



INTERACTIVE HPC WITH JUPYTERLAB

EDIH (European-Data-Innovation-Hub) -Workshop, Part 1

2023-10-23 | JENS HENRIK GÖBBERT (J.GOEBBERT@FZ-JUELICH.DE)

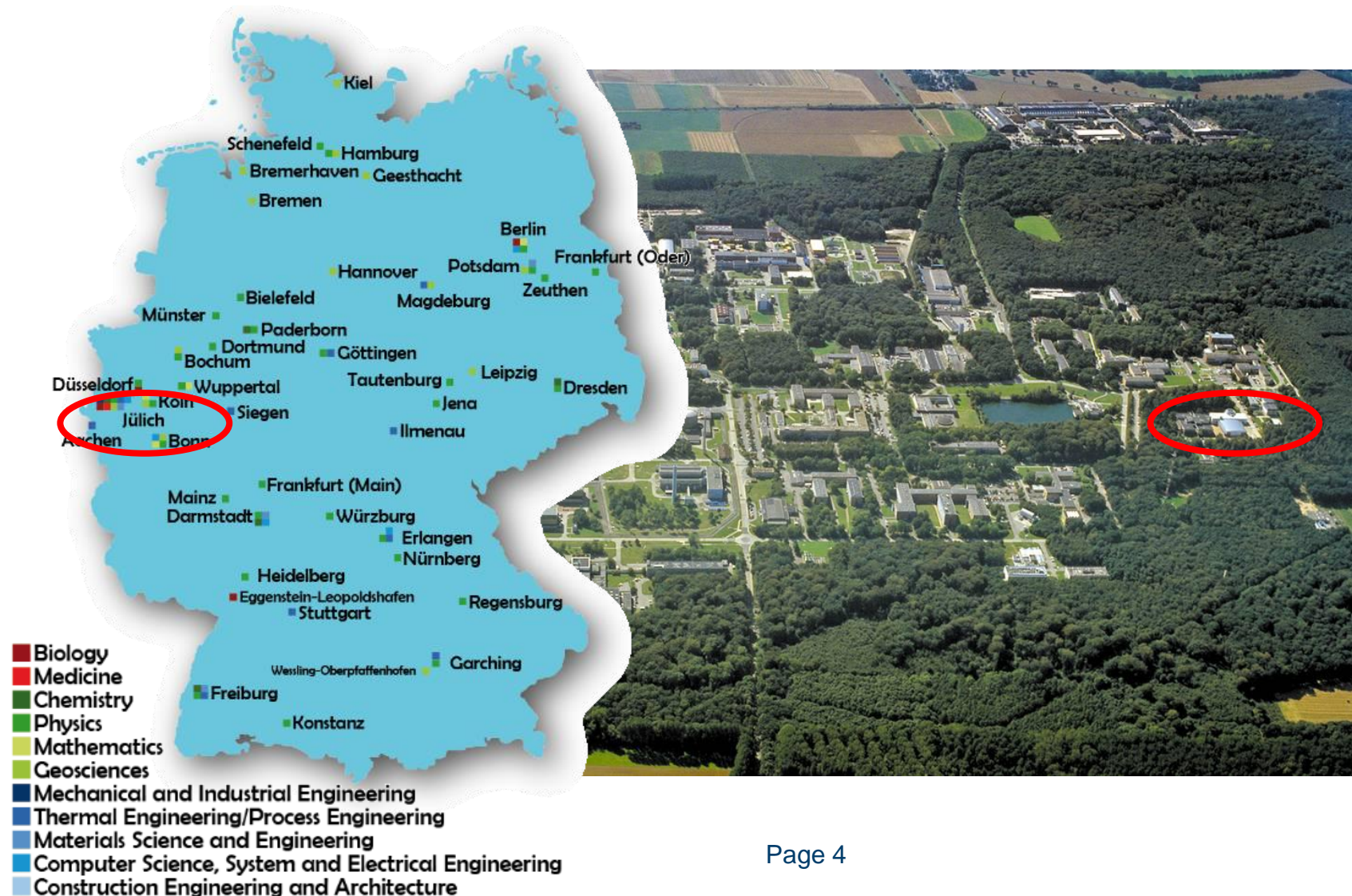
WELCOME

- Hello !
- Live document for this class
 - https://gitlab.jsc.fz-juelich.de/hedgedoc/S46PXGluSiuMv1Vgw_UU_Q
- Class repository
 - **<https://gitlab.jsc.fz-juelich.de/jupyter4jsc/workshop-2023.10-jupyter4hpc>**
- This is an short version of the 3-day workshop, the material of which you can find here:
 - <https://gitlab.jsc.fz-juelich.de/jupyter4jsc/training-2023.04-jupyter4hpc>

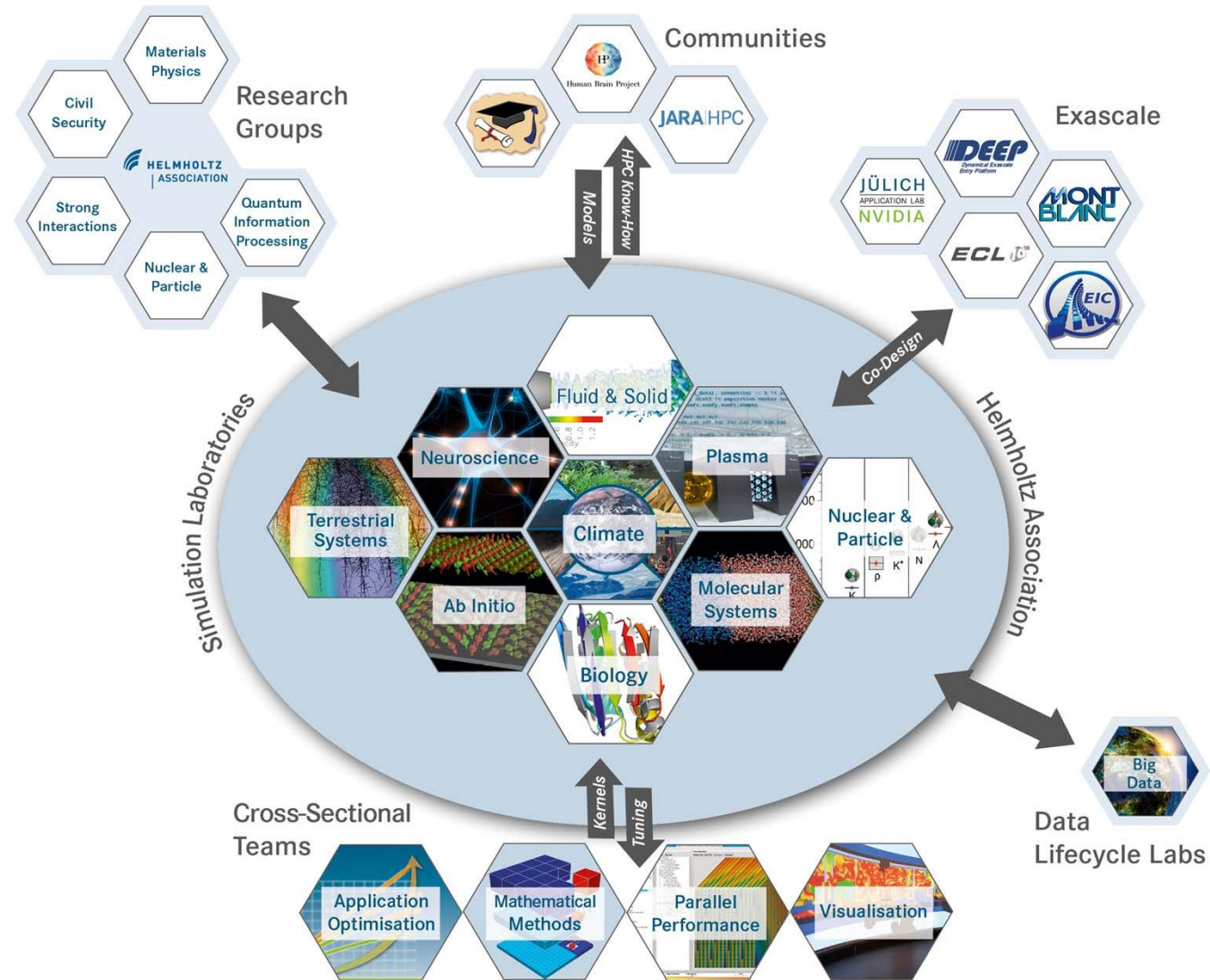
JÜLICH SUPERCOMPUTING CENTRE

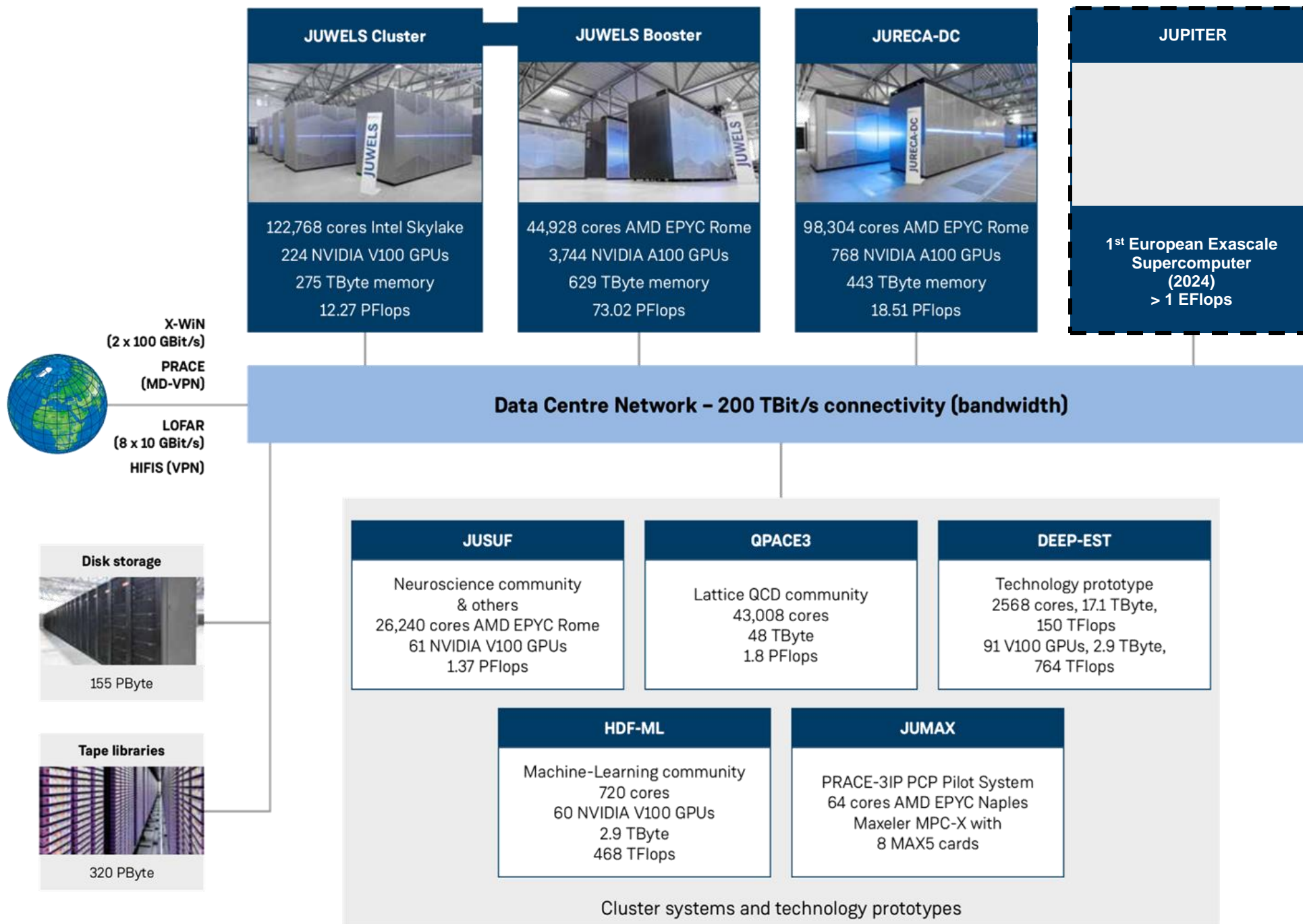
JÜLICH SUPERCOMPUTING CENTRE (JSC)

TIER-0/1 HPC RESOURCES OF THE HIGHEST PERF. CLASS

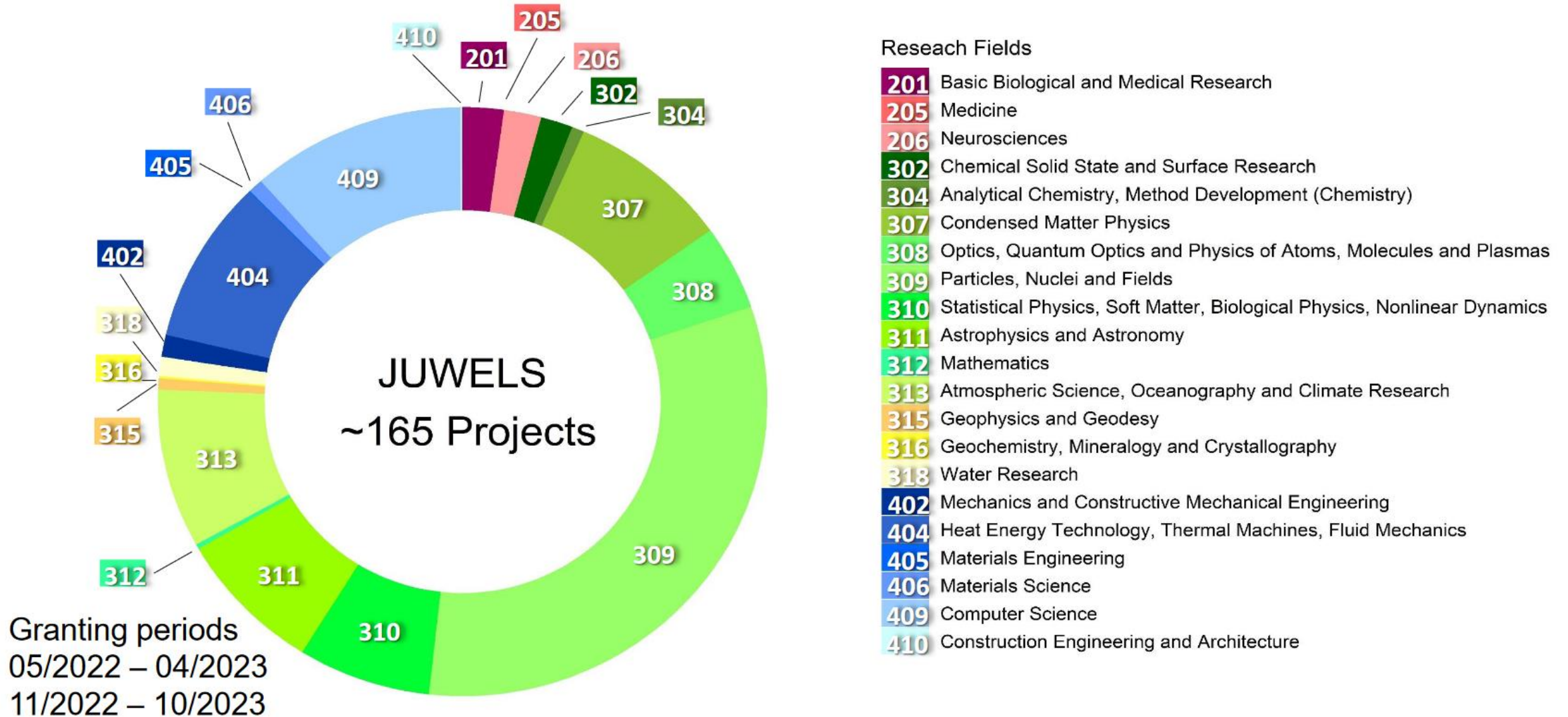


DOMAIN SPECIFIC SUPPORT

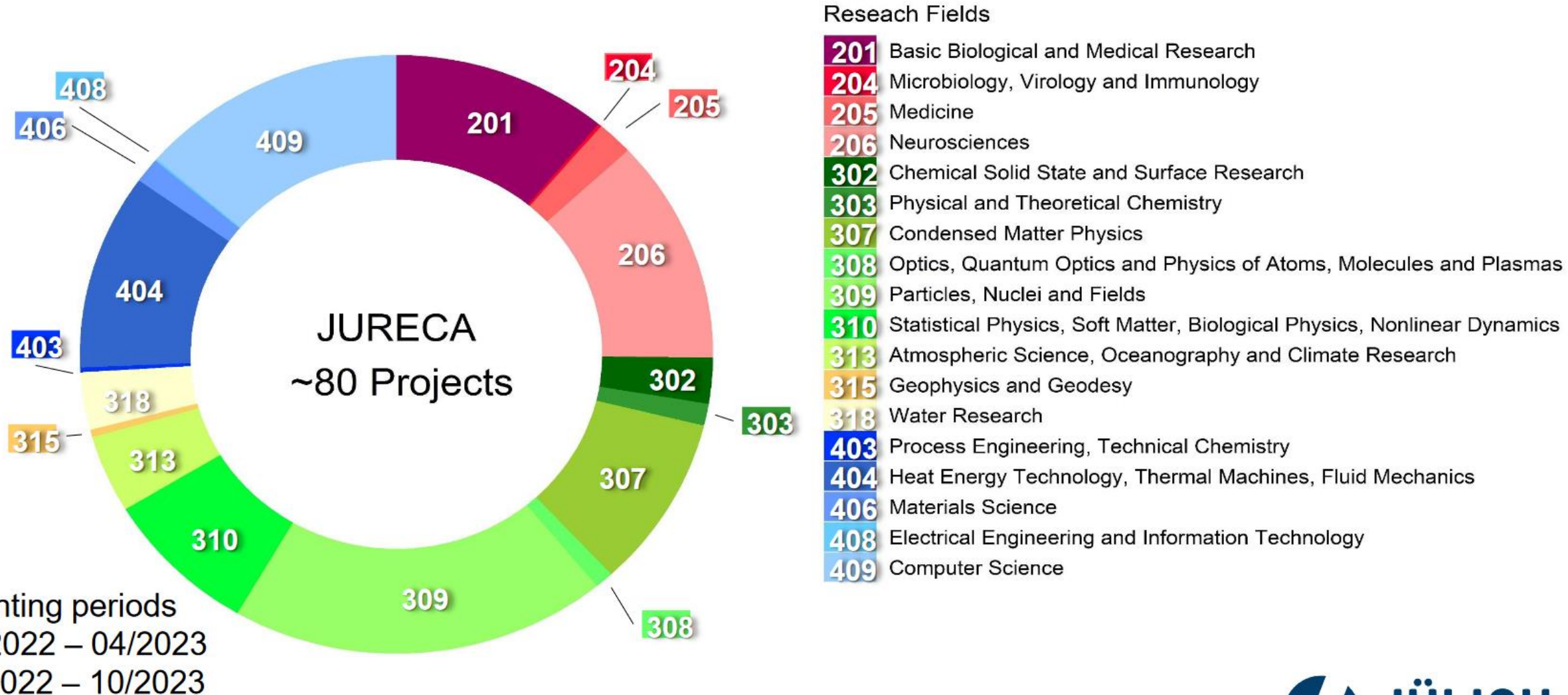




RESEARCH FIELDS ON JUWELS (CLUSTER + BOOSTER)



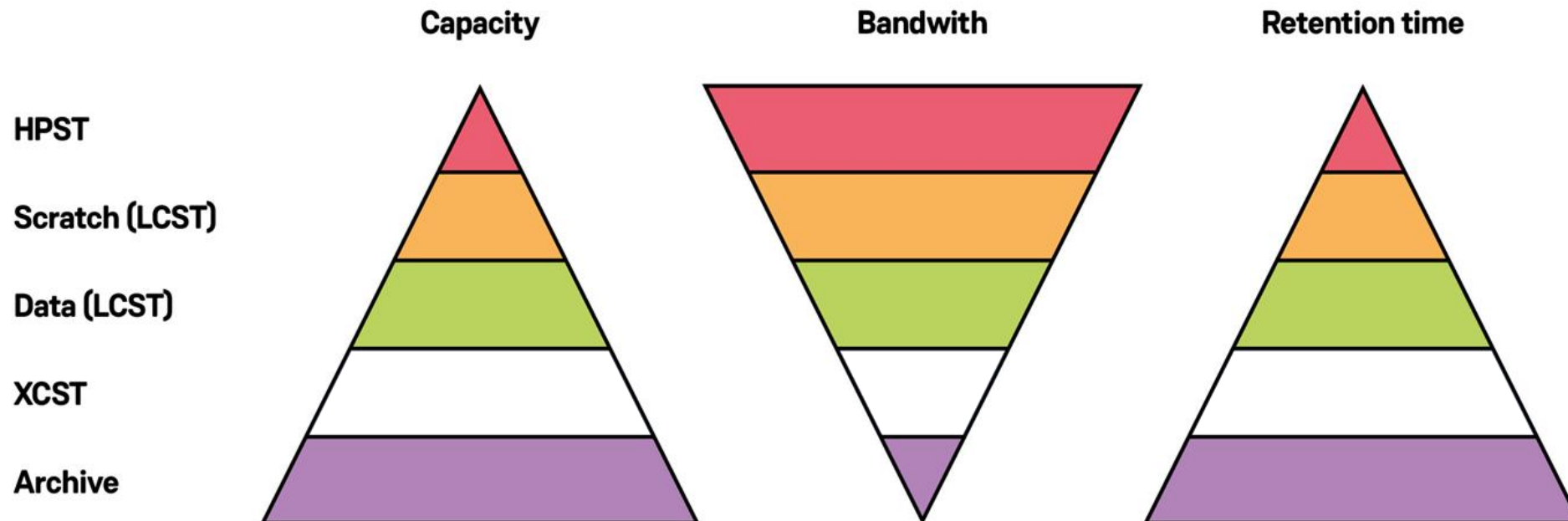
RESEARCH FIELDS ON JURECA (CLUSTER + BOOSTER)





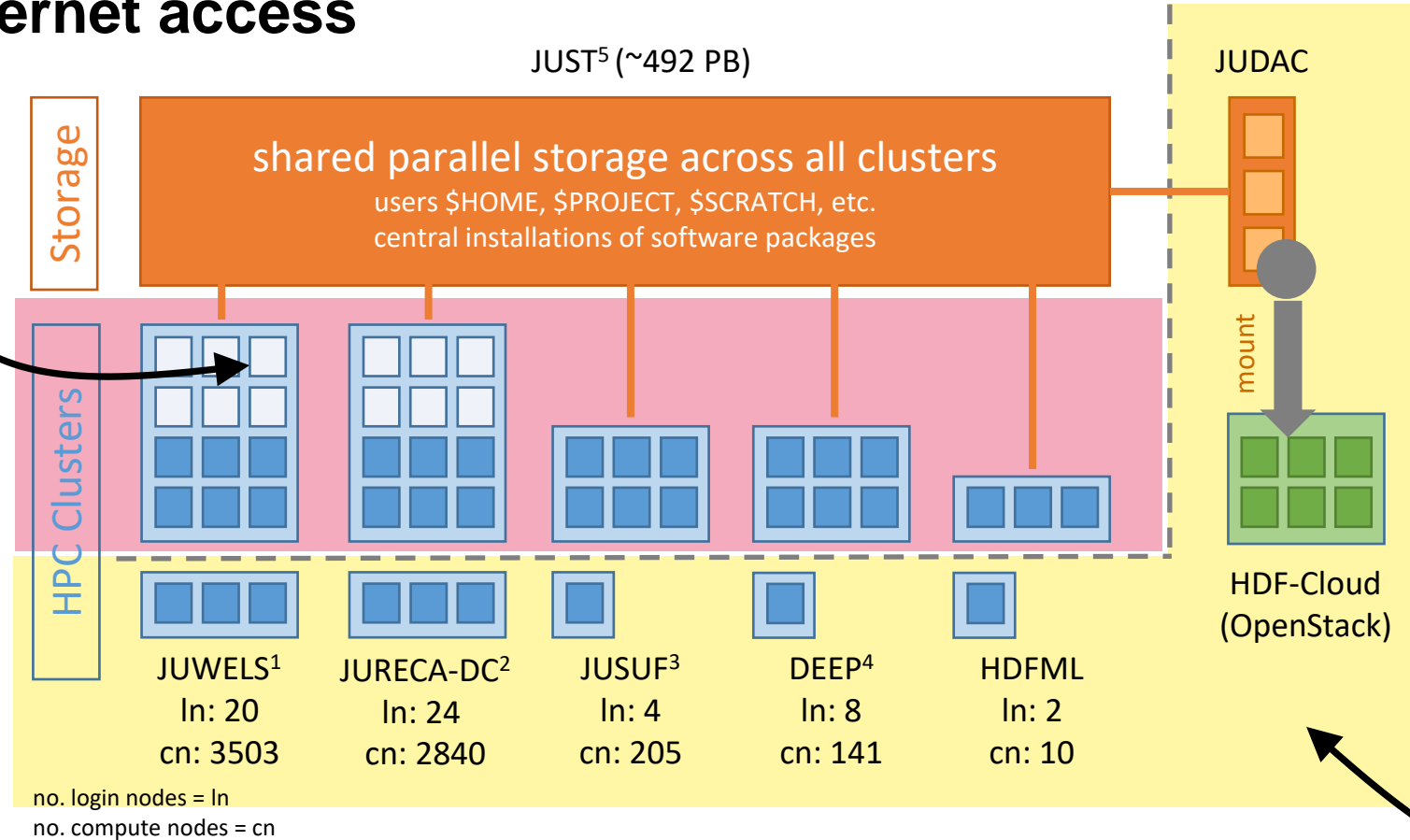
JUST (Juelich Storage cluster)

- One central storage infrastructure for HPC
- Total gross capacity
 - NVMe disks: ~2 PB
 - Spinning disks: ~180 PB
 - Tape: ~310 PB
- Software:
 - IBM Spectrum Scale
 - IBM Spectrum Protect
 - DDN Infinite Memory Engine
- **Project partners: DDN, IBM, Lenovo, ProCom**



SUMMARY – COMPUTE RESOURCES @ JSC

NO internet access



[1] <https://apps.fz-juelich.de/jsc/hps/juwels/configuration.html>

[2] <https://apps.fz-juelich.de/jsc/hps/jureca/configuration.html>

[3] <https://apps.fz-juelich.de/jsc/hps/jusuf/cluster/configuration.html>

[4] https://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/DEEP-EST/_node.html

[5] https://www.fz-juelich.de/ias/jsc/EN/Expertise/Datamanagement/OnlineStorage/JUST/Configuration/Configuration_node.html

ACCESS TO COMPUTE RESOURCES

PRE-ACCESS TODOS

1) Register & Login

- ✓ <https://judoor.fz-juelich.de>

2) Join the project „paj2206“

- ✓ Wait to get joined by the project PI

3) Sign usage agreement

- ✓ Wait for creation of HPC accounts

4) Check Connected Services:

- ✓ jupyter-jsc

The screenshot shows the JU JÜLICH SUPERCOMPUTING CENTRE user interface. At the top, there is a navigation bar with 'JU Your account', 'Mentoring', a search bar, and 'Detailed Statistics'. The main content area is divided into several sections:

- Account:** Fields for Salutation, E-mail address (with a checkmark), Telephone, and Address.
- Mentored projects:** A section for managing projects.
- Systems:** A table listing systems and their status:
 - judac:** Managed by training2109, with a green checkmark and 'Usage agreement confirmed on 18.04.2021'.
 - jureca:** JURECA-DC_GPU, managed by training2211, with a red X and the message 'You need to sign the usage agreement to access this system'.
- Projects:** A section for managing projects, showing 'Interactive High-Performance Computing with Jupyter @ JSC' with a green checkmark and 'training2211'.
- Software:** A section for managing software.
- Connected Services:** A section for managing services, showing 'trac', 'llview', 'jards', 'gitlab', and 'jupyter-jsc' with a green checkmark.

For more details, please visit
https://gitlab.jsc.fz-juelich.de/hedgedoc/S46PXGluSiuMv1Vgw_UU_Q?view#Getting-Access

PRE-ACCESS TODOS

1) I

2) J

3) S

4) C

<https://judoor.fz-juelich.de>



PRE-ACCESS TODOS

1) Register & Login

- ✓ <https://judoor.fz-juelich.de>

2) Join the project „paj2206“

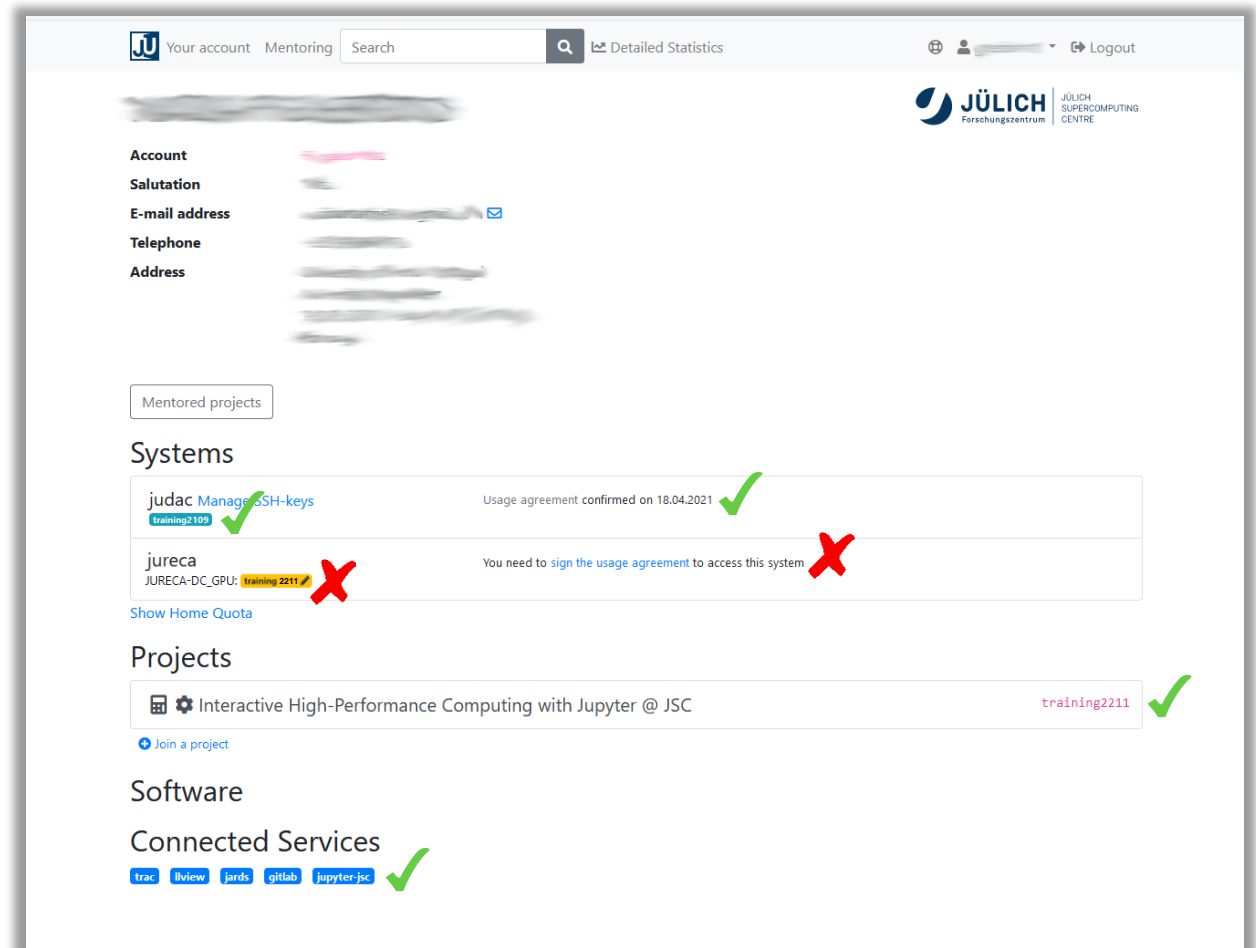
- ✓ Wait to get joined by the project PI

3) Sign usage agreement

- ✓ Wait for creation of HPC accounts

4) Check Connected Services:

- ✓ jupyter-jsc



For more details, please visit
https://gitlab.jsc.fz-juelich.de/hedgedoc/S46PXGluSiuMv1Vgw_UU_Q?view#Getting-Access

PRE-ACCESS TODOS

1) I

2) J

3) S

4) C

<https://judoor.fz-juelich.de>

Project id = „paj2206“

PRE-ACCESS TODOS

1) Register & Login

- ✓ <https://judoor.fz-juelich.de>

2) Join the project „paj2206“

- ✓ Wait to get joined by the project PI

3) Sign usage agreement

- ✓ Wait for creation of HPC accounts

4) Check Connected Services:

- ✓ jupyter-jsc

The screenshot shows the JU JÜLICH SUPERCOMPUTING CENTRE user interface. At the top, there's a navigation bar with 'JU Your account', 'Mentoring', a search bar, and 'Detailed Statistics'. The main content area is divided into several sections:

- Account:** Fields for Salutation, E-mail address (with a checkmark), Telephone, and Address.
- Mentored projects:** A section for managing projects.
- Systems:** A table listing systems and their status:
 - judac:** Managed by SH-keys, usage agreement confirmed on 18.04.2021 (green checkmark).
 - jureca:** JURECA-DC_GPU: training 2211 (red X), with a message: 'You need to sign the usage agreement to access this system' (red X).
- Projects:** A section for managing projects, showing 'Interactive High-Performance Computing with Jupyter @ JSC' (green checkmark).
- Software:** A section for managing software.
- Connected Services:** A section for managing services, showing 'trac', 'llview', 'jards', 'gitlab', and 'jupyter-jsc' (green checkmark).

For more details, please visit
https://gitlab.jsc.fz-juelich.de/hedgedoc/S46PXGluSiuMv1Vgw_UU_Q?view#Getting-Access

MOTIVATION

MOTIVATION

your thinking, your reasoning, your insides, your ideas

“It is all about using and building a machinery **interface between** computational researchers and data, supercomputers, laptops, cloud **and** your thinking, your reasoning, your insides, your ideas about a problem.”

Fernando Perez, Berkely Institute for Data Science
Founder of Project Jupyter

JUPYTER NOTEBOOK

creating reproducible computational narratives

Markdown Cells

Code Cells

Fourier transform

Fourier transforms are one of the universal tools in computational physics, which appear over and over again in different contexts. SciPy provides functions for accessing the classic [FFTPACK](#) library from NetLib, which is an efficient and well tested FFT library written in FORTRAN. The SciPy API has a few additional convenience functions, but overall the API is closely related to the original FORTRAN library.

To use the `fftpack` module in a python program, include it using:

```
[41]: from numpy.fft import fftfreq
      from scipy.fftpack import *
```

To demonstrate how to do a fast Fourier transform with SciPy, let's look at the FFT of the solution to the damped oscillator:

$$\frac{d^2x}{dt^2} + 2\zeta\omega_0 \frac{dx}{dt} + \omega_0^2 x = 0$$

where x is the position of the oscillator, ω_0 is the frequency, and ζ is the damping ratio. To write this second-order ODE on standard form we introduce $p = \frac{dx}{dt}$:

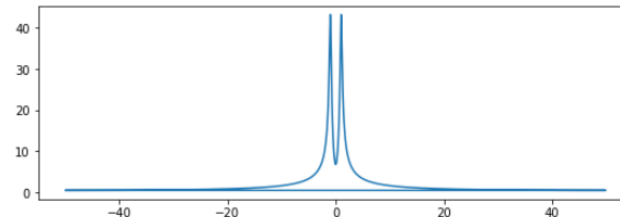
```
[42]: N = len(t)
      dt = t[1]-t[0]
      dt
```

```
[42]: 0.01001001001001001
```

```
[43]: # calculate the fast fourier transform
      # y2 is the solution to the under-damped oscillator from the previous section
      F = fft(y2[:,0])

      # calculate the frequencies for the components in F
      w = fftfreq(N, dt)
```

```
[44]: fig, ax = plt.subplots(figsize=(9,3))
      ax.plot(w, abs(F));
```



Output

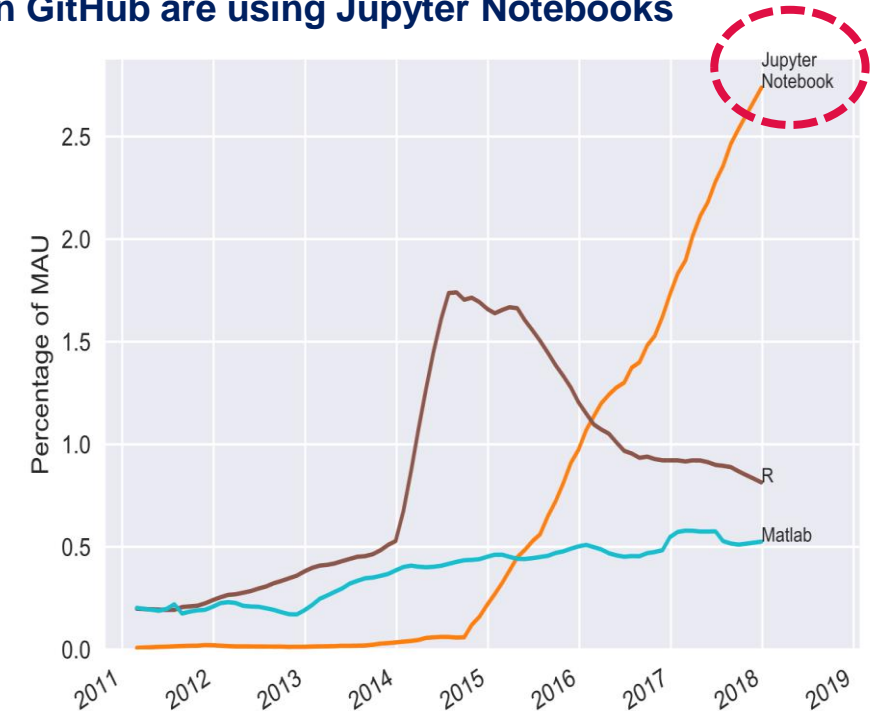
Output

MOTIVATION

Rise of Jupyter's popularity

- In 2007, Fernando Pérez and Brian Granger announced „**IPython**: a system for interactive scientific computing“ [1]
- In 2014, Fernando Pérez announced a spin-off project from IPython called **Project Jupyter**.
 - IPython continued to exist as a Python shell and a kernel for Jupyter, while the Jupyter notebook moved under the Jupyter name.
- In 2015, GitHub and the Jupyter Project announced native rendering of Jupyter notebooks file format (.ipynb files) on the **GitHub**
- In 2017, the **first JupyterCon** was organized by O'Reilly in New York City. Fernando Pérez opened the conference with an inspiring talk. [2]
- In 2018, **JupyterLab** was announced as the next-generation web-based interface for Project Jupyter.
- In 2019, JupyterLab 1.0 ...
In 2020, JupyterLab 2.0 ...
In 2021, JupyterLab 3.0 ...
In 2023, JupyterLab 4.0 released in May 2023.

Counting how many Monthly Active Users (MAU) on GitHub are using Jupyter Notebooks

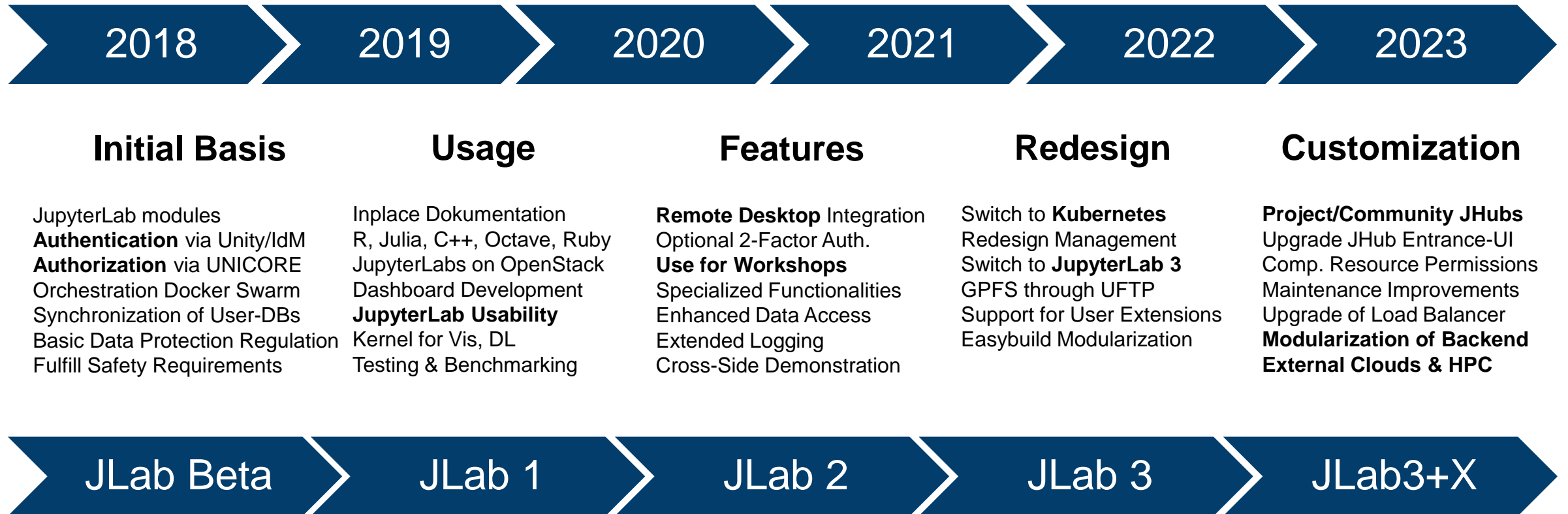


<https://www.benfrederickson.com/ranking-programming-languages-by-github-users/>
<https://github.com/benfred/github-analysis>

[1] Pérez F, Granger BE (2007) IPython: a system for interactive scientific computing. Comput Sci Eng 9(3):21–29

[2] Pérez F, Project Jupyter: From interactive Python to open science -> <https://www.youtube.com/watch?v=xuNj5paMuow>

HISTORY OF JUPYTERLAB AT JSC



HISTORY OF JUPYTERLAB AT JSC

2020

2023

Initi

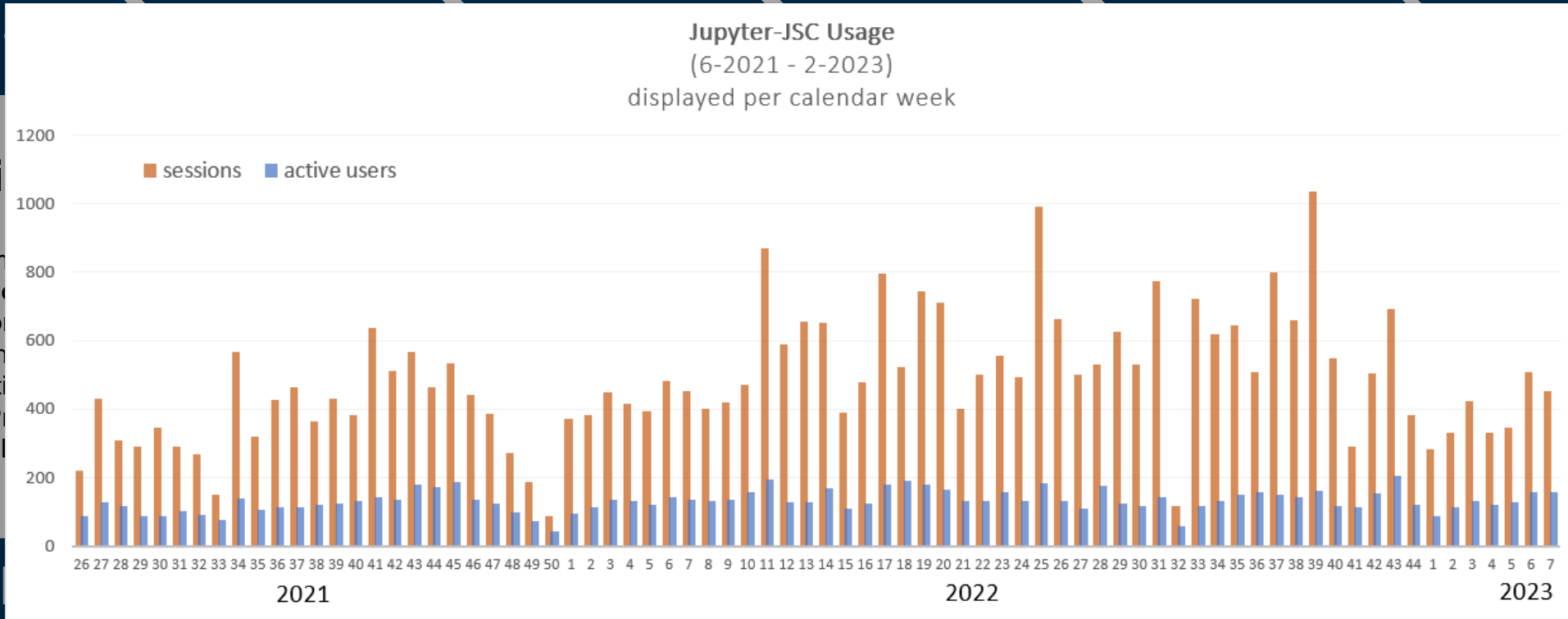
ization

JupyterLab m
Authenticati
Authorization
Orchestration
Synchronizati
Basic Data P
Fulfill Safety I

Community JHubs
Entrance-UI
Permissions
Improvements
Balancer
of Backend

JLa

B+X



TERMINOLOGY

TERMINOLOGY

What is JupyterLab

JupyterLab

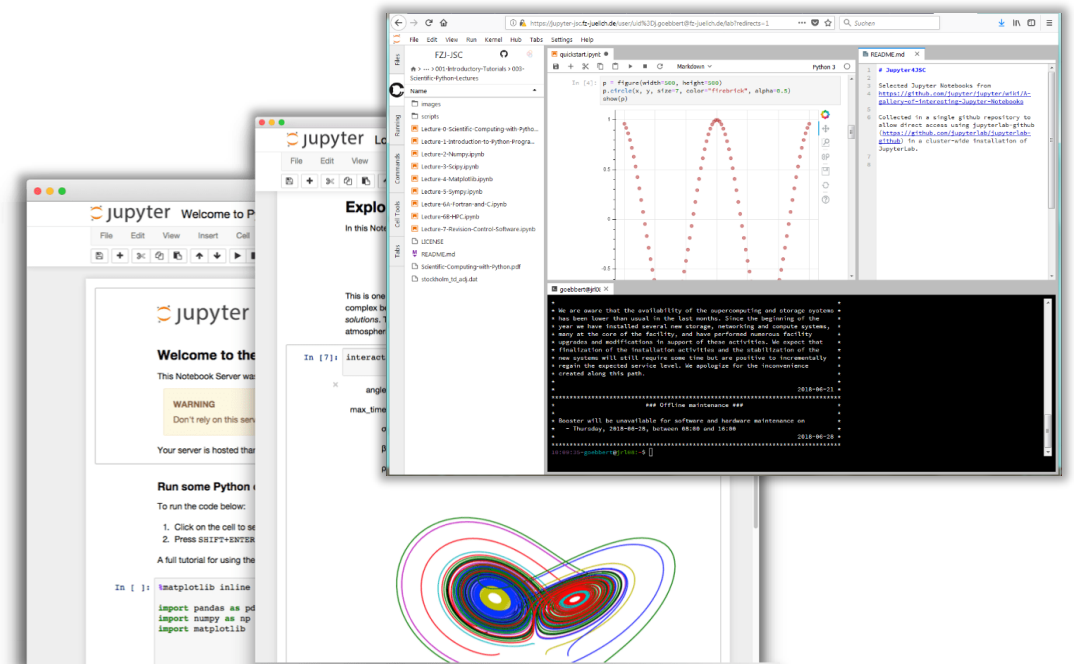
- **Interactive** working environment in the web browser
- For the creation of **reproducible** computer-aided narratives
- Very **popular** with researchers from all fields
- Jupyter = Julia + Python + R

Multi-purpose working environment

- Language agnostic
- Supports execution environments (“kernels”)
 - For dozens of languages: Python, R, Julia, C++, ...
- Extensible software design („extensions“)
 - many server/client plug-ins available
 - Eg. in-browser-terminal and file-browsing

Document-Centered Computing (“notebooks”)

- Combines code execution, rich text, math, plots and rich media.
- All-in-one document called Jupyter Notebook



<https://jupyterlab.readthedocs.io>

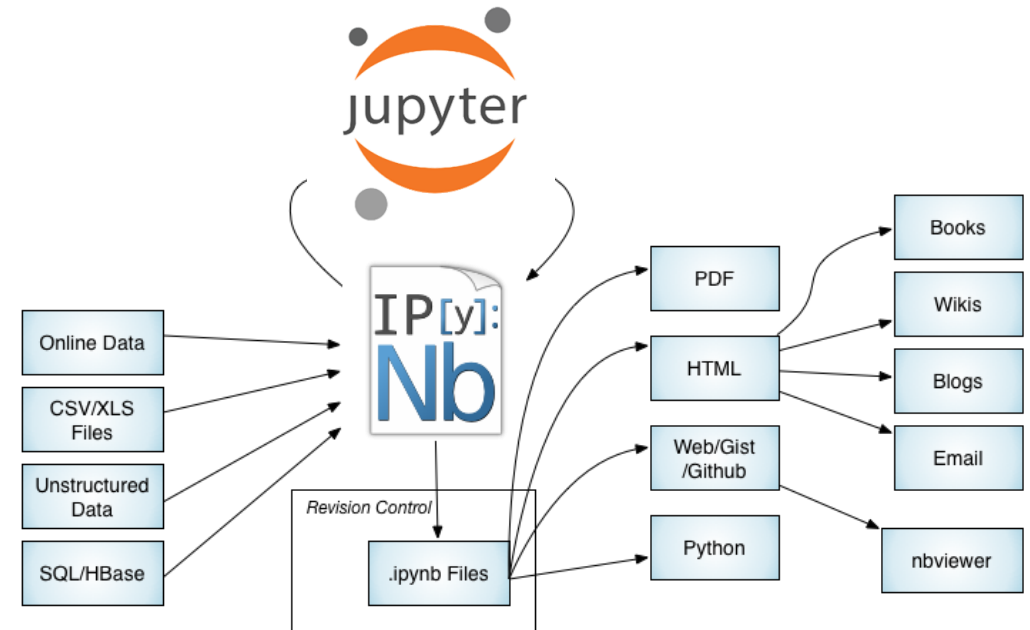
TERMINOLOGY

What is a Jupyter Notebook?

Jupyter Notebook

A notebook document (file extension .ipynb) is a document that can be rendered in a web browser

- It is a file, which stores your work in JSON format
- Based on a set of open standards for interactive computing
- Allows development of custom applications with embedded interactive computing.
- Can be extended by third parties
- Directly convertible to PDF, HTML, LaTeX ...
- Supported by many applications such as GitHub, GitLab, etc..



<https://jupyter-notebook.readthedocs.io/>

<https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks>

TERMINOLOGY

What is a Jupyter Kernel?

Jupyter Kernel

A “kernel” refers to the separate process which executes code cells within a Jupyter notebook.

Jupyter Kernel

- **run code** in different programming languages and environments.
- can be **connected to** a notebook (one at a time).
- **communicates** via ZeroMQ with the JupyterLab.
- Multiple **preinstalled** Jupyter Kernels can be found on our clusters
 - Python, R, Julia, Bash, C++, Ruby, JavaScript
 - Specialized kernels for visualization, quantum-computing
- You can easily **create your own kernel** which for example runs your specialized virtual Python environment.



<https://jupyter-notebook.readthedocs.io/>
<https://github.com/jupyter/jupyter/wiki/Jupyter-kernels>
<https://zeromq.org>

TERMINOLOGY

What is a JupyterLab Extension?

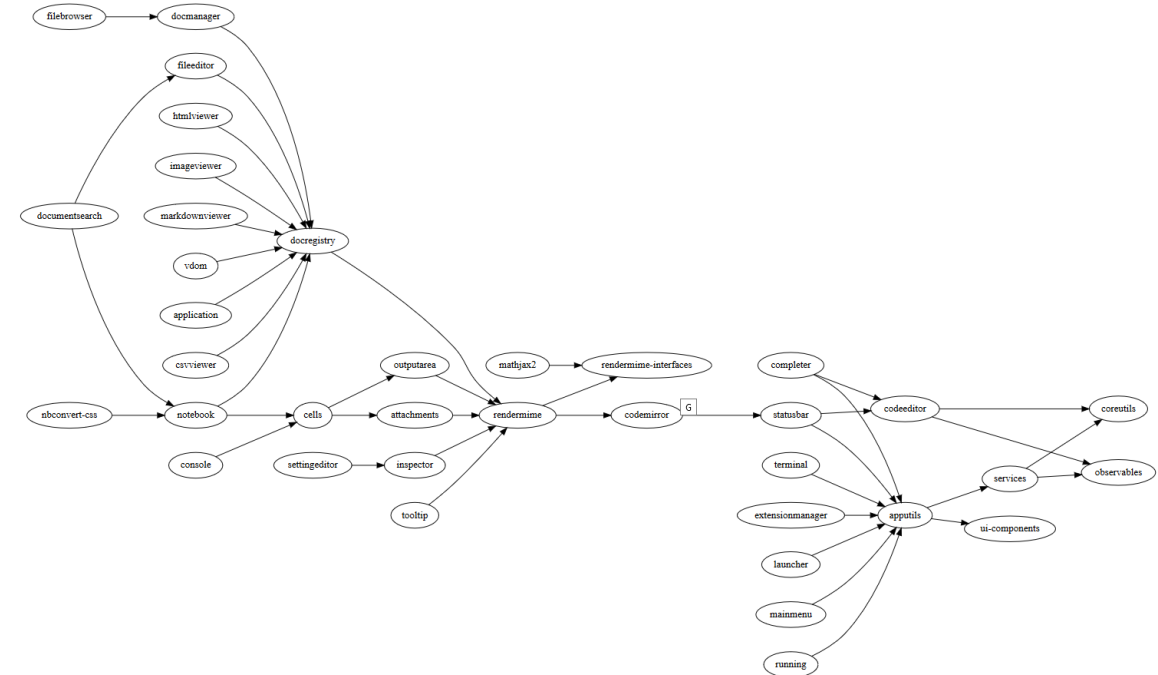
JupyterLab Extension

JupyterLab extensions can customize or enhance any part of JupyterLab.

JupyterLab Extensions

- provide new file viewers, editors, themes
 - provide renderers for rich outputs in notebooks
 - add items to the menu or command palette
 - add keyboard shortcuts
 - add settings in the settings system.
-
- Extensions can even provide an API for other extensions to use and can depend on other extensions.

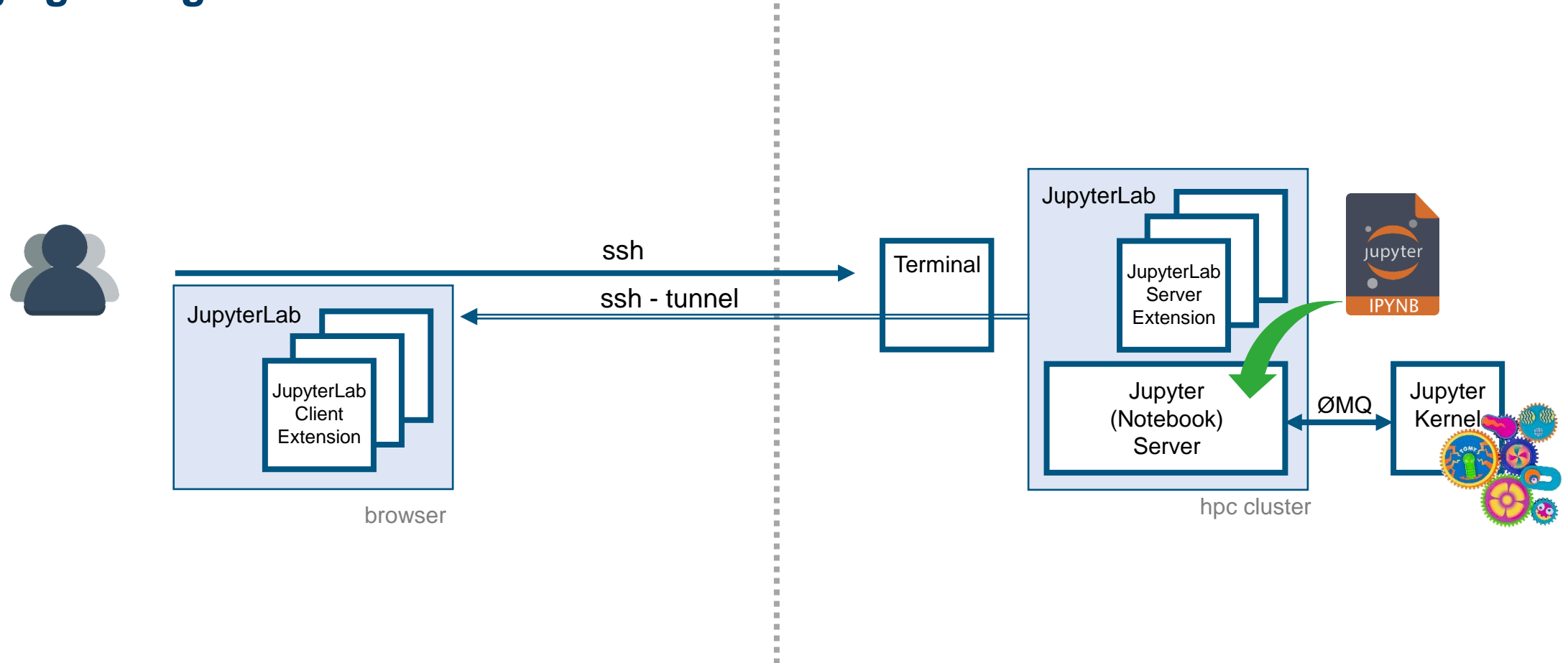
The whole JupyterLab itself is simply a **collection of extensions** that are no more powerful or privileged than any custom extension.



<https://jupyterlab.readthedocs.io/en/stable/user/extensions.html>
<https://github.com/topics/jupyterlab-extension>

TERMINOLOGY

Bringing all together



PRE-ACCESS TODOS

1) Register & Login

- ✓ <https://judoor.fz-juelich.de>

2) Join the project „paj2206“

- ✓ Wait to get joined by the project PI

3) Sign usage agreement

- ✓ Wait for creation of HPC accounts

4) Check Connected Services:

- ✓ jupyter-jsc

The screenshot shows the JU JÜLICH SUPERCOMPUTING CENTRE user interface. At the top, there is a navigation bar with 'JU Your account', 'Mentoring', a search bar, and 'Detailed Statistics'. The main content area is divided into several sections:

- Account:** Fields for Salutation, E-mail address (with a checkmark), Telephone, and Address.
- Mentored projects:** A section for managing projects.
- Systems:** A table listing systems and their status:
 - judac:** Managed by training2109, with a green checkmark and 'Usage agreement confirmed on 18.04.2021'.
 - jureca:** JURECA-DC_GPU: training2211, with a red X and 'You need to sign the usage agreement to access this system'.
- Projects:** A section for managing projects, showing 'Interactive High-Performance Computing with Jupyter @ JSC' with a green checkmark and 'training2211'.
- Software:** A section for managing software.
- Connected Services:** A section for managing services, showing 'trac', 'lview', 'jards', 'gitlab', and 'jupyter-jsc' with a green checkmark.

For more details, please visit
https://gitlab.jsc.fz-juelich.de/hedgedoc/S46PXGluSiuMv1Vgw_UU_Q?view#Getting-Access

PRE-ACCESS TODOS

1) I

2) J

3) S

4) C

<https://judoor.fz-juelich.de>

RECORDED WITH
SCREENCAST  MATIC

PRE-ACCESS TODOS

1) Register & Login

- ✓ <https://judoor.fz-juelich.de>

2) Join the project „paj2206“

- ✓ Wait to get joined by the project PI

3) Sign usage agreement

- ✓ Wait for creation of HPC accounts

4) Check Connected Services:

- ✓ jupyter-jsc

The screenshot shows the Jülich Supercomputing Centre (JSC) user portal. At the top, there is a navigation bar with 'JU Your account', 'Mentoring', a search bar, and 'Detailed Statistics'. The Jülich logo and 'JÜLICH SUPERCOMPUTING CENTRE' are on the right. The main content area is divided into sections: 'Account' (Salutation, E-mail address, Telephone, Address), 'Mentored projects', 'Systems', 'Projects', 'Software', and 'Connected Services'. The 'Systems' section lists 'judac' (with a green checkmark and 'Usage agreement confirmed on 18.04.2021') and 'jureca' (with a red X and 'You need to sign the usage agreement to access this system'). The 'Projects' section shows 'Interactive High-Performance Computing with Jupyter @ JSC' (with a green checkmark). The 'Connected Services' section lists 'trac', 'llview', 'jards', 'gitlab', and 'jupyter-jsc' (with a green checkmark).

For more details, please visit
https://gitlab.jsc.fz-juelich.de/hedgedoc/S46PXGluSiuMv1Vgw_UU_Q?view#Getting-Access

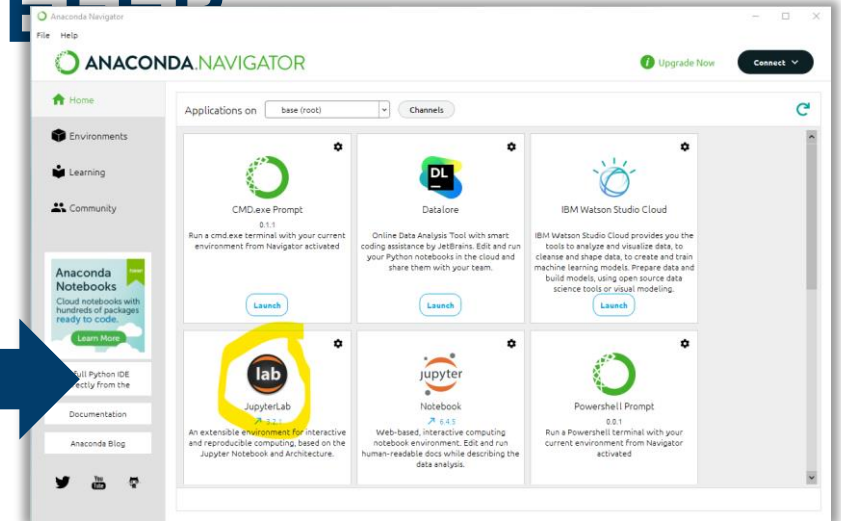
INSTALLATION

JUPYTERLAB - WHEREVER YOU PREFER

Local, Remote, Browser-only

Local installation:

- JupyterLab installed using conda, mamba, pip, pipenv or docker.
→ https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html



JUPYTERLAB - WHEREVER YOU PREFER

Local, Remote, Browser-only

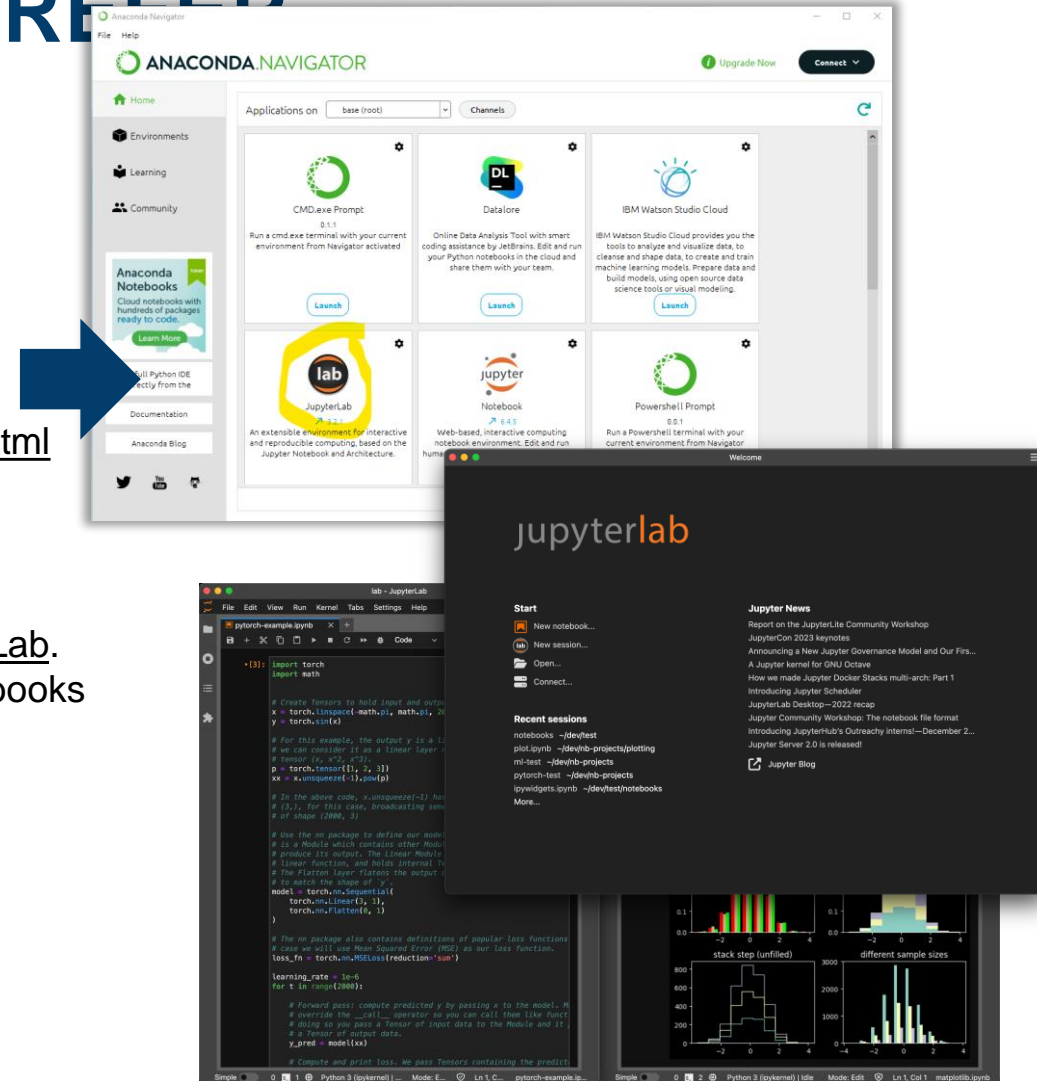
Local installation:

- **JupyterLab** installed using conda, mamba, pip, pipenv or docker.
→ https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html
- **JupyterLab** installed as normal desktop application = **JupyterLab Desktop**
→ <https://github.com/jupyterlab/jupyterlab-desktop/releases>

JupyterLab Desktop is the cross-platform desktop application for JupyterLab.

It is probably the quickest and easiest way to get started with Jupyter notebooks on your personal computer, with the flexibility for advanced use cases.

(Windows, macOS, Debian/Ubuntu, RedHat/Fedora)



JUPYTERLAB - WHEREVER YOU PREFER

Local, Remote, Browser-only

Local installation:

- **JupyterLab** installed using conda, mamba, pip, pipenv or docker.
→ https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html
- **JupyterLab** installed as normal desktop application = **JupyterLab Desktop**
→ <https://github.com/jupyterlab/jupyterlab-desktop/releases>

Remote (cluster) installation:

- **JupyterLab** installed on a remote server and accessed through the browser
 - in \$HOME (e.g. using pip or miniconda)
 - system-wide (e.g. with Easybuild, Spark) by the admins.



Tunnel the new JupyterLab to your local machine

Linux or Mac:
If your operating system is Linux or Mac user:

```
ssh -N -L <LOCAL_PORT>:<JLAB_NODE>:<JLAB_PORT> <USERID>@<LOGIN_NODE>.fz-juelich.de  
# example: ssh -N -L 8888:jwels04:8888 goebbert1@jwels01.fz-juelich.de  
  
# if you want to tunnel to jwels04 only, then you should set JLAB_NODE to "localhost"
```

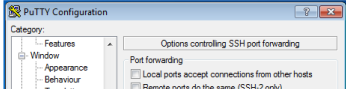
Attention:

- LOGIN_NODE - Hostname of login node from the view of your local machine
- JLAB_NODE - Hostname of the node running JupyterLab from the view of LOGIN_NODE
- LOCAL_PORT - port on your local machine
- JLAB_PORT - port on the node running JupyterLab

Windows: In case your operating system is Windows, the setup of the tunnel depends on your ssh client. Here a short overview on how-to setup a tunnel with **PUTTY** is given.

It is assumed that PuTTY is already configured in a way that a general ssh connection to JUWELS is possible. That means that host name, user name and the private ssh key (using PuTTY's Pageant) are correctly set. You already made a first connection to JUWELS using PuTTY.

To establish the ssh tunnel start PuTTY and enter the "SSH->tunnels" tab in the PuTTY configuration window before connecting to JUWELS. You have to enter the source port (eg. <LOCAL_PORT> = 8888) and the destination (eg. jwels01.fz-juelich.de:8888) and then press add. After pressing add, the tunnel should appear in the list of forwarded ports and you can establish the tunnel by pressing the open button.



JUPYTERLAB - WHEREVER YOU PREFER

Local, Remote, Browser-only

Local installation:

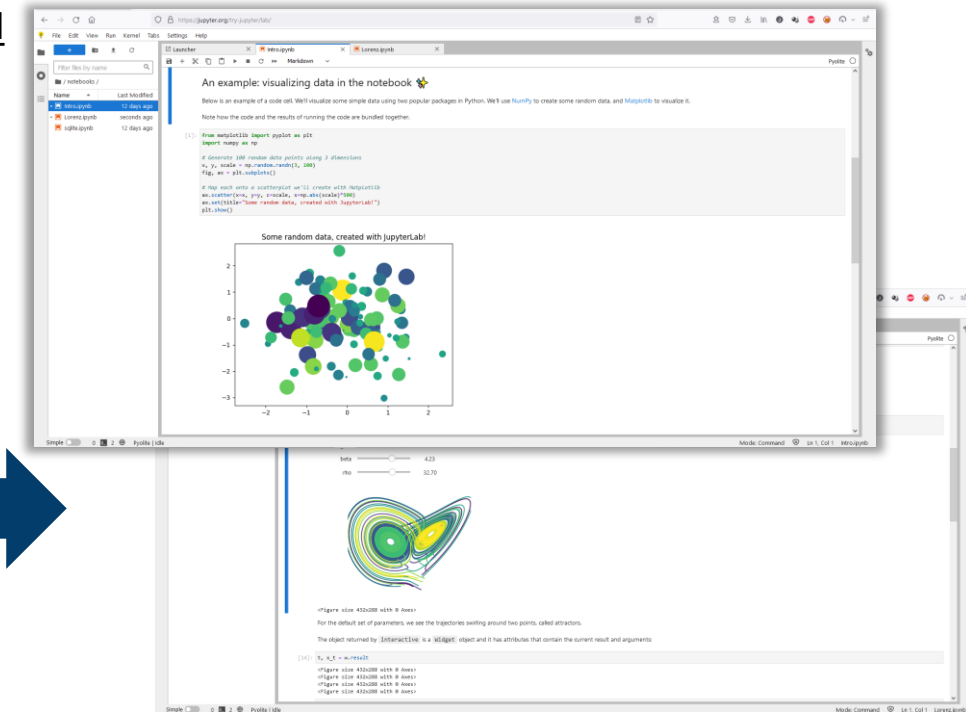
- **JupyterLab** installed using conda, mamba, pip, pipenv or docker.
→ https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html
- **JupyterLab** installed as normal desktop application = **JupyterLab Desktop**
→ <https://github.com/jupyterlab/jupyterlab-desktop/releases>

Remote (cluster) installation:

- **JupyterLab** installed on a remote server and accessed through the browser
 - in \$HOME (e.g. using pip or miniconda)
 - system-wide (e.g. with Easybuild, Spark) by the admins.

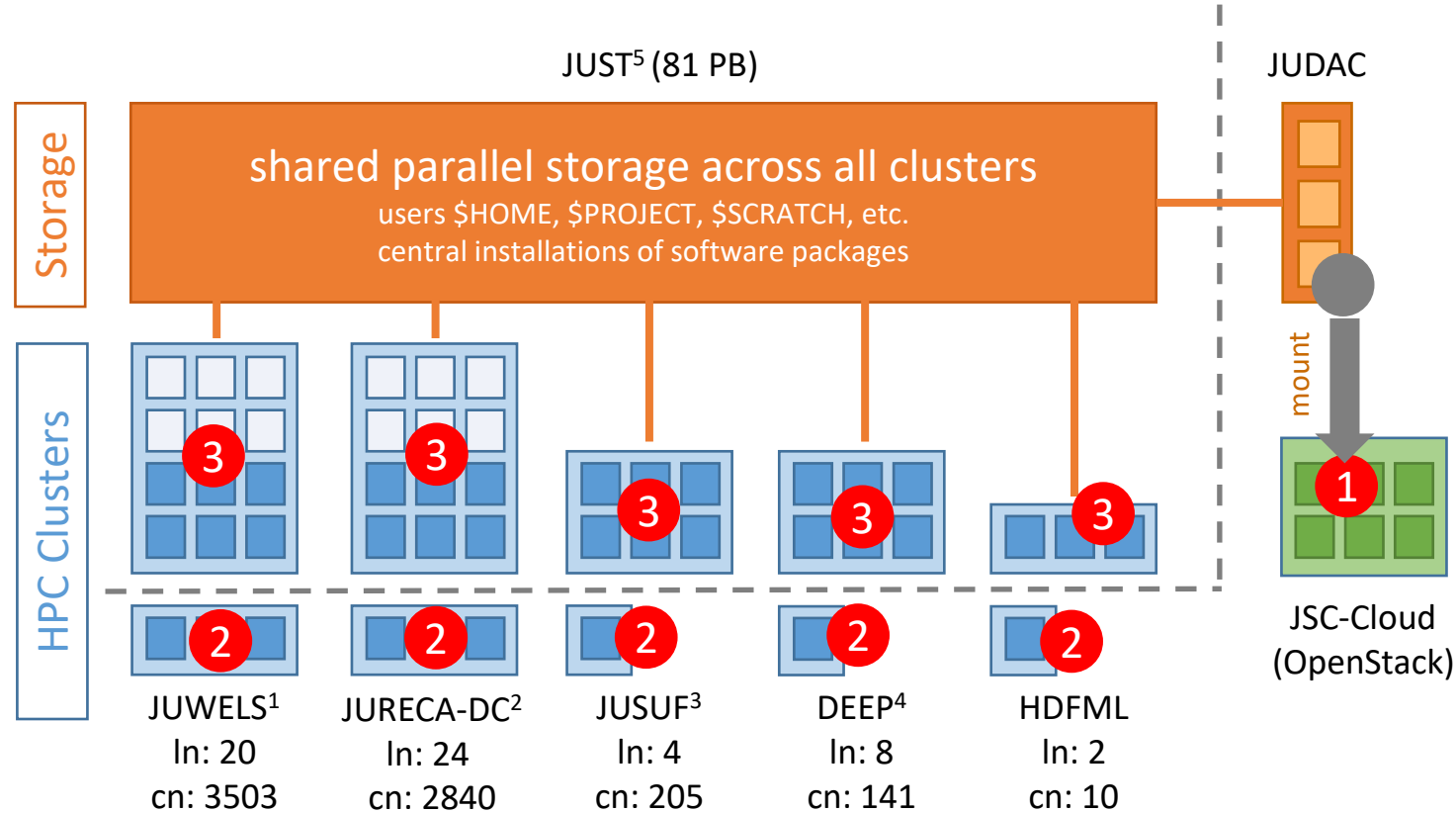
Browser-only installation (limited feature set):

- **JupyterLab** local with server + client in your browser = **JupyterLite**
Includes a browser-ready Python environment named Pyodide.
→ <https://jupyter.org/try-jupyter/lab>



START & LOGIN

JUPYTERLAB EVERYWHERE



no. login nodes = ln
no. compute nodes = cn

[1] <https://apps.fz-juelich.de/jsc/hps/juwels/configuration.html>

[2] <https://apps.fz-juelich.de/jsc/hps/jureca/configuration.html>

[3] <https://apps.fz-juelich.de/jsc/hps/jusuf/cluster/configuration.html>

[4] https://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/DEEP-EST/_node.html

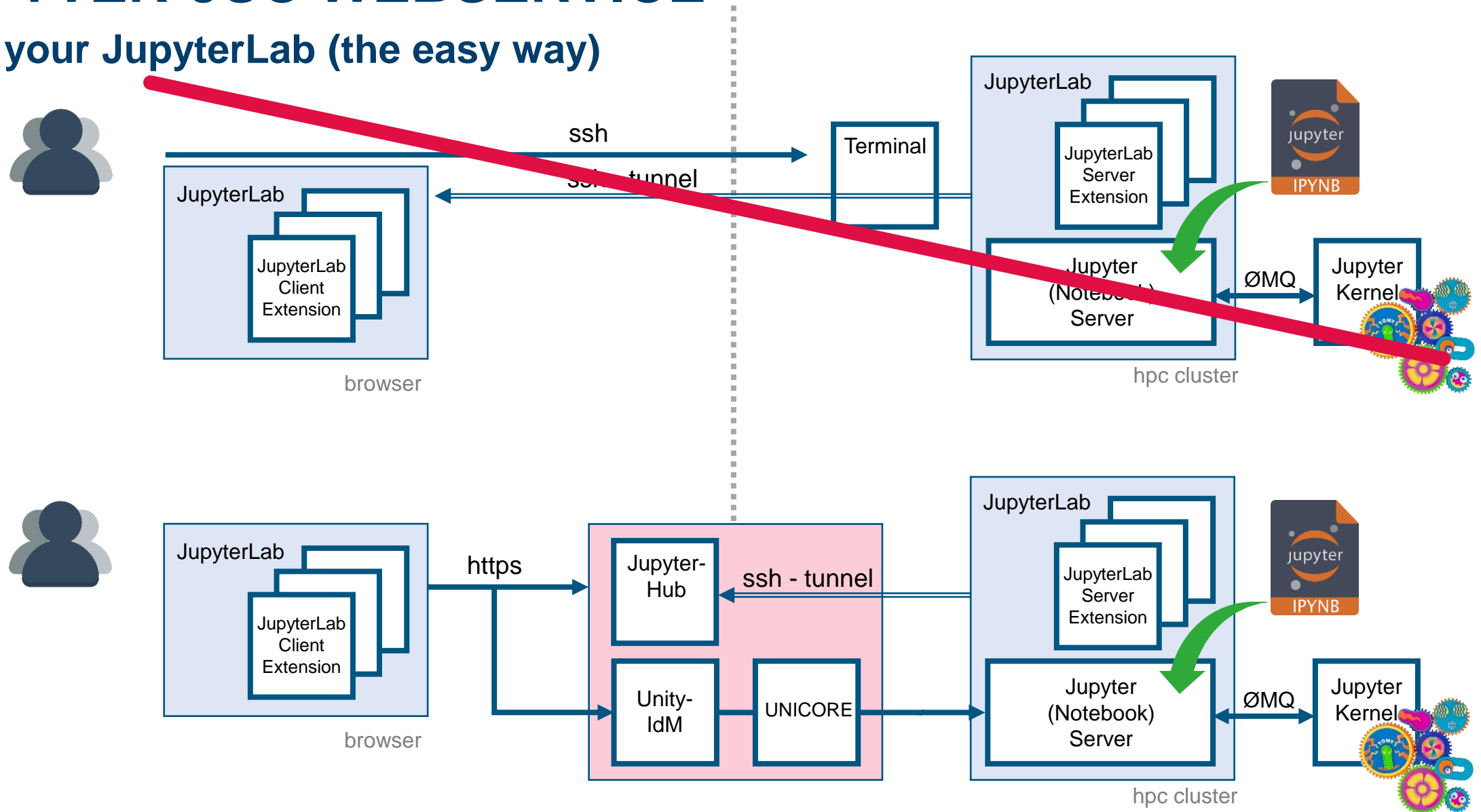
[5] https://www.fz-juelich.de/ias/jsc/EN/Expertise/Datamanagement/OnlineStorage/JUST/Configuration/Configuration_node.html

JupyterLab everywhere

- 1 JupyterLab on cloud
- 2 JupyterLab on login nodes
- 3 JupyterLab on compute nodes

JUPYTER-JSC WEBSERVICE

Start your JupyterLab (the easy way)



JUPYTER-JSC WEBSERVICE

Start your JupyterLab

JUPYTER-JSC WEBSERVICE

Your server is starting up...

You will be redirected automatically when it's ready for you.

Name	System	Partition	Project	Status	Actions
juwels_cluster	JUWELS	devel	ccsys	70%	Cancel

JupyterLab + New

You can configure your existing JupyterLab's by expanding the corresponding table row.

Name	System	Partition	Project	Status	Actions
hdfcloud_3.3	HDF-Cloud	N/A	N/A		Start
juwelsbooster_login	JUWELS	LoginNodeBooster	ccstdf		Start
juwels_cluster	JUWELS	devel	ccsys	30%	Open Cancel

Supercomputing in Your Browser

We are pleased to bring "Supercomputing in your browser". Jupyter-JSC is designed to provide you with the rich high performance computing (HPC) ecosystem to the world's most popular software: web browsers. JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible to support a wide range of workflows in data science, scientific computing, and machine learning. Read more.

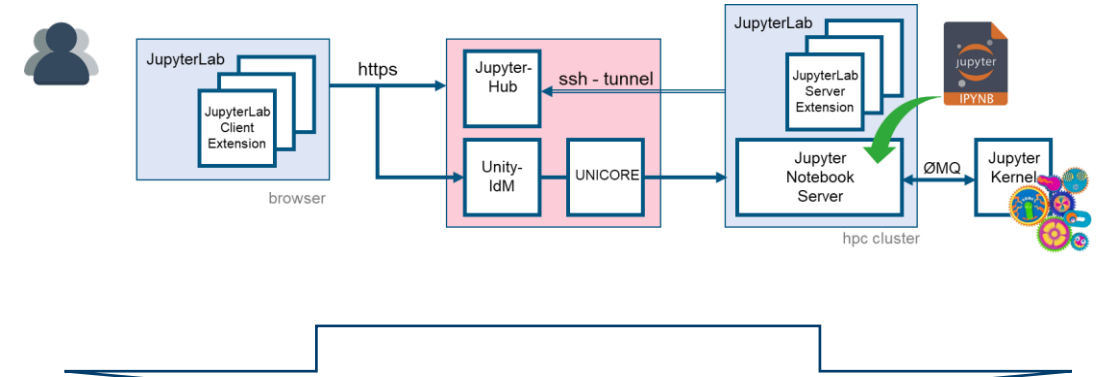
Please use your JSC account to log in or register if you have not already done so. It's also possible to log in via Helmholtz AAI.

[Login](#) [Register](#)

JUPYTER-JSC JUWELS JURECA JUSUF DEEP HDFML HDF-Cloud

© Forschungszentrum Jülich | [Imprint](#) | [Privacy Policy](#) | [Support](#) | [Terms of Service](#)

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES



GPU DASHBOARDS

- GPU Utilization
- GPU Memory
- GPU Resources
- PCIe Throughput
- WLink Throughput
- WLink Timeline
- Matching Resources

```
1 import math
2 import numpy as np
3 from numba import cuda
4 import numba.cuda.jit as jit
5 import sys
6
7 len(cuda.gpus)
8
9 cuda.gpus[0].name
10
11 b"Tesla V100-SXM2-10GB"
12
13 @cuda.jit
14 def mandelbrot_numba(numbas, iterations):
15     # matrix indices
16     i, j = cuda.grid(2)
17     size = numbas[0]
18     # skip through outside the matrix.
19     if i > size or j > size:
20         return
21     # Run the calculation.
22     z = 0 + 0j
23     for n in range(iterations):
24         z = z**2 + 1j * (size - i)
```

GPU Memory: 332.40 MB

GPU Utilization

GPU Resources

PCIe Throughput

WLink Throughput

WLink Timeline

Matching Resources

Variables

- numba: module
- np: module
- cuda: module
- jit: module
- mandelbrot_numba: numba.cuda.compiler.Dispatcher
- size: 400
- iterations: 100
- my_numpy_array: array

Breakpoints

- hmp/jupyterlab_30146/402220956.py 4

Source

JUPYTER-JSC WEBSERVICE

Control Panel

A. Jupyter-JSC – Add new JupyterLab

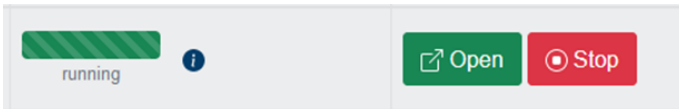


B. Configuration Dialog

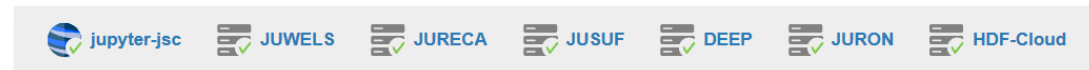
- set Name, Type, System, Account, Project, Partition

C. Jupyter-JSC – Actions

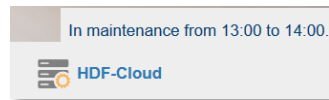
- Open/Stop a running JupyterLab
- Change/Delete **configuration**



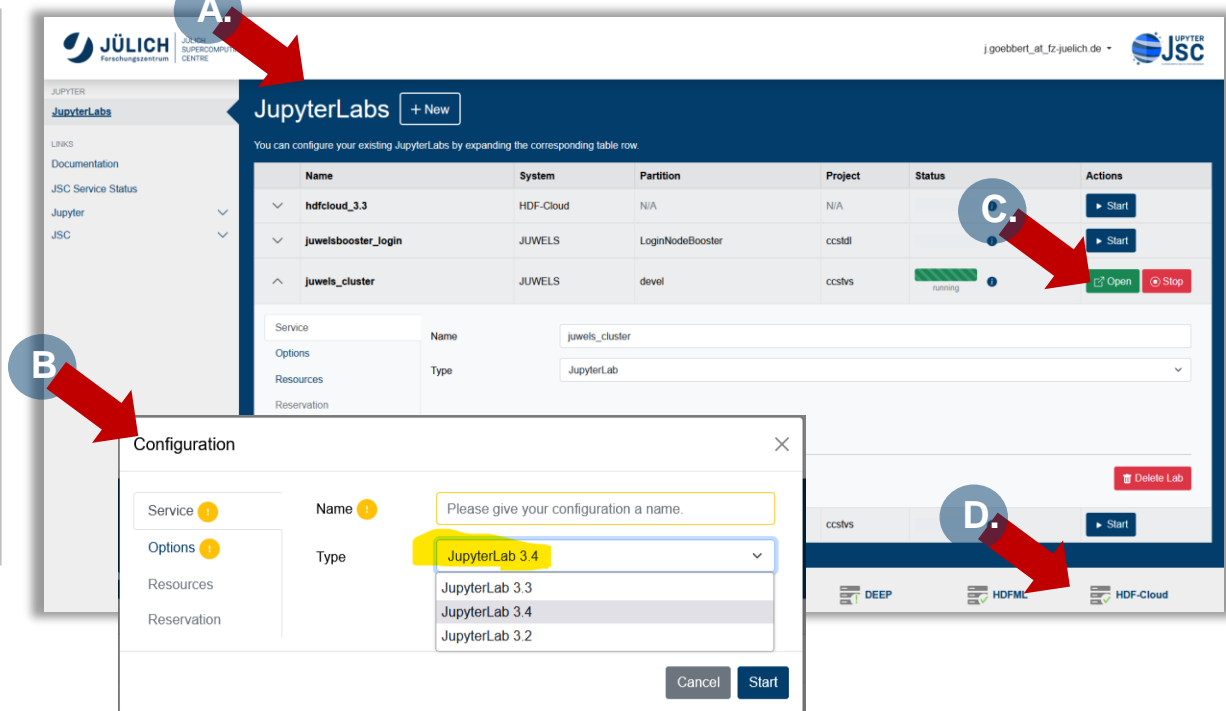
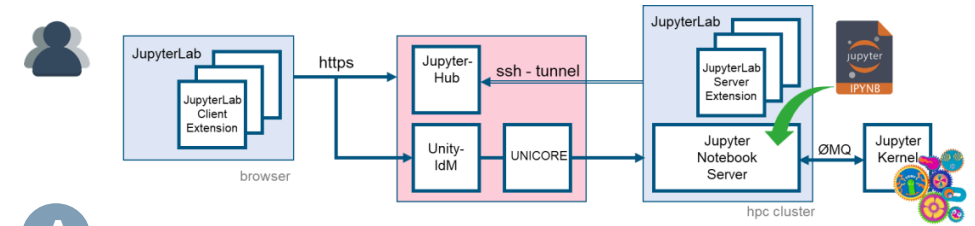
D. Jupyter-JSC -- Statusbar



- Upcoming maintenance (mouse hover for details)

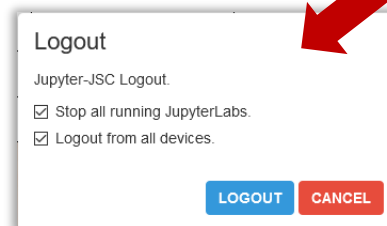


- System offline



E. Jupyter-JSC – Logout

Logout will ask what you want to do with the running JupyterLabs – be careful what you answer!



JUPYTER-JSC WEBSERVICE

JupyterLab Configuration

Jupyter-JSC – Configuration

Available options **depend on**

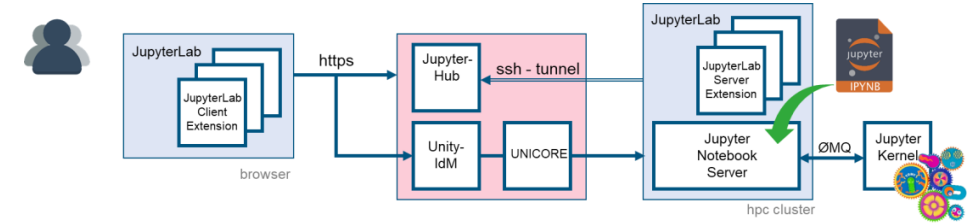
- user account settings visible in judoor.fz-juelich.de
- system specific usage agreement on JuDoor is signed (!!!)
- currently available systems in all of your projects

Basic options

- Version:
multiple versions of JupyterLab are installed
- System:
JUWELS, JURECA, JUSUF, DEEP, HDFML, HDF-Cloud
- Account:
In general users only have a single account
- Project:
project which have access to the selected system
- Partition:
partition which are accessible by the project
(this includes the decision for LoginNode and ComputeNode)

Extra options

- Partition == compute Resources
- Kernel and Extensions non-default JupyterKernel, Extensions, Proxies



Name	System	Partition	Project	Status	Actions
NEW JUPYTERLAB					

Lab Config

Resources

Kernels and Extensions

Name

Version

System

Account

Project

Partition

Give your lab a name

JupyterLab - 3.6

JUWELS

goebbert1

ccstdl

LoginNode

Login Nodes

LoginNode

LoginNodeBooster

LoginNodeVis

Compute Nodes

batch

devel

develgpu

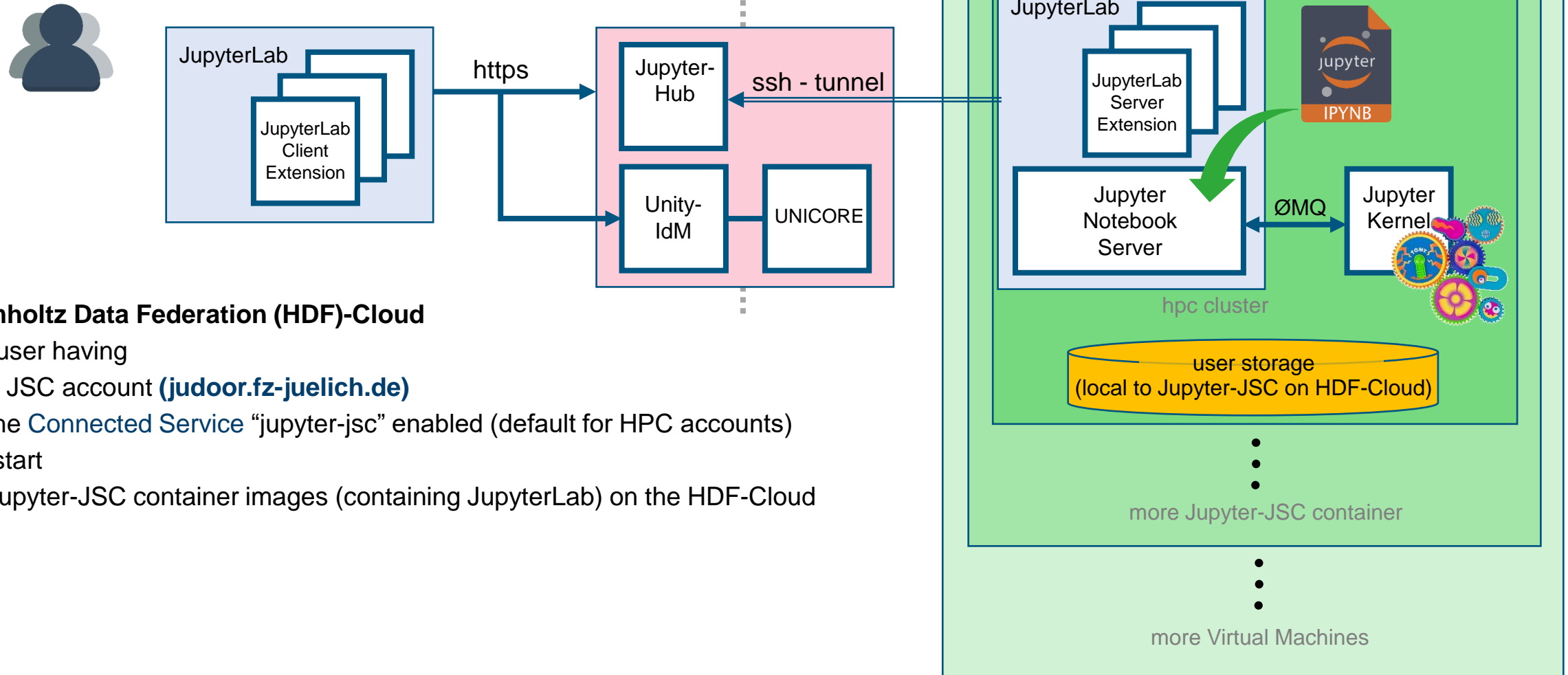
gpu

mem192

Start

JUPYTER-JSC WEBSERVICE

System: HDF-Cloud

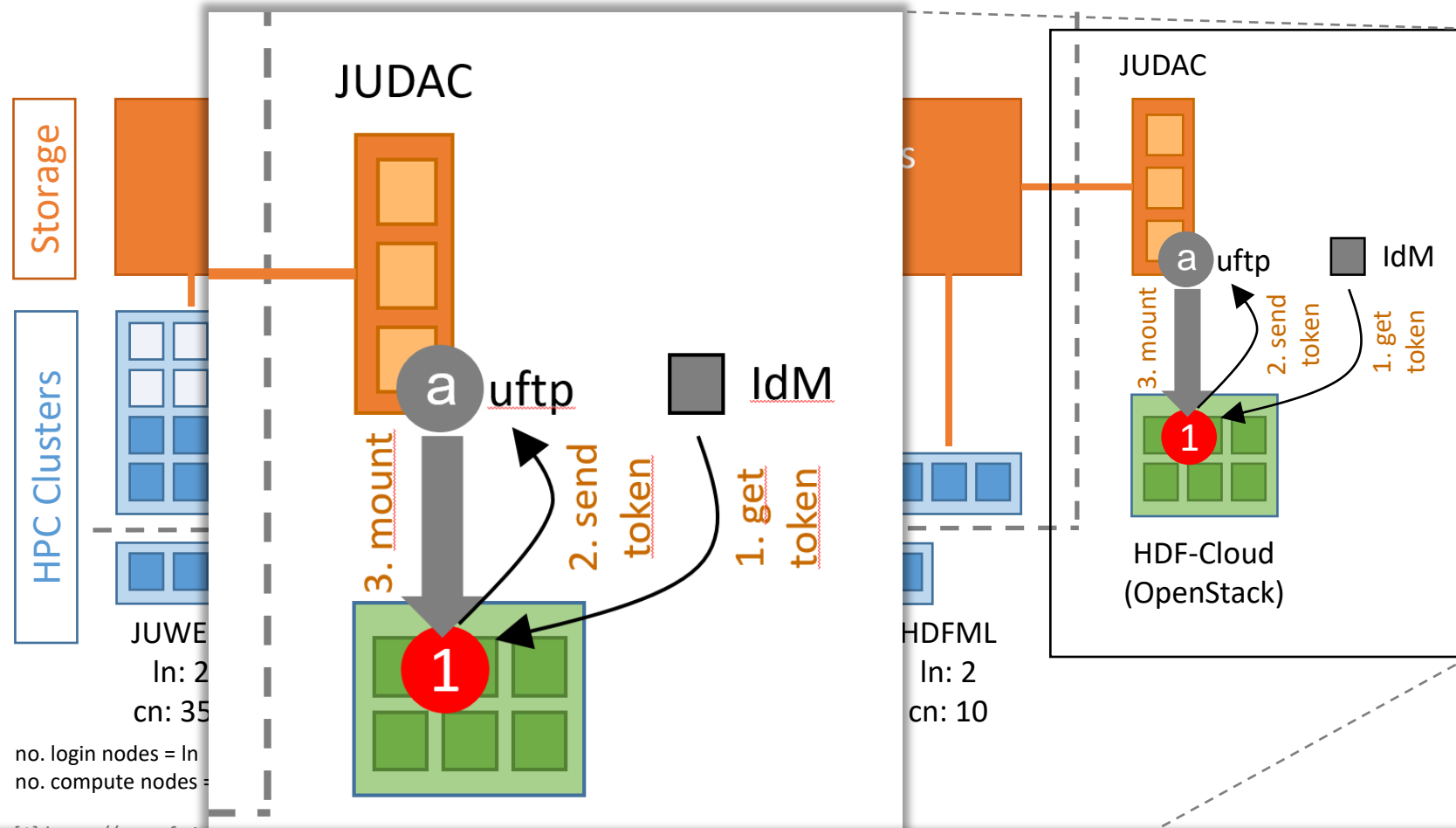


Helmholtz Data Federation (HDF)-Cloud

Any user having

- a JSC account (judoor.fz-juelich.de)
 - the [Connected Service](#) "jupyter-jsc" enabled (default for HPC accounts)
- can start
- Jupyter-JSC container images (containing JupyterLab) on the HDF-Cloud

HOW TO MOUNT GPFS ON HDF-CLOUD



https://gitlab.jsc.fz-juelich.de/jupyter4jsc/training-2023.04-jupyter4hpc/-/blob/main/day2_hpcenv/7_cloud-hpc_challenges/1-hdf-cloud_mount-hpc-storage.ipynb

JUPYTER-JSC WEBSERVICE

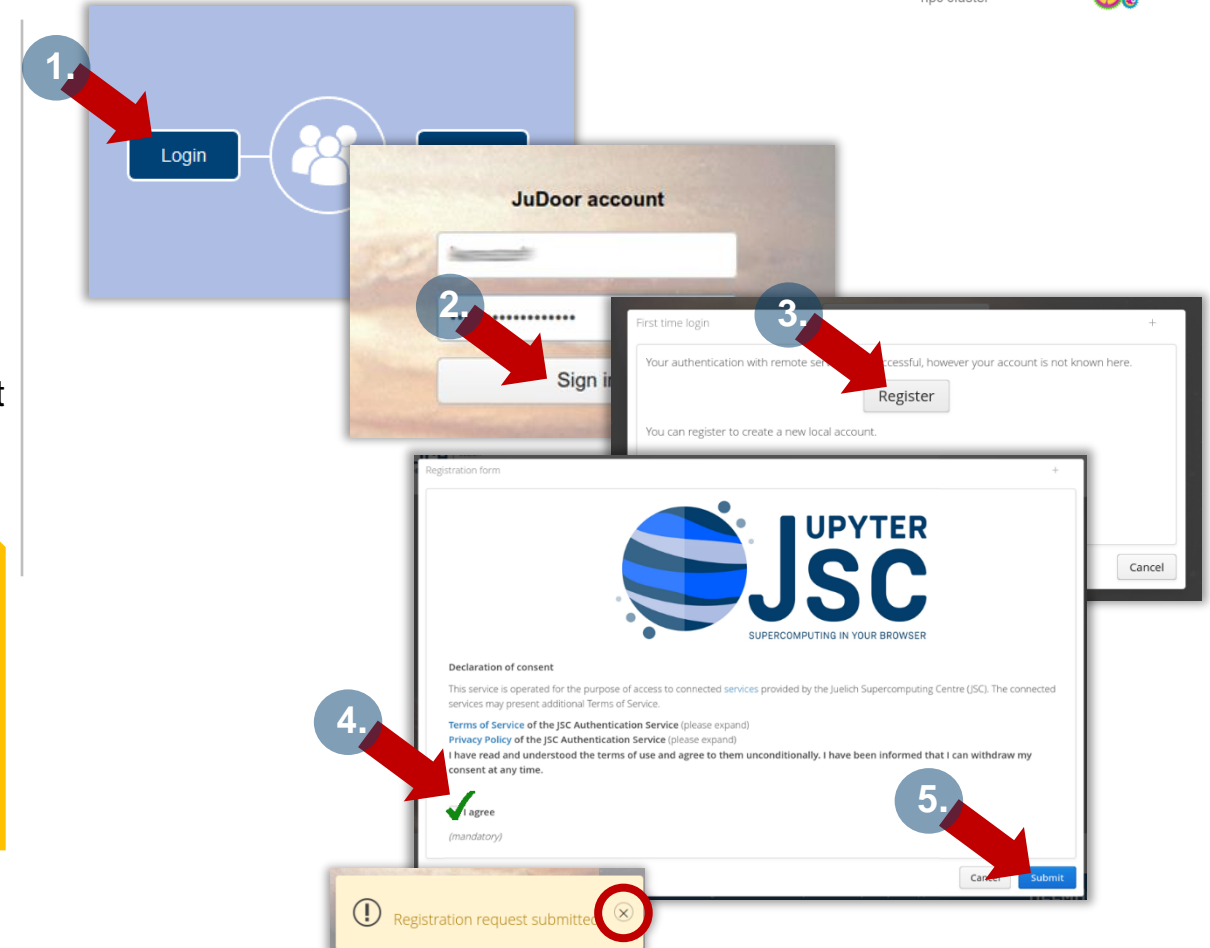
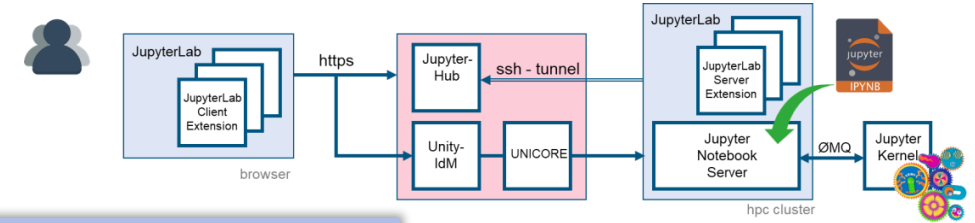
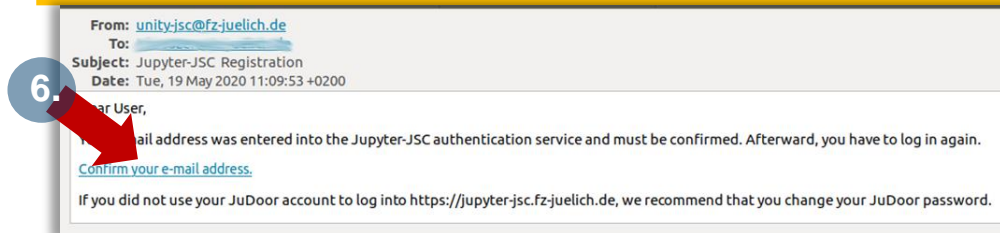
First time login

=> <https://jupyter-jsc.fz-juelich.de>

Jupyter-JSC first time login

- Requirements:
 - Registered at judoor.fz-juelich.de
 - (check "Connected Services" = jupyter-jsc)
 - Project membership + signed systems usage agreement
 - Waited ~10 minutes

1. Login at <https://jupyter-jsc.fz-juelich.de>
2. Sign in with your JSC account
3. Register to Jupyter-JSC
4. Accept usage agreement
5. Submit the registration
6. Wait for email and confirm your email address



JUPYTER-JSC WEBSERVICE

First time

=> <https://jupyter-jsc.fz-juelich.de>

Jupyter

- Re

First check on
<https://judoor.fz-juelich.de>
if you are ready for Jupyter-JSC.

1. L
2. S
3. R
4. A
5. S
6. W

6

From: unit
To:
Subject: Jupy
Date: Tue,
Dear User,

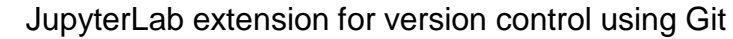
RECORDED WITH
SCREENCASTOMATIC

Your e-mail address was entered into the Jupyter-JSC authentication service and must be confirmed. Afterward, you have to log in again.
[Confirm your e-mail address.](#)
If you did not use your JuDoor account to log into <https://jupyter-jsc.fz-juelich.de>, we recommend that you change your JuDoor password.

JUPYTERLAB EXTENSIONS

Installed by default at Jupyter-JSC

3d plotting for Python in the Jupyter notebook based on IPython widgets using WebGL



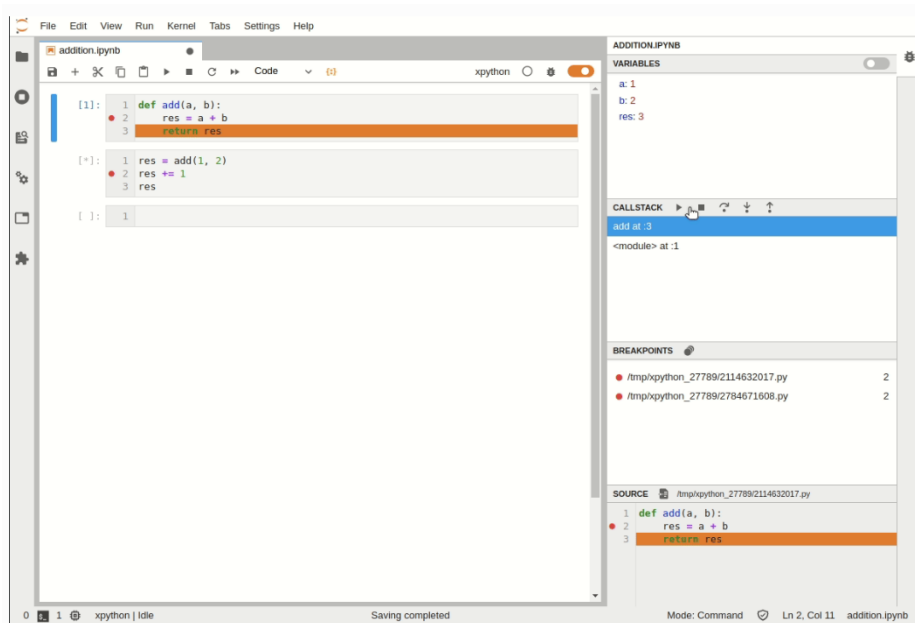
JUPYTERLAB EXTENSIONS

Installed by default at Jupyter-JSC

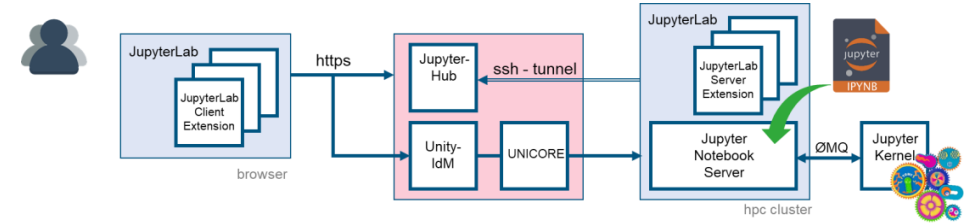
JupyterLab - Visual Debugger

JupyterLab >= 3 ships with a Debugger front-end by default.

This means that notebooks, code consoles and files can now be debugged from JupyterLab directly! For the debugger to be enabled and visible, a kernel with support for debugging is required.

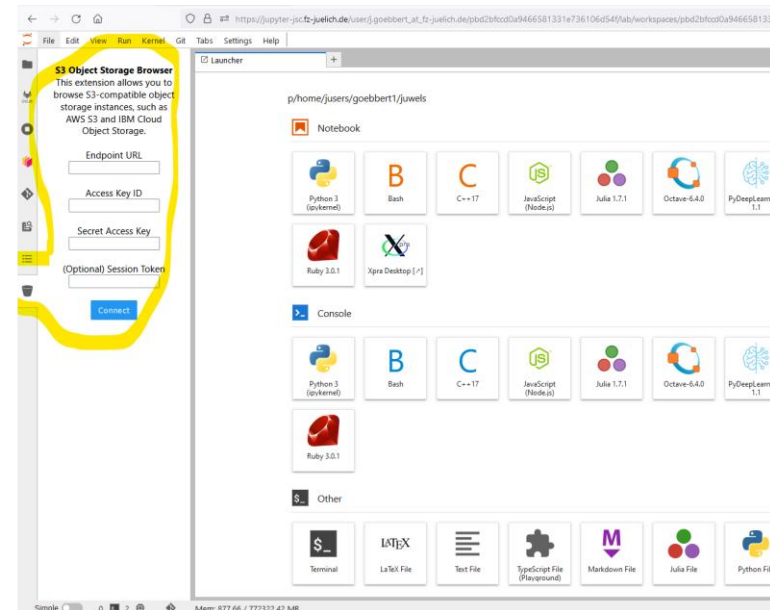


<https://jupyterlab.readthedocs.io/en/stable/user/debugger.html>



JupyterLab-S3-browser

A JupyterLab extension for browsing S3-compatible object storage



<https://github.com/IBM/jupyterlab-s3-browser>

JUPYTERLAB EXTENSIONS

Installed by default at Jupyter-JSC

PyThreeJS

A Python / ThreeJS bridge utilizing the Jupyter widget infrastructure.
<https://threejs.org> - lightweight, 3D library with a default WebGL renderer.

```
In [9]: f = """
function f(origu,origv) {
  // scale u and v to the ranges I want: [0, 2*pi]
  var u = 2*Math.PI*origu;
  var v = 2*Math.PI*origv;

  var x = Math.sin(u);
  var y = Math.cos(v);
  var z = Math.cos(u*v);

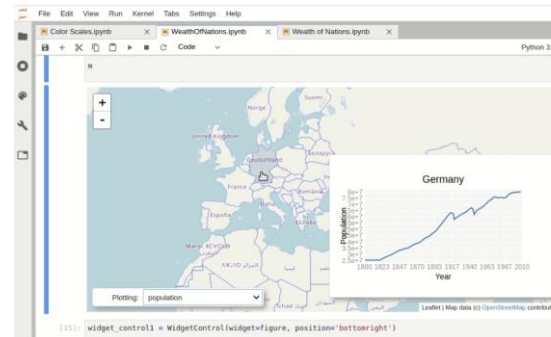
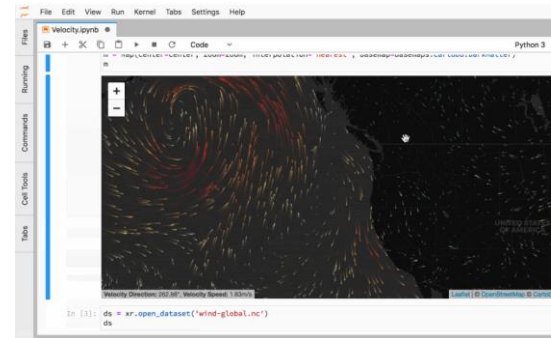
  return new THREE.Vector3(x,y,z)
}
"""
surf_g = ParametricGeometry(func=f);
surf = Mesh(geometry=surf_g, material=LambertMaterial(color='green', side='FrontSide'))
surf2 = Mesh(geometry=surf_g, material=LambertMaterial(color='yellow', side='BackSide'))
scene = Scene(children=[surf, surf2, AmbientLight(color='#777777')])
c = PerspectiveCamera(position=[5, 5, 3], up=[0, 0, 1],
                      children=[DirectionalLight(color='white',
                                                  position=[3, 5, 1],
                                                  intensity=0.6)])
renderer = Renderer(camera=c, scene=scene, controls=[OrbitControls(controlling=c)])
display(renderer)
```



<https://github.com/jupyter-widgets/pythreejs>

IPyLeaflet

A Jupyter / Leaflet bridge enabling interactive maps in the Jupyter notebook.



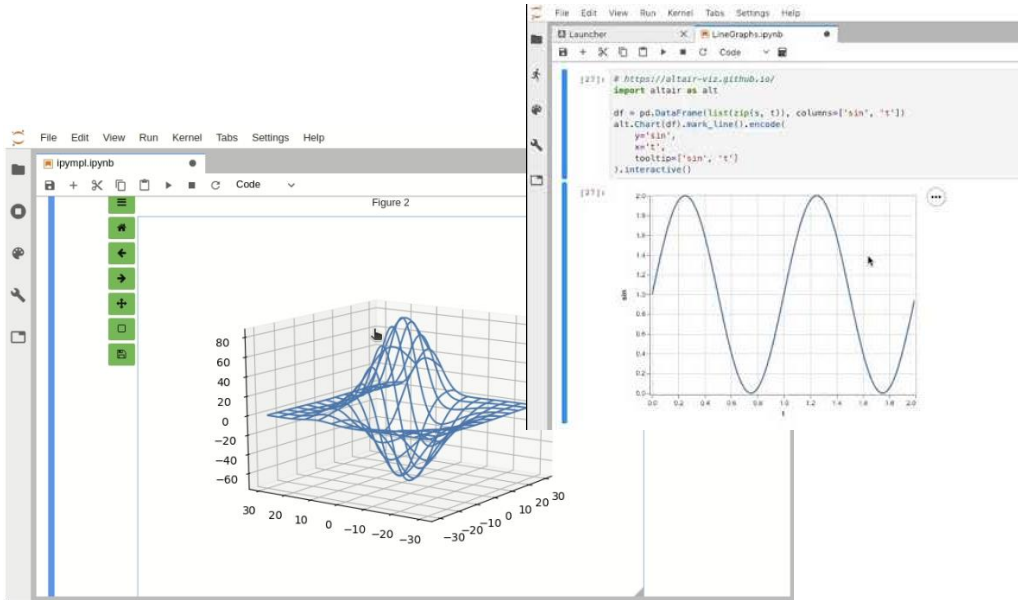
<https://github.com/jupyter-widgets/ipyleaflet>

JUPYTERLAB EXTENSIONS

Installed by default at Jupyter-JSC

IPyMPL - matplotlib

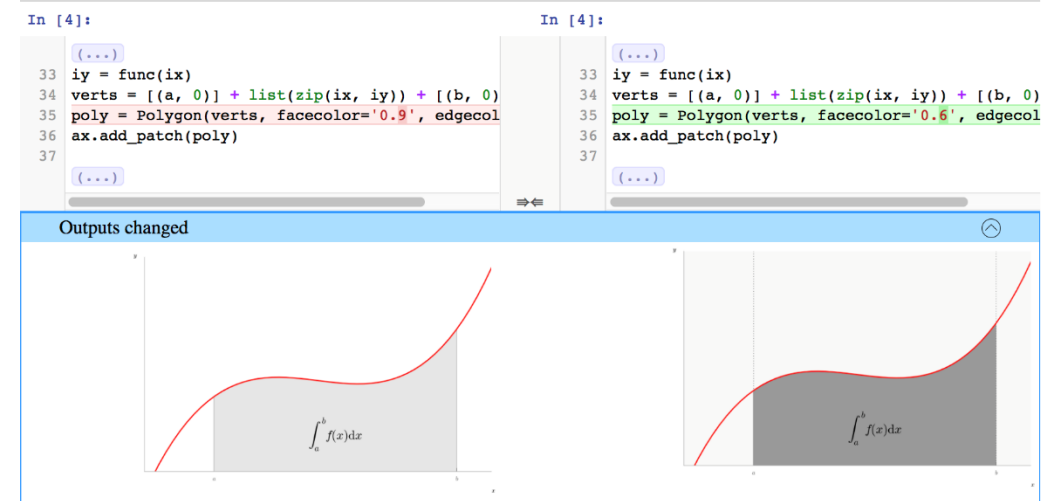
Leveraging the Jupyter interactive widgets framework, ipympl enables the interactive features of matplotlib in the Jupyter notebook and in JupyterLab.



<https://github.com/matplotlib/ipympl>

NBDime

Tools for diffing and merging of Jupyter notebooks.



<https://github.com/jupyter/nbdime>

JUPYTERLAB EXTENSIONS

Installed by default at Jupyter-JSC

Plotly

JupyterLab extension for the interactive and browser-based graphing library Plotly.

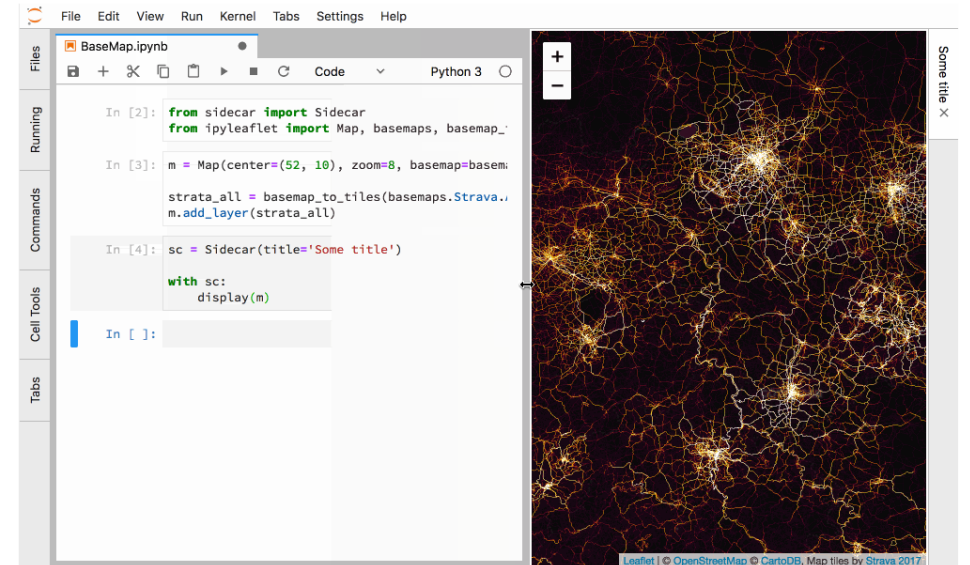
<https://plotly.com/python/>



<https://github.com/plotly/plotly.py>

JupyterLab-Sidecar

A sidecar output widget for JupyterLab.



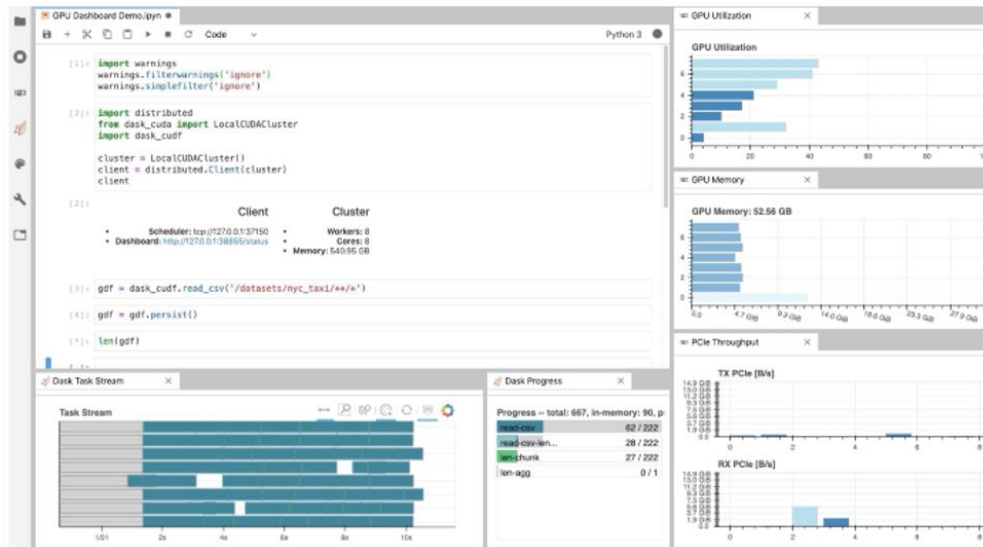
<https://github.com/jupyter-widgets/jupyterlab-sidecar>

JUPYTERLAB EXTENSIONS

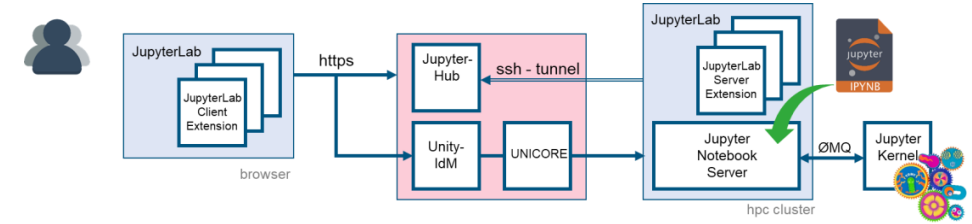
Installed by default at Jupyter-JSC

NVDashboard

NVDashboard is an open-source package for the real-time visualization of NVIDIA GPU metrics in interactive Jupyter Lab environments.

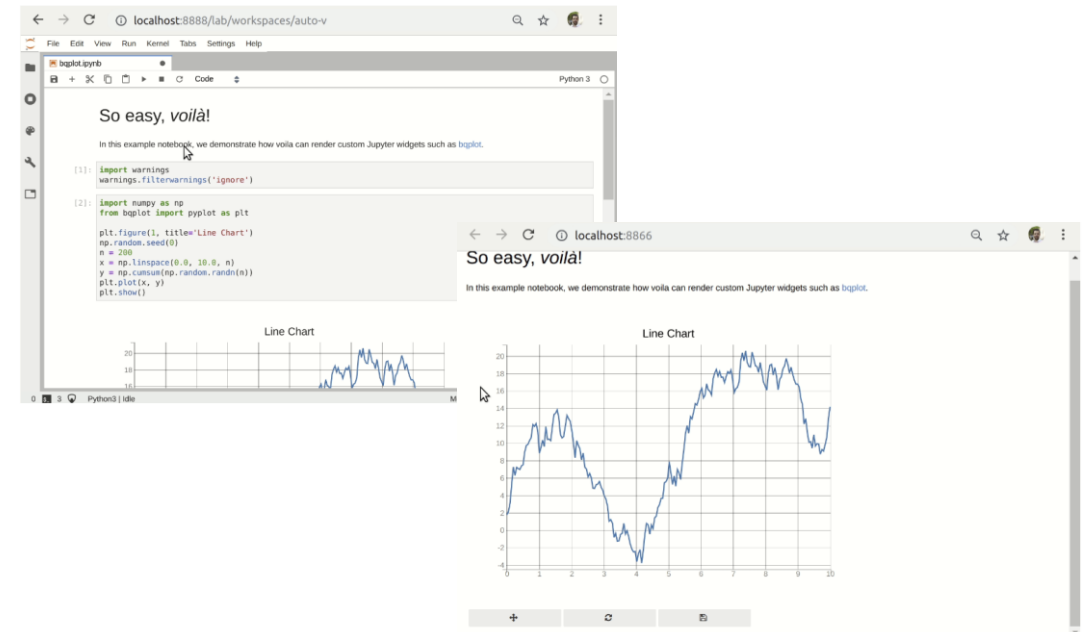


<https://github.com/rapidsai/jupyterlab-nvdashboard>
<https://developer.nvidia.com/blog/gpu-dashboards-in-jupyter-lab/>



Voilà

Voilà turns Jupyter notebooks into standalone web applications.



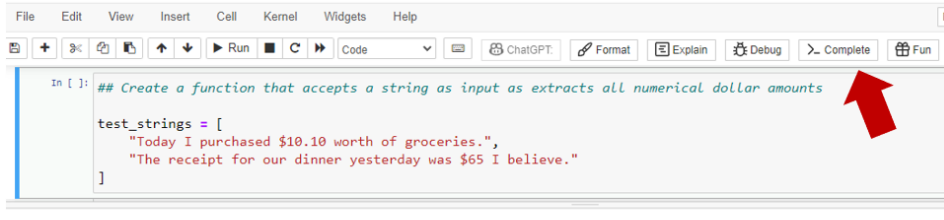
<https://github.com/voila-dashboards/voila>

JUPYTERLAB EXTENSIONS

... more useful extensions

ChatGPT for Jupyter

A browser extension to provide various helper functions in Jupyter Notebooks and Jupyter Lab, powered by ChatGPT.



The screenshot shows the JupyterLab interface. At the top is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. Below it is a toolbar with various icons. A red arrow points to the 'Complete' button, which is located next to the 'ChatGPT' button. The code cell below contains the following text:

```
In [ ]: ## Create a function that accepts a string as input as extracts all numerical dollar amounts

test_strings = [
    "Today I purchased $10.10 worth of groceries.",
    "The receipt for our dinner yesterday was $65 I believe."
]
```

ChatGPT - Complete Code

Here is the code that extracts all numerical dollar amounts from a string:

```
import re

def extract_dollar_amounts(string):
    dollar_amounts = re.findall(r'\$\d+(?:\.\d+)?', string)
    return dollar_amounts

test_strings = [
    "Today I purchased $10.10 worth of groceries.",
    "The receipt for our dinner yesterday was $65 I believe."
]

for test_string in test_strings:
    dollar_amounts = extract_dollar_amounts(test_string)
    print(f"Dollar amounts in '{test_string}': {dollar_amounts}")
```

This code will output the following:

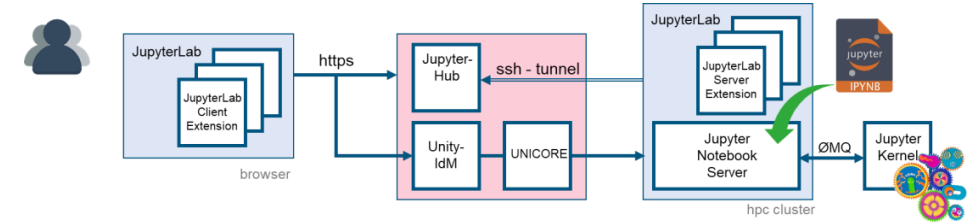
```
Dollar amounts in 'Today I purchased $10.10 worth of groceries.': ['$10.10']
Dollar amounts in 'The receipt for our dinner yesterday was $65 I believe.': ['$65']
```

<https://github.com/TiesdeKok/chat-gpt-jupyter-extension>

JUPYTERLAB EXTENSIONS

Installed by default at Jupyter-JSC

Extensions	old version	new version	type
Core			
@jupyterlab/server-proxy	v2.1.0	v3.1.0	prebuild
@jupyter-widgets/jupyterlab-manager	v2.0.0	v3.0.1	prebuild
jupyterlab-datawidgets	v6.3.0	v7.0.0	source
UI Enhancements			
@jlab-enhanced/recents		v3.0.1	prebuild
@jlab-enhanced/favorites	v2.0.0	v3.0.0	prebuild
jupyterlab-topbar-extension	v0.5.0	v0.6.1	
jupyterlab-system-monitor	v0.6.0	v0.8.0	prebuild
@jupyter-server/resource-usage		v0.6.0	n/a
jupyterlab-theme-toggle	v0.5.0	v0.6.1	source
jupyterlab-controlbtn	jupyterlab-control	v0.5.0	n/a
@jupyterlab/toc	v4.0.0	integrated into JupyterLab 3	
Developer Tools			
@jupyterlab/git	v0.23.3	v0.32.4	prebuild
jupyterlab-gitlab	v2.0.0	v3.0.0	prebuild
@krassowski/jupyterlab-lsp	v2.1.3	v3.9.0	prebuild
nbdime-jupyterlab	v2.1.0	v3.1.0	prebuild
@ryantam626/jupyterlab_code_formatter	v1.3.8	v1.4.10	prebuild
@jimbarr/jupyterlab_spellchecker	v0.2.0	v0.7.2	prebuild
jupyterlab-nvdashboard		v0.6.0	prebuild



Data Visualization

jupyter-matplotlib	v0.7.4	v0.9.0	prebuild
@bokeh/jupyter_bokeh	v2.0.4	v3.0.4	prebuild
jupyterlab-plotly	v4.14.3	v5.3.1	
bqplot	v0.5.22	v0.5.32	prebuild
@pyviz/jupyterlab_pyviz	v1.0.4	v2.1.0	prebuild
jupyter-leaflet	v0.13.3	v0.14.0	prebuild
ipyvolume	v0.6.0-alpha.5	v0.6.0-alpha.8	prebuild
jupyter-threejs	v2.2.0	v2.3.0	prebuild
@jupyter-widgets/jupyterlab-sidecar	v0.5.0	v0.6.1	prebuild

Framework Integrations

dask-labextension	v3.0.0	v5.1.0	prebuild
@jupyterlab/latex	v2.0.1	v3.1.0	prebuild
jupyter-webrtc	v0.5.0	v0.6.0	prebuild

Dashboard Development

jupyter-vue	v1.5.0	v1.6.1	
jupyter-vuetify	v1.6.1	v1.8.1	
@voila-dashboards/jupyterlab-preview	v1.1.0	v2.1.0-alpha.2	prebuild
jupyterlab-dash	v0.4.0	v0.4.0	prebuild

Welcome

jupyterlab_iframe	v0.3.0	v0.4.0	source
jupyterlab-tour		v3.1.3	prebuild

CONCLUSION

Why Jupyter is so popular among Data Scientists

JupyterLab ...

- ... is a **web-based platform for interactive computing and data analysis** that is well-suited to the needs of research software engineers.
- ... provides researchers with a **comprehensive environment** for working with code, text, multimedia, and data, making it an ideal tool for a wide range of research tasks.
- ... is designed to be **flexible and customizable**, and can be modified to suit the specific needs and workflows of individual researchers.
- ... supports the creation of **reproducible research** through its support for Jupyter notebooks.
- ... supports **collaboration and sharing** of research work through its support for sharing notebooks, dashboards, and other elements of a research project.
- ... provides a wide range of **extensions and plugins** that can be used to integrate other tools and services into the environment.
- ... is an **open-source project**, which means that researchers have access to the source code and can contribute to its development.

