

Interactive Supercomputing with Jupyter Notebooks

Tutorial on Python for HPC

CSCS-USI Summer School 2021

Tim Robinson and Rafael Sarmiento (ETH Zurich / CSCS)

July 26, 2021

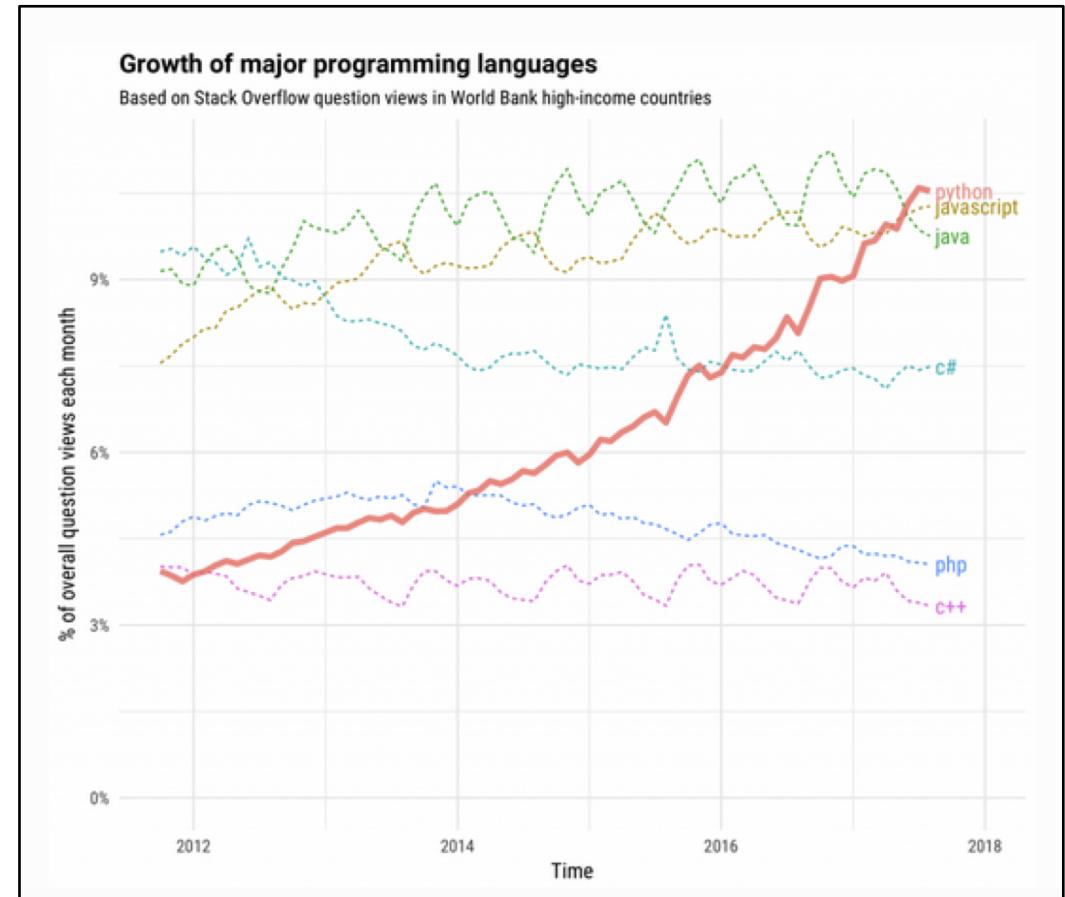
Schedule for today

- 09:00 – 10:30 Introduction to Jupyter [Notebook, Lab, Hub, ...]
- 10:30 – 11:00 Break
- 11:00 – 12:00 Python for HPC
- 12:00 – 13:00 Lunch
- 13:00 – 13:30 Networking on Spatial Chat (?)
- 13:30 – 15:30 Python for HPC (cont.)
- 15:30 – 16:00 Break
- 16:00 – 17:00 Q&A (Zoom)

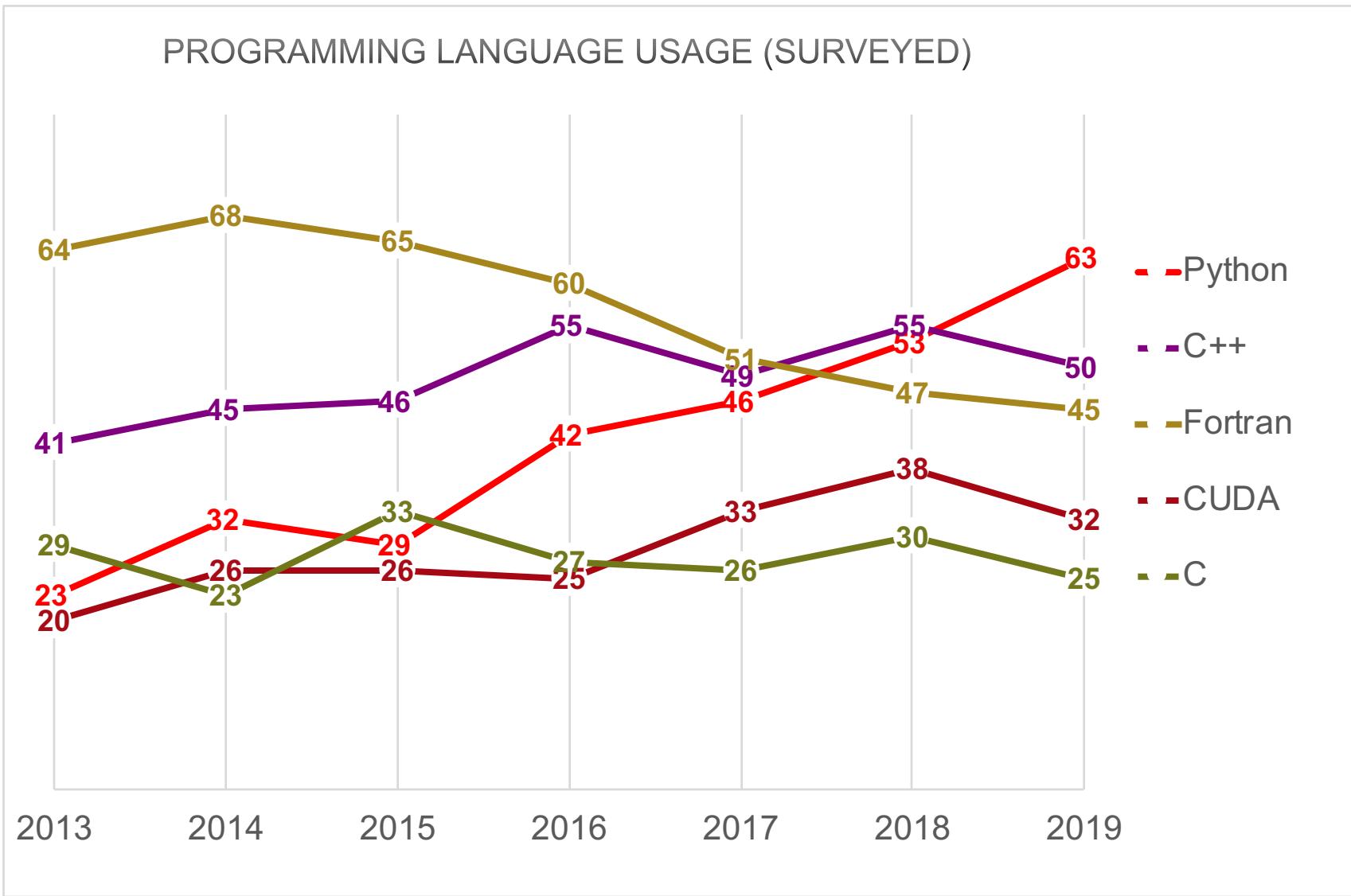
The rise of Python



- Python has grown to become a dominant language both in data analytics and general programming
- Rise fueled by computational libraries like Numpy, Pandas, and Scikit-Learn and libraries for visualization, interactive notebooks, collaboration, etc.
- Python long used as glue, for pre- and/or post-processing.. but increasingly used for simulation as well



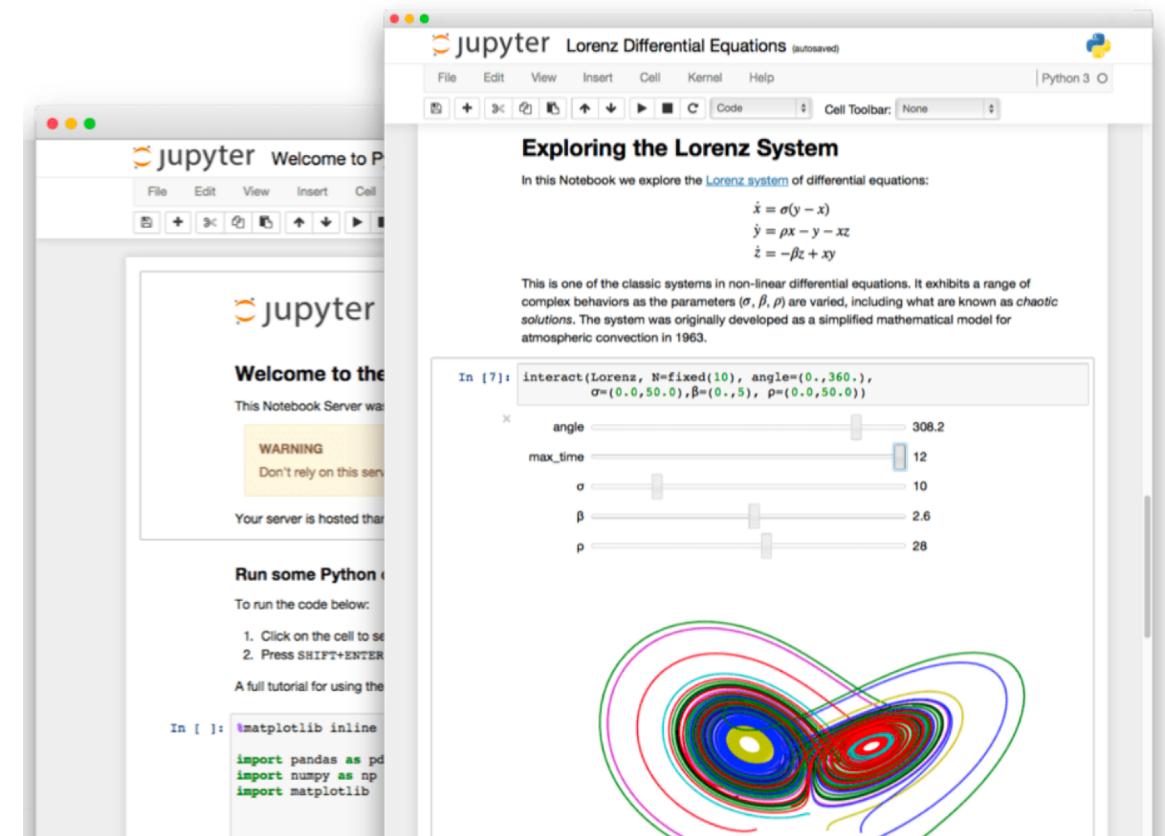
Python usage at CSCS



Jupyter Notebook

- Jupyter Notebook is an open-source web app for creating reproducible computational narratives
- Documents can contain live code, equations, narrative text, visualizations, and rich media
- Documents can be shared or exported to PDF, HTML, LaTeX, etc.
- Working environment includes
 - File browser
 - Terminal
 - Support for many languages: Python, R, Julia, C++, ...
 - Many server/client plugins
- Notebook is attached to a “kernel”, which does the actual computation

IP[y]:
IPython



JupyterLab

- JupyterLab is the next-generation user interface for Jupyter
- Higher degree of interaction between notebooks, documents, text editors and other activities (arrange with tabs/splitters)
- More advanced interactive development environment
- Uses the same notebook document format
- JupyterLab is the default interface at CSCS
- Classic notebook interface is still accessible by modifying the URL from /lab to /tree
 - Or: Help -> Launch Classic Notebook

The screenshot shows a Jupyter Notebook titled "Exploring the Lorenz System of Differential Equations". The code cell contains the Lorenz equations:

```
x = σ(y - x)
y = ρx - y - xz
z = -βz + xy
```

The output cell shows the numerical integration and plotting code:

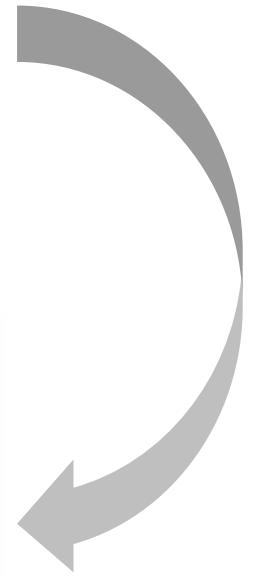
```
In [13]: %matplotlib inline
In [14]: from ipywidgets import interact, interactive
         from IPython.display import clear_output, display, HTML
In [15]: import numpy as np
         from scipy import integrate
         from matplotlib import pyplot as plt
         from mpl_toolkits.mplot3d import Axes3D
         from matplotlib.colors import cnames
         from matplotlib import animation
```

Computing the trajectories and plotting the result

```
We define a function that can integrate the differential equations numerically and then plot the solutions. This function has arguments that control the parameters of the differential equation ( $\sigma$ ,  $\beta$ ,  $\rho$ ), the numerical integration ( $N$ ,  $max\_time$ ) and the visualization ( $angle$ ).
```

```
In [16]: def solve_lorenz(N=10, angle=0., max_time=4.0, sigma=10.0, beta=8./3., rho=28.0):
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
```

The screenshot shows the JupyterLab interface with a tab for "lorenz.ipynb". The code and output are identical to the Jupyter Notebook version above, showing the Lorenz attractor plot.



JupyterLab Extensions

- JupyterLab is designed as a customizable, extensible environment
- Extensions can provide new themes, file viewers and editors, and renderers for rich output
- >100 GitHub repos tagged “jupyterlab-extensions”
- Examples include:
 - Memory and CPU usage, simple profiling
 - Dask dashboard
 - Tensorboard
 - GPU dashboards





CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETHzürich

Using notebooks in the cloud...

Sharing static notebooks

<https://nbviewer.jupyter.org/>

nbviewer

A simple way to share Jupyter Notebooks

Enter the location of a Jupyter Notebook to have it rendered here:

URL | GitHub username | GitHub username/repo | Gist ID

Go!

Programming Languages

IPython



IRuby



IJulia

An IJulia Preview

This notebook is a preview demo of IJulia: a [Julia language](#) backend combined with the [IPython](#) interactive environment. This combination allows you to interact with the Julia language using IPython's powerful [graphical notebook](#), which combines code, formatted text, math, and multimedia in a single document.



• Note: this is a preview, because it relies on pre-release bleeding-edge versions of Julia, IPython, and several Julia packages, as explained on the [Julia GitHub page](#), and functionality is evolving rapidly. We hope to have a more polished release soon.

Basic Julia interaction

Sharing live notebooks

<https://mybinder.org/>



Turn a Git repo into a collection of interactive notebooks

Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.

Build and launch a repository

GitHub repository name or URL

GitHub repository name or URL

GitHub ▾

Git branch, tag, or commit

Git branch, tag, or commit

Path to a notebook file (optional)

Path to a notebook file (optional)

File ▾

launch

Copy the URL below and share your Binder with others:

Fill in the fields to see a URL for sharing your Binder.

copy

Copy the text below, then paste into your README to show a binder badge:

copy

launch binder



CSCS

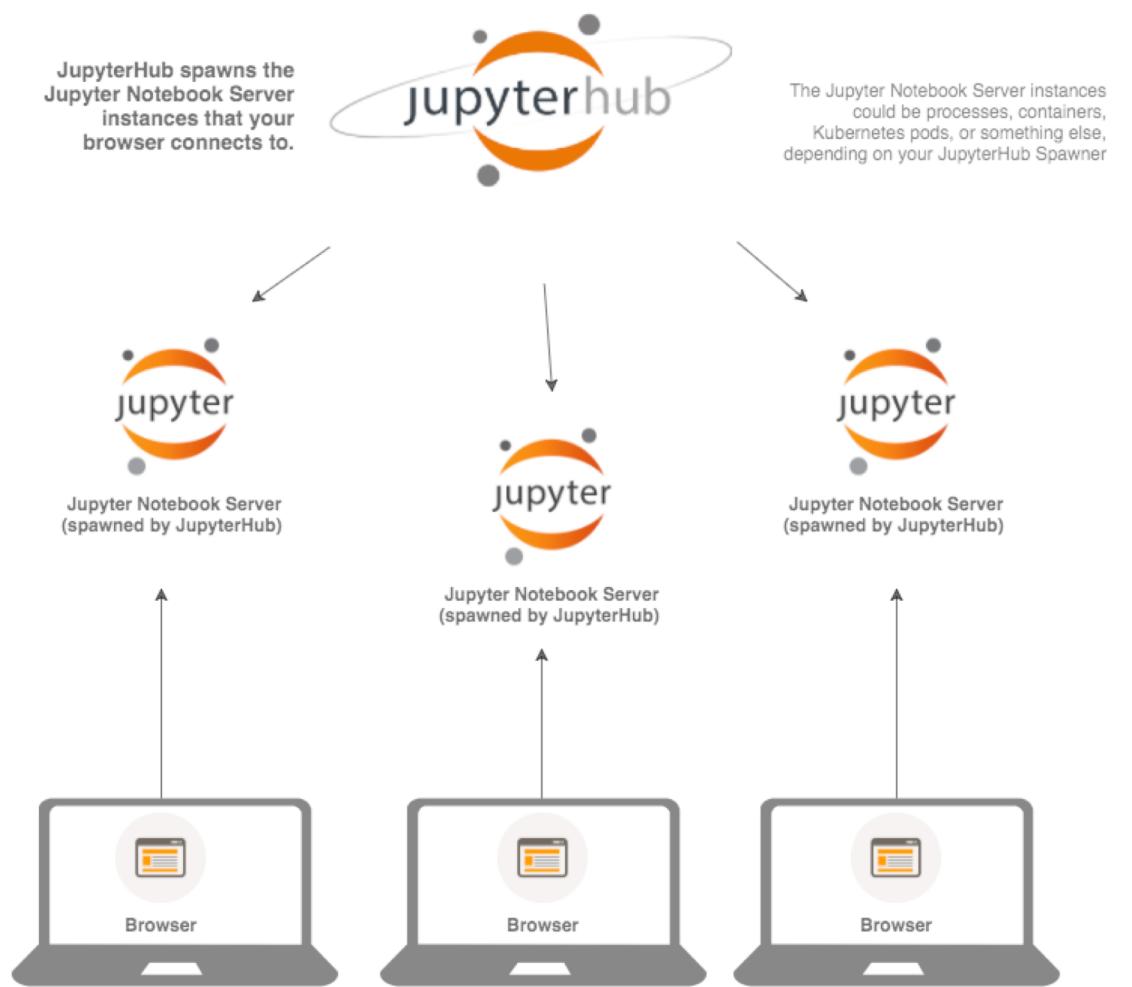
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETHzürich

Using notebooks on Piz Daint: JupyterHub

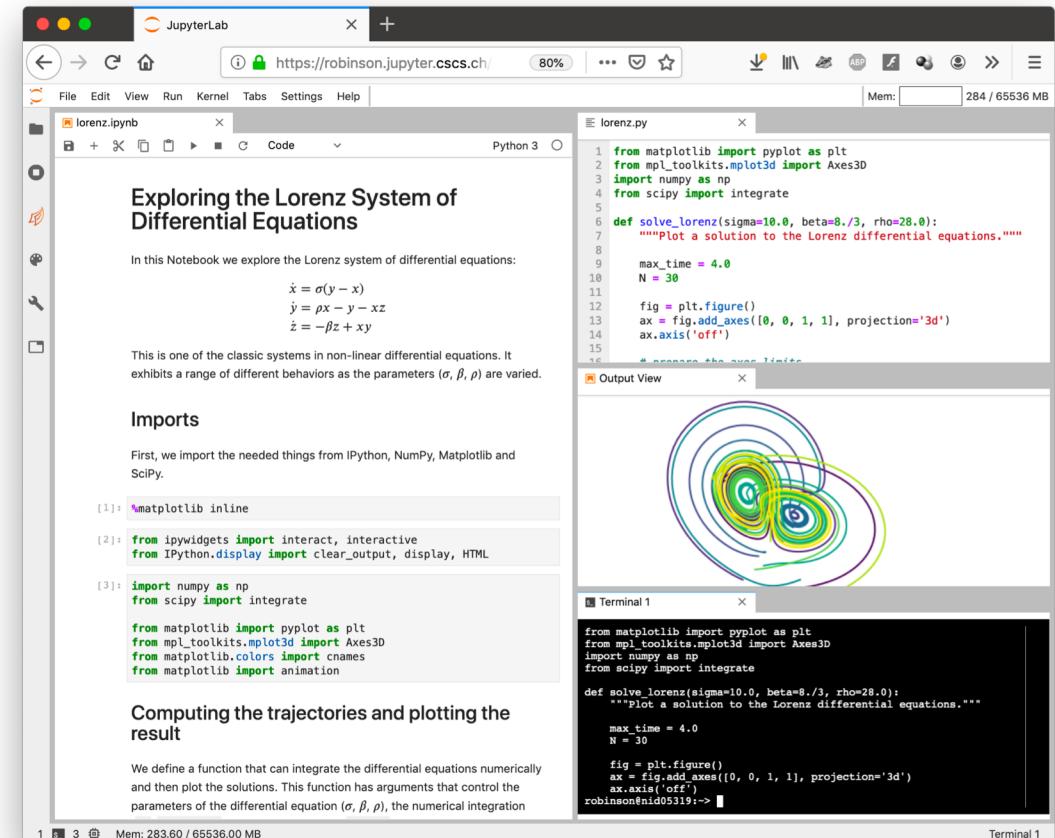
JupyterHub

- 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
- Main components:
 - **Hub** (tornado process)
 - **configurable http proxy** (node-http-proxy) that receives the requests from the client's browser
 - multiple **single-user Jupyter notebook servers** (Python/IPython/tornado) that are launched and monitored by **Spawners**
 - an **authentication class** that manages how users can access the system



JupyterHub for Piz Daint

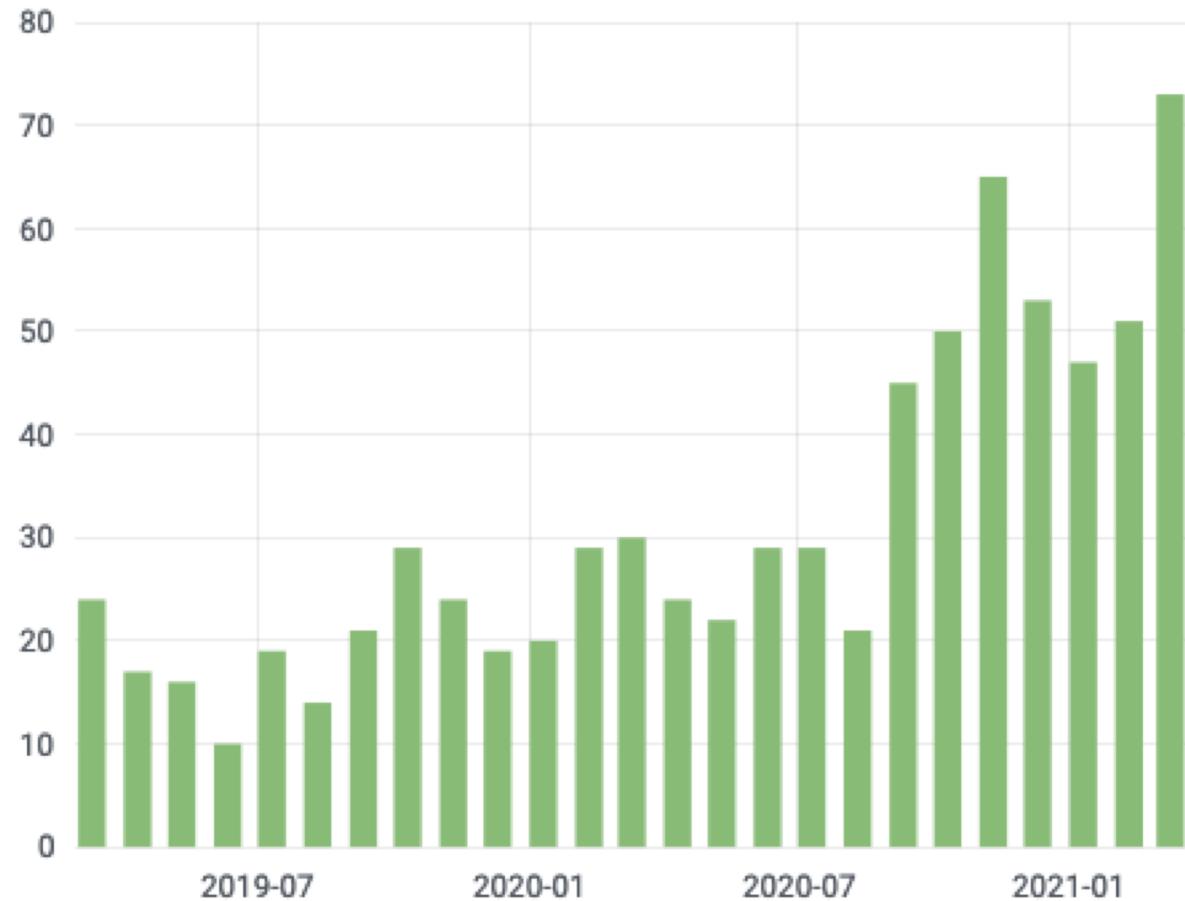
- JupyterHub for Piz Daint is accessible at:
<https://jupyter.cscs.ch>
- Available to anyone with a compute allocation on Piz Daint
- Servers are launched on the compute nodes (GPU or multicore)
- You have dedicated access to the full compute capability of the node(s)
- Special Slurm reservations for interactive computing grow and shrink automatically according to demand
- In most cases, users will wait less than five minutes for a notebook server



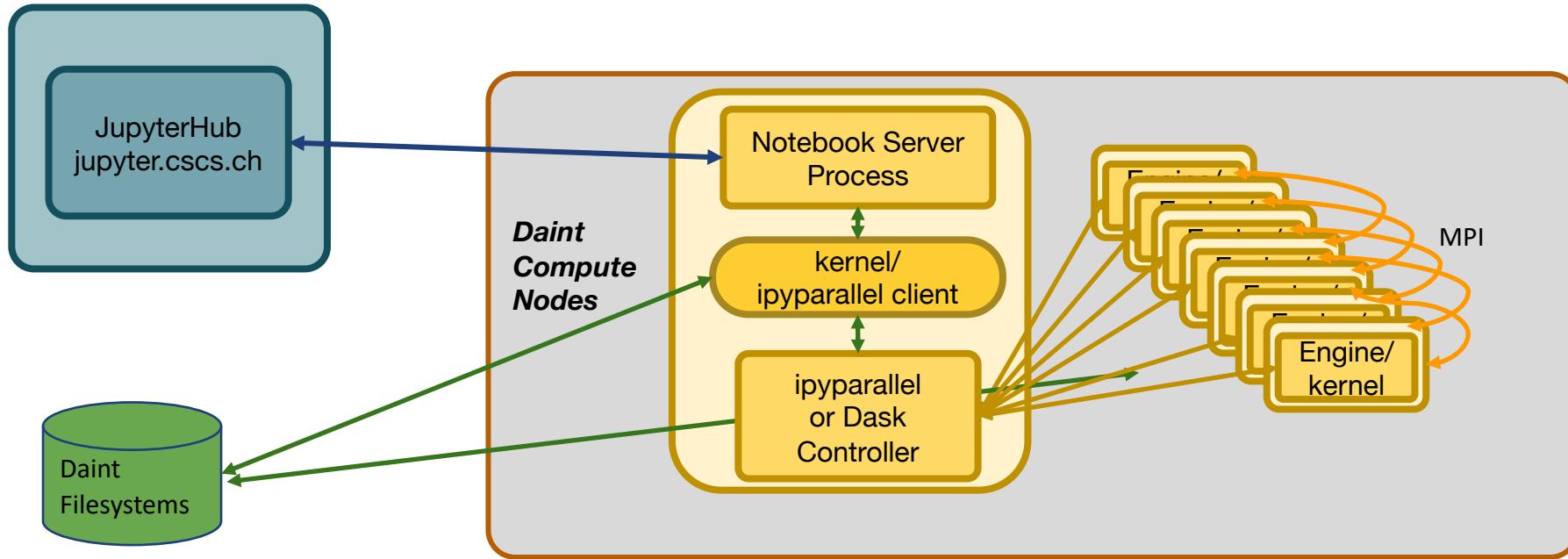
The screenshot shows a JupyterLab interface with the following components:

- Top Bar:** Shows the title "JupyterLab" and the URL "https://robinson.jupyter.cscs.ch".
- Notebook Area:** A tab labeled "lorenz.ipynb" is active. The content of the notebook includes:
 - A section titled "Exploring the Lorenz System of Differential Equations" with a description of the Lorenz system.
 - A section titled "Imports" listing imports for matplotlib, ipywidgets, numpy, and scipy.
 - A section titled "Computing the trajectories and plotting the result" describing a function to solve the Lorenz equations.
- Code Area:** A tab labeled "lorenz.py" contains the Python code for solving the Lorenz equations, which is identical to the code in the notebook.
- Output View:** Displays a 3D plot of the Lorenz attractor, showing three interlocking spiral trajectories in red, green, and blue.
- Terminal 1:** A terminal window showing the command used to run the script: "robinson@nid05339:~\$ python lorenz.py".

JupyterHub monthly users on Piz Daint



JupyterHub on Piz Daint

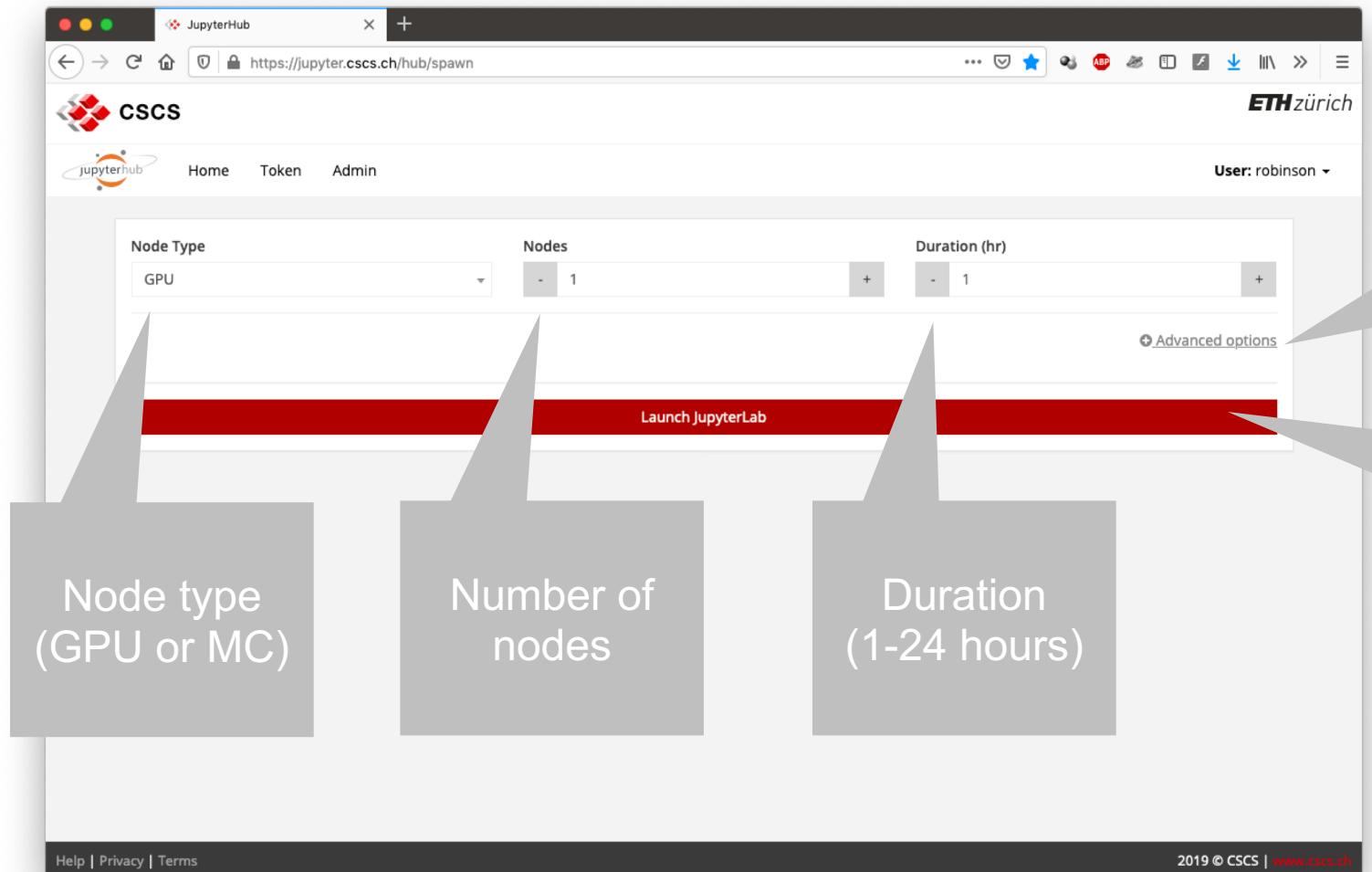


Challenges with interactive computing on HPC

- Shared parallel filesystems
- Network connectivity
- Slurm
- Warm swapping
- All kinds of other possible issues - usually these are temporal!

- Debugging problems:
 - Check for any maintenance announcements on the spawner page
 - Check your own internet connectivity
 - Check for old JupyterHub sessions in browser tabs (especially if kernels are not starting or are slow to connect or restart)
 - Inspect the Jupyter log file in your \$SCRATCH directory

Basic options



Advanced options

The screenshot shows the JupyterHub spawn interface on a web browser. A large callout bubble labeled "Reservation" points to the "Advanced Reservation" section of the form.

Node Type: GPU

Nodes: 1

Duration (hr): 1

Queue: Dedicated Queue (Max. 4 Nodes)

Project Id (leave empty for default): (empty input field)

Advanced Reservation: (empty input field)

JupyterLab Version: 1.1.1

Start IPyParallel Cluster with MPI Support? No Yes

MPI Processes Per Node (default: one process per virtual core): 1

Start Distributed Dask Cluster? No Yes

Dask Tasks Per Node (default: one task per node): 1

ⓘ | the number of threads = ncores / nprocesses

[Help](#) | [Privacy](#) | [Terms](#)

2019 © CSCS | www.cscs.ch

JupyterLab interface

Standard output and stderr written to:

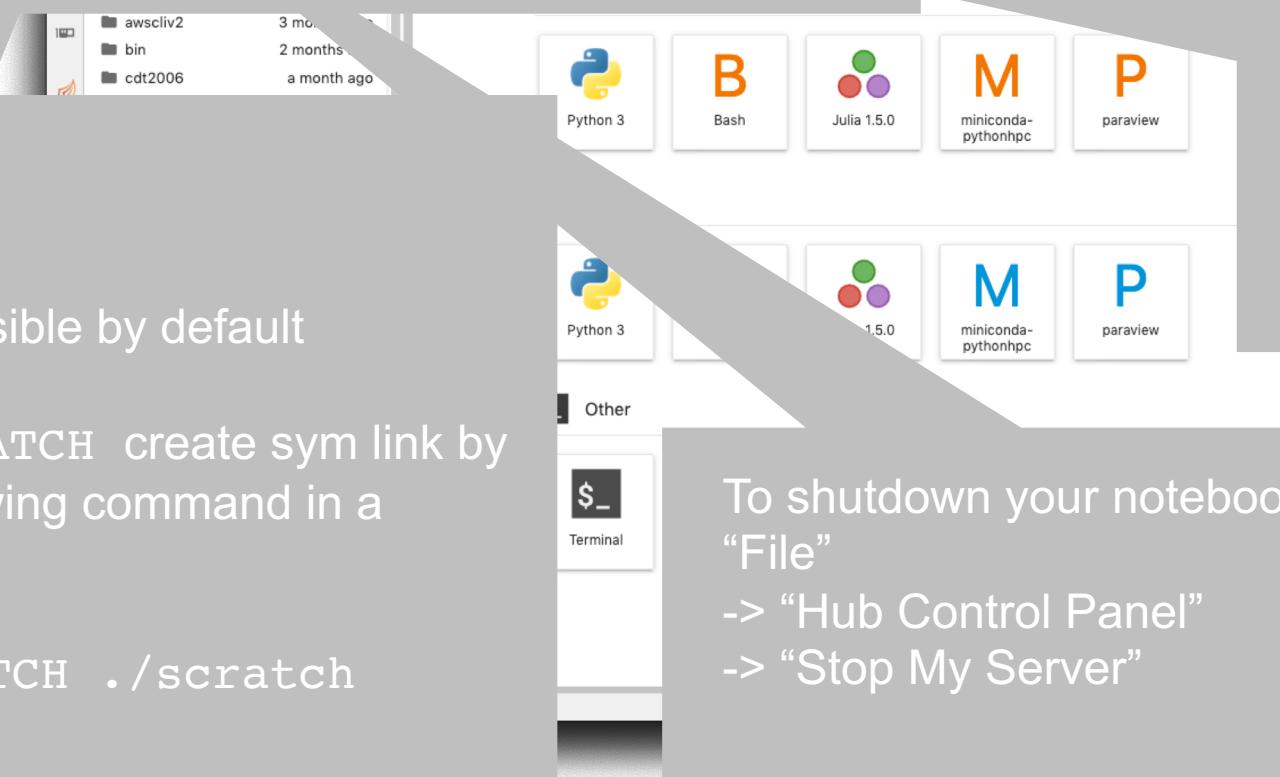
```
$SCRATCH/jupyterhub_slurmspawner_<jobid>.log
```

File Browser

`$HOME` is accessible by default

To access `$SCRATCH` create sym link by issuing the following command in a notebook cell:

```
!ln -s $SCRATCH ./scratch
```



Main Workspace

Launch notebooks with default and custom kernels

Launch a terminal

Text editors, etc.

To shutdown your notebook server:
“File”
-> “Hub Control Panel”
-> “Stop My Server”



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETHzürich

Demo of JupyterLab interface



Let's get going...

Please open a browser (**Firefox** or **Chrome**, no guarantees for the others...) and visit...

<https://jupyter.cscs.ch>