



JUPYTERLAB - SUPERCOMPUTING IN YOUR BROWSER

Introduction to Jupyter-JSC at Jülich Supercomputing Centre

2024-11-12 | JENS H. GÖBBERT

(J.GOEBBERT@FZ-JUELICH.DE)

TIM KREUZER

(T.KREUZER@FZ-JUELICH.DE)

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 $p = \frac{dx}{dt}$:

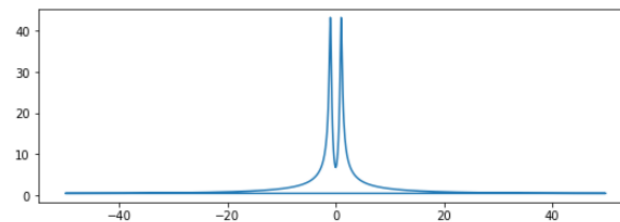
```
[42]: N = len(t)
      dt = t[1]-t[0]
      dt
```

```
[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

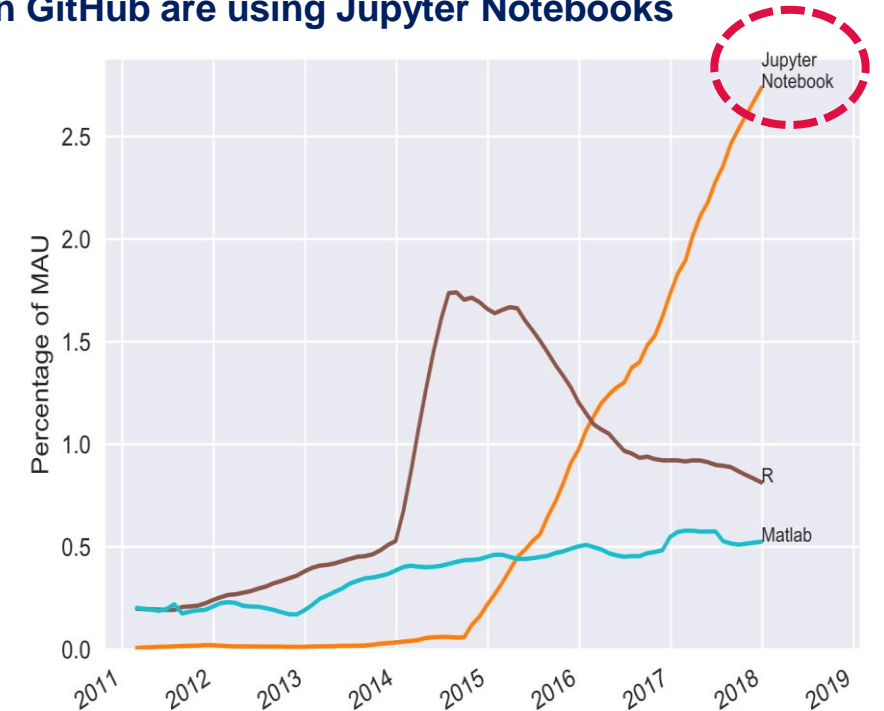
Output

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 ...

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



Initial Basis

JupyterLab modules
Authentication via Unity/IdM
Authorization via UNICORE
 Orchestration Docker Swarm
 Synchronization of User-DBs
 Basic Data Protection Regulation
 Fulfill Safety Requirements

Usage

Inplace Dokumentation
 R, Julia, C++, Octave, Ruby
 JupyterLabs on OpenStack
 Dashboard Development
JupyterLab Usability
 Kernel for Vis, DL
 Testing & Benchmarking

Features

Remote Desktop Integration
 Optional 2-Factor Auth.
Use for Workshops
 Specialized Functionalities
 Enhanced Data Access
 Extended Logging
 Cross-Side Demonstration

Redesign

Switch to **Kubernetes**
 Redesign Management
 Switch to **JupyterLab 3**
 GPFS through UFTP
 Support for User Extensions
 Easybuild Modularization

Customization

Project/Community JHubs
 Upgrade JHub Entrance-UI
 Comp. Resource Permissions
 Maintenance Improvements
 Upgrade of Load Balancer
Modularization of Backend
External Clouds & HPC



HISTORY OF JUPYTERLAB AT JSC

2

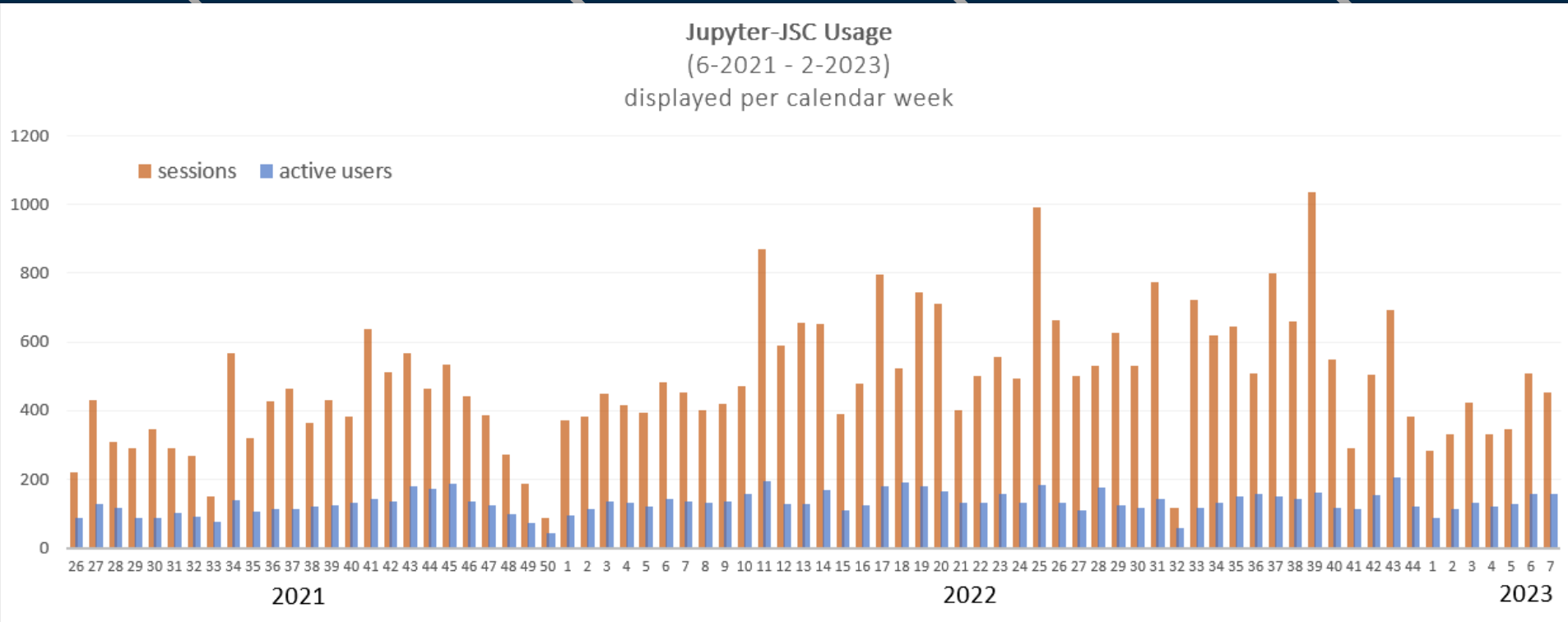
22

Initi

ization

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

Community JHubs
 Entrance-UI
 e Permissions
 improvements
 d Balancer
of Backend



JLa

B+X

TERMINOLOGY

TERMINOLOGY

What is JupyterLab

JupyterLab

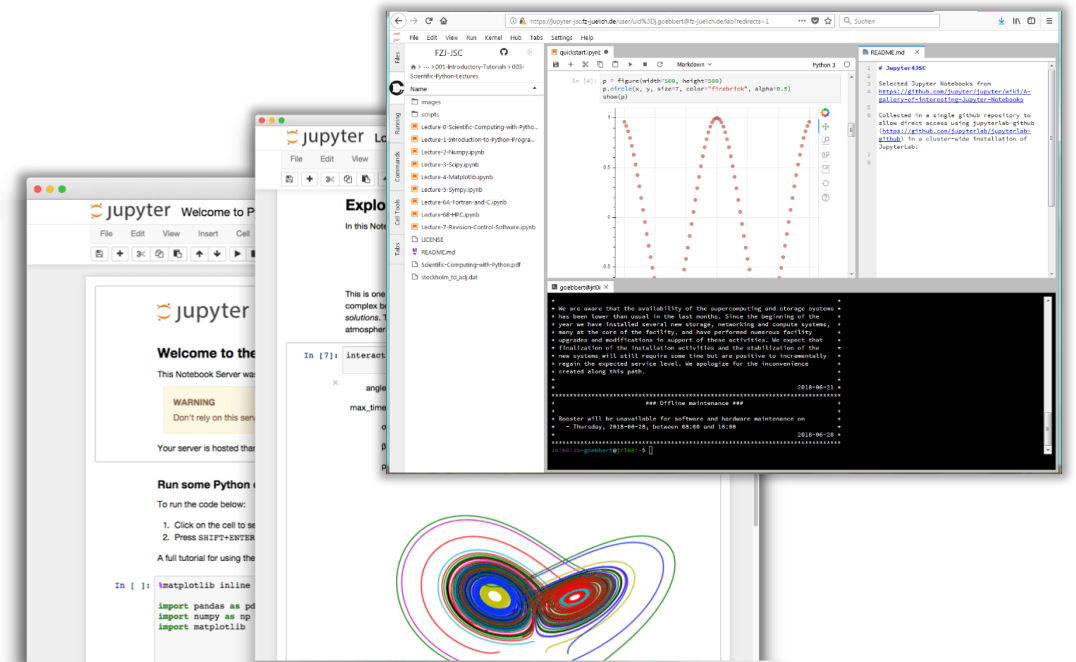
- **Interactive** working environment in the web browser
- For the creation of **reproducible** computer-aided narratives
- Very **popular** with researchers from all fields
- Jupyter = Julia + Python + R

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>

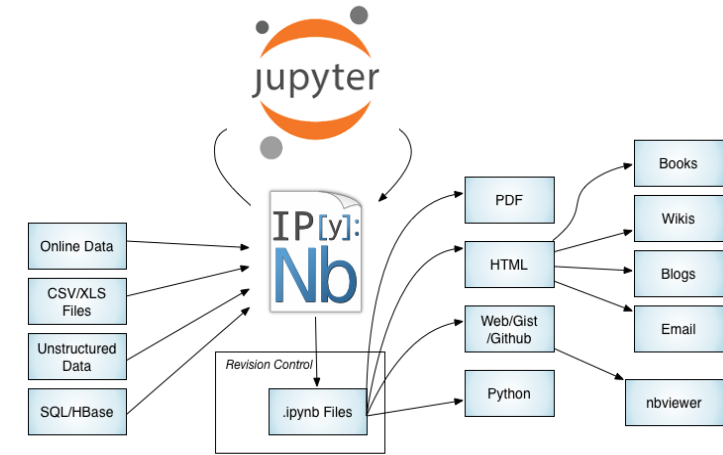
TERMINOLOGY

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..



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[1:,0:1])
      # calculate the frequencies for the components in F
      w = fftfreq(N, dt)
      fig, ax = plt.subplots(figsize=(9,3))
      ax.plot(w, abs(F));
```

Output

Output

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?

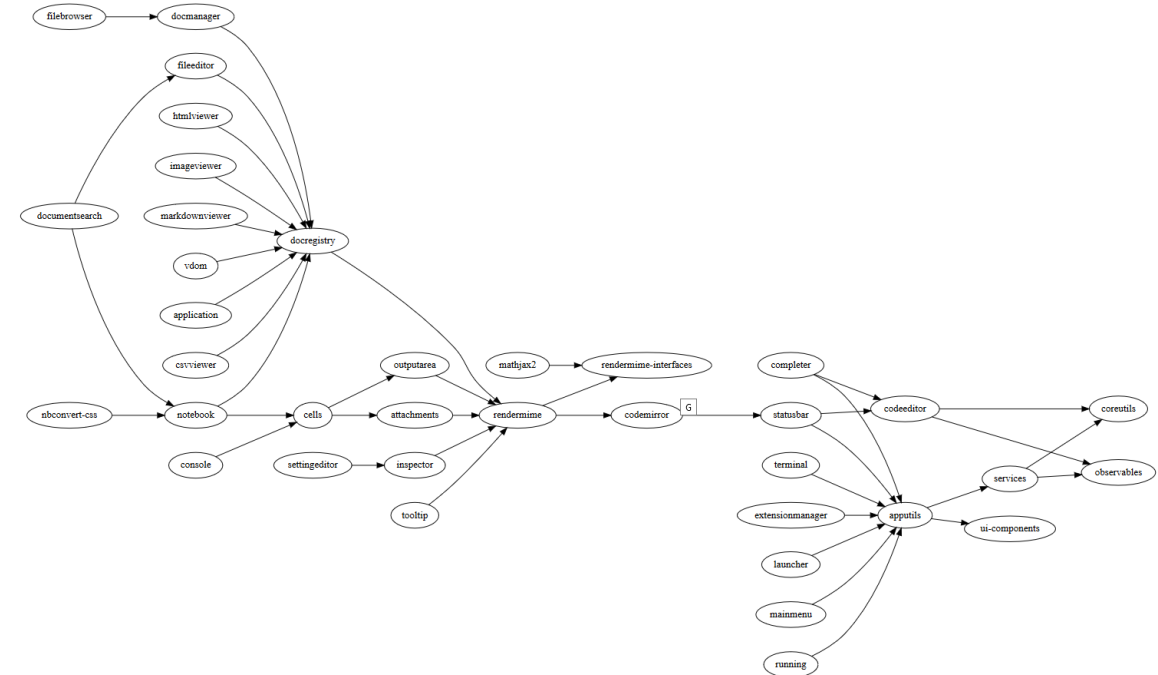
JupyterLab Extension

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.

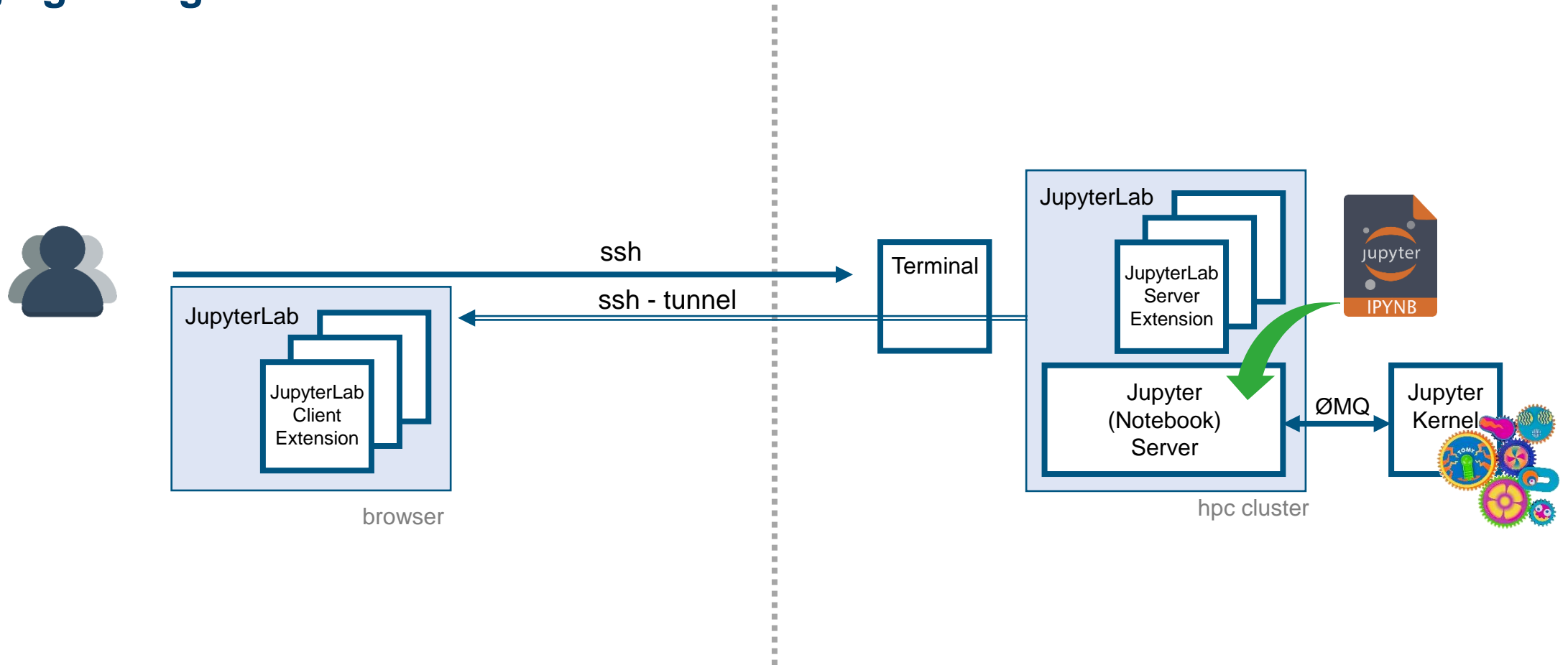
The whole JupyterLab itself is simply a **collection of extensions** that are no more powerful or privileged than any custom extension.



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

TERMINOLOGY

Bringing all together



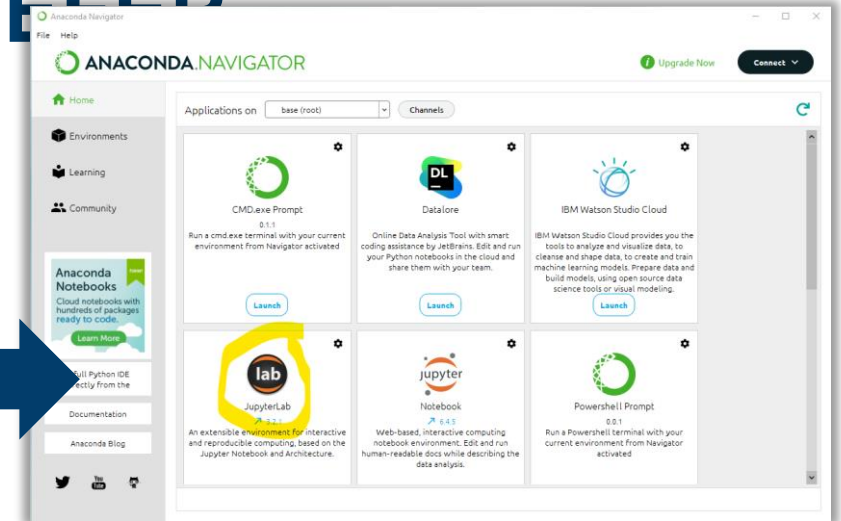
INSTALLATION

JUPYTERLAB - WHEREVER YOU PREFER

Local, Remote, Browser-only

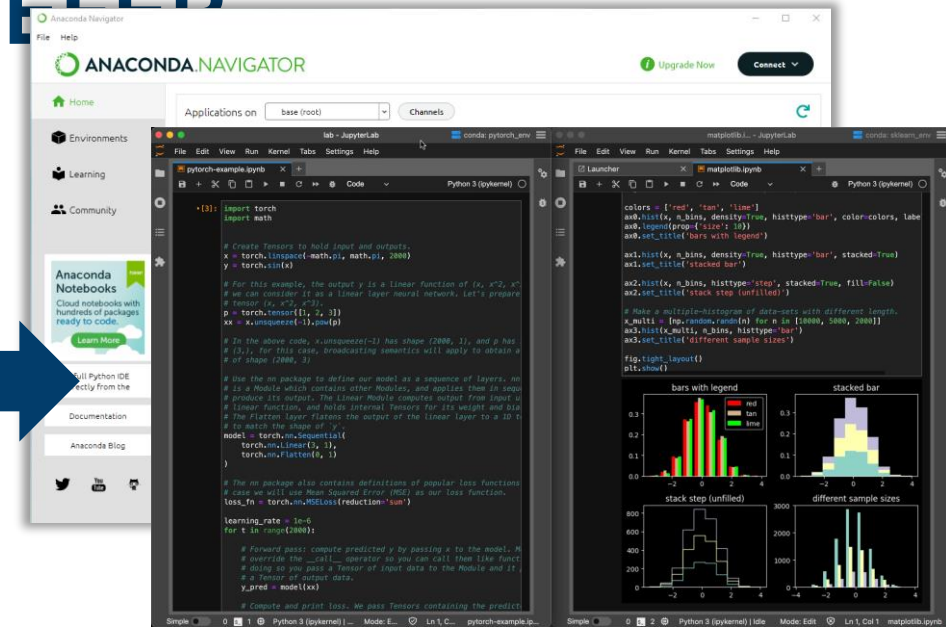
Local installation:

- **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

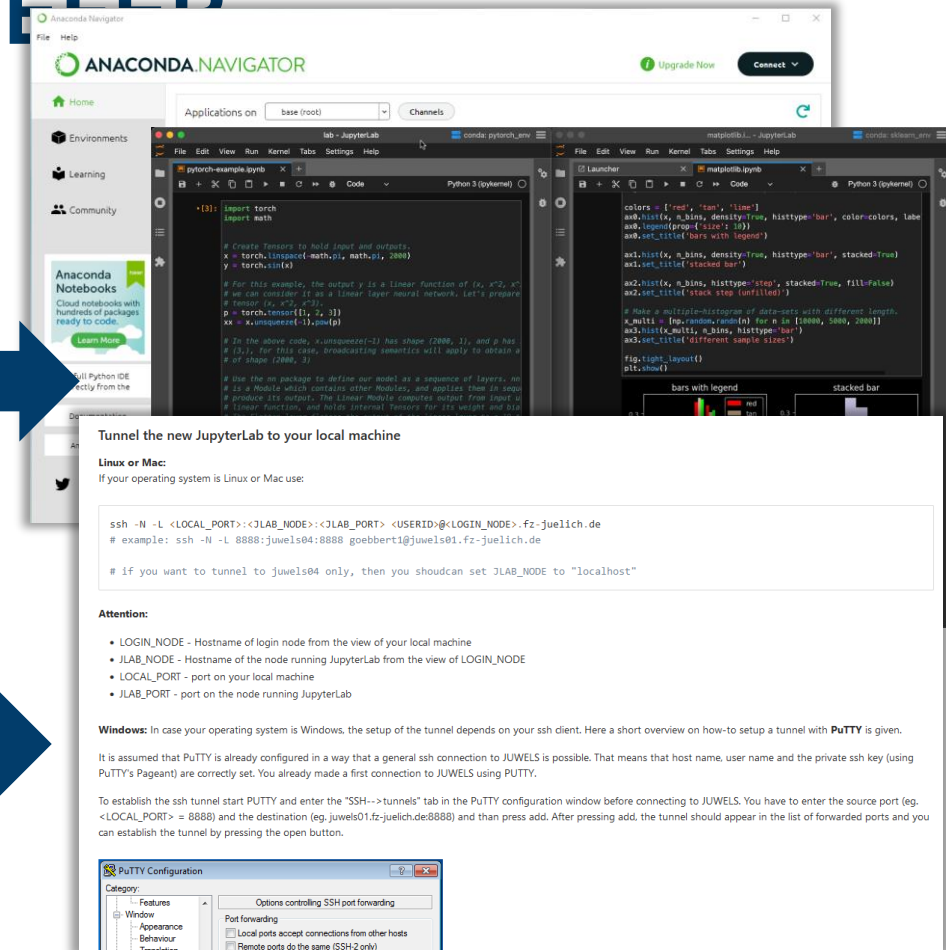


Local installation:

- 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 - WHEREVER YOU PREFER

Local, Remote, Browser-only



Local installation:

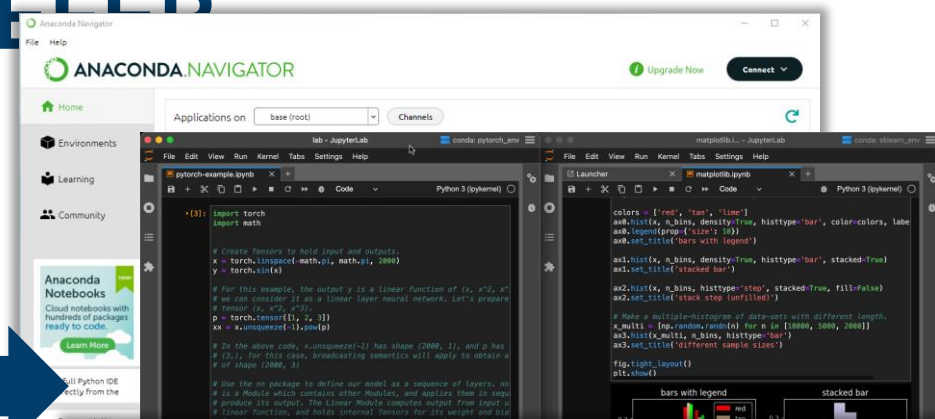
- **JupyterLab** installed using conda, mamba, pip, pipenv or docker.
→ https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html
- **JupyterLab** installed as normal desktop application = **JupyterLab Desktop**
→ <https://github.com/jupyterlab/jupyterlab-desktop/releases>

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.

JUPYTERLAB - WHEREVER YOU PREFER

Local, Remote, Browser-only



Local installation:

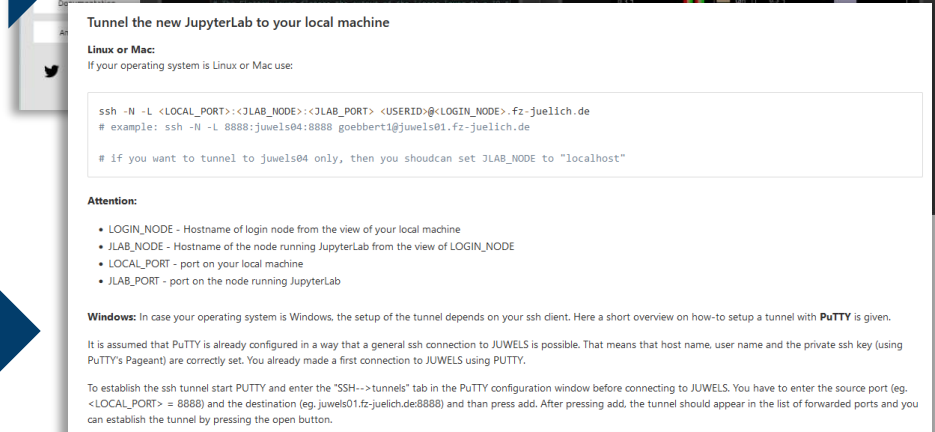
- **JupyterLab** installed using conda, mamba, pip, pipenv or docker.
→ https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html
- **JupyterLab** installed as normal desktop application = **JupyterLab Desktop**
→ <https://github.com/jupyterlab/jupyterlab-desktop/releases>

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):

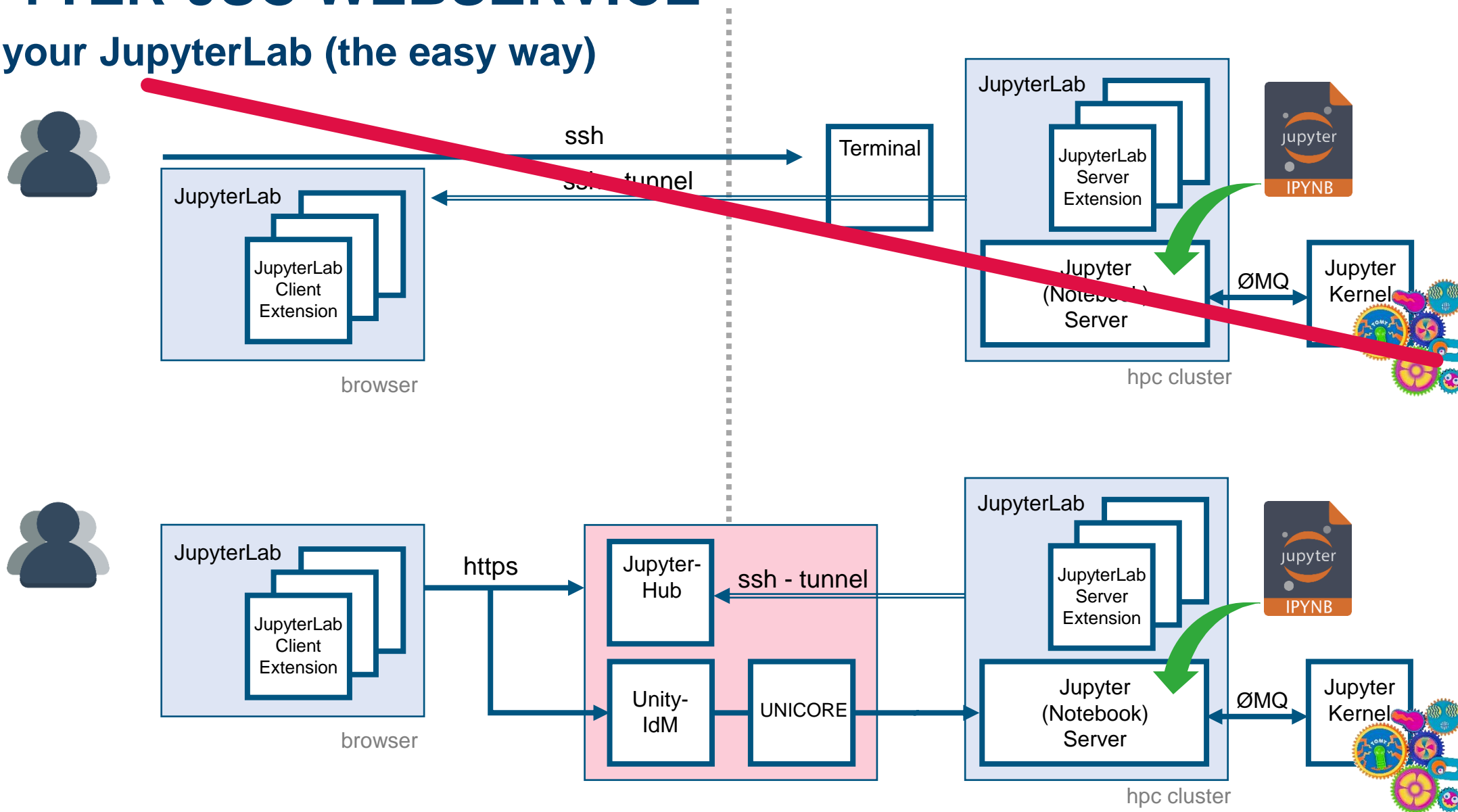
- **JupyterLab** local with server + client in your browser = **JupyterLite**
Includes a browser-ready Python environment named Pyodide.
→ <https://jupyter.org/try-jupyter/lab>



START & LOGIN

JUPYTER-JSC WEBSERVICE

Start your JupyterLab (the easy way)



PRE-ACCESS TODOS

1) Register & Login

- ✓ <https://judoor.fz-juelich.de>

2) Join a project

- ✓ Wait to get joined by the project PI

3) Sign usage agreement

- ✓ Wait for creation of HPC accounts

4) Check Connected Services:

- ✓ jupyter-jsc

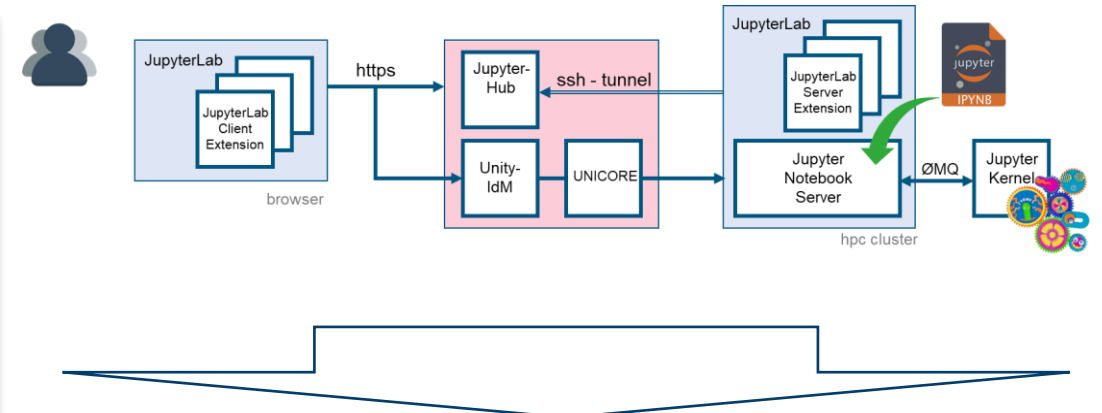
The screenshot shows the JU account dashboard. At the top, there is a navigation bar with 'JU Your account Mentoring Search' and 'Detailed Statistics' on the right. Below the navigation bar, the user's account details are listed: Account, Salutation, E-mail address (with a checkmark), Telephone, and Address. A 'Mentored projects' section is empty. The 'Systems' section lists two systems: 'judac' with 'Usage agreement confirmed on 18.04.2021' (marked with a green checkmark) and 'jureca' with 'You need to sign the usage agreement to access this system' (marked with a red X). The 'Projects' section shows one project: 'Interactive High-Performance Computing with Jupyter @ JSC' (marked with a green checkmark). The 'Software' section is empty. The 'Connected Services' section lists 'trac', 'llview', 'jards', 'gitlab', and 'jupyter-jsc' (marked with a green checkmark).

JUPYTER-JSC WEBSERVICE

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

The image shows three overlapping screenshots of the JupyterLab web interface. The top screenshot displays a "Your server is starting up..." message with a progress bar and logs. The middle screenshot shows a table of "NEW JUPYTERLAB" instances with columns for Name, Configuration, Status, and Actions. The bottom screenshot is a landing page titled "Supercomputing in Your Browser" with a "Jupyter-JSC" logo and a "Login" button.

Name	Configuration	Status	Actions
NEW JUPYTERLAB			
+			
jujuf_login_3.6	System JUSUF Partition LoginNode	Project ccsvs	Start
jujews-3.6	System HANDEL Partition LoginNode	Project ccsvs	Start



The image shows a screenshot of a JupyterLab notebook interface. The notebook contains Python code for matrix multiplication using NumPy and GPU resources. The code is as follows:

```
[1]: import math
[2]: import numpy as np
[3]: from numba import cuda
[4]: import matplotlib.pyplot as plt
[5]: matplotlib inline

[2]: len(cuda.gpus)

[3]: len(cuda.gpus)

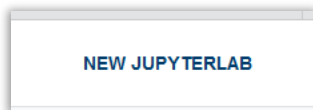
[4]: @cuda.jit
def mandelbrot_number(iterations):
    # matrix indices
    i, j = cuda.grid(2)
    size = math.ceil(4)
    # skip threads outside the matrix.
    if i > size or j > size:
        return
    # Run the simulation.
    c = 1.2 - 0.5j, size * j
    z = 0
    for n in range(iterations):
        z = z**2 + c
```

The interface also shows a "GPU DASHBOARDS" sidebar with metrics for GPU Utilization, GPU Memory, GPU Resources, PCIe Throughput, WLink Throughput, WLink Timeout, and Machine Resources. A "GPU Memory" graph is visible at the bottom.

JUPYTER-JSC WEBSERVICE

Control Panel

A. New JupyterLab

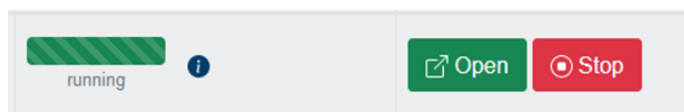


B. Configuration Dialog

- Lab Config: set Name, Version, System, Account, Project, Partition
- Resources: if running on a compute node
- Kernels and Extensions: Optional addons

C. Actions

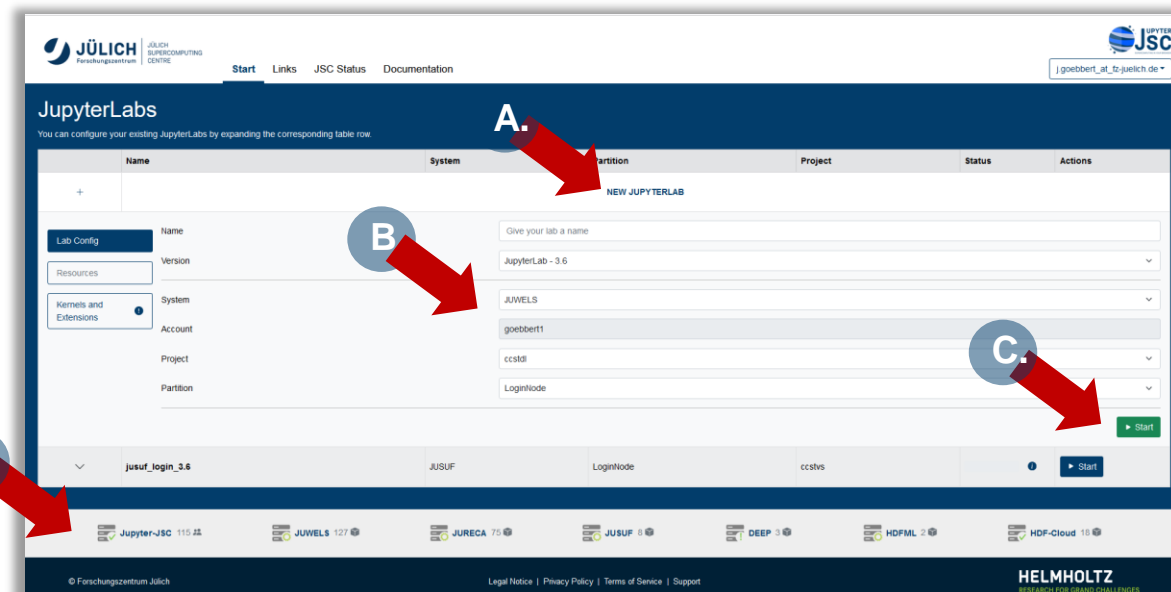
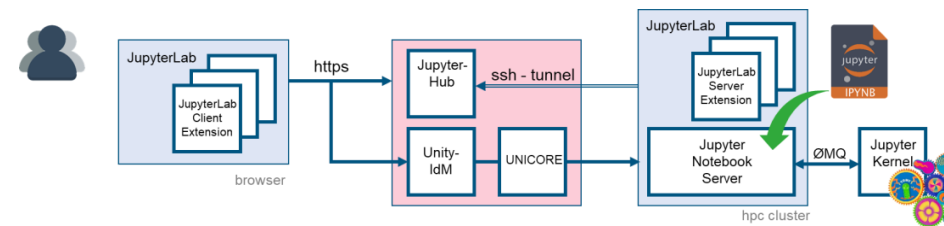
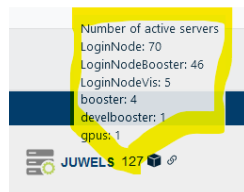
- Start/Open/Stop a JupyterLab
- Change/Delete **configuration**



D. Statusbar

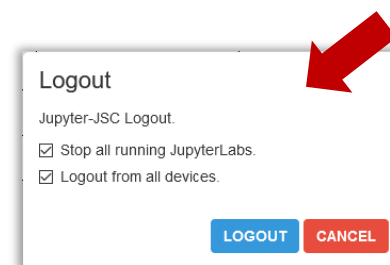


- Shows, (hover to get more details)
 - Number of active users in the last 24h
 - Number of running JupyterLabs
- Click to see system status page



E. 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**

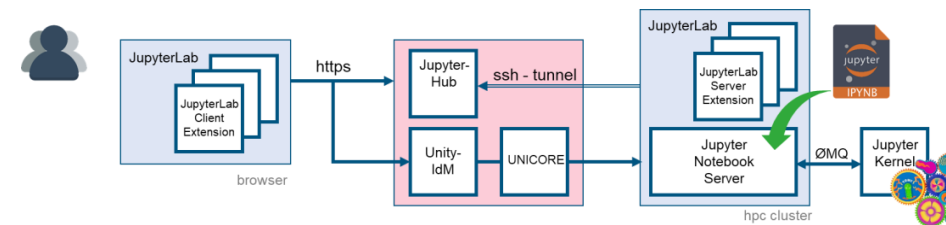
- user account settings visible in judoor.fz-juelich.de
- system specific usage agreement on JuDoor is signed (!!!)
- currently available systems in all of your projects

Basic options

- Version:
multiple versions of JupyterLab are installed
- System:
JUWELS, JURECA, JUSUF, DEEP, JSC-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					
Name		Give your lab a name			
Version		JupyterLab - 3.6			
System		JUWELS			
Account		goebbert1			
Project		ccstdl			
Partition		LoginNode			

Login Nodes

- LoginNode
- LoginNodeBooster
- LoginNodeVis

Compute Nodes

- batch
- devel
- develgpu
- gpu
- mem192

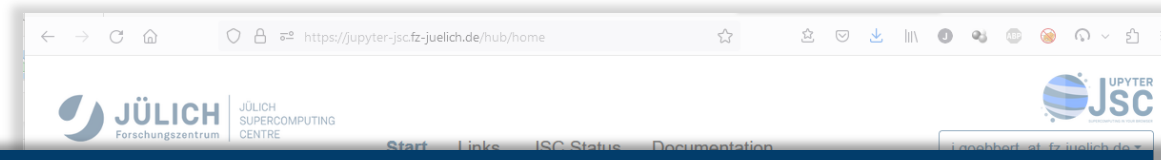
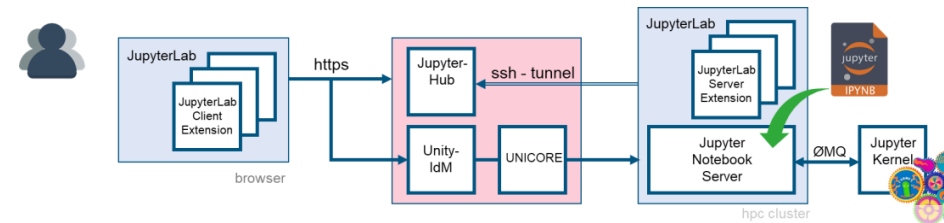
JUPYTER-JSC WEBSERVICE

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 (!!!)



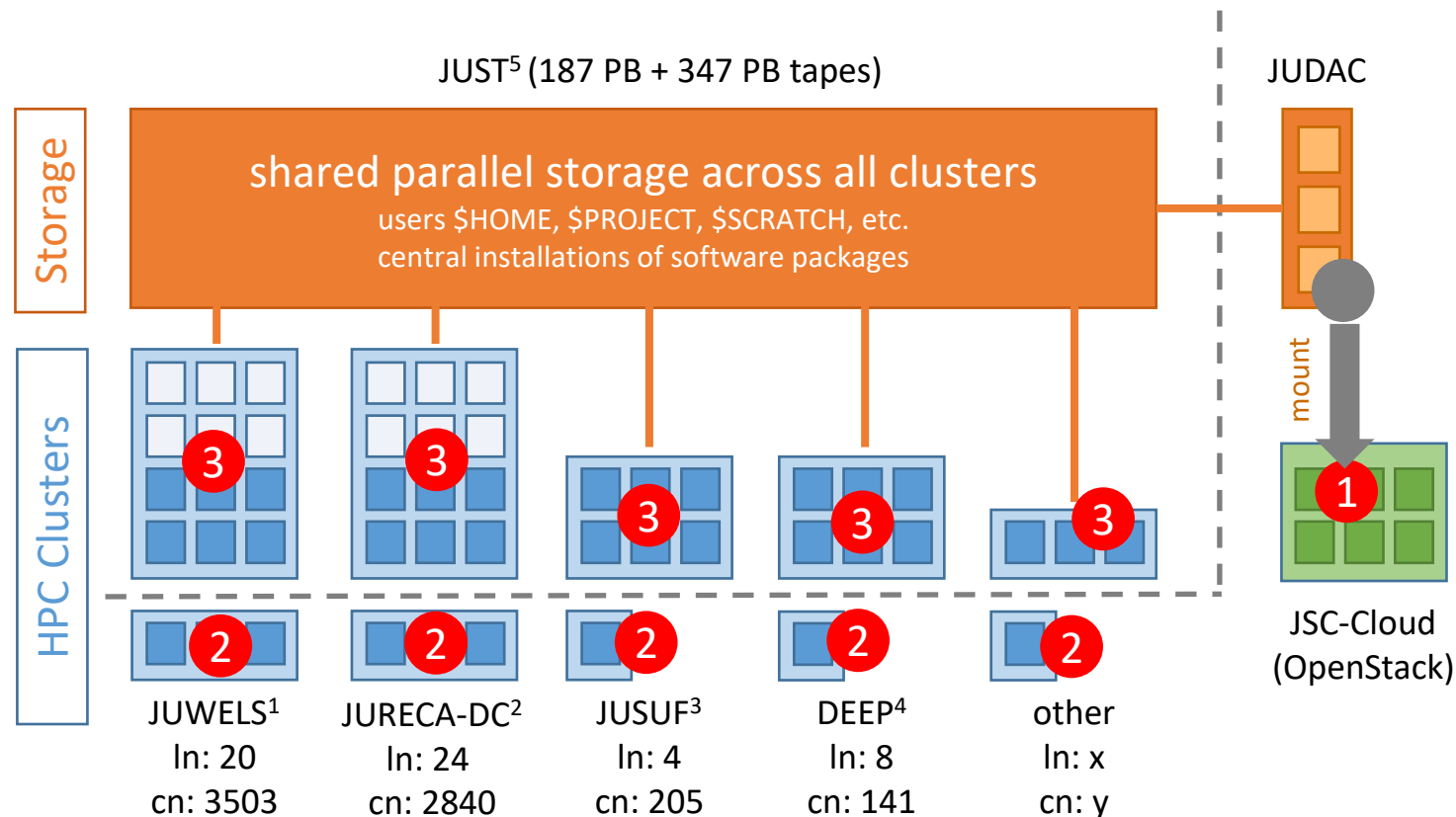
	Name	System	Partition	Project	Status	Actions
^	jusuf_login_3.4	JUSUF	LoginNode	ccstdl	Error ⓘ	<button>⌂ Retry</button>
Service	<ul style="list-style-type: none"> ▶ 2023_03_14 14:04:23.887: Sending request to backend service to start your service on JUSUF. ▼ 2023_03_14 14:04:33.724: Setup ssh port-forwarding. Create ssh tunnel with system user ljupyter. JupyterHub will then be able to connect to JupyterLab at jsfl05i:52243 ▼ 2023_03_14 14:04:34.044: Cancel in progress We're stopping your service. This may take a few seconds. ▼ 2023_03_14 14:04:34.044: 2023_03_14 14:04:34.044: Could not setup tunnel Request identification: d2f8bd07a10f4534a9897f568ef3cbcb 					
Options						
Resources						
Reservation						
Kernels and Extensions						
Logs						

Extra options

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

LoginNode
LoginNodeBooster
LoginNodeVis
Compute Nodes
batch
devel
develgpu
gpu
mem192

JUPYTERLAB EVERYWHERE



JupyterLab everywhere

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

no. login nodes = In

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>

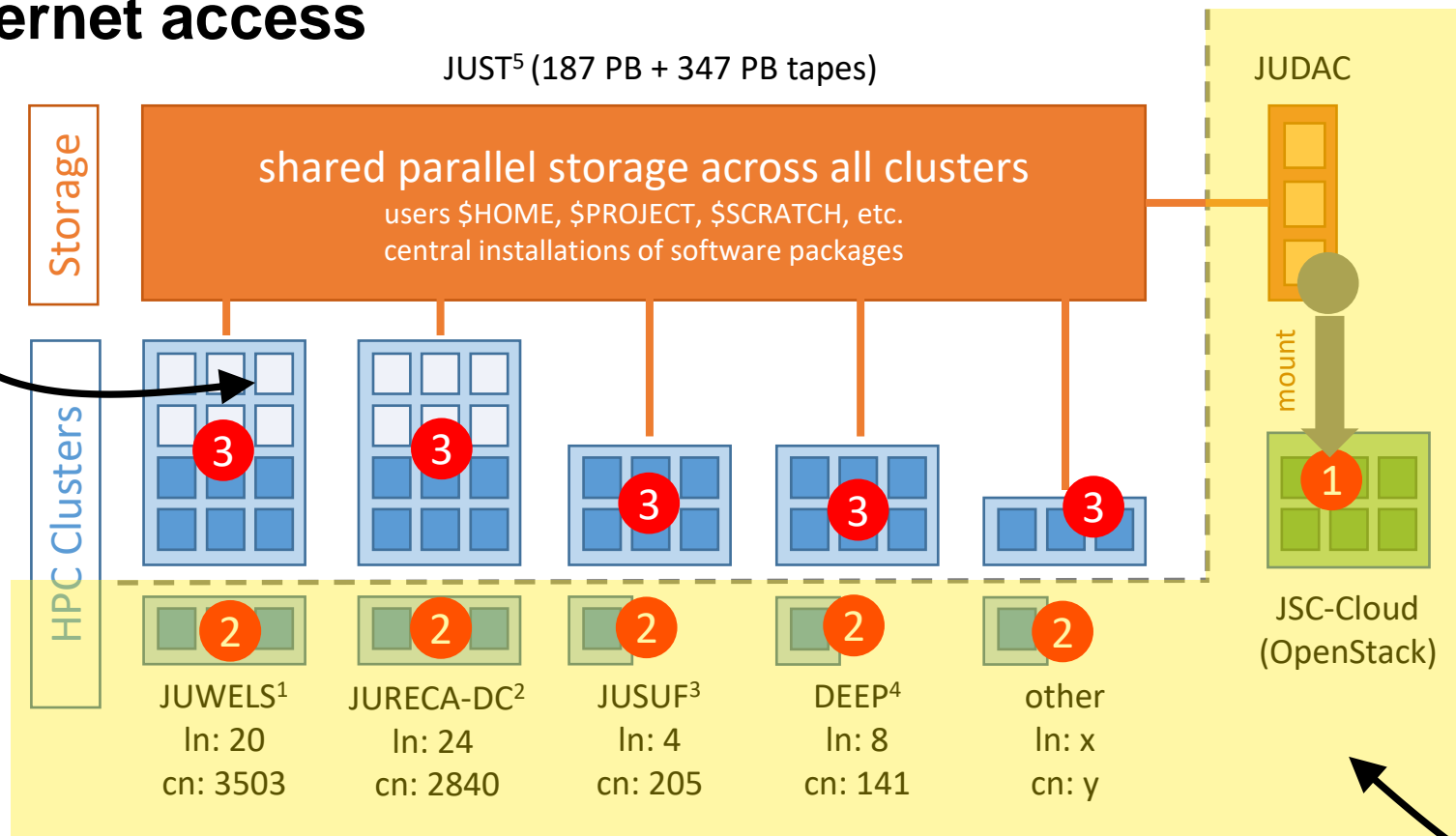
[3] <https://apps.fz-juelich.de/jsc/hps/jusuf/configuration.html>

[4] https://www.fz-juelich.de/en/ias/jsc/systems/prototype-systems/deep_system

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

JUPYTERLAB EVERYWHERE

NO internet access



JupyterLab everywhere

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

internet access

no. login nodes = In
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>

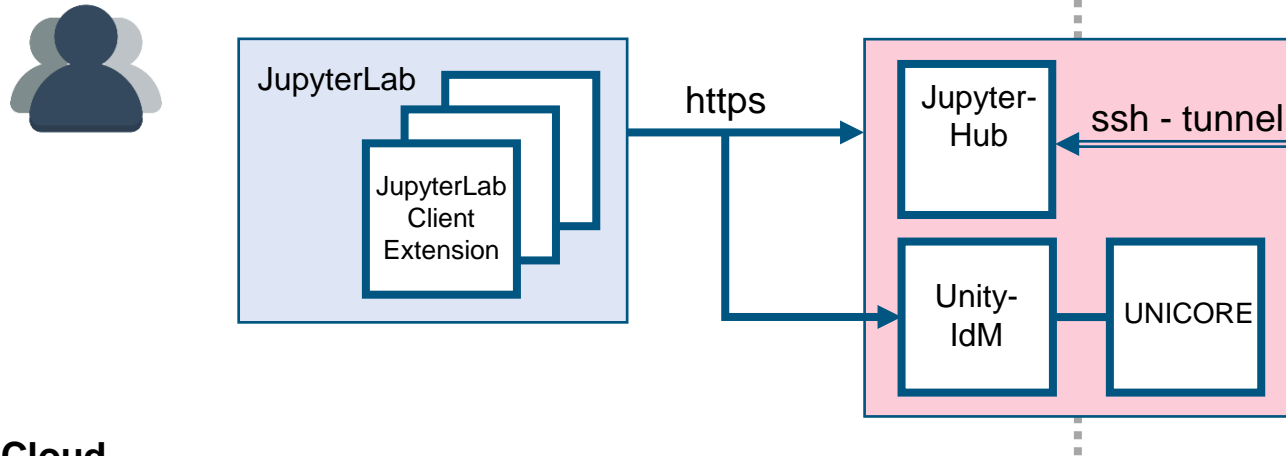
[3] <https://apps.fz-juelich.de/jsc/hps/jusuf/configuration.html>

[4] https://www.fz-juelich.de/en/ias/jsc/systems/prototype-systems/deep_system

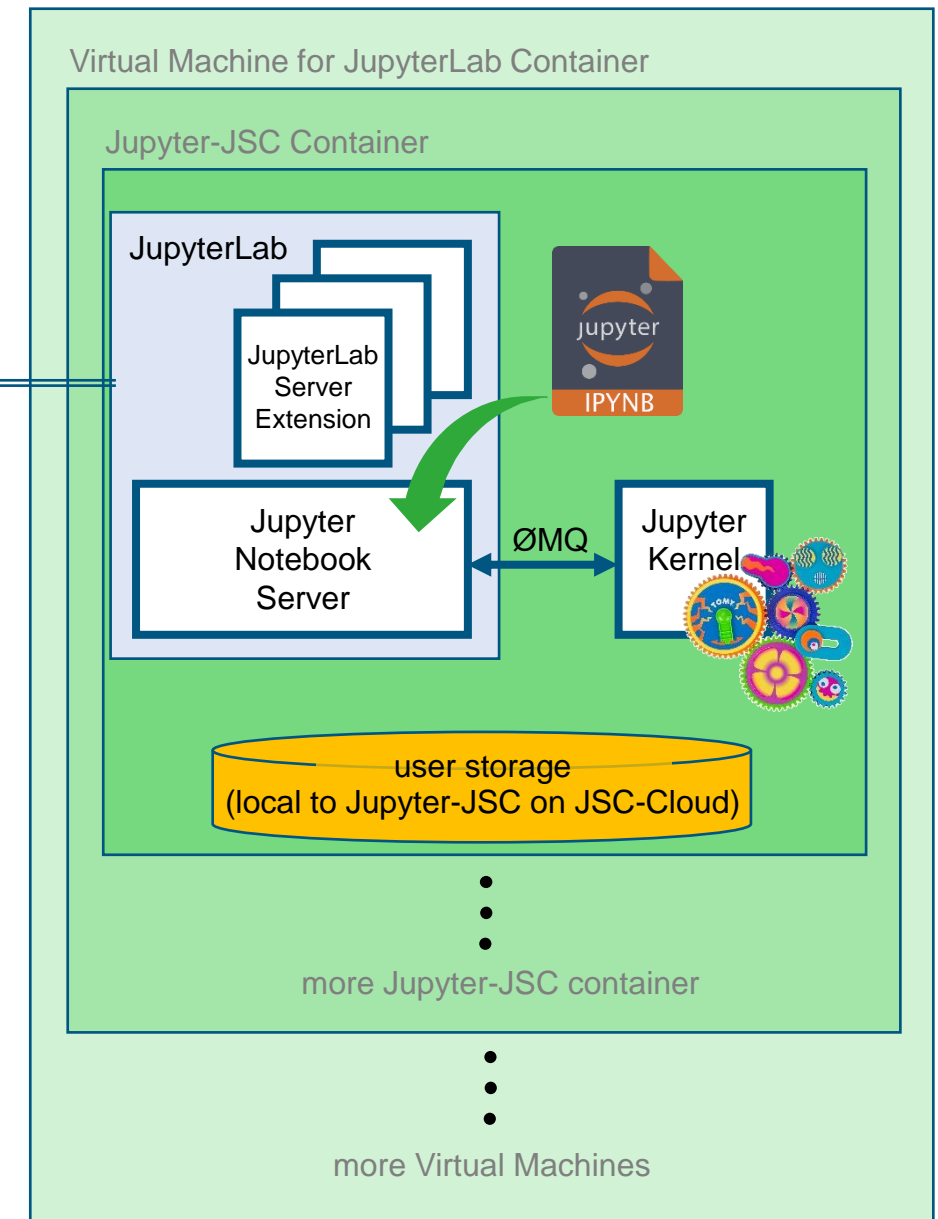
[5] <https://apps.fz-juelich.de/jsc/hps/just/configuration.html>

JUPYTER-JSC WEBSERVICE

System: JSC-Cloud



JSC-Cloud – OpenStack Cluster for running Virtual Machines



JSC-Cloud

Any user having

- a JSC account (**judoor.fz-juelich.de**)
- and the **Connected Service** “jupyter-jsc” enabled (default for users with HPC accounts or fz-juelich.de email address)

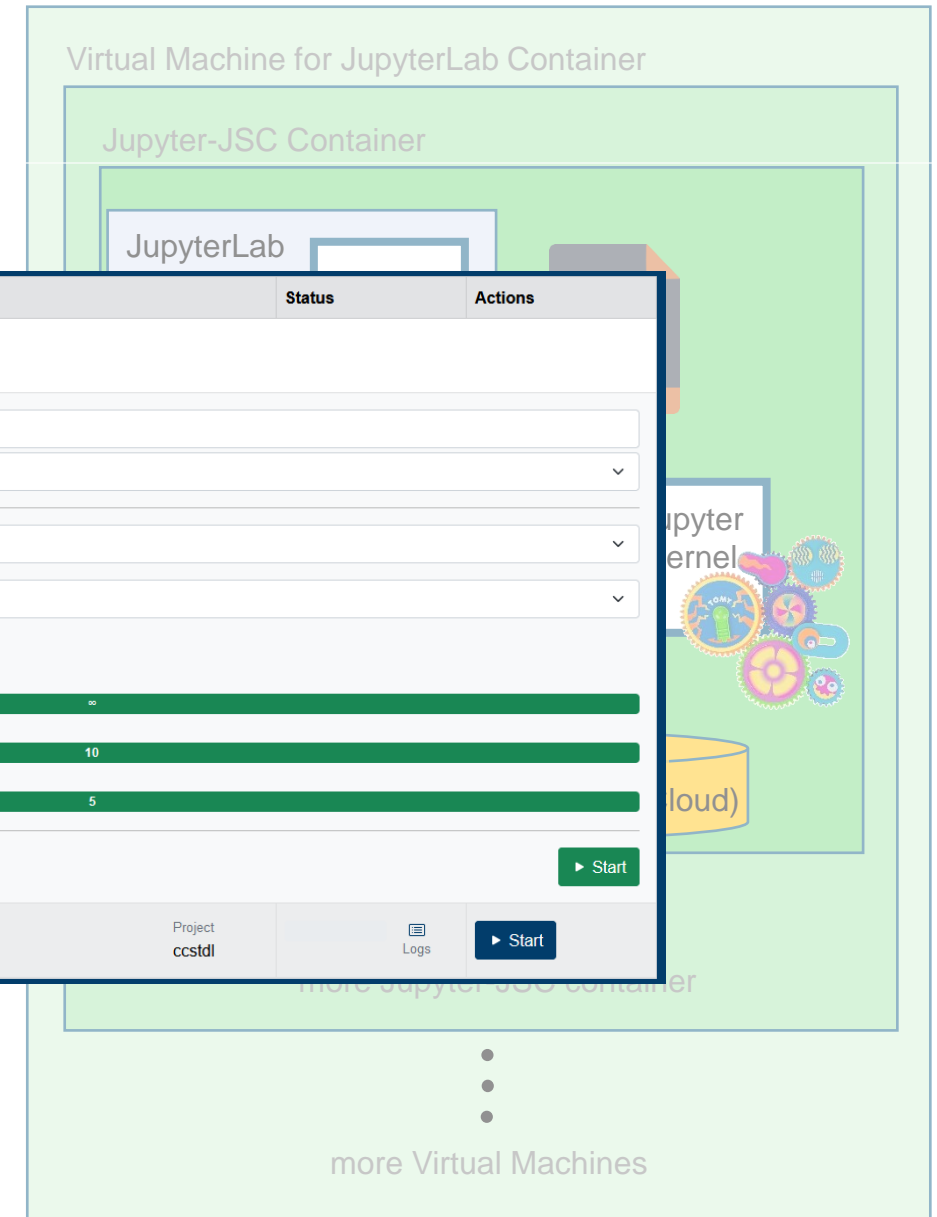
can start

- Jupyter-JSC container images (containing JupyterLab) on the JSC-Cloud

JUPYTER-JSC WEBSERVICE

System: JSC-Cloud

JSC-Cloud – OpenStack Cluster for running Virtual Machines



The screenshot shows the 'NEW JUPYTERLAB' configuration page. The 'System' dropdown menu is open, and 'JSC-Cloud' is selected and circled in yellow. A yellow arrow points to this selection. The 'Flavor' dropdown is set to '2GB RAM, 1VCPU, 5 days'. Below the configuration fields, there is a table of 'Available Flavors' with a legend for their status.

Flavor	Status
2GB RAM, 1VCPU, 5 days	Free
4GB RAM, 1VCPUs, 2 days	Used
8GB RAM, 2VCPUs, 10 hours	Limit exceeded

JSC-Cloud

Any user h

- a JSC
- and the
- (default

can start

- Jupyter-JSC container images (containing JupyterLab) on the JSC-Cloud

JUPYTER-JSC WEBSERVICE

System: JSC-Cloud

Custom Docker image on the JSC-Cloud

- Requirements:
 - Custom docker image must start a Jupyter server
- Examples:
 - <https://jupyter-docker-stacks.readthedocs.io/en/latest/using/selecting.html>

The screenshot displays the JupyterLab configuration page on the JSC-Cloud website. The page title is "NEW JUPYTERLAB". The configuration form includes the following fields and values:

- Name:** myCustomDocker
- Version:** Custom Docker image (starting a Jupyter server)
- Image:** e.g. jupyter/datascience-notebook
- Private repository:**
- Image Repository:** URL for your image registry
- Username:** Please enter your username
- Password:** Please enter your password
- Mount user data:**
- User data path:** /mnt/userdata
- System:** JSC-Cloud
- Flavor:** 4GB RAM, 1VCPU, 2 days

The "Available Flavors" section shows the following status:

- 4GB RAM, 1VCPU, 2 days: 3 Free, 47 Used
- 2GB RAM, 1VCPU, 5 days: 7 Free, 0 Used
- 8GB RAM, 2VCPU, 10 hours: 5 Free, 0 Used

A "Start" button is located at the bottom right of the configuration form.

JUPYTER-JSC WEBSERVICE

System: JSC-Cloud

Custom Docker image on the JSC-Cloud

- Requirements:
 - Custom docker image must start a Jupyter server
- Examples:
 - <https://jupyter-docker-stacks.readthedocs.io/en/latest/using/selecting.html>

Repo2Docker (Binder) on the JSC-Cloud

- Requirements:
 - Repository compatible with repo2docker
- Details:
 - <https://repo2docker.readthedocs.io/en/latest/configuration/index.html>

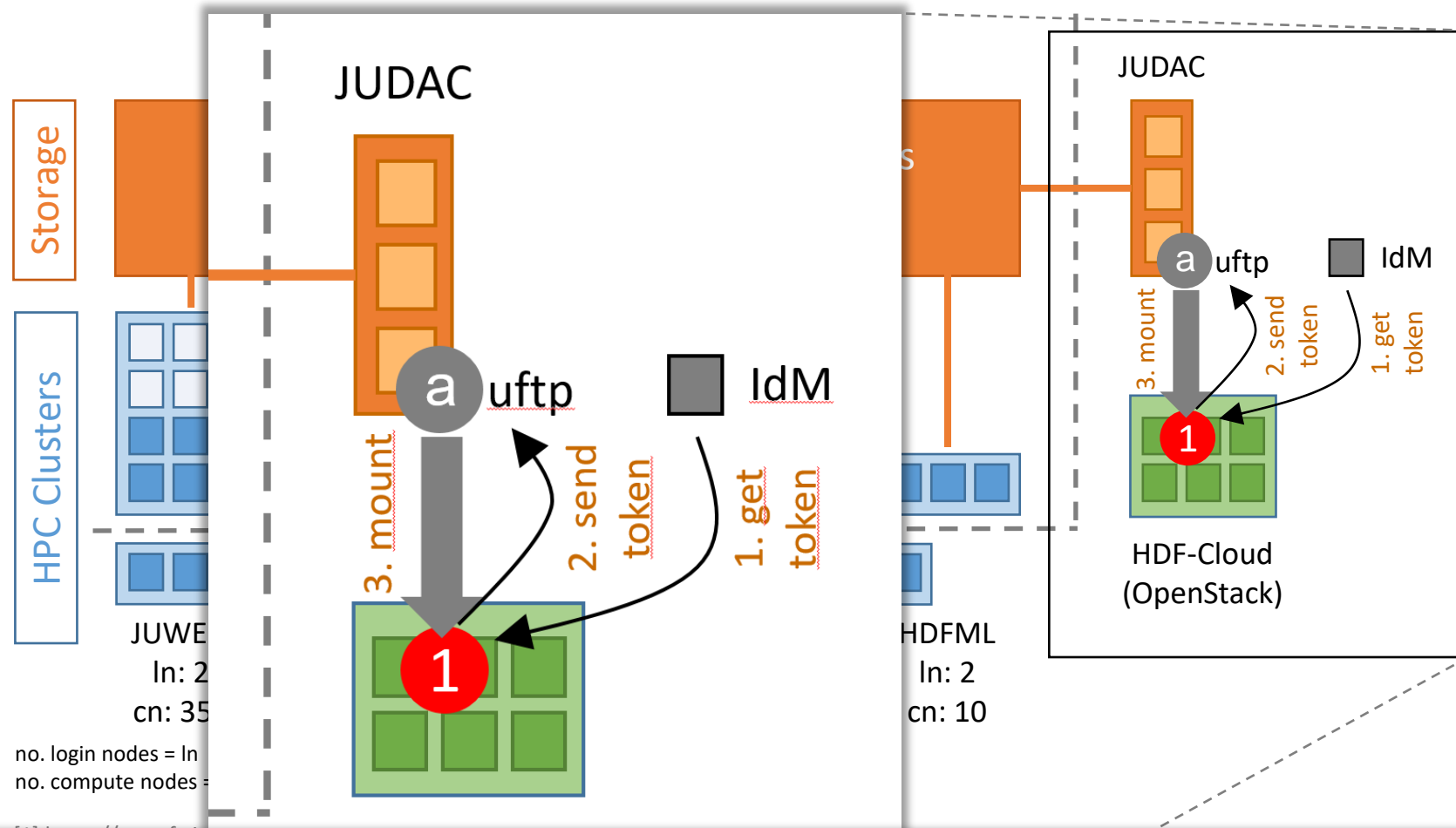
The screenshot displays the JupyterLab configuration interface on the JSC-Cloud website. The page title is "NEW JUPYTERLAB". The configuration form includes the following fields:

- Name:** myRepo2docker
- Version:** Repo2docker (Binder) - beta
- Repository:** GitHub
- GitHub repository name or URL:** GitHub repository name or URL
- Git ref (branch, tag, or commit):** HEAD
- Path to a notebook file (optional):** Path to a notebook file (optional)
- Notebook Type:** File
- System:** JSC-Cloud
- Flavor:** 4GB RAM, 1VCPUs, 2 days

The "Available Flavors" section shows a table with the following data:

Flavor	Free	Used	Limit exceeded
4GB RAM, 1VCPUs, 2 days	3	47	
2GB RAM, 1VCPU, 5 days	7		
8GB RAM, 2VCPUs, 10 hours	5		

HOW TO MOUNT GPFS ON HDF-CLOUD



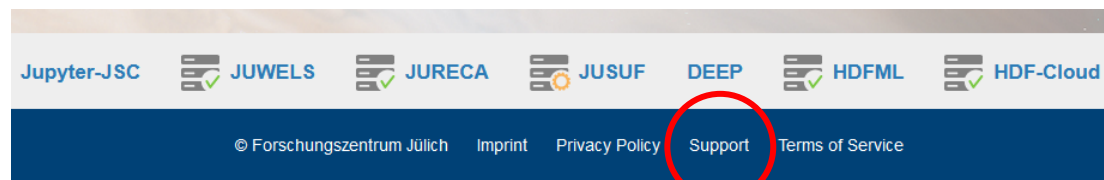
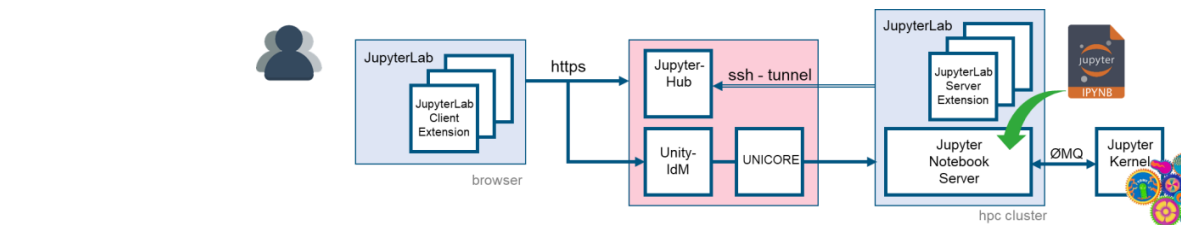
https://gitlab.jsc.fz-juelich.de/jupyter4jsc/training-2024.04-jupyter4hpc/-/blob/main/day2_hpcenv/7_cloud-hpc_challenges/1-hdf-cloud_mount-hpc-storage.ipynb

JUPYTER-JSC SECRETS

Very important to know

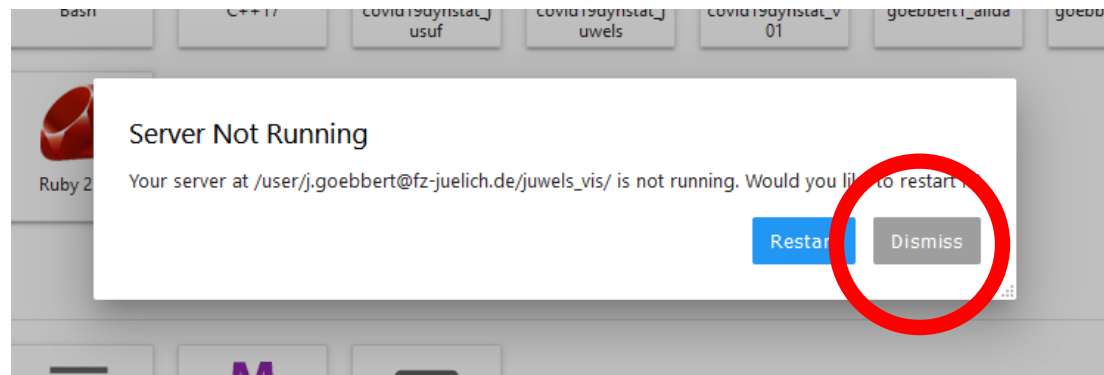
Secret 1: Support button

- Let us know, if something does not work.
We can only fix it, if we know it.

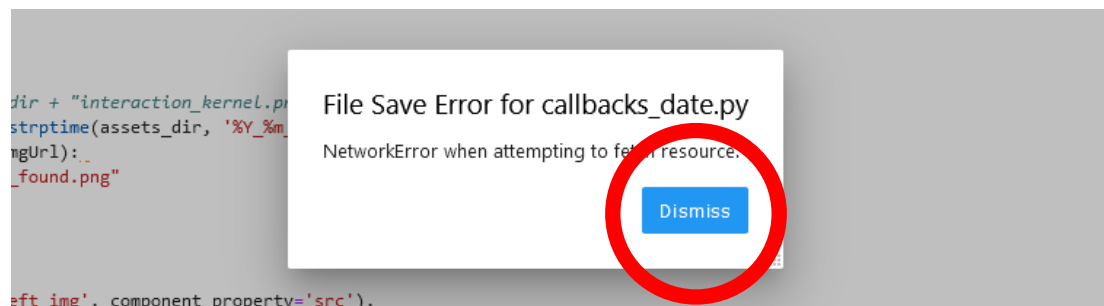


Secret 2: Reload on connection loss

- “Server Not Running”
means, that your browser just lost connection
=> **Just hit “Dismiss” !!!**
(as soon as you are online again)



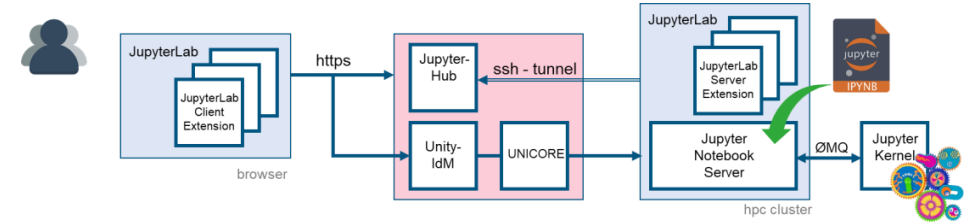
- “File Save Error for <...>”
means, that your browser just lost connection
=> **Just hit “Dismiss” !!!**
(as soon as you are online again)



You can **always** safely hit the “Reload” button of your browser, if the connection to JupyterLab ever gets lost.
(it will just restart JupyterLab on the browser-site)

JUPYTER-JSC SECRETS

For experts only 😊

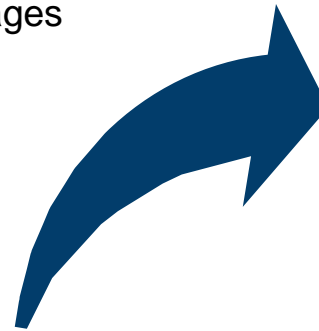


Secret 3: Jupyter-JSC logs

- Jupyter-Lab gets started by UNICORE on our HPC systems
- On startup UNICORE created the directory `$SCRATCH_<project>/unicore-jobs/<random-hash>/`
 - In the terminal of a running JupyterLab, this directory is `$JUPYTER_LOG_DIR`
- In this directory you find
 - `stdout` -> terminal output of jupyterlab messages
 - `stderr` -> terminal output of jupyterlab error messages
 - `.start` -> details how your JupyterLab got started

Secret 4: change to a different JupyterLab version

- In `.start` you can see, that
 - `$HOME/.jupyter/start_jupyter-jsc.sh` is used to prepare the environment for JupyterLab. This script must ensure that the command `jupyter` is available in `$PATH`.



```
#!/bin/bash
```

```
module purge  
module load Stages/2024  
module load GCCcore/.12.3.0  
module load Jupyter-bundle/20240520
```

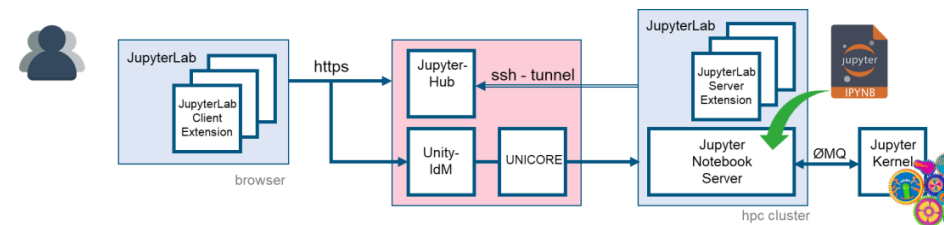
Switch to a customized JupyterLab with
`$HOME/.jupyter/start_jupyter-jsc.sh`

It enables you to switch to an older/newer/other version of JupyterLab, if the default one gives you trouble or is missing features.

https://gitlab.jsc.fz-juelich.de/jupyter4jsc/j4j_notebooks/-/blob/documentation/03-HowTos/Create_JupyterKernel_general.ipynb

JUPYTER-JSC WEBSERVICE

Some comments about the UI



Annotations on the JupyterLab interface:

- open filebrowser
- open launcher
- tutorials & examples
- sidebar with core and extensions features
- indicates active notebook cell
- type of active notebook cell
- no close, but go back to Jupyter-JSC's control panel
- memory consumption (keep an eye on that!)
- Type of Jupyter kernel this notebook is connected to (click to change)
- notebook cell

JUPYTERLAB EXTENSIONS

JUPYTERLAB EXTENSIONS

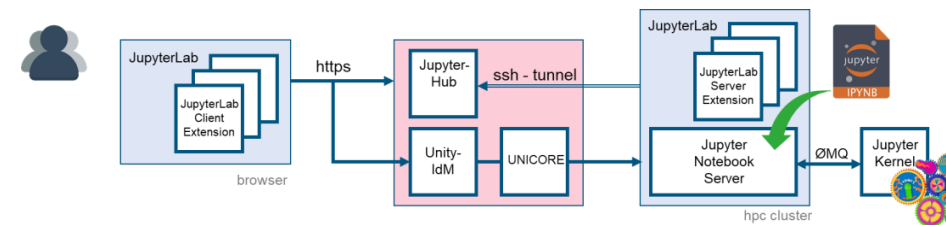
Some general information

List the installed JupyterLab extensions

- Open the Launcher
- Start a Terminal
- Run command `jupyter labextension list`

Extensions are installed in JupyterLab's Application Directory, which

- stores any information that JupyterLab persists
 - including settings and built assets of extensions
- default location is `<sys-prefix>/share/jupyter/lab`
- can be relocated by setting `$JUPYTERLAB_DIR`
 - contains the JupyterLab static assets
 - (e.g. `static/index.html`)
 - **JupyterLab < 3:**
any change requires a rebuild of the whole JupyterLab to take effect!
 - **JupyterLab >= 3:**
introduced prebuild extensions, which are loaded at startup time



```
[goebbert1@jlogin04 jureca]$ jupyter labextension list
JupyterLab v3.2.1
/p/software/jurecadc/stages/2020/software/Jupyter/2021.3.2-gccoreml-10.3.0-2021.2.0-Py
jupyterlab-iframe v0.4.0 enabled OK
jupyter-leaflet v0.14.0 enabled OK
ipyvolume v0.6.0-alpha.8 enabled OK
jupyterlab-system-monitor v0.8.0 enabled OK (python, jupyterlab-system-monitor)
jupyterlab-gitlab v3.0.0 enabled OK (python, jupyterlab-gitlab)
jupyterlab-topbar-extension v0.6.1 enabled OK (python, jupyterlab-topbar)
dask-labextension v5.1.0 enabled OK (python, dask_labextension)
jupyterlab-plotly v5.3.1 enabled OK
jupyter-vue v1.6.1 enabled OK
...
Other labextensions (built into JupyterLab)
app dir: /p/software/jurecadc/stages/2020/software/Jupyter/2021.3.2-gccoreml-10.3.0-2021.2.0-python-3.8.5/share/jupyter/lab
jupyterlab-dash v0.4.0 enabled OK
jupyterlab-theme-toggle v0.6.1 enabled OK
[goebbert1@jlogin04 jureca]$
```

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

Hint: JupyterLab Playground

A JupyterLab extension to write and load simple JupyterLab plugins inside JupyterLab.

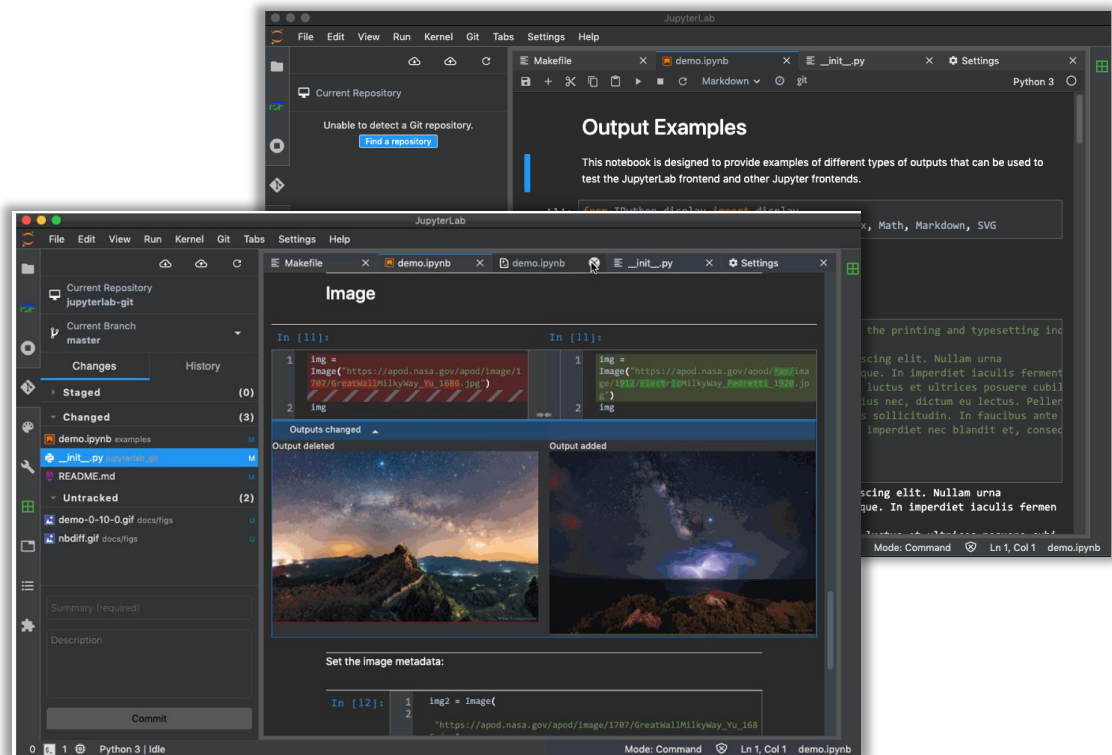
<https://github.com/jupyterlab/jupyterlab-plugin-playground>

JUPYTERLAB EXTENSIONS

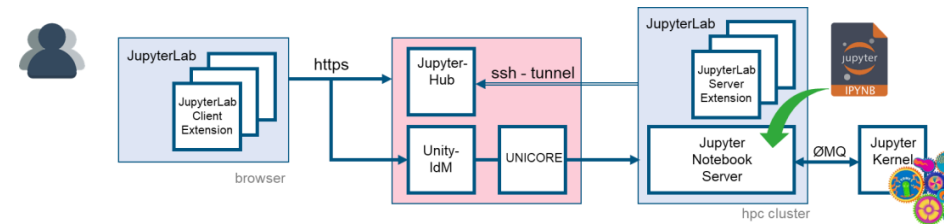
Installed by default at Jupyter-JSC

JupyterLab-Git

JupyterLab extension for version control using Git

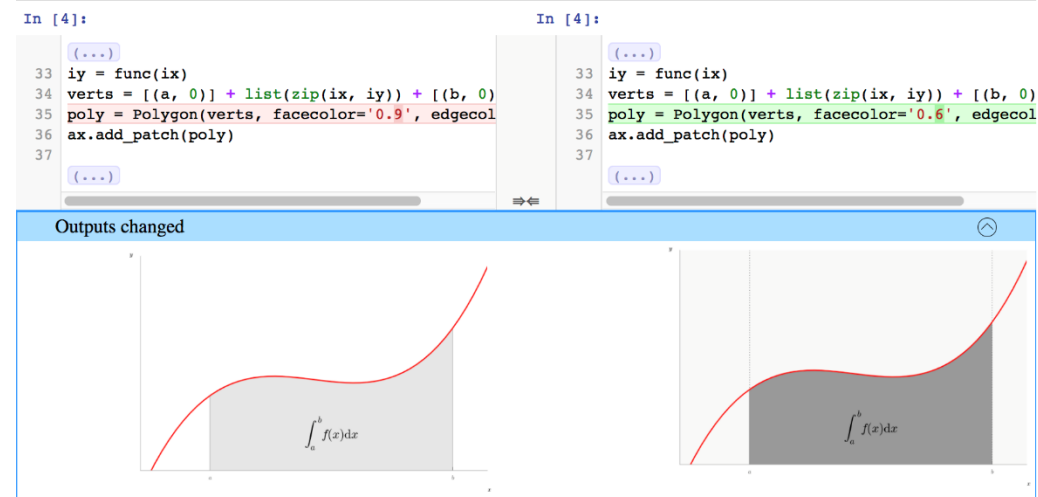


<https://github.com/jupyterlab/jupyterlab-git>



NBDime

Tools for diffing and merging of Jupyter notebooks.



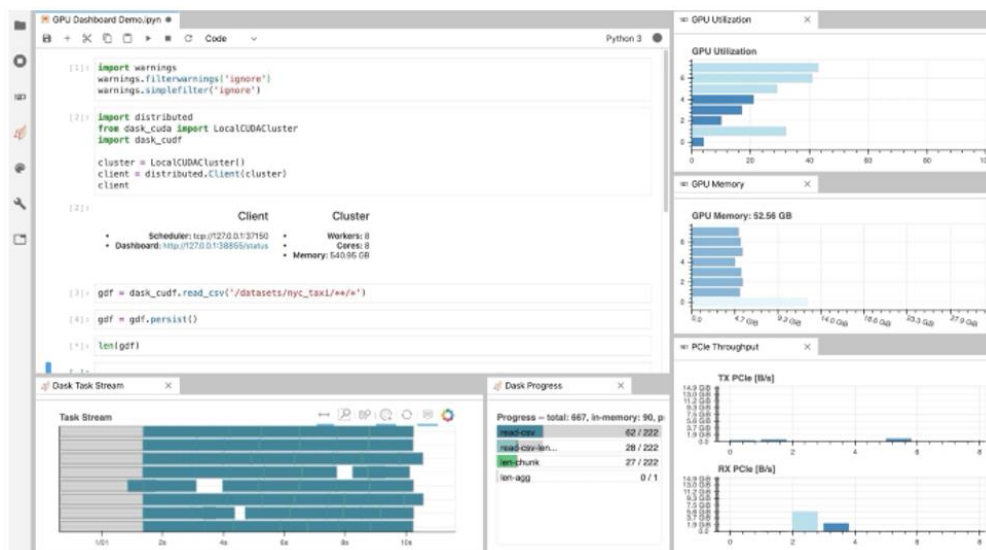
<https://github.com/jupyter/nbdime>

JUPYTERLAB EXTENSIONS

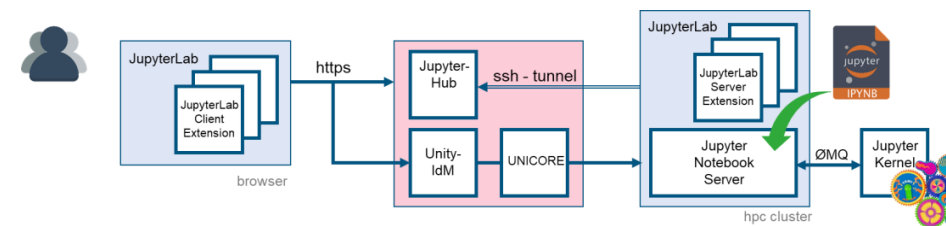
Installed by default at Jupyter-JSC

NVDashboard

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

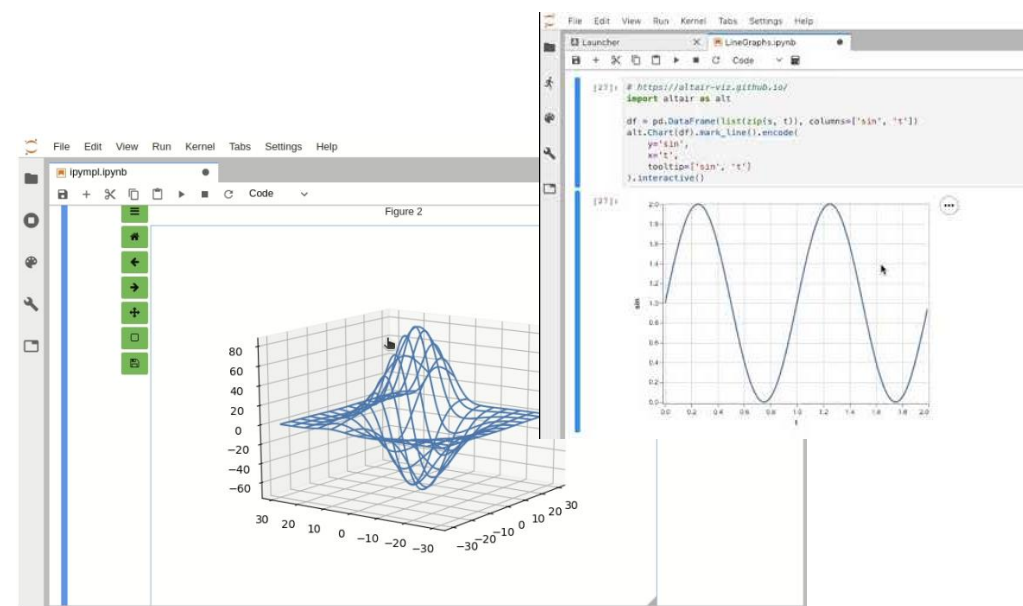


<https://github.com/rapidsai/jupyterlab-nvdashboard>
<https://developer.nvidia.com/blog/gpu-dashboards-in-jupyter-lab/>



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>

JUPYTERLAB EXTENSIONS

Installed by default at Jupyter-JSC

Core packages	Version	Link	Description
jupyterlab	4.2.1	https://jupyterlab.readthedocs.io	-
notebook	7.2.0	https://jupyter-notebook.readthedocs.io	Jupyter Notebook as a Jupyter Server extension
jupyterlab_server	2.27.1	https://jupyterlab-server.readthedocs.io	Server components for JupyterLab applications
jupyterhub	4.1.5	https://jupyterhub.readthedocs.io	Multi-user server for Jupyter notebooks

Optional extension	Version	Link	Description
jupyterlab-nvdashboard	0.10.0	https://github.com/rapidsai/jupyterlab-nvdashboard , eb-file	A JupyterLab extension for displaying dashboards of GPU usage.
jupyter-slurm-provisioner	0.6.0	https://github.com/FZJ-JSC/jupyter-slurm-provisioner , eb-file	Allows to start Jupyter kernels as a SLURM job remote from the Jupyter server
nglview	3.1.2	http://nglviewer.org/nglview/latest/ , eb-file	Jupyter widget to interactively view molecular structures and trajectories
jupyter-ai	2.15.0	https://jupyter-ai.readthedocs.io/ , eb-file	A generative AI extension for JupyterLab

Core Kernels	Version	Link	Description
Bash	0.9.3	https://github.com/takuyver/bash_kernel , eb-file	A bash kernel for IPython
Cling (C++)	20231018	https://github.com/root-project/cling/ , eb-file	Jupyter kernel for the C++ programming language
Julia	1.9.5	https://github.com/JuliaPy/pyjulia , eb-file	python interface to julia
LFortran	0.30.0	https://lfortran.org/ , eb-file	Modern interactive LLVM-based Fortran compiler
Octave	8.4.0	https://www.octave.org/ , eb-file	Scientific Programming Language - Powerful mathematics-oriented syntax with built-in 2D/3D plotting and visualization tools
R	4.3.2	https://irkernel.github.io , eb-file	R kernel for Jupyter
Ruby	3.2.2	https://github.com/SciRuby/iruby , eb-file	Ruby kernel for Jupyter

Community Kernels	Version	Link	Description
DeepLearning	2024.3	eb-file	Python kernel incl. a collection of extra modules/packages for Deep Learning
PyEarthSystem	2024.3	eb-file	Python kernel incl. a collection of extra modules/packages for the Earth System community
QuantumComputing	2024.5	eb-file	Python kernel incl. a collection of extra modules/packages for the quantum computing community
Visualization	2024.3	eb-file	Python kernel incl. a collection of extra modules/packages for visualization

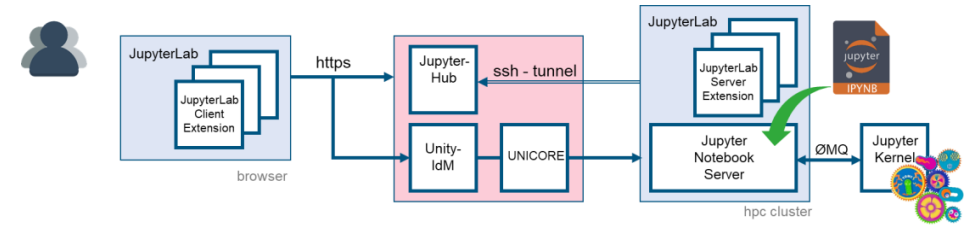
JupyterLab Applications	Version	Link	Description
Xpra	5.0.8	https://xpra.org eb-file	Remote desktop for X11 applications in the browser

Core Extensions	Version	Link	Description
jupyter-server-proxy	4.1.2	https://jupyter-server-proxy.readthedocs.io	Jupyter notebook server extension to proxy web services.
jupyterlab-lsp	5.1.0	https://github.com/jupyter-lsp/jupyterlab-lsp	Coding assistance for JupyterLab using Language Server Protocol
ipympi	0.9.4	https://matplotlib.org/ipympi/	Interactive features of matplotlib in Jupyter
ipyleaflet	0.19.1	https://ipyleaflet.readthedocs.io	Interactive maps in the Jupyter notebook
bqplot	0.13.0rc0	https://bqplot.github.io/bqplot/	Plotting library for IPython/Jupyter notebooks
jupyterlab_gitlab	4.0.0	https://github.com/jupyterlab-contrib/jupyterlab-gitlab	A JupyterLab extension for browsing GitLab repositories
jupyterlab_git	0.50.1	https://github.com/jupyterlab/jupyterlab-git	A Git extension for JupyterLab
nbdime	4.0.1	https://nbdime.readthedocs.io/	Tools for diffing and merging of Jupyter notebooks.
jupyterlab_latex	4.0.0	https://github.com/jupyterlab/jupyterlab-latex	JupyterLab extension for live editing of LaTeX documents
plotly	5.22.0	https://plotly.com/python/	Python graphing library for interactive, publication-quality graphs.
jupyter_bokeh	4.0.4	https://github.com/bokeh/jupyter_bokeh	An extension for rendering Bokeh content in JupyterLab notebooks
panel	1.3.8	https://panel.holoviz.org/	The powerful data exploration & web app framework for Python
holoviews	1.18.3	https://holoviews.org/	With Holoviews, your data visualizes itself.
jupyterlab_h5web	12.1.0	https://github.com/silx-kit/jupyterlab-h5web	Open and explore HDF5 files in JupyterLab. Can handle very large (TB) sized files, and datasets of any dimensionality
ipyparallel	8.8.0	https://ipyparallel.readthedocs.io	IPython Parallel: Interactive Parallel Computing in Python
dask_labextension	7.0.0	https://github.com/dask/dask-labextension	JupyterLab extension for Dask
voila	0.5.7	https://voila.readthedocs.io	Voilà turns Jupyter notebooks into standalone web applications
nbdev	2.3.25	https://nbdev.fast.ai/	Create delightful software with Jupyter Notebooks
sidecar	0.7.0	https://github.com/jupyter-widgets/jupyterlab-sidecar	A sidecar output widget for JupyterLab
dash	2.17.0	https://plotly.com/dash	Data Apps & Dashboards for Python. No JavaScript Required.
jupyterlab-spellchecker	0.8.4	https://github.com/jupyterlab-contrib/spellchecker	Spellchecker for JupyterLab notebook markdown cells and file editor.
jupyterlab-favorites	3.2.1	https://github.com/NERSC/jupyterlab-favorites	Add the ability to save favorite folders to JupyterLab for quicker browsing
jupyterlab-resource-usage	1.0.2	https://github.com/jupyter-server/jupyter-resource-usage	JupyterLab extension to for monitoring your own resource Usage
jupyterlab-tour	4.0.1	https://github.com/jupyterlab-contrib/jupyterlab-tour	A JupyterLab UI tour built on jupyterlab-tutorial and react-joyride.
papermill	2.5.0	https://papermill.readthedocs.io	Parameterize, execute, and analyze notebooks
pyunicore	1.0.0	https://github.com/HumanBrainProject/pyunicore	UNICORE REST bindings for python

JUPYTER KERNEL

JUPYTER KERNEL

How to create your own Jupyter Kernel



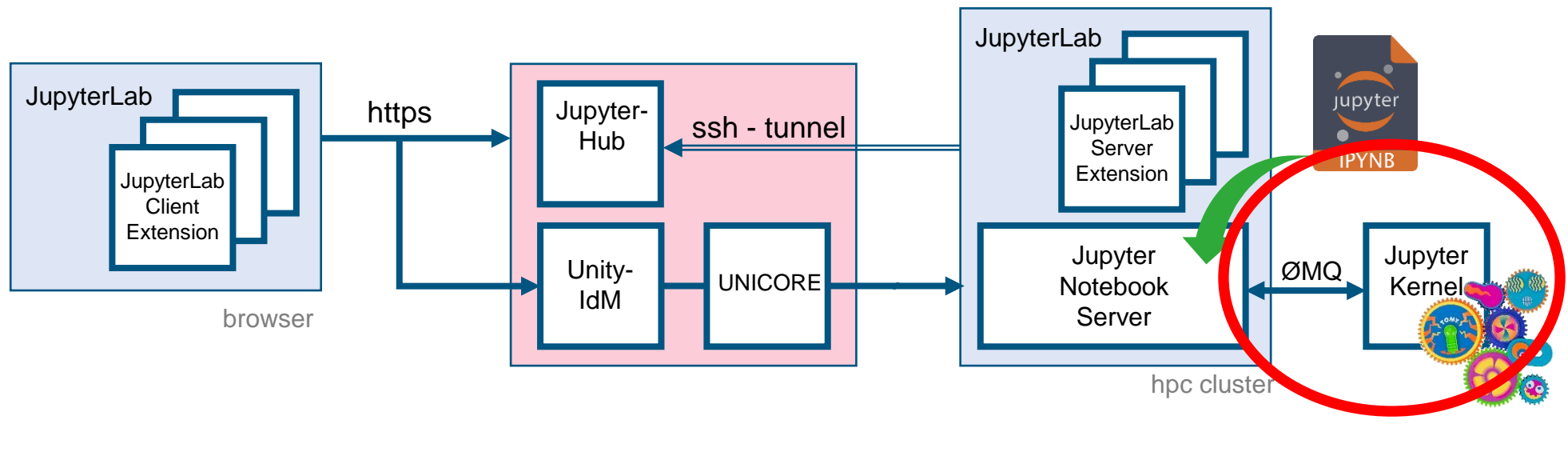
Jupyter Kernel

A “kernel” refers to the separate process

w/

Ju

-
-
-
-



You can easily **create your own kernel** which for example runs your specialized virtual Python environment.

<https://github.com/jupyter/jupyter/wiki/Jupyter-kernels>

JUPYTER KERNEL

How to create your own Jupyter Kernel

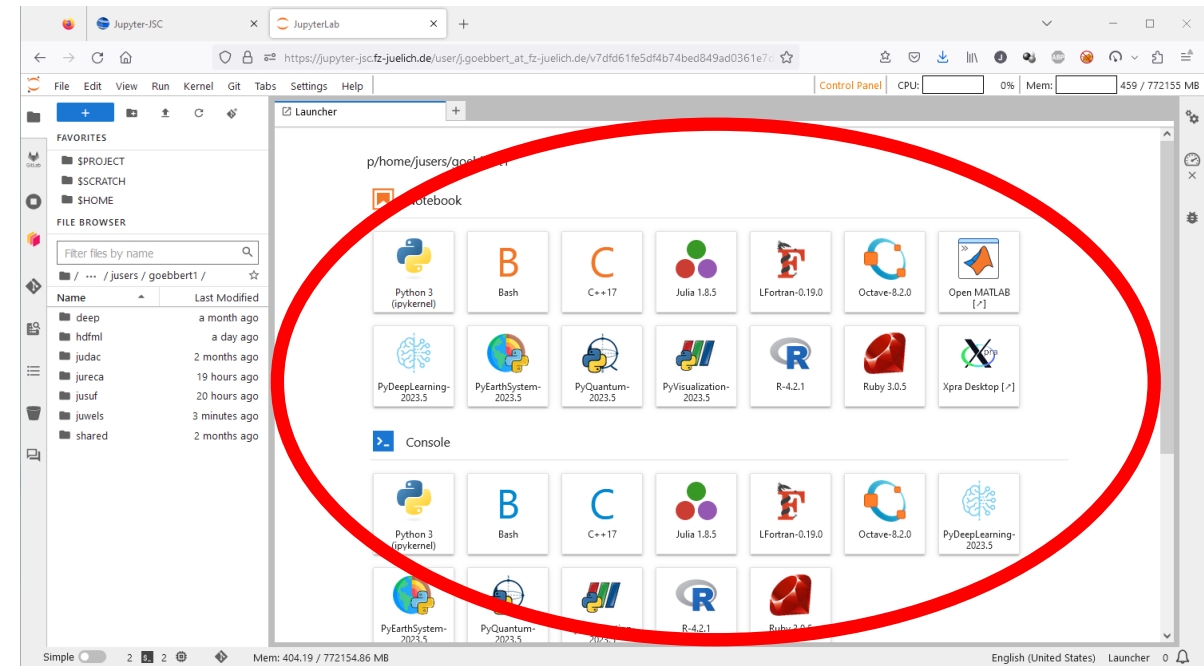
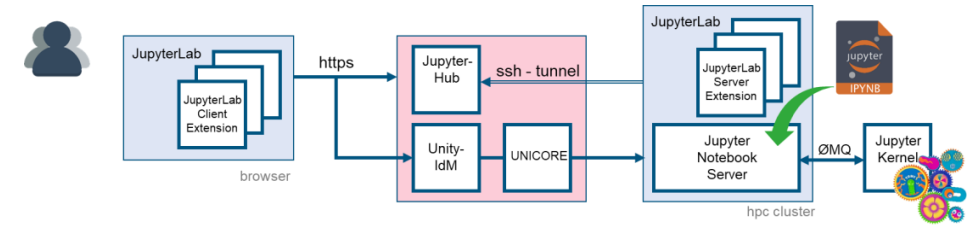
Jupyter Kernel

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://github.com/jupyter/jupyter/wiki/Jupyter-kernels>

JUPYTER KERNEL

How to create your own Jupyter Kernel

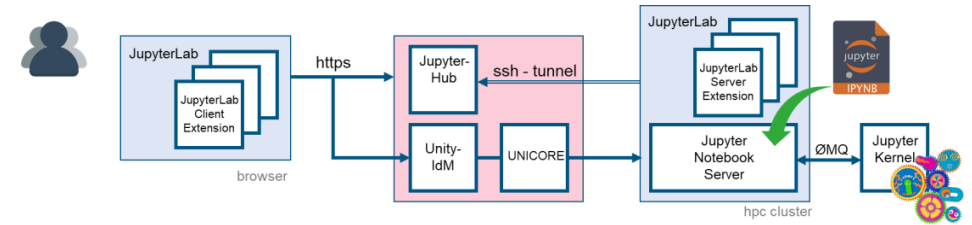
Jupyter Kernel

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.



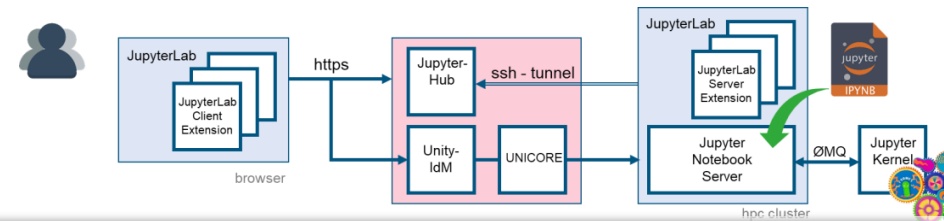
Building your own Jupyter kernel is a three step process

1. Create/Pimp new **virtual Python environment**
`venv`
2. Create/Edit **launch script** for the Jupyter kernel
`kernel.sh`
3. Create/Edit Jupyter **kernel configuration**
`kernel.json`

<https://github.com/jupyter/jupyter/wiki/Jupyter-kernels>

JUPYTER KERNEL

How to create your own 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 Jupyter Notebook Server

The screenshot shows a JupyterLab interface. On the left, a file browser displays a directory structure with files like 'Create_JupyterKernel_conda.ipynb', 'Create_JupyterKernel_general.ipynb' (highlighted), 'Create_JupyterKernel_pyenv.ipynb', and others. The main editor window shows a notebook titled 'Create your own Jupyter Kernel'. The notebook content includes an introduction, an attention box stating 'This notebook is meant to run out of a JupyterLab on JSC's HPC systems.', and a three-step process for building a kernel: 1. Create/Pimp new virtual Python environment (venv), 2. Create/Edit launch script for the Jupyter kernel (kernel.sh), and 3. Create/Edit Jupyter kernel configuration (kernel.json). The 'Settings' section is partially visible, showing 'Set the kernel name' with a note 'must be lower case'.

- Create_JupyterKernel_conda.ipynb
- Create_JupyterKernel_general.ipynb**
- Create_JupyterKernel_pyenv.ipynb

<https://github.com/jupyter/jupyter/wiki/Jupyter-kernels>

https://gitlab.version.fz-juelich.de/jupyter4jsc/j4j_notebooks/-/blob/master/001-Jupyter/Create_JupyterKernel_general.ipynb

JUPYTER KERNEL

Run your Jupyter kernel configuration

Run your Jupyter Kernel

1. <https://jupyter-jsc.fz-juelich.de>
2. Choose system where your Jupyter kernel is installed in `~/ .local/share/jupyter/kernels`
3. Select your kernel in the launch pad or click the kernel name.

One of the many alternatives: Conda

Base your Jupyter Kernel on a Conda environment.

https://gitlab.version.fz-juelich.de/jupyter4jsc/j4j_notebooks/-/blob/master/001-Jupyter/Create_JupyterKernel_conda.ipynb

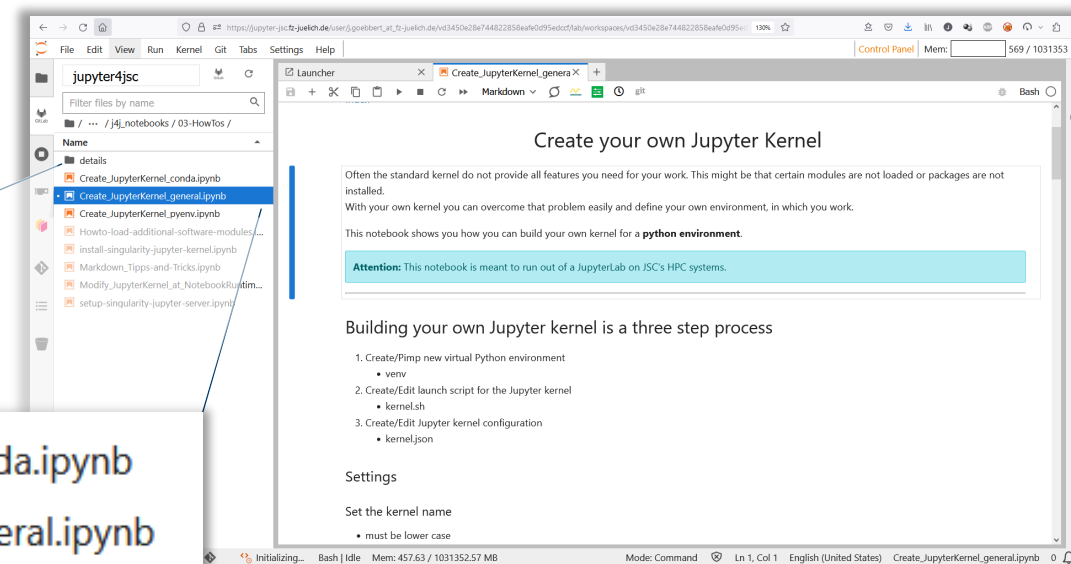
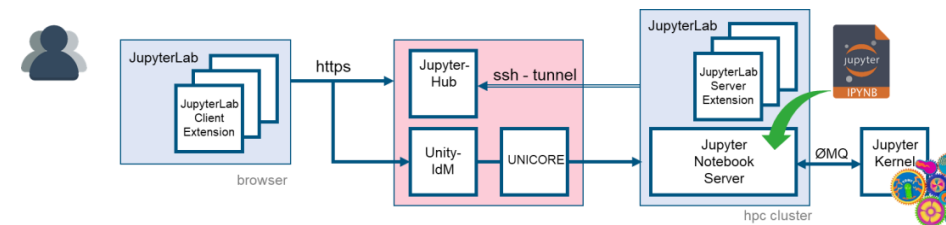
Your kernel is **independent of the software stage** in which JupyterLab is located.

Jupyter kernel are **NOT limited** to Python at all!

The kernel-endpoint just needs to talk the Jupyter's kernel protocol (in general over ZeroMQ).

E.g.

- IRkernel for R (<https://github.com/IRkernel/IRkernel>)
- IJulia.jl (<https://github.com/JuliaLang/IJulia.jl>)

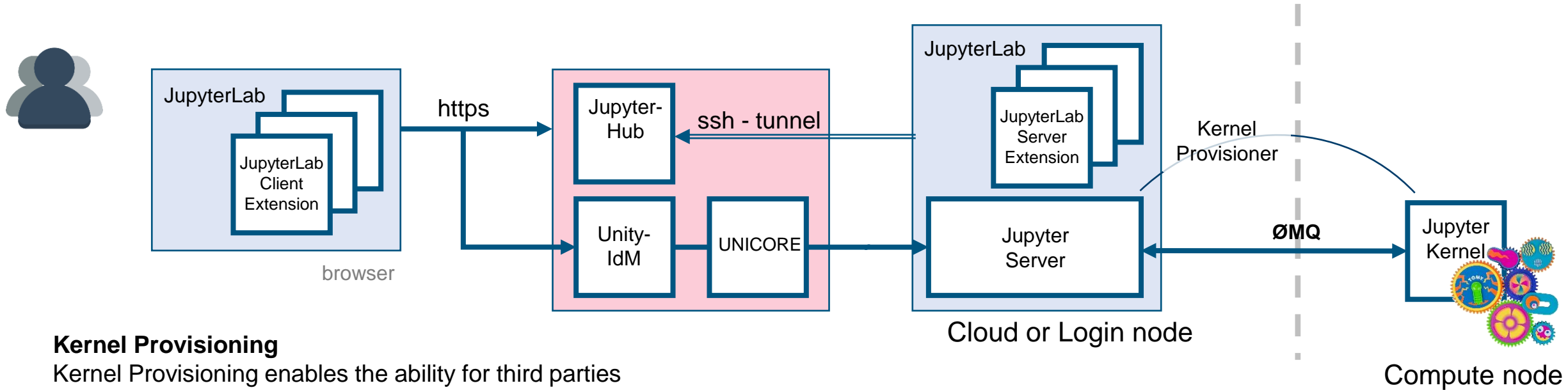


- 📄 Create_JupyterKernel_conda.ipynb
- 📄 Create_JupyterKernel_general.ipynb
- 📄 Create_JupyterKernel_pyenv.ipynb

SLURM WRAPPED KERNELS WITH SLURM-PROVISIONER

REMOTE JUPYTER KERNELS

Running multiple Jupyter kernels separate on the HPC system



Kernel Provisioning

Kernel Provisioning enables the ability for third parties to **manage the lifecycle of a kernel's runtime environment**.

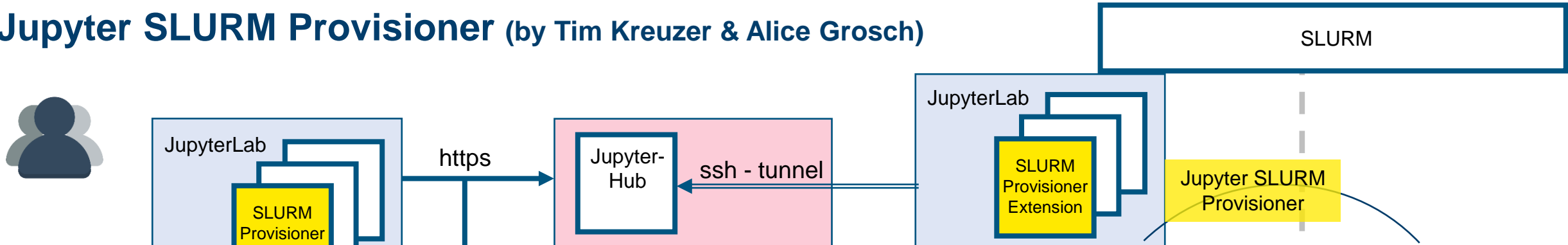
By implementing and configuring a *kernel provisioner*, third parties have the ability to **provision kernels for different environments**, typically managed by resource managers like Kubernetes, Hadoop YARN, Slurm, etc.

The kernel provisioner optionally extends the current **metadata stanza within the kernel.json** to include the specification of the kernel provisioner name, along with an optional config stanza

```
[..]  
"metadata": {  
  "kernel_provisioner": {  
    "provisioner_name": "slurm-provisioner",  
    "config": {  
      "kernel_argv": "Python",  
      "project": "zam",  
      "partition": "batch",  
      "nodes": 1,  
      "runtime": 3600,  
    }  
  }  
},
```


REMOTE JUPYTER KERNELS

Jupyter SLURM Provisioner (by Tim Kreuzer & Alice Grosch)



Slurm wrapped kernels allow you to run kernels on compute nodes while your Jupyter Server runs on a login node.

This has the advantage that when your allocation on the compute node(s) ends, **only the kernel is stopped**, but your JupyterLab server keeps running. You will only have to restart the kernel, not your entire JupyterLab instance.



```
[4]: !hostname
jsfc080
[ ]:
```

```
kreuzer1@jsf102:~/slurm-pr...
[kreuzer1@jsf102 slurm-provisioner]$ hostname
jsf102.jusuf
[kreuzer1@jsf102 slurm-provisioner]$
```

```
[goebbert1@jsf101 jusuf]$ jupyter kernelspec provisioners
Available kernel provisioners:
local-provisioner      jupyter_client.provisioning:LocalProvisioner
slurm-provisioner     jupyter_slurm_provisioner:SlurmProvisioner
```

<https://github.com/FZJ-JSC/jupyter-slurm-provisioner>
<https://github.com/FZJ-JSC/jupyter-slurm-provisioner-extension>

REMOTE HYBRID KERNELS

Jupy

The screenshot shows the JupyterLab interface. The browser address bar displays the URL: `https://jupyter-jsc.fz-juelich.de/user/j.goebbert_at_fz-juelich.de/be5f53dfad7b4b4f91b2f5ac9c6342d8/lab/workspaces/be5f53dfad7b4b4f91b2f5ac9c6342d8`. The interface includes a top menu bar with options like File, Edit, View, Run, Kernel, Git, Tabs, Settings, and Help. A status bar at the top right shows CPU usage at 0% and memory usage at 196 / 257468 MB. On the left, there is a sidebar with sections for 'Current Configuration' (with a 'Configure' button), 'Kernel Allocations' (stating 'There are no allocations available.'), and a 'Configure Slurm' panel. The main area is titled 'Launcher' and shows the path `p/home/jusers/goebbert1/jusuf`. Below this, a 'Notebook' section displays a grid of 24 kernel options, each with an icon and a label: Python 3 (ipykernel), Bash, C++17, covid19dynstat_j usuf, covid19dynstat_j uwels, esm2022_kernel, goebbert1_kerne l_jsonmerge, goebbert1_pyferr et, goebbert1_works hop2, JavaScript (Node.js), Julia 1.7.1, my_env, Octave-6.4.0, PyDeepLearning-1.1, PyQuantum-3.0, PyVisualization-1.0, R-4.1.2, Ruby 3.0.1, Slurm Wrapper, unseen_kernel, and Xpra Desktop [↗]. A 'Try the Welcome Tour.' notification is visible in the bottom right corner. The bottom status bar shows 'Mem: 195.93 / 257467.88 MB' and 'English (United States) Launcher'.

Configure Slurm
Select allocation for s
New
Select kernel for slurm
Custom Python
Select project for slurm
ccstvs
Select partition for slurm
develgpu
Nodes [1-2]:
GPUs [1-4]:
Runtime (min) [10-12]

Jupyter
Kernel
compute node

RECORDED WITH
SCREENCAST
Simple

Mem: 195.93 / 257467.88 MB

Try the Welcome Tour. X
Start now Don't show me again

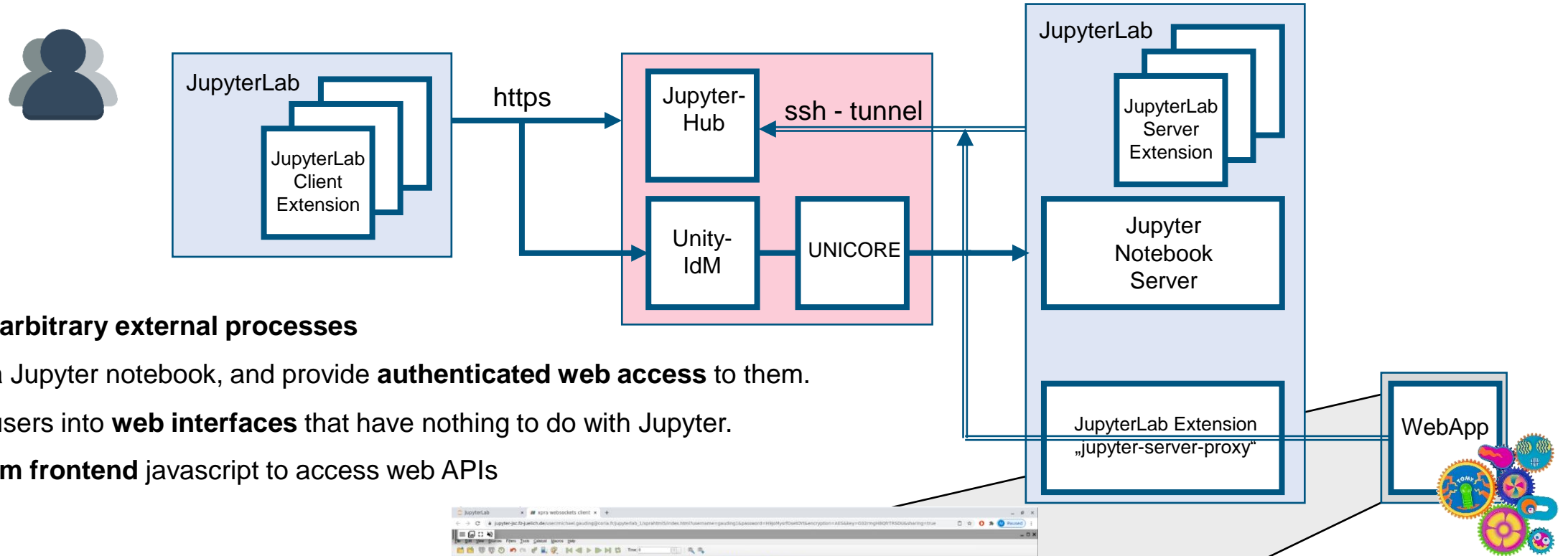
<https://github.com/FZJ-JSC/jupyter-slurm-provisioner>
<https://github.com/FZJ-JSC/jupyter-slurm-provisioner-extension>

slurm-provisioner jupyter_slurm_provisioner:SlurmProvisioner
Forschungszentrum

JUPYTER SERVER PROXY

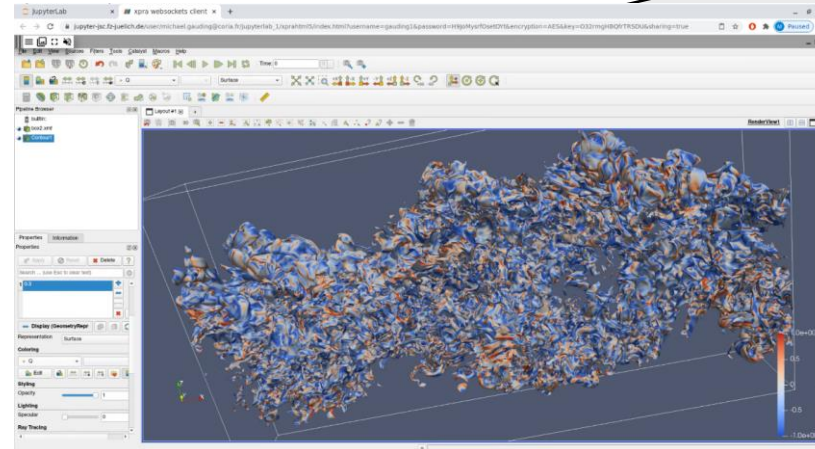
JUPYTERLAB – WEBSERVICE PROXY

Extension: jupyter-server-proxy



Allows to run **arbitrary external processes**

- alongside a Jupyter notebook, and provide **authenticated web access** to them.
- launching users into **web interfaces** that have nothing to do with Jupyter.
- **access from frontend javascript** to access web APIs

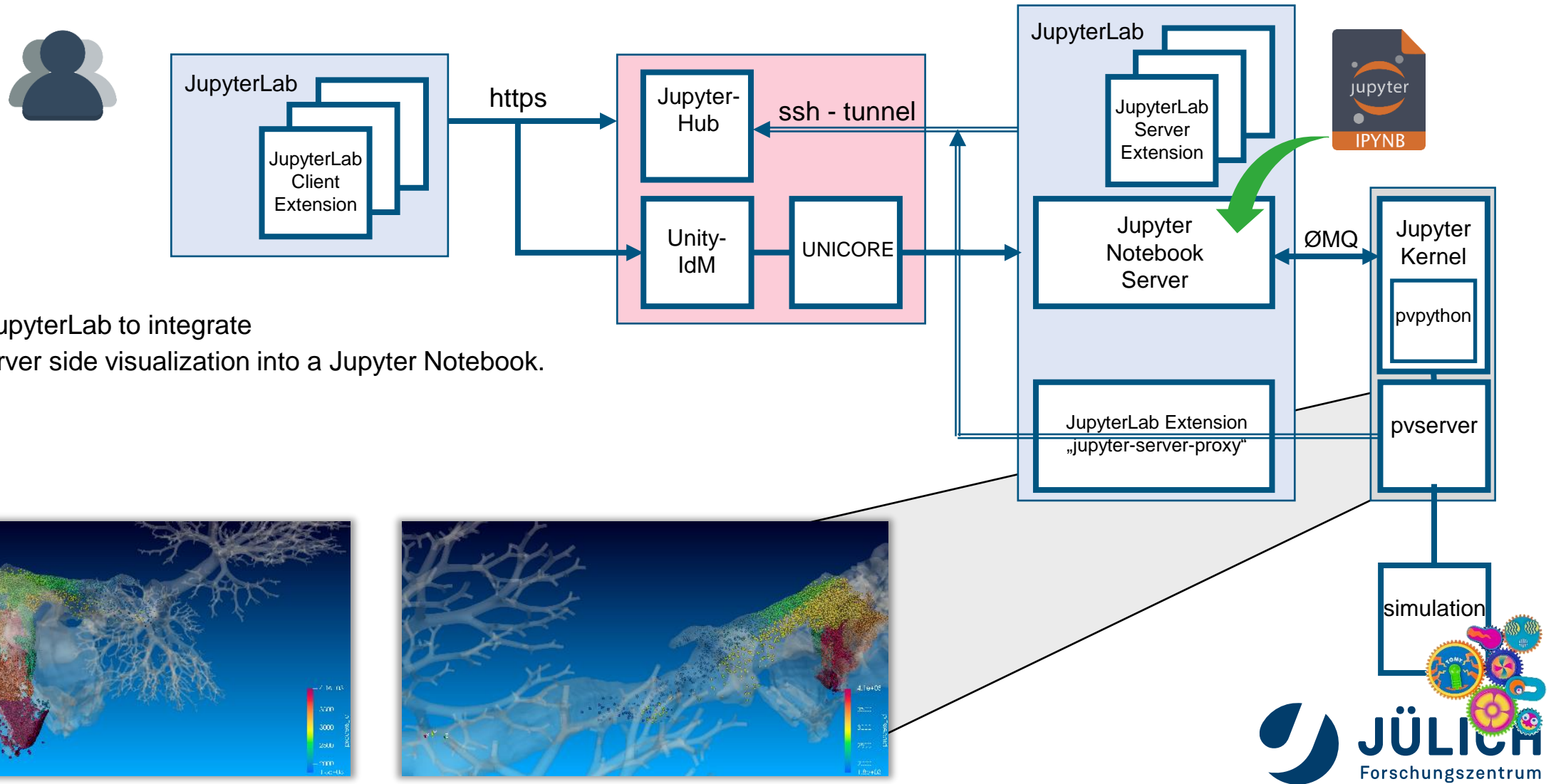


Turbulent mixing with variable density, subset of 1939x600x3584 grid points, Michael Gauding, CORIA

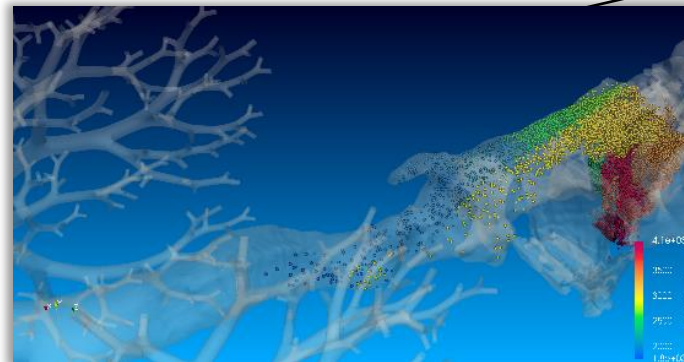
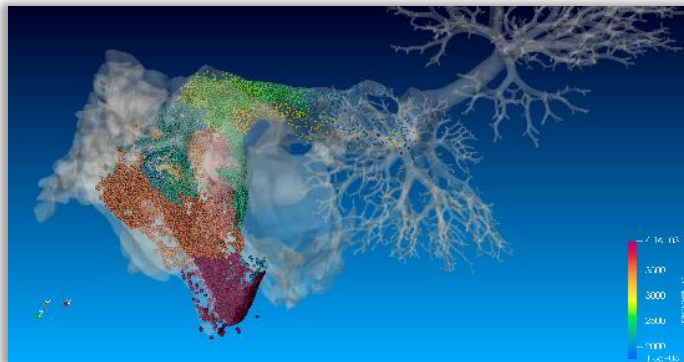
<https://github.com/jupyterhub/jupyter-server-proxy>

JUPYTERLAB – WEBSERVICE PROXY

Extension: jupyter-server-proxy



How to use JupyterLab to integrate interactive server side visualization into a Jupyter Notebook.



PORT TUNNELING – WEBSERVICE PROXY

Extension: jupyter-server-proxy

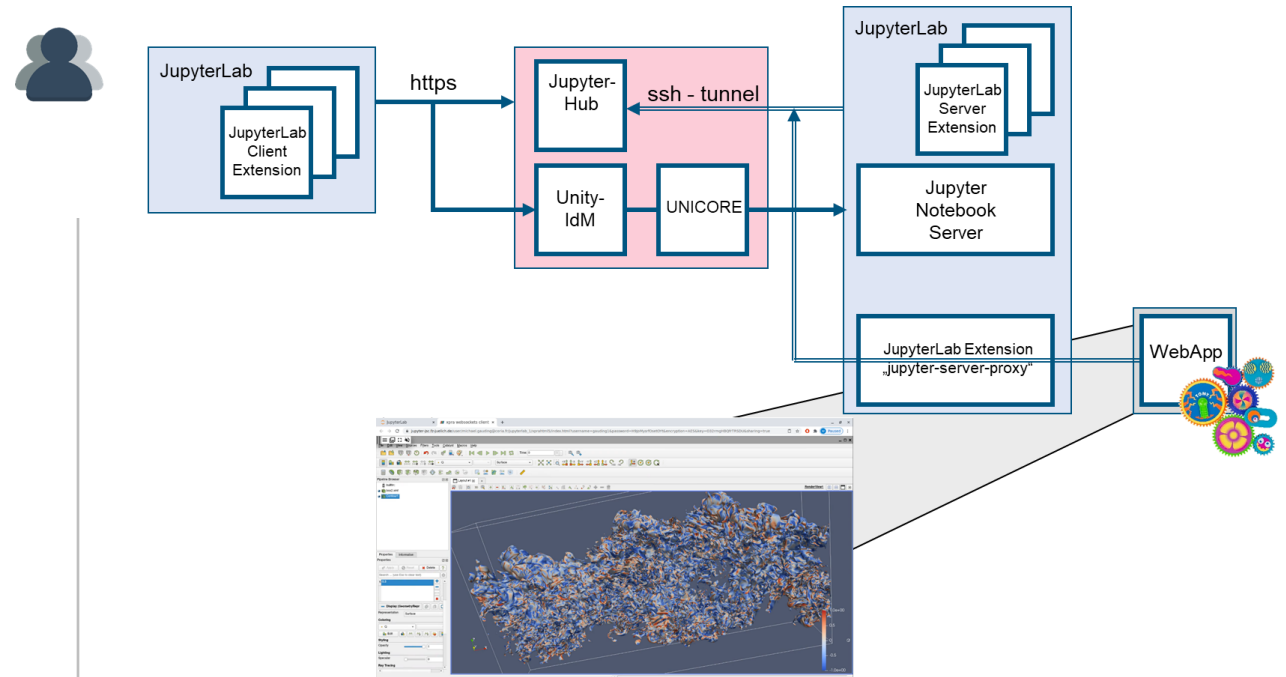
Accessing Arbitrary Ports or Hosts from the Browser

If you have a web-server running on the server listening on <port>, you can access it through the notebook at **<notebook-base>/proxy/<port>**

The URL will be rewritten to remove the above prefix.

You can disable URL rewriting by using **<notebook-base>/proxy/absolute/<port>** so your server will receive the full URL in the request.

This works for all ports listening on the local machine.



Example:

`https://jupyter-jsc.fz-juelich.de/user/j.goebbert@fz-juelich.de/juwels_login/proxy/<port>`

`https://jupyter-jsc.fz-juelich.de/user/j.goebbert@fz-juelich.de/juwels_login/proxy/<host>:<port>`

Upcoming: Support proxying to a server process via a Unix socket (#337)

<https://jupyter-server-proxy.readthedocs.io/en/latest/arbitrary-ports-hosts.html>

JUPYTER SERVER PROXY EXAMPLES

JUPYTERLAB – REMOTE DESKTOP

Run your X11-Applications in the browser

Jupyter-JSC gives you easy access to a remote desktop

1. <https://jupyter-jsc.fz-juelich.de>
2. Click on “Xpra”

Xpra - X Persistent Remote Applications

is a tool which runs X clients on a remote host and directs their display to the local machine.

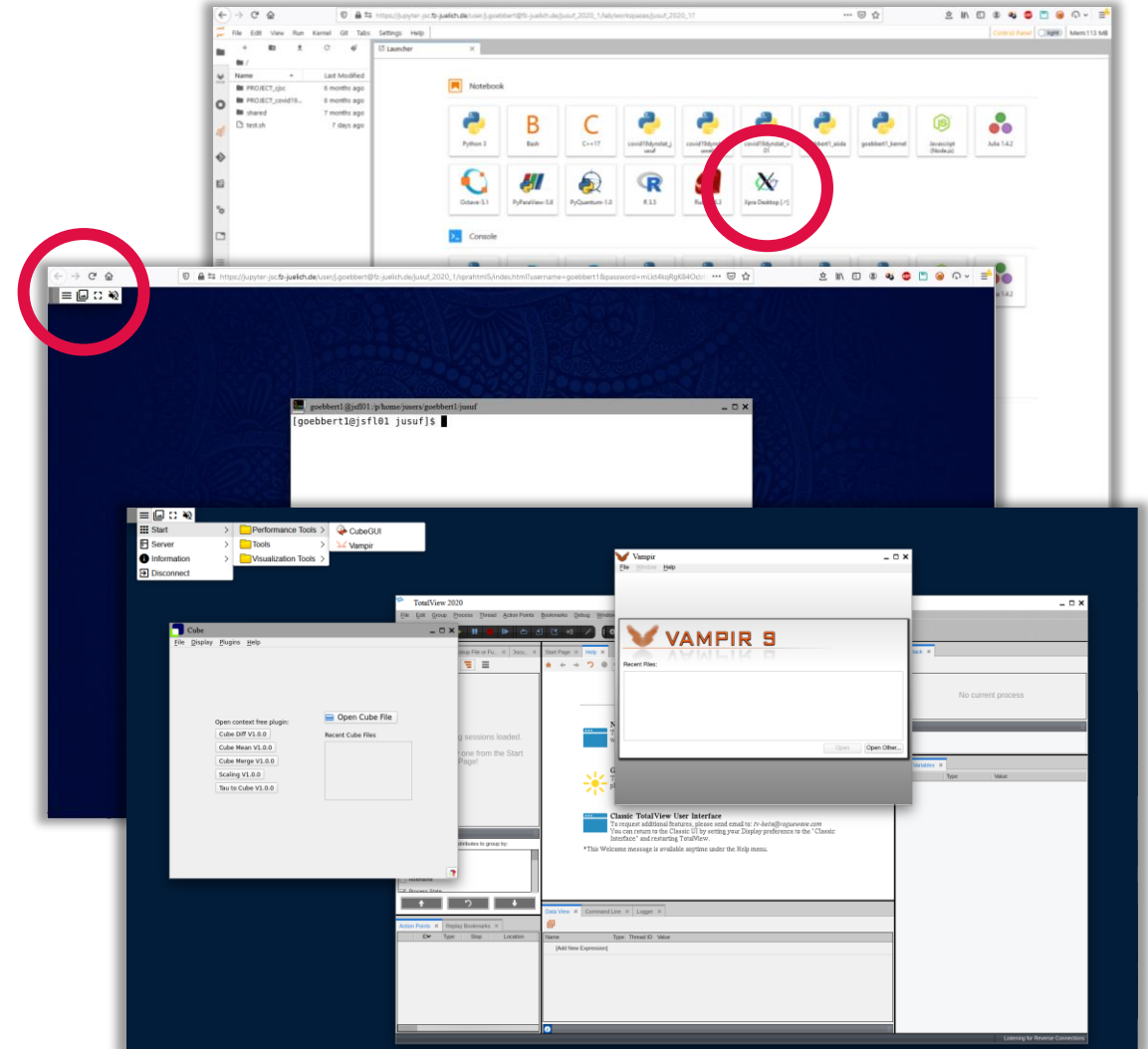
- Runs in a browser
- allows dis-/reconnection without disrupting the forwarded application
- <https://xpra.org>

The remote desktop will run on the same node as your JupyterLab does (this includes compute nodes).

It gets killed, when you stop your JupyterLab session.

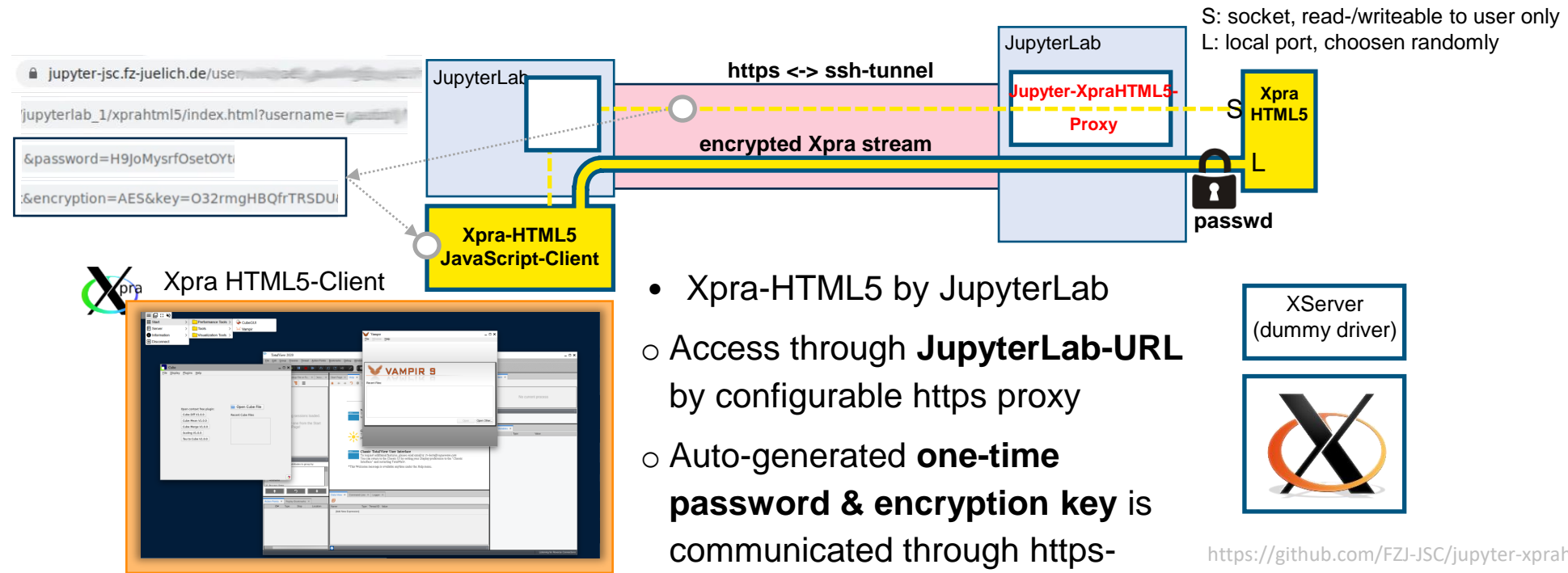
Hint:

- CTRL + C -> CTRL + Insert
- CTRL + V -> SHIFT + Insert



JUPYTERLAB – REMOTE DESKTOP

Run your X11-Applications in the browser



- Xpra-HTML5 by JupyterLab
- Access through **JupyterLab-URL** by configurable https proxy
- Auto-generated **one-time password & encryption key** is communicated through https-proxy

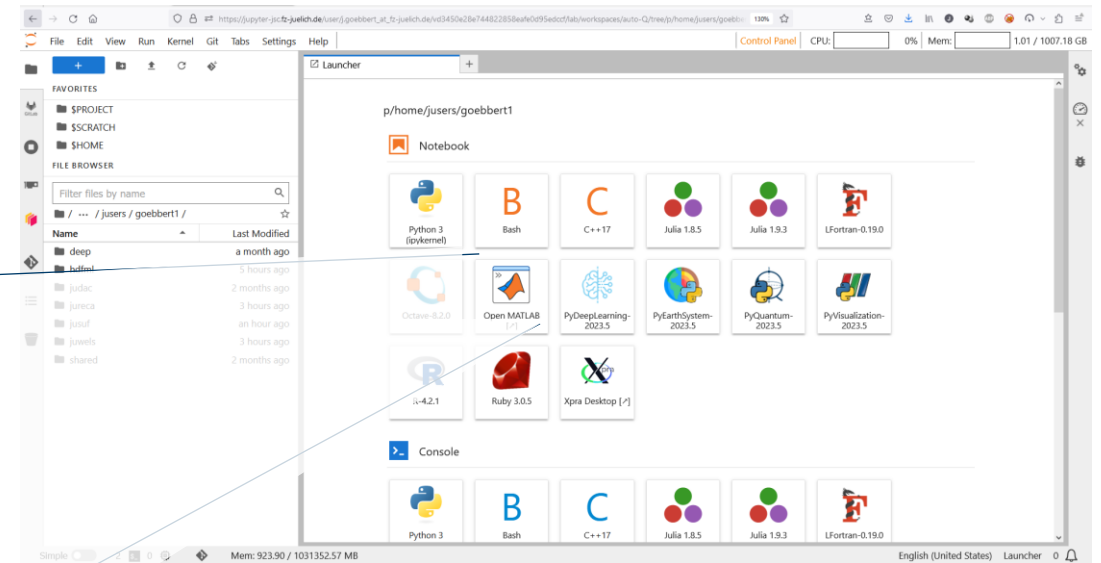
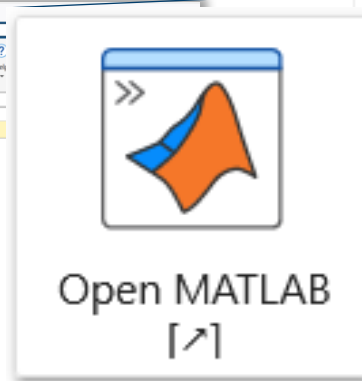
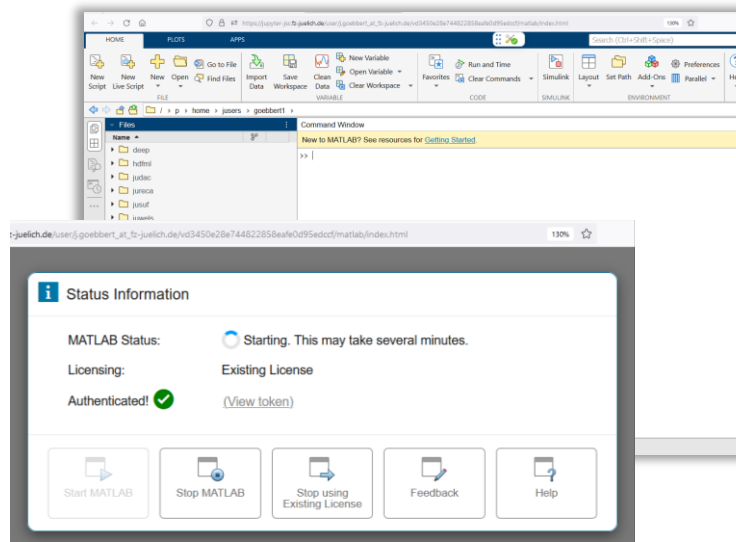
JUPYTERLAB – MATLAB

Web-based GUI for MATLAB

MATLAB – Web-based GUI

Based on an existing connection to the HPC system, MATLAB can be accessed in the browser.

- From here- you can connect directly to the cluster [2]
- Integrates MATLAB the HPC resources into the workflow (partool) [3].



[1] <https://www.fz-juelich.de/en/ias/jsc/services/user-support/software-tools/matlab>

[2] <https://de.mathworks.com/help/parallel-computing/remotecclusteraccess.html>

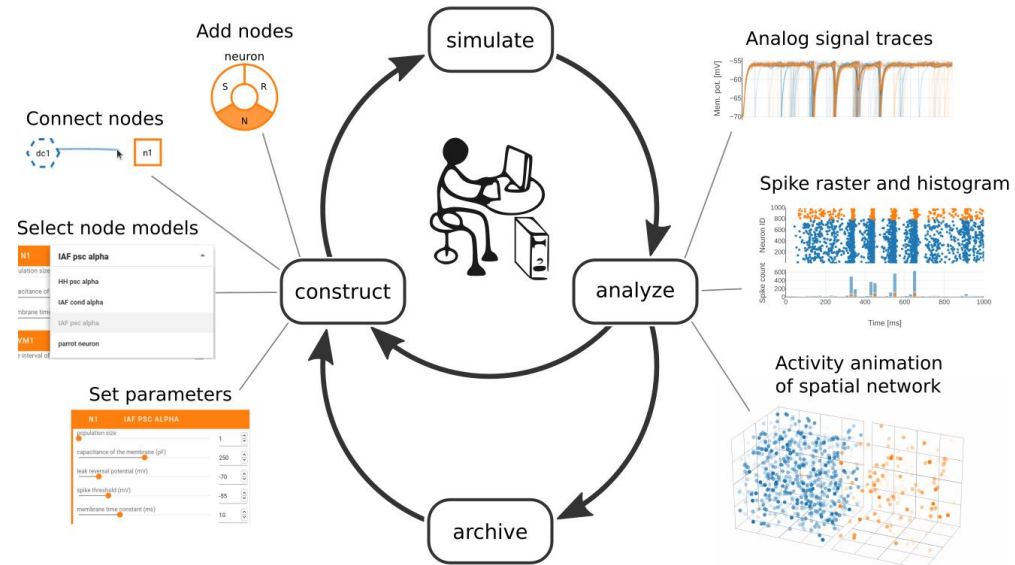
[3] <https://de.mathworks.com/products/parallel-computing.html>

JUPYTERLAB – NEST DESKTOP

Web-based GUI for Neuroscientists using NEST

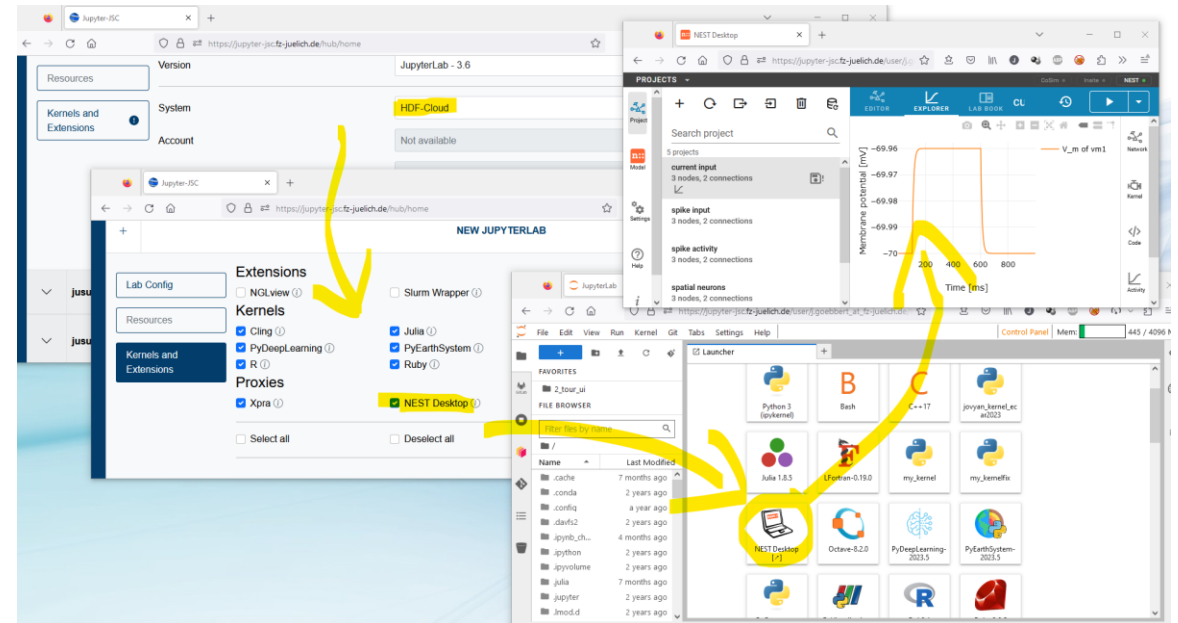
NEST-Desktop

NEST Desktop is a web-based GUI application for NEST Simulator, an advanced simulation tool for the computational neuroscience.



Jupyter-JSC gives you easy access to a NEST-Desktop

With Jupyter-JSC using Jupyter-Server-Proxy authenticated & authorized users get secure access to the WebUI of NEST-Desktop running NEST-simulations on HPC.



Plugin for Jupyter-Server-Proxy: [jupyter-xprahtml5-proxy](https://github.com/jhgoebbert/jupyter-nestdesktop-proxy)
<https://github.com/jhgoebbert/jupyter-nestdesktop-proxy>

[1] <https://nest-desktop.readthedocs.io>
[2] <https://www.nest-simulator.org>

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.

QUESTIONS?



Training course: <https://gitlab.jsc.fz-juelich.de/jupyter4jsc/training-2024.04-jupyter4hpc>