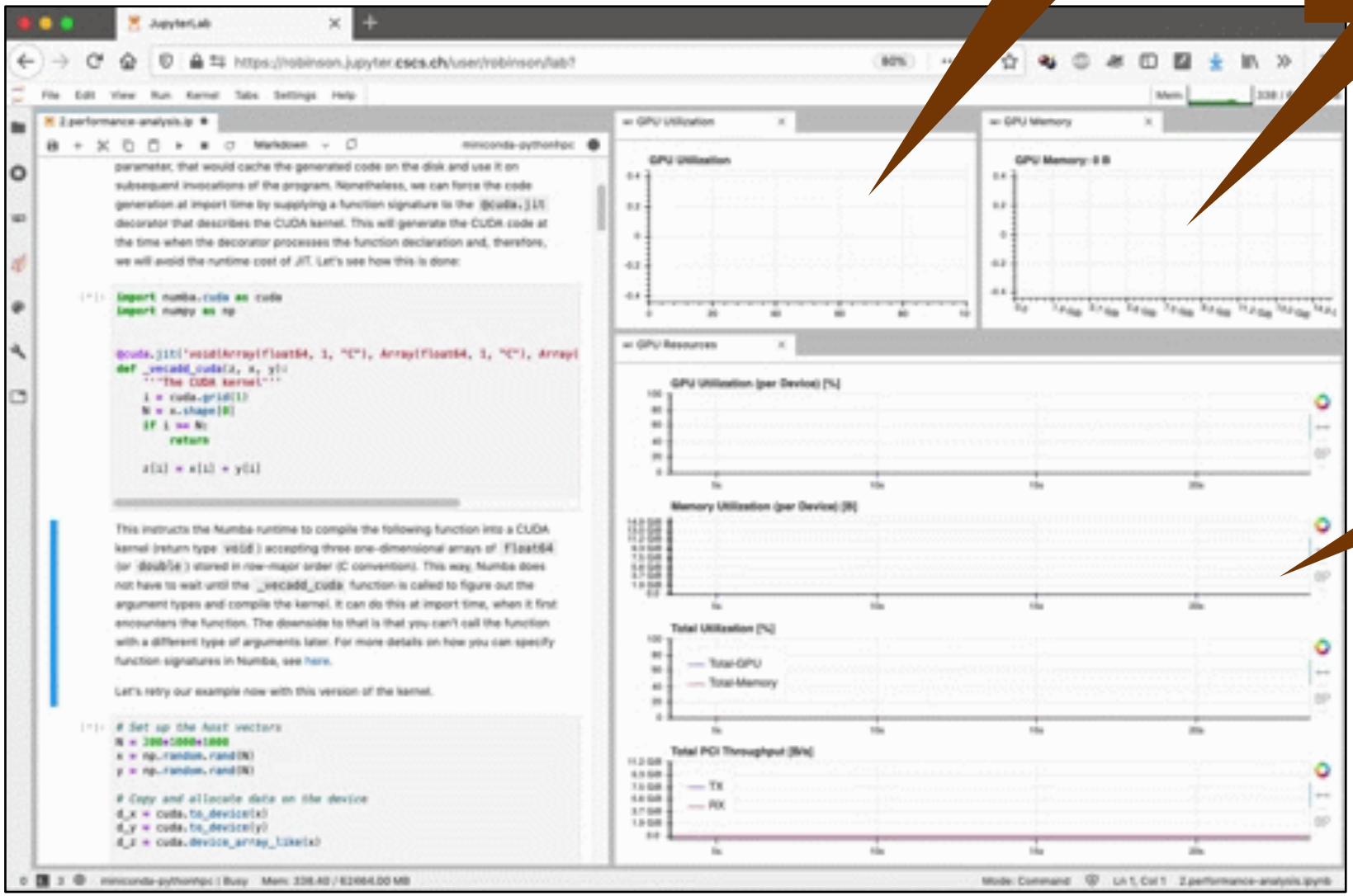
Project Id (account)

JupyterLab version

Launching notebook

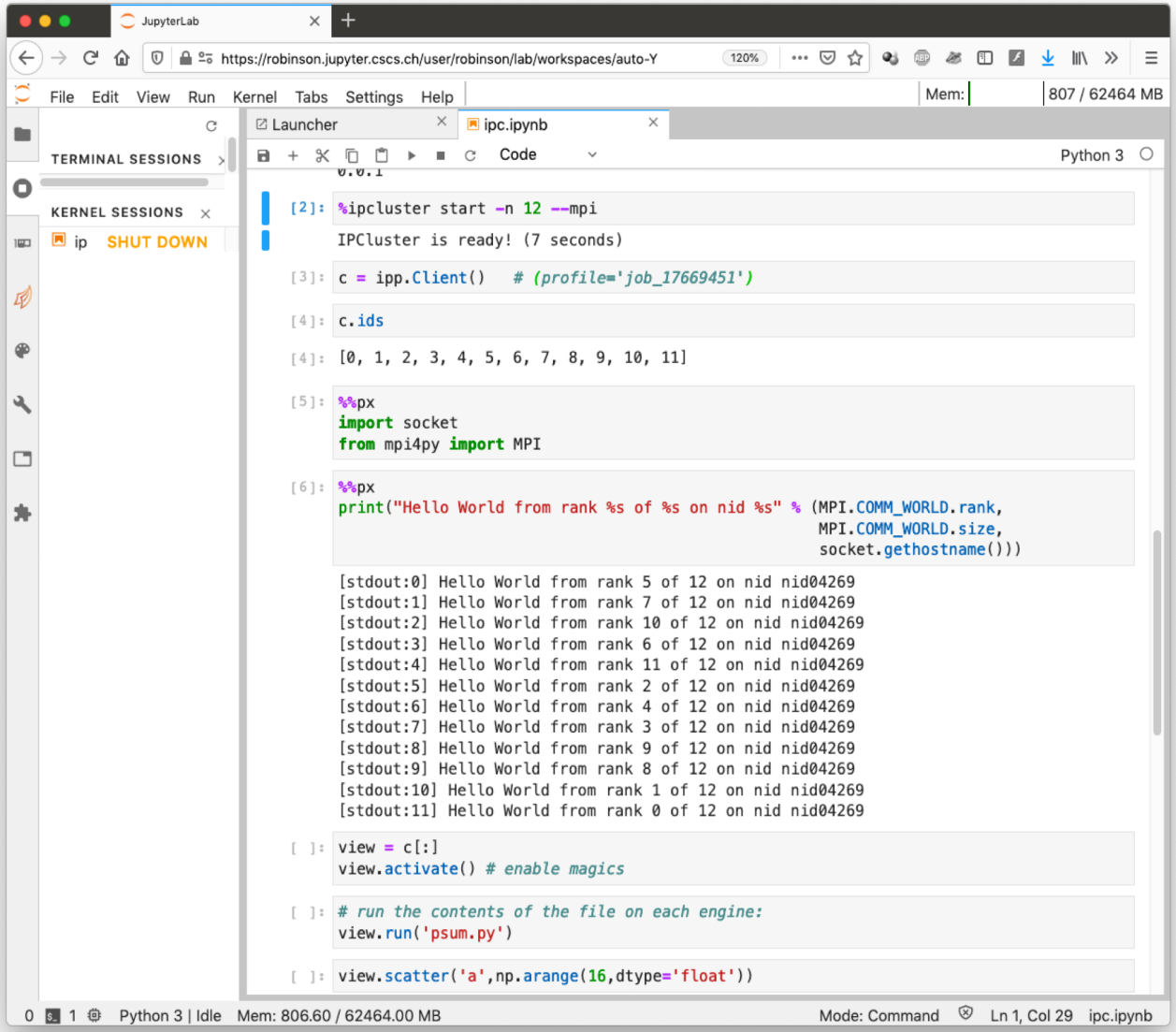


GPU dashboards

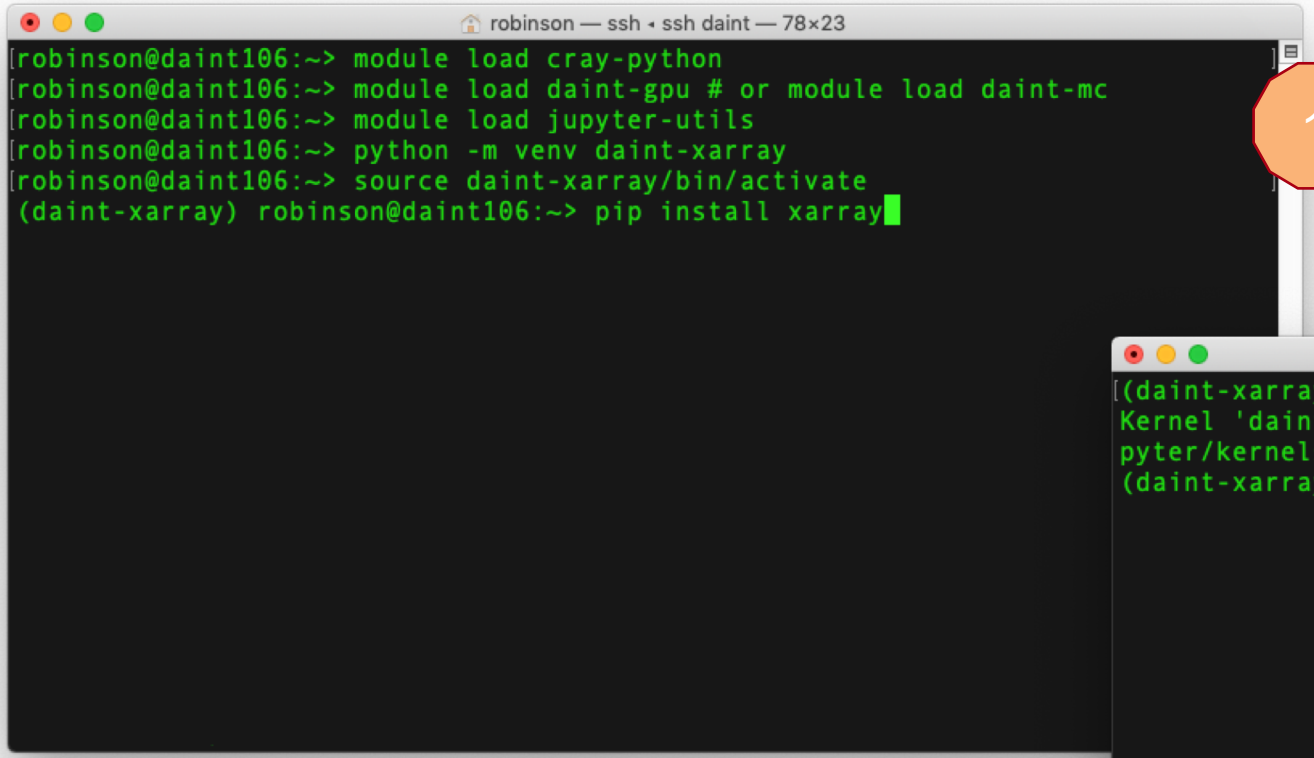


IPCluster magic

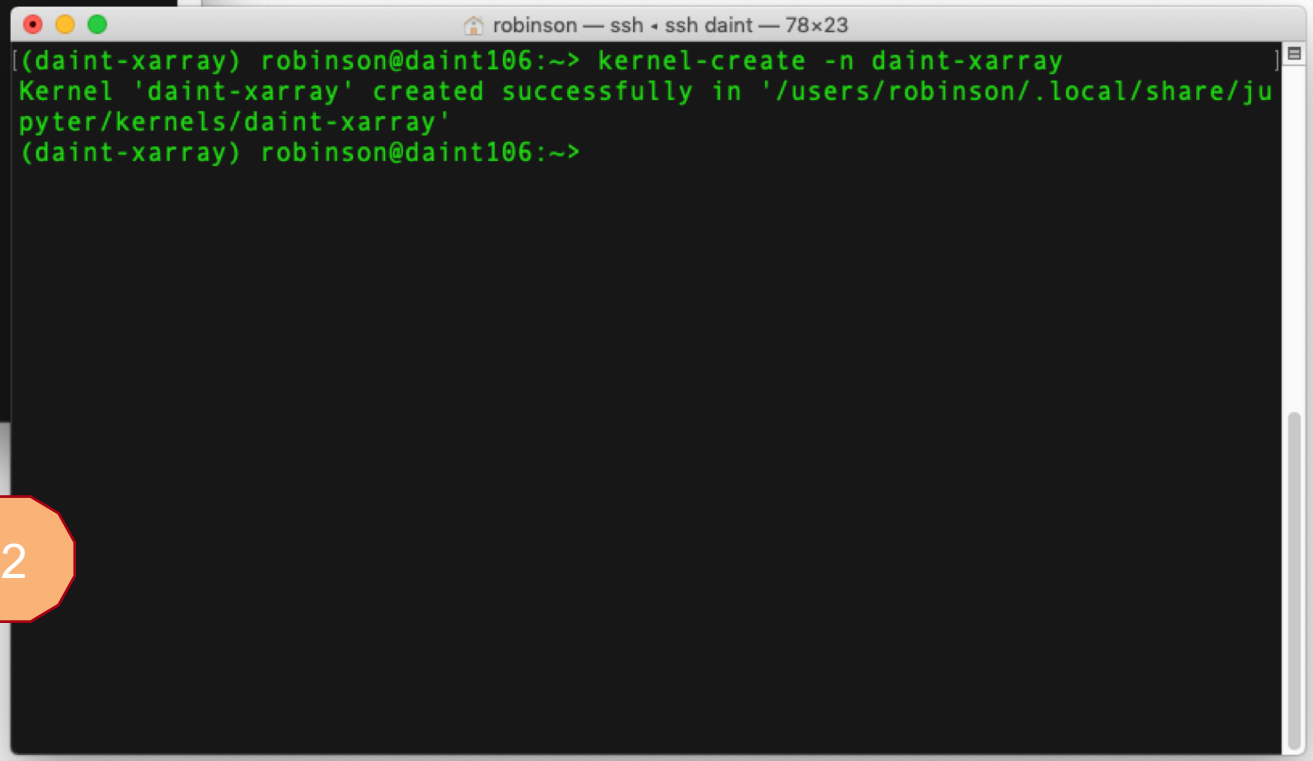
- IPCluster magic commands for automatically starting and stopping ipyparallel clusters (ipcontroller and ipengines)
- Enables the use of MPI directly in the notebook
- Developed at CSCS, inspired by NERSC's %ipcluster_magics



Creating and installing a custom kernel

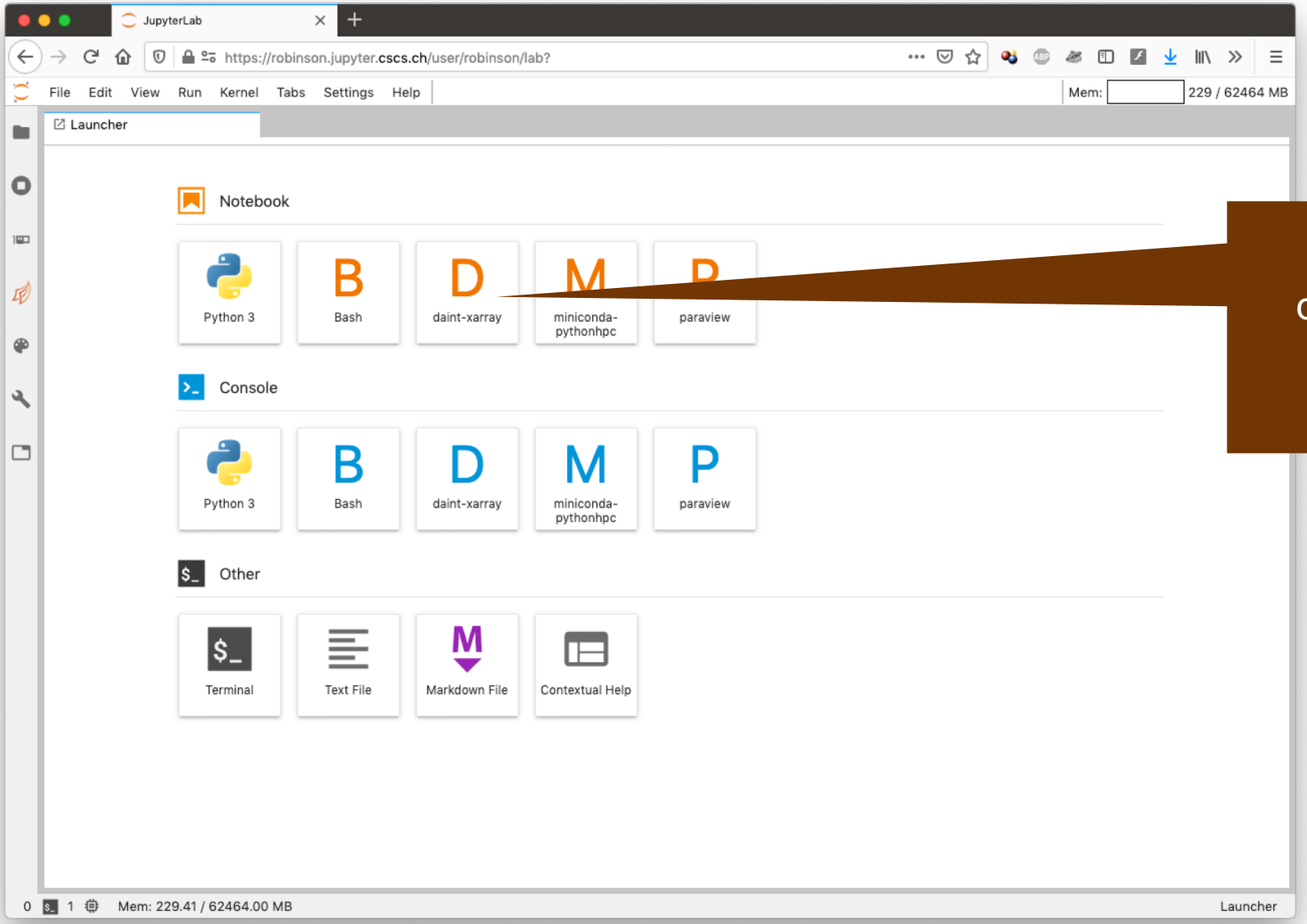


1 Create a virtualenv with your packages



2 Create a JupyterLab kernel from the virtualenv

Using the newly created kernel

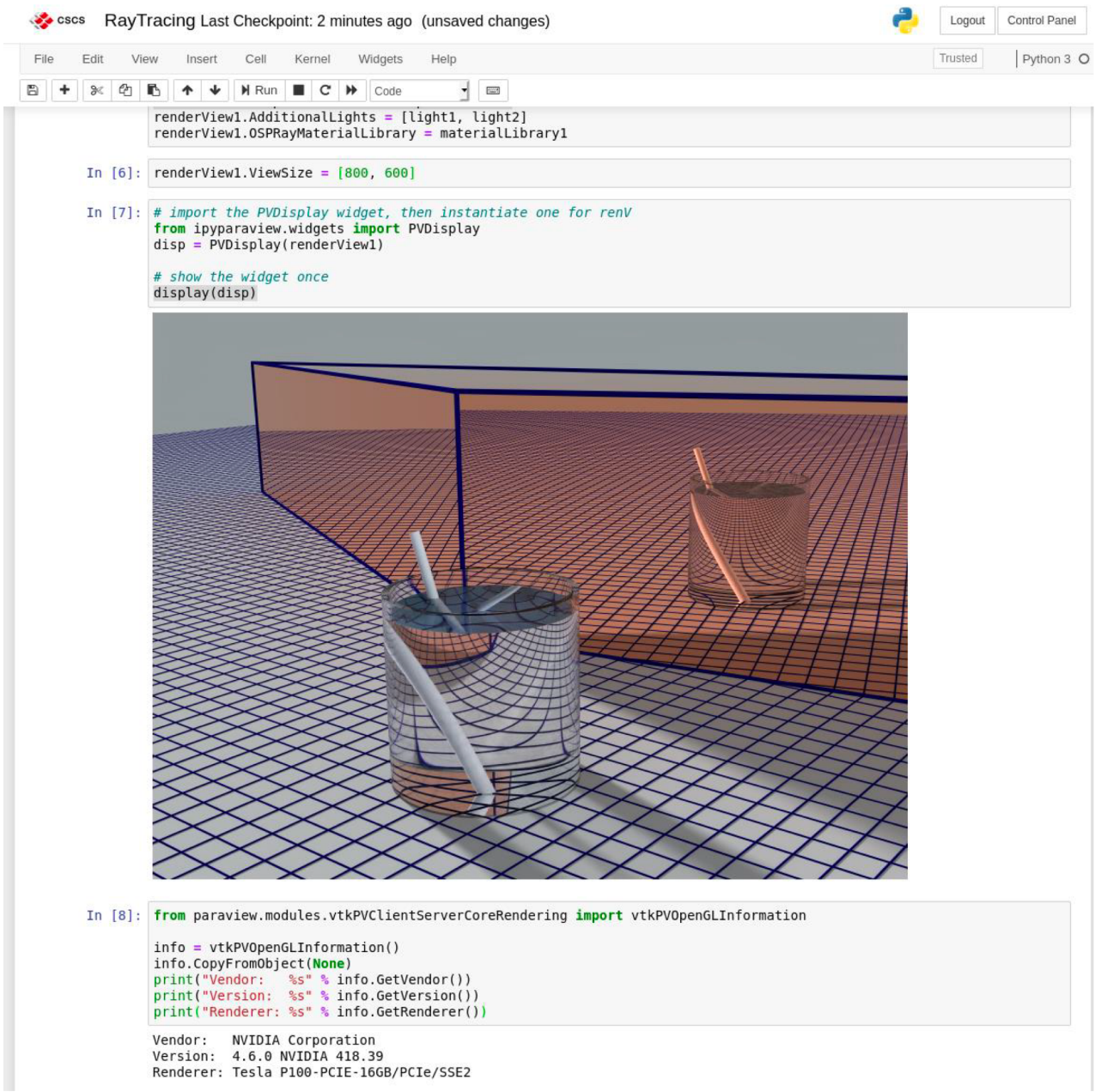


daint-xarray kernel will appear immediately

Integration improvement

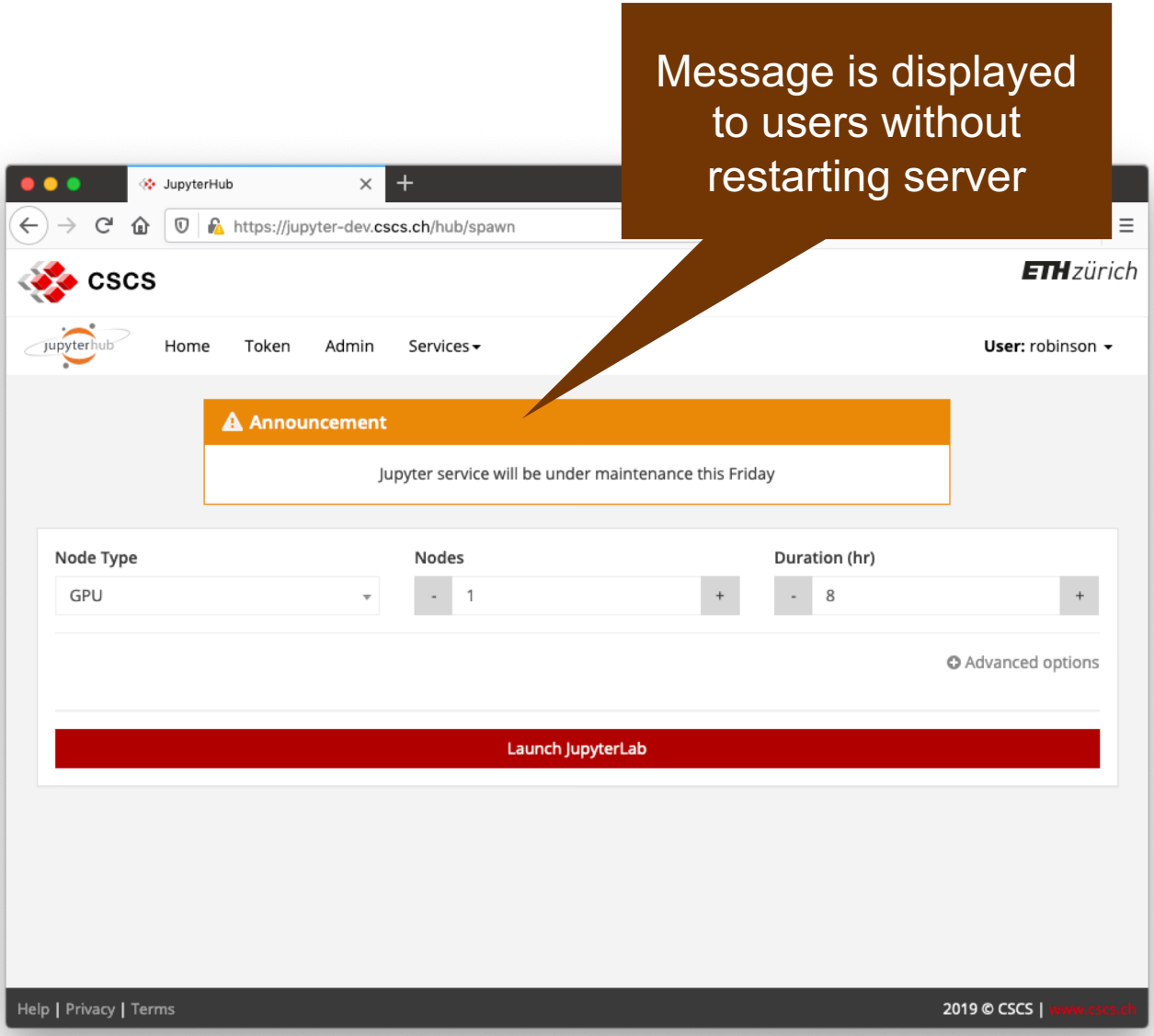
- IPyParaview is an iPython widget for server-side ParaView rendering in Jupyter

- Adding support for Julia notebook



JupyterHub announcement service

- We need to communicate with our users about status, upcoming outages, or share news
- Adding an announcement required system administrator intervention, and a restart of the JupyterHub server, which killed all user's notebook servers!
- JupyterHub recently added support for "services"
- We deployed an announcement service that is managed via a user interface
- Announcements can be added by user support staff directly
- No longer requires a restart of the JupyterHub server



Future plans

Automatization of deployment

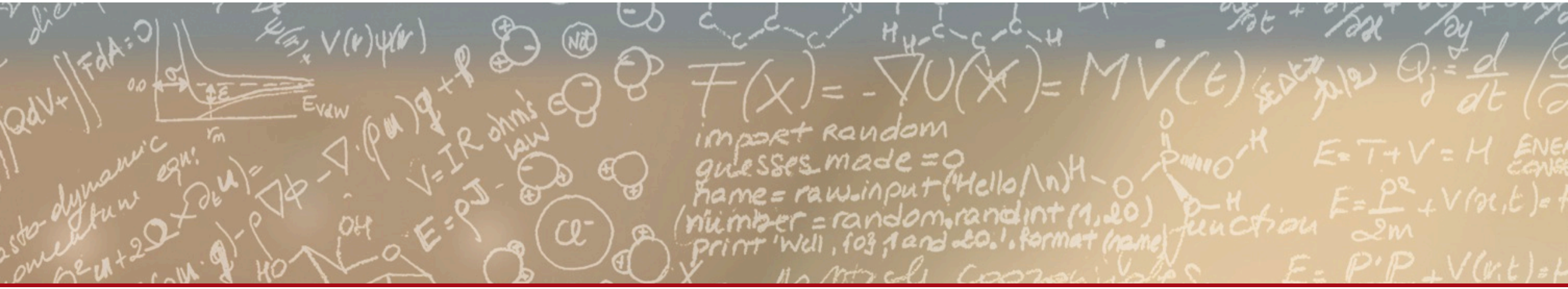
- Integrate Jupyter into the SRE process being developed
- Containerization of the Jupyter stack
- Automatic deployment of JupyterHub
- Environment configuration and management
- Improved unit testing and use case testing
- Better integration in IAM (for federation)

Next features in JupyterLab

- User-defined JupyterLab stack
  - Containerized JupyterLab stacks can help reproducibility
  - Stacks be customized by users (e.g. installing specific lab extensions)

- Sharing notebook with a link on Piz Daint
  - Make notebooks accessible on-demand from a link, exchange/reproducibility
  - Useful to move notebook close to data generated by an HPC simulation
  - Binder-like service for HPC
  - GitHub repo + requirements.txt/environment.yml → automatic deployment of a notebook
  - Container + Kubernetes automatic deployment

- Zero to JupyterHub
  - Instantiate a JupyterHub infrastructure on-demand
  - Uses Kubernetes and containers
  - JupyterHub as a Service



Thanks in particular to Tim Robinson