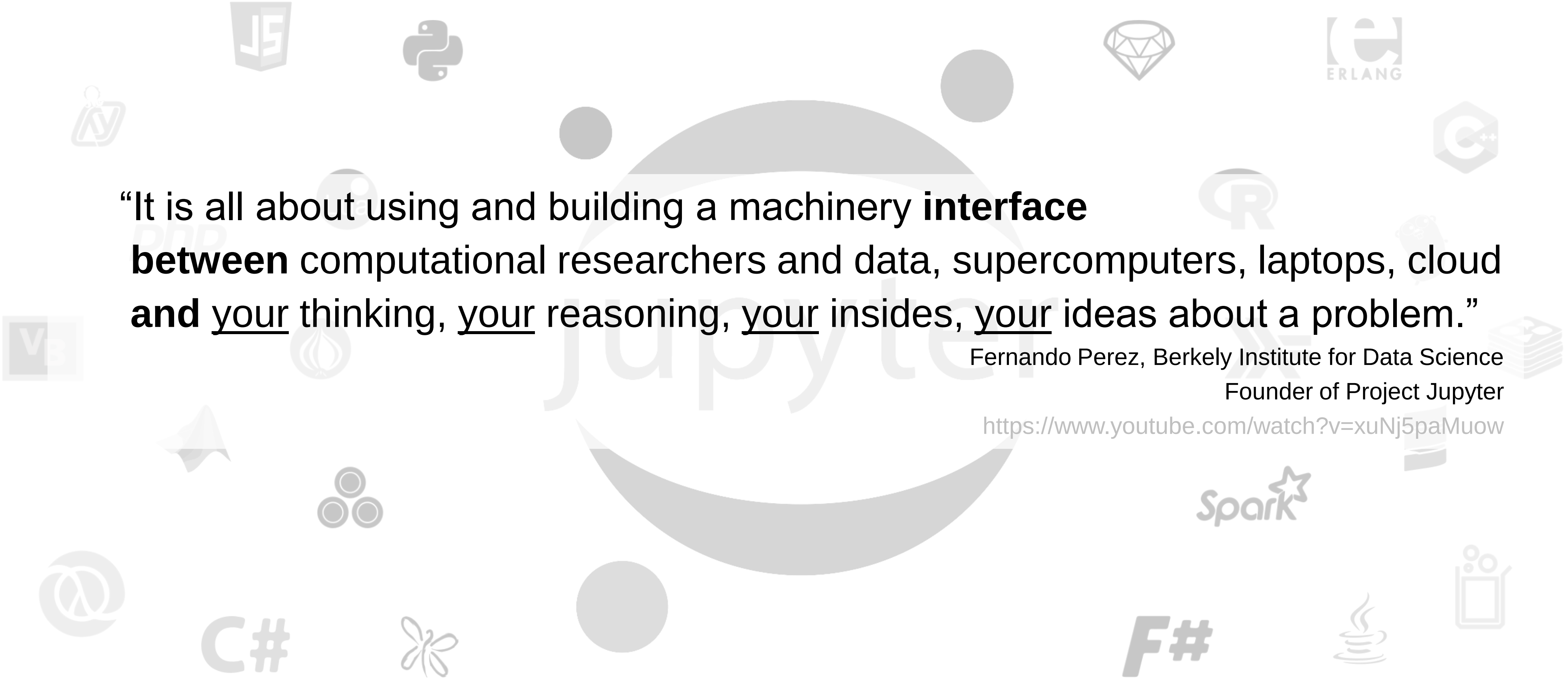
JUPYTERLAB - SUPERCOMPUTING IN YOUR BROWSER

Training course "Introduction to the usage and programming of supercomputer resources in Jülich"

2022-11-21 I JENS H. GÖBBERT (J.GOEBBERT@FZ-JUELICH.DE)
TIM KREUZER (T.KREUZER@FZ-JUELICH.DE)
ALICE GROSCH (A.GROSCH@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
https://www.youtube.com/watch?v=xuNj5paMuow

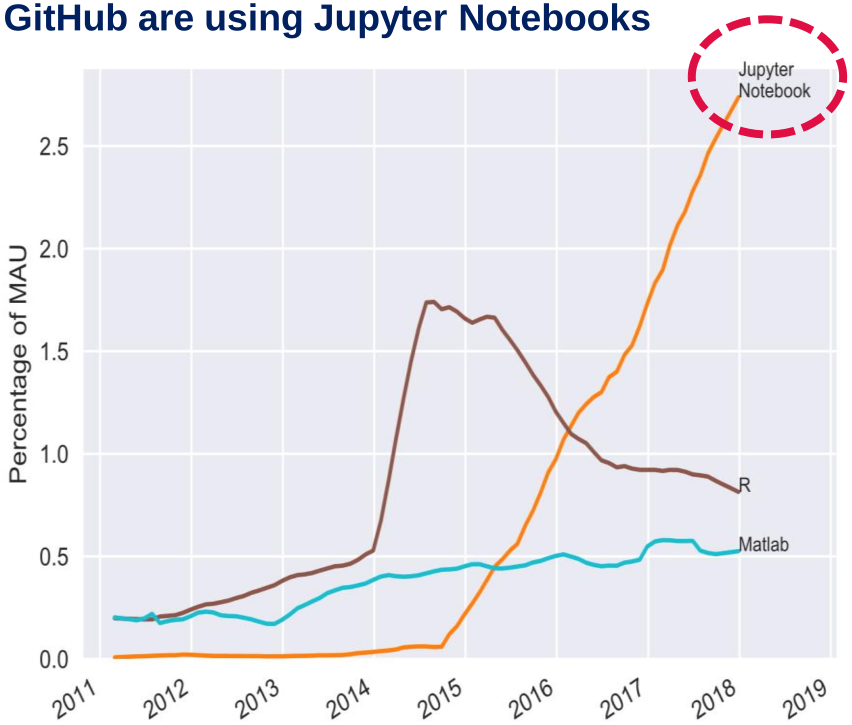
https://jupyter.org

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 2022, JupyterLab 4.0 planned end of year

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

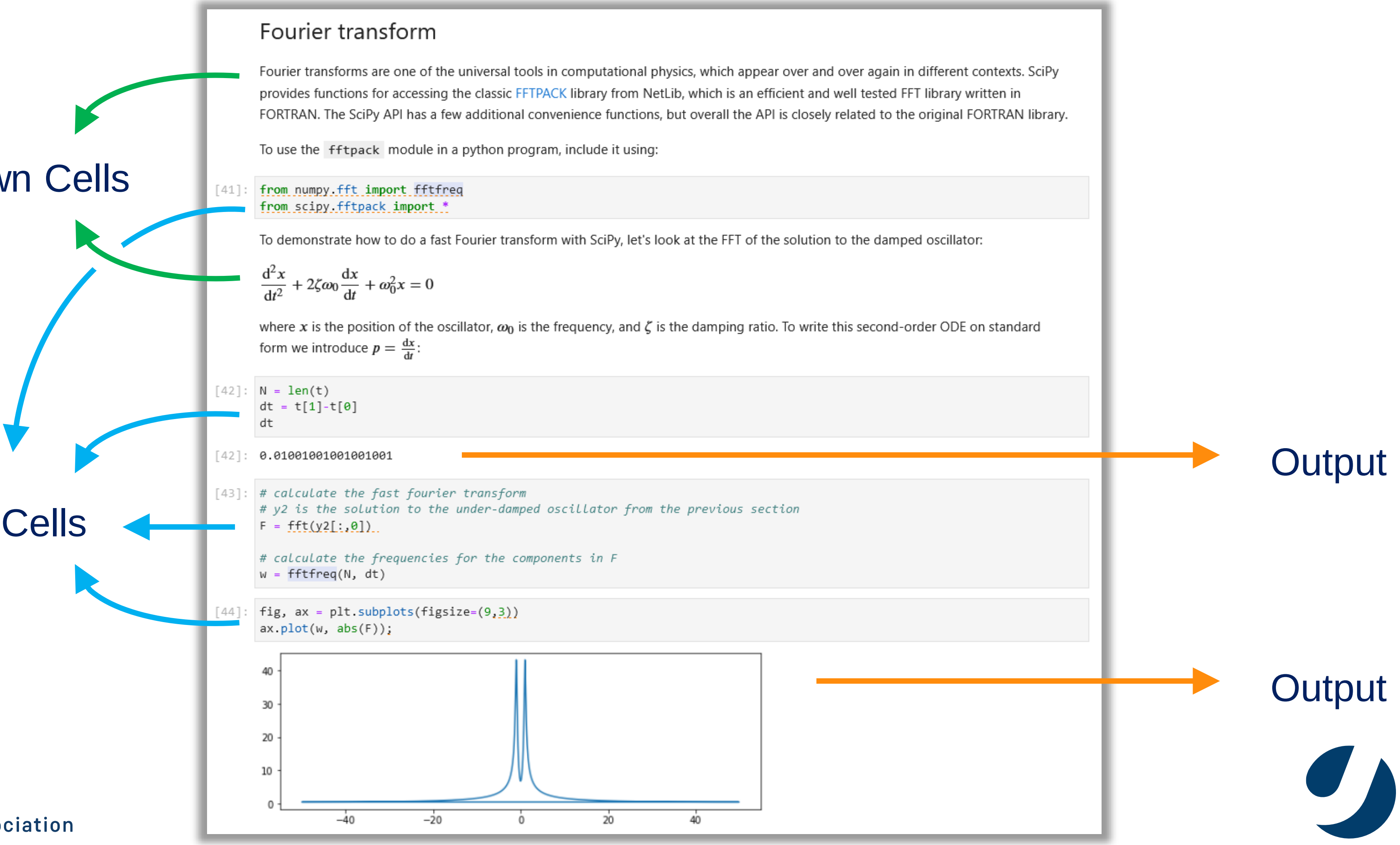


https://www.benfrederickson.com/ranking-programming-languages-by-github-users/
https://github.com/benfred/github-analysis

[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

JUPYTER NOTEBOOK

creating reproducible computational narratives



Markdown Cells

Code Cells

Output

Output

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:

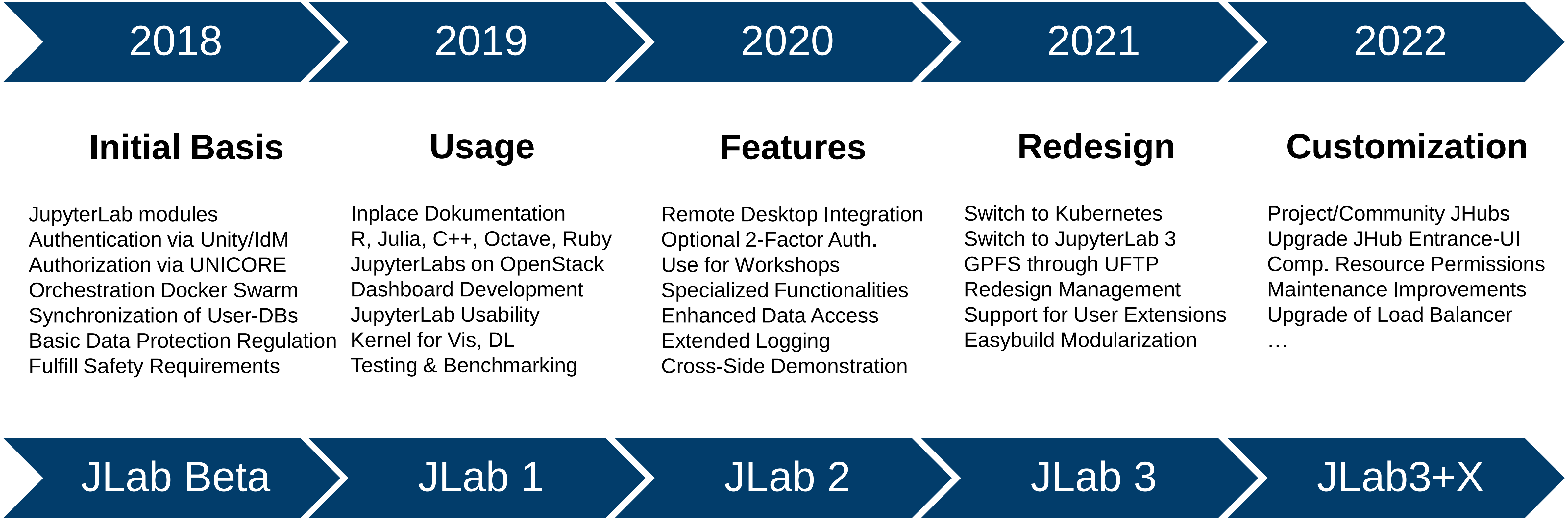
$$\frac{d^2x}{dt^2} + 2\zeta\omega_0\frac{dx}{dt} + \omega_0^2 x = 0$$

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));
```

JÜLICH Forschungszentrum

HISTORY OF JUPYTERLAB AT JSC



2018	2019	2020	2021	2022
Initial Basis	**Usage**	**Features**	**Redesign**	**Customization**
JupyterLab modules Authentication via Unity/IdM Authorization via UNICORE Orchestration Docker Swarm Synchronization of User-DBs Basic Data Protection Regulation Fulfill Safety Requirements	Inplace Dokumentation R, Julia, C++, Octave, Ruby JupyterLabs on OpenStack Dashboard Development JupyterLab Usability Kernel for Vis, DL Testing & Benchmarking	Remote Desktop Integration Optional 2-Factor Auth. Use for Workshops Specialized Functionalities Enhanced Data Access Extended Logging Cross-Side Demonstration	Switch to Kubernetes Switch to JupyterLab 3 GPFS through UFTP Redesign Management Support for User Extensions Easybuild Modularization	Project/Community JHubs Upgrade JHub Entrance-UI Comp. Resource Permissions Maintenance Improvements Upgrade of Load Balancer …
JLab Beta	JLab 1	JLab 2	JLab 3	JLab3+X

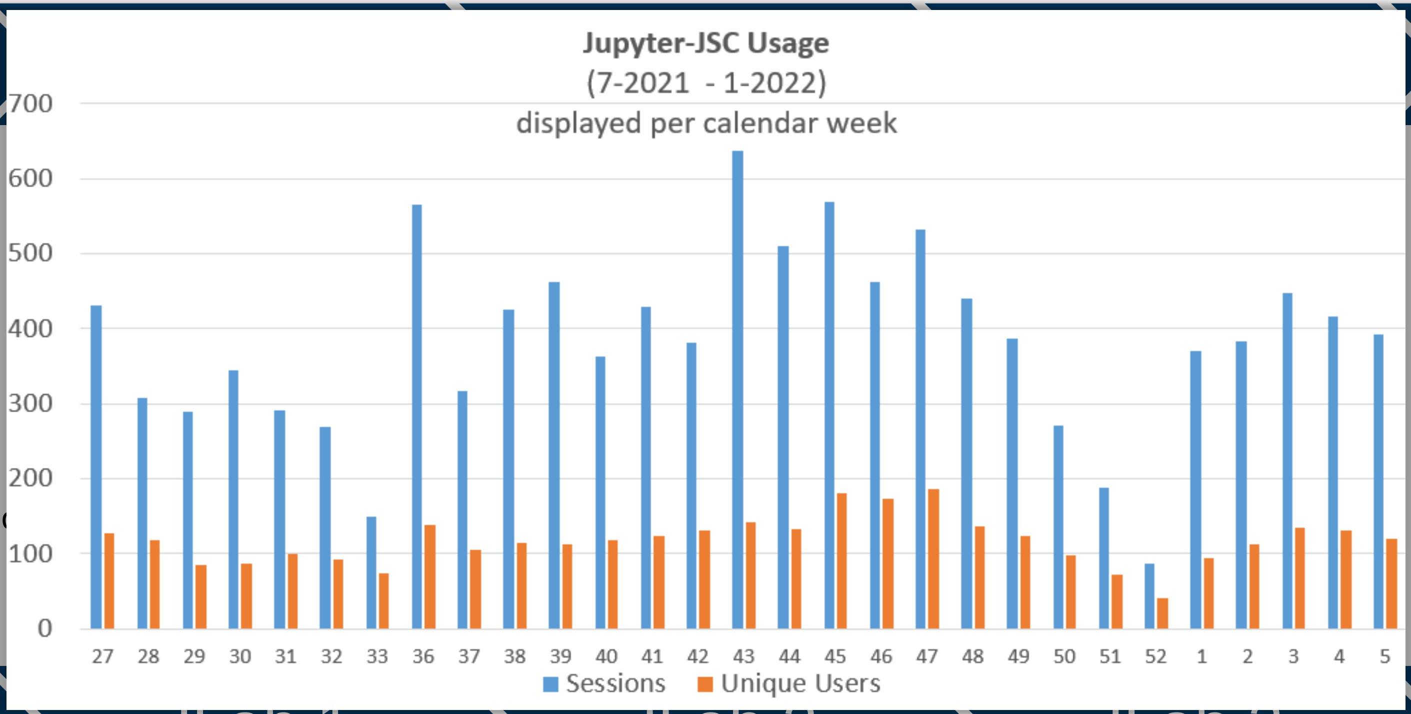
HISTORY OF JUPYTERLAB AT JSC

2018

2022

Initial Basis

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



Customization

Project/Community JHubs
Upgrade JHub Entrance-UI
Comp. Resource Permissions
Maintenance Improvements
Upgrade of Load Balancer
…

JLab Beta

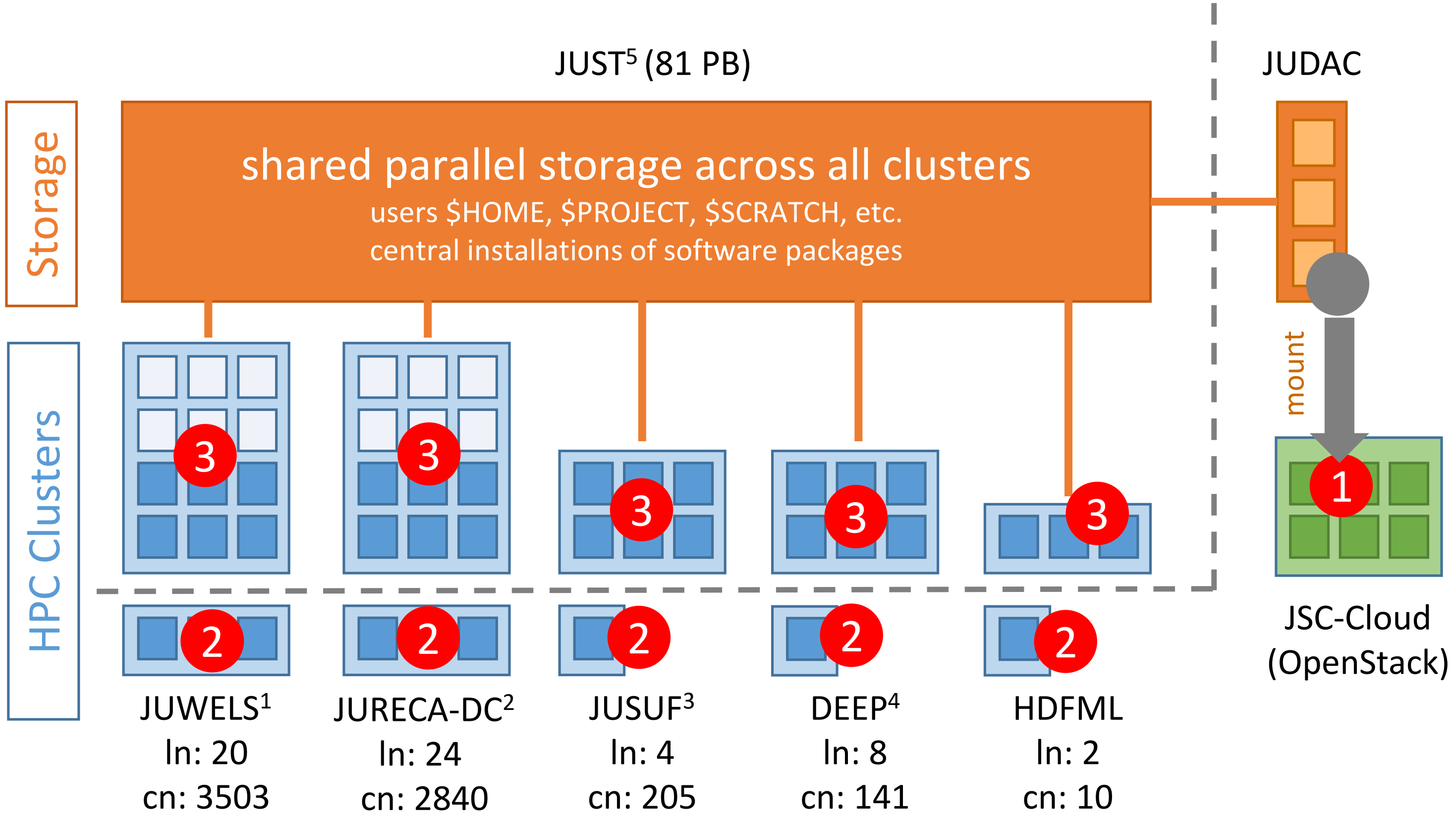
JLab 1

JLab 2

JLab 3

JLab3+X

JUPYTERLAB EVERYWHERE



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

JupyterLab everywhere

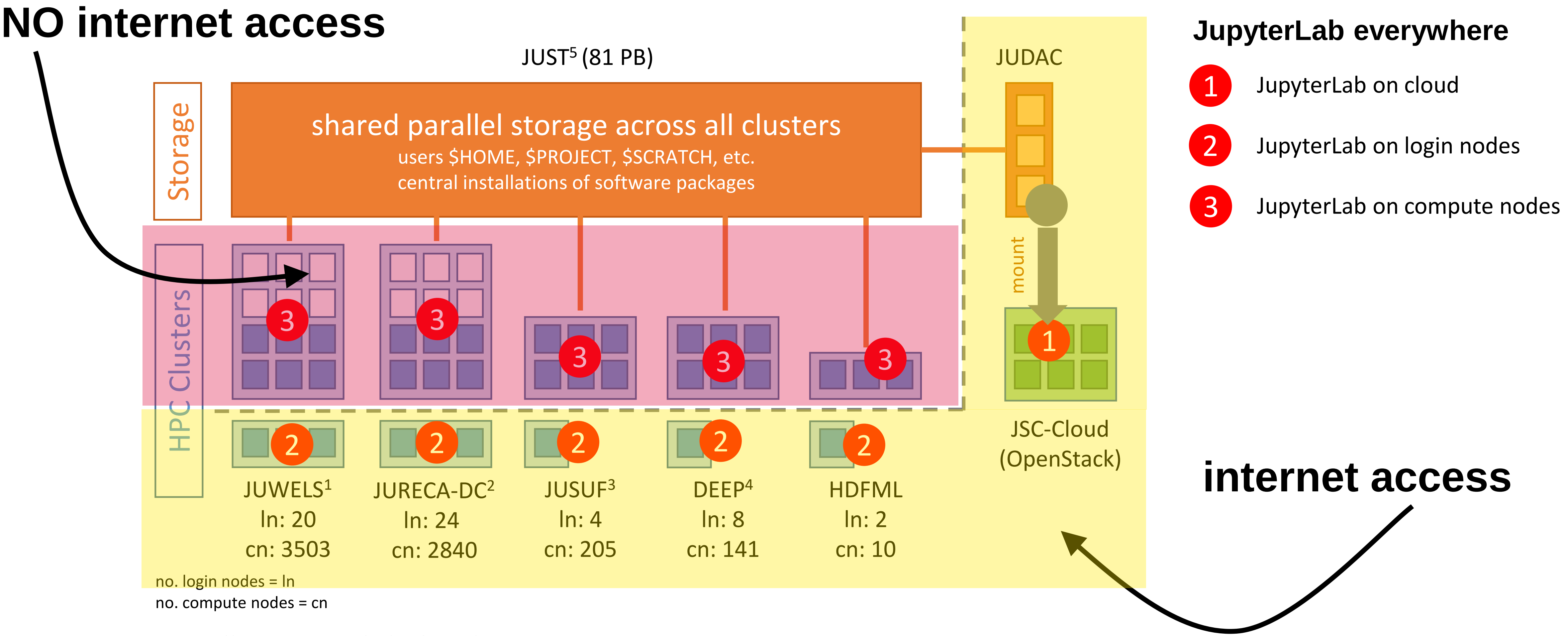
1 JupyterLab on cloud

2 JupyterLab on login nodes

3 JupyterLab on compute nodes

[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



NO internet access

JUST[5] (81 PB)

Storage

shared parallel storage across all clusters

users $HOME, $PROJECT, $SCRATCH, etc.

central installations of software packages

JUDAC

mount

HPC Clusters

JSC-Cloud (OpenStack)

JUWELS[1]	JURECA-DC[2]	JUSUF[3]	DEEP[4]	HDFML
ln: 20	ln: 24	ln: 4	ln: 8	ln: 2
cn: 3503	cn: 2840	cn: 205	cn: 141	cn: 10

no. login nodes = ln

no. compute nodes = cn

internet access

JupyterLab everywhere

1 JupyterLab on cloud

2 JupyterLab on login nodes

3 JupyterLab on compute nodes

[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

TERMINOLOGY

What is JupyterLab

JupyterLab

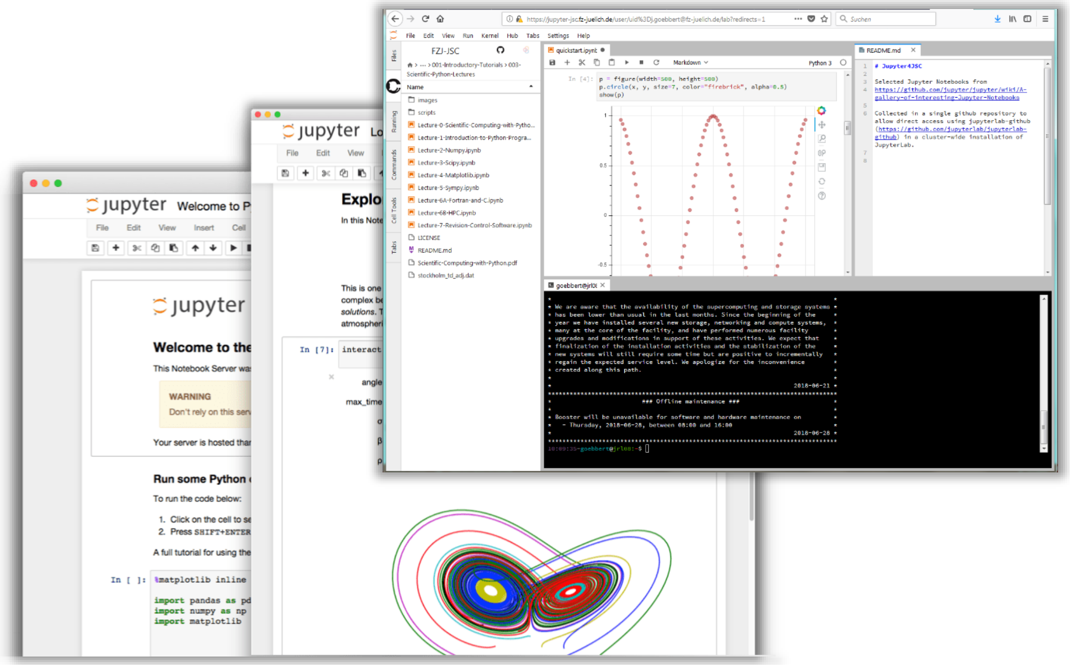
- **Interactive** working environment in the web browser
- For the creation of **reproducible** computer-aided narratives
- Very **popular** with researchers from all fields
- Jupyter = <u>Ju</u>lia + <u>Pyt</u>hon + <u>R</u>

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



```
In [ ]: %matplotlib inline

import pandas as pd
import numpy as np
import matplotlib
```

```
In [7]: interact
```

```
https://jupyter-jsc.fz-juelich.de/user/uid%3Dj.goebbert@fz-juelich.de/lab?redirects=1
```

```
In [4]: p = figure(width=500, height=500)
        p.circle(x, y, size=7, color="firebrick", alpha=0.5)
        show(p)
```

```
# Jupyter4JSC

Selected Jupyter Notebooks from
https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks

Collected in a single github repository to
allow direct access using jupyterlab-github
(https://github.com/jupyterlab/jupyterlab-github) in a cluster-wide installation of
JupyterLab.
```

```
*                                                                     *
* We are aware that the availability of the supercomputing and storage systems *
* has been lower than usual in the last months. Since the beginning of the    *
* year we have installed several new storage, networking and compute systems, *
* many at the core of the facility, and have performed numerous facility     *
* upgrades and modifications in support of these activities. We expect that   *
* finalization of the installation activities and the stabilization of the    *
* new systems will still require some time but are positive to incrementally  *
* regain the expected service level. We apologize for the inconvenience       *
* created along this path.                                                    *
*                                                                             *
*                                                                  2018-06-21 *
*******************************************************************************
*                          ### Offline maintenance ###                        *
*                                                                             *
* Booster will be unavailable for software and hardware maintenance on        *
*   - Thursday, 2018-06-28, between 08:00 and 16:00                           *
*                                                                  2018-06-28 *
*******************************************************************************
10:09:35-goebbert@jrl08:~$
```

https://jupyterlab.readthedocs.io

TERMINOLOGY

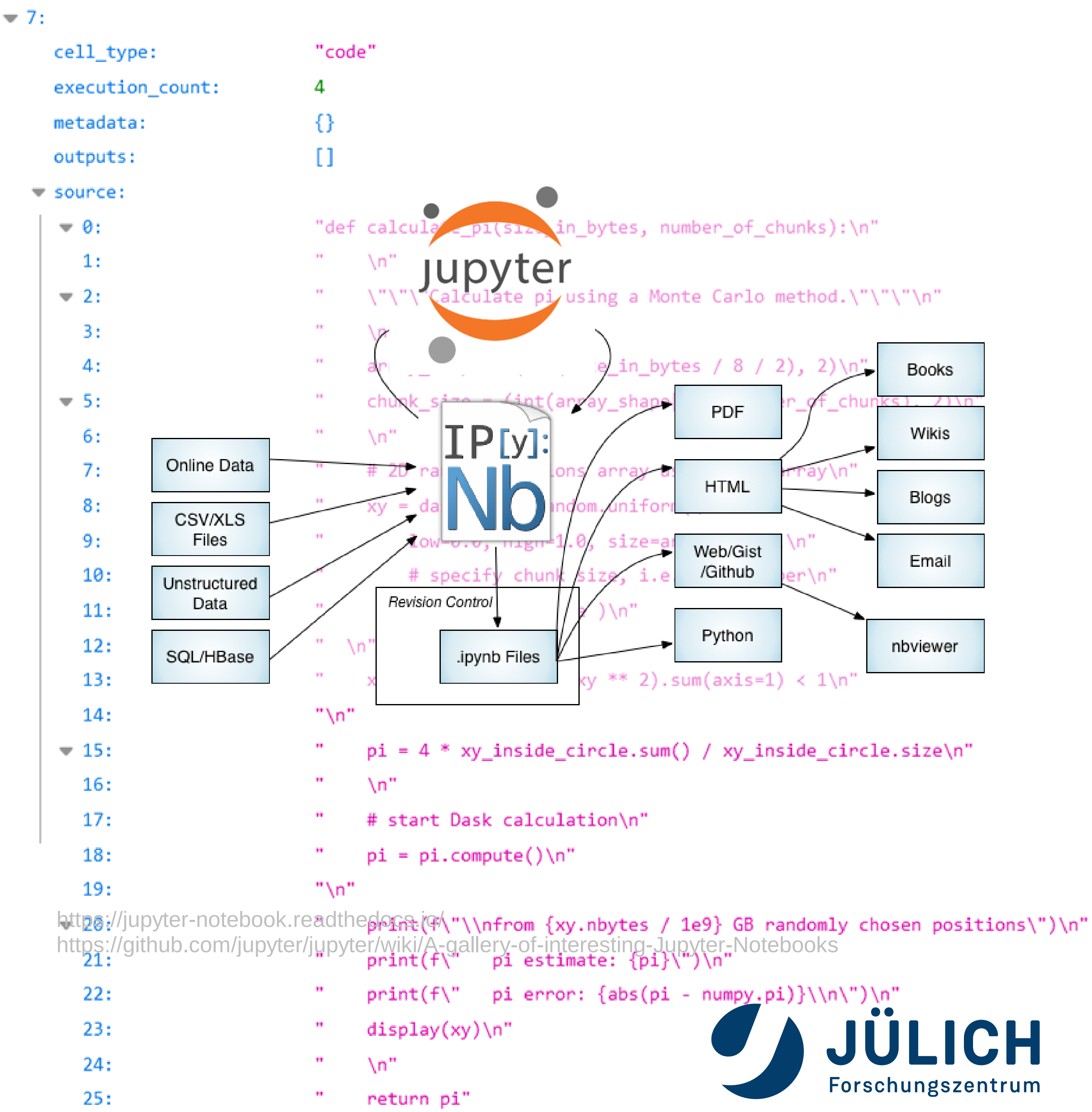
What is a Jupyter Notebook?

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..



```
▼ 7:
    cell_type:          "code"
    execution_count:    4
    metadata:           {}
    outputs:            []
  ▼ source:
    ▼ 0:    "def calcula  _pi(siz  in_bytes, number_of_chunks):\n"
      1:    "    \n"
    ▼ 2:    "    \"\"\" Calculate pi using a Monte Carlo method.\"\"\"\n"
      3:    "    \n"
      4:    "    ar                      e_in_bytes / 8 / 2), 2)\n"
    ▼ 5:    "    chunk_si     (int(ar  ay_shap              r_of_chunks),  2)\n"
      6:    "    \n"
      7:    "    # 2D ra              ons  array             rray\n"
      8:    "    xy = da             andom.unifor
      9:    "    low=0.0, high=1.0, size=a                  \n"
     10:    "        # specify chunk size, i.e              er\n"
     11:    "                                   )\n"
     12:    "  \n"
     13:    "    x                             xy ** 2).sum(axis=1) < 1\n"
     14:    "\n"
   ▼ 15:    "    pi = 4 * xy_inside_circle.sum() / xy_inside_circle.size\n"
     16:    "    \n"
     17:    "    # start Dask calculation\n"
     18:    "    pi = pi.compute()\n"
     19:    "\n"
   ▼ 20:    "    print(f\"\\nfrom {xy.nbytes / 1e9} GB randomly chosen positions\")\n"
     21:    "    print(f\"   pi estimate: {pi}\")\n"
     22:    "    print(f\"   pi error: {abs(pi - numpy.pi)}\\n\")\n"
     23:    "    display(xy)\n"
     24:    "    \n"
     25:    "    return pi"
```

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 deep learning, 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?

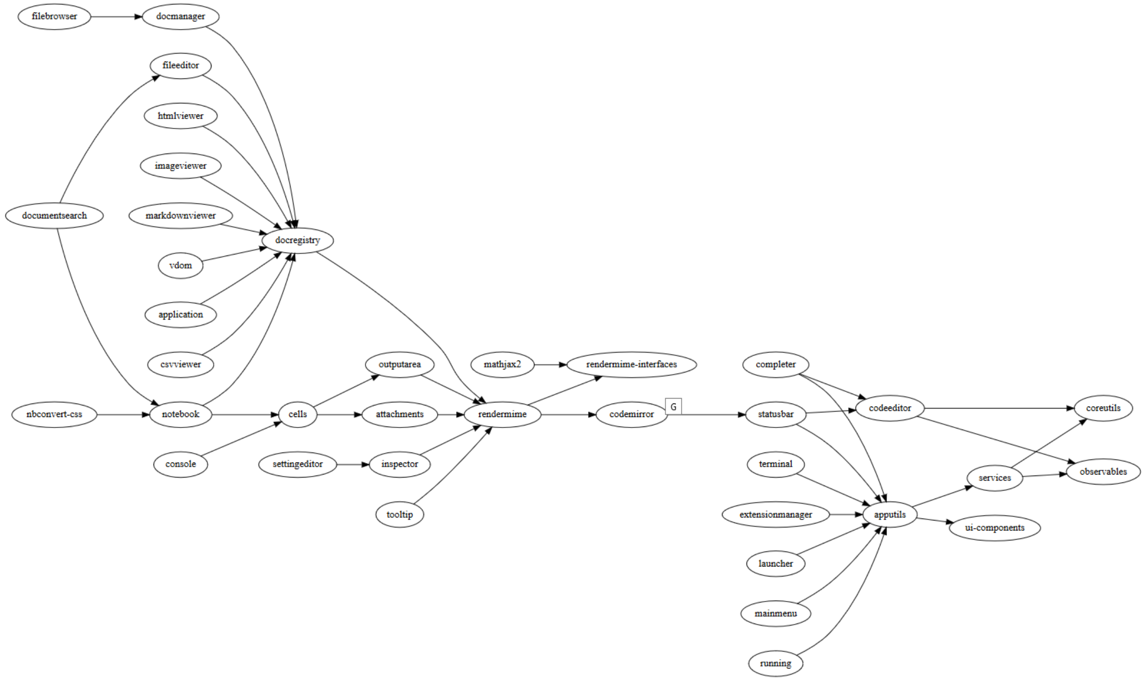
JupyterLab Extension

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

With JupyterLab 3 **prebuild extensions** were introduced (in contrast to **source extensions**).

You can now (technically) extend a compiled JupyterLab 3+ with without the need to rebuild.
=> That allows a separation of a system-wide JupyterLab core installation and extensions installed by a user.

JUPYTERLAB - WHEREVER YOU PREFER

Local, Remote, Browser-only

Local installation:

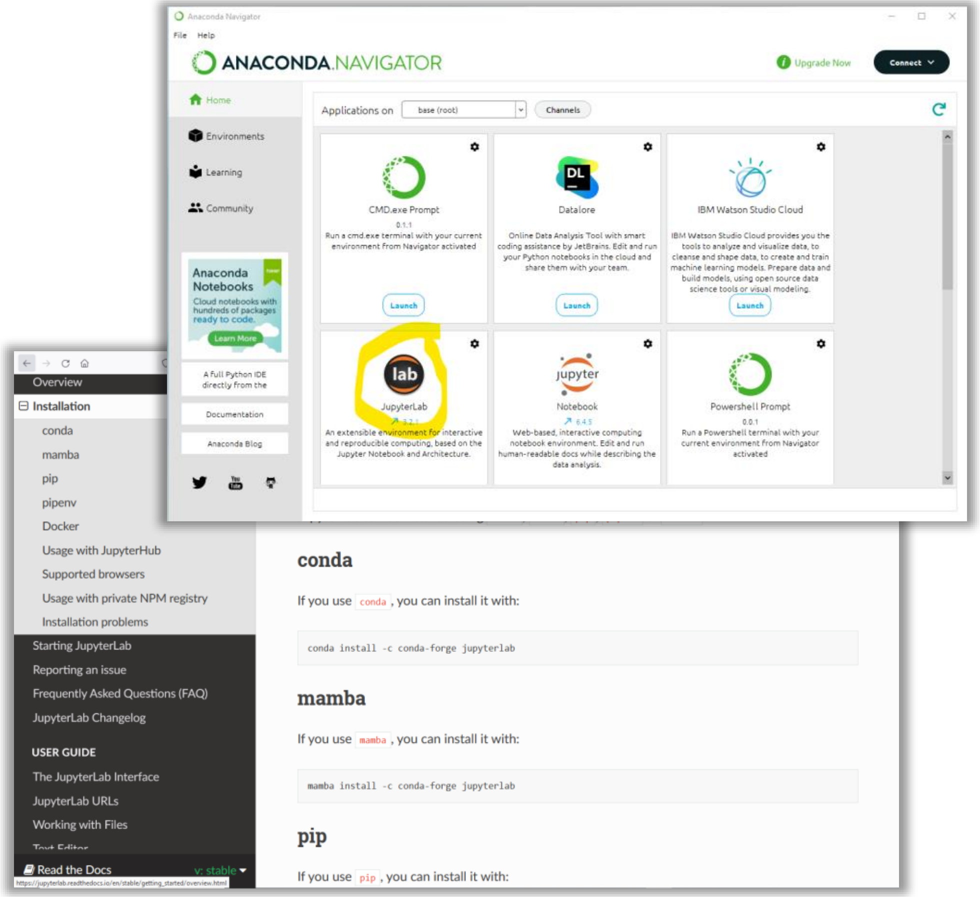
- **JupyterLab** installed using conda, mamba, pip, pipenv or docker.
  https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html

Remote (cluster) installation:

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

Browser-only installation (experimental!):

- **JupyterLab** local with server + client components in your browser
  JupyterLite includes a browser-ready Python environment named Pyodide.
  https://blog.jupyter.org/jupyter-everywhere-f8151c2cc6e8
  https://jupyter.org/try-jupyter/lab



CMD.exe Prompt: Run a cmd.exe terminal with your current environment from Navigator activated

Datalore: Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

IBM Watson Studio Cloud: IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

JupyterLab: An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook: Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt: Run a Powershell terminal with your current environment from Navigator activated

conda

If you use `conda`, you can install it with:

```
conda install -c conda-forge jupyterlab
```

mamba

If you use `mamba`, you can install it with:

```
mamba install -c conda-forge jupyterlab
```

pip

If you use `pip`, you can install it with:

JUPYTERLAB - WHEREVER YOU PREFER

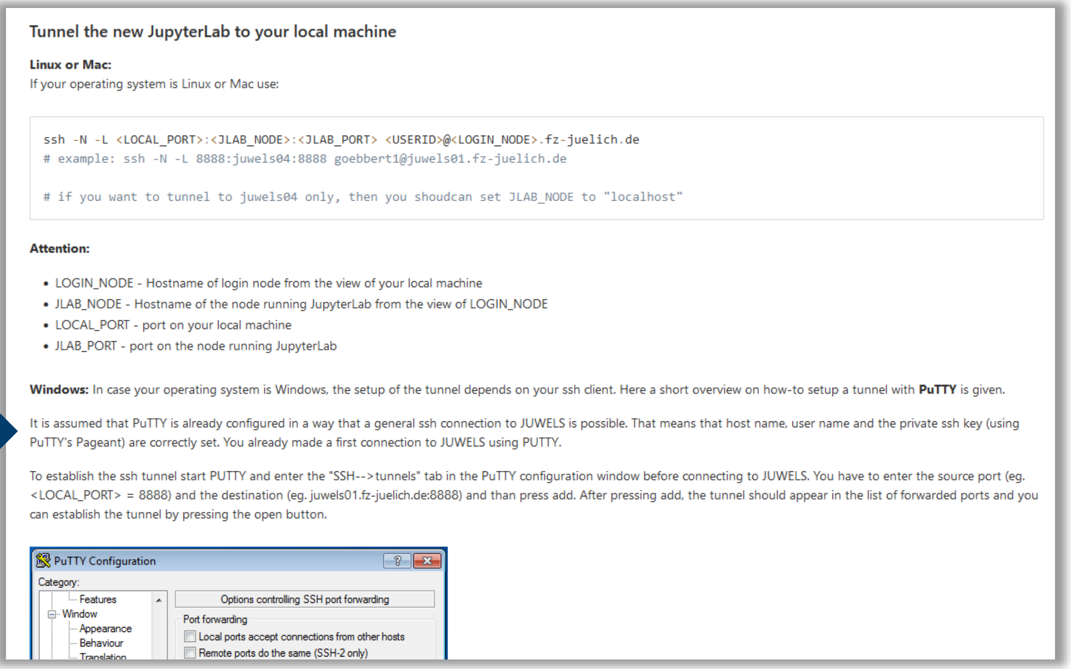
Local, Remote, Browser-only

Local installation:

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

Remote (cluster) installation:

- **JupyterLab** installed on a remote server
  - 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 use:

```
ssh -N -L <LOCAL_PORT>:<JLAB_NODE>:<JLAB_PORT> <USERID>@<LOGIN_NODE>.fz-juelich.de
# example: ssh -N -L 8888:juwels04:8888 goebbert1@juwels01.fz-juelich.de

# if you want to tunnel to juwels04 only, then you shoudcan 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. juwels01.fz-juelich.de:8888) and than 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.

□ Local ports accept connections from other hosts
□ Remote ports do the same (SSH-2 only)

Browser-only installation (experimental!):

- **JupyterLab** local with server + client components in your browser
  JupyterLite includes a browser-ready Python environment named Pyodide.
  https://blog.jupyter.org/jupyter-everywhere-f8151c2cc6e8
  https://jupyter.org/try-jupyter/lab

JUPYTERLAB - WHEREVER YOU PREFER

Local, Remote, Browser-only

Local installation:

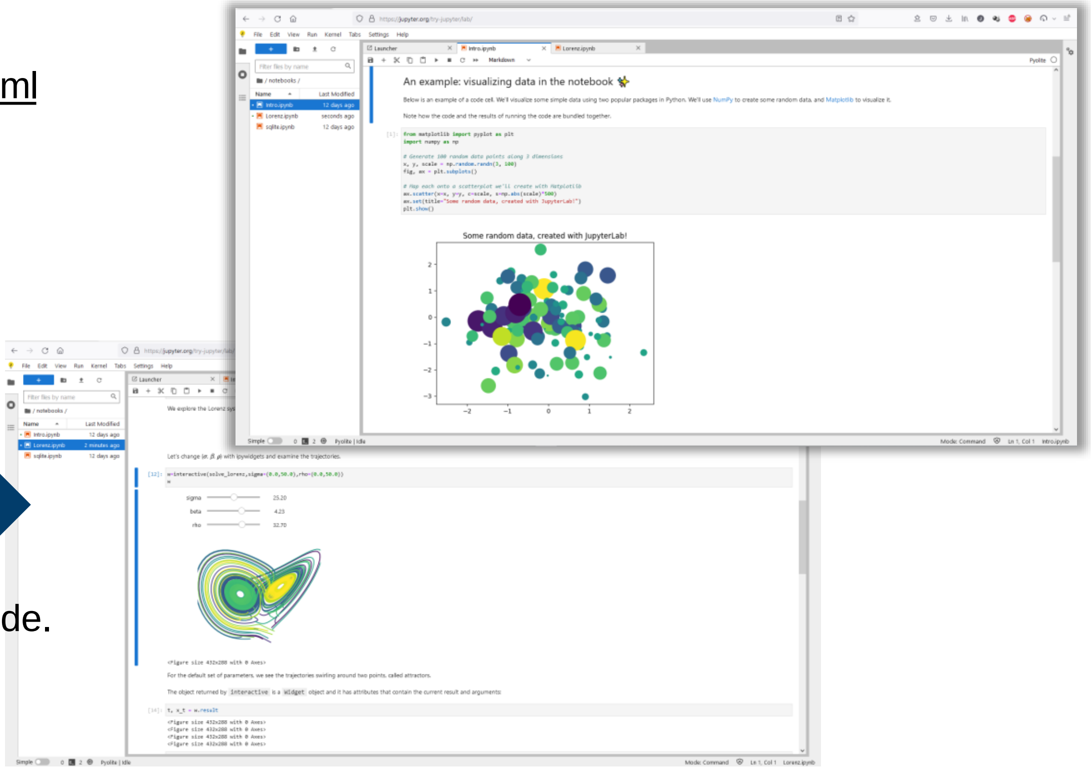
- **JupyterLab** installed using conda, mamba, pip, pipenv or docker.
  https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html

Remote (cluster) installation:

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

Browser-only installation (experimental!):

- **JupyterLab** local with server + client components in your browser
  JupyterLite includes a browser-ready Python environment named Pyodide.
  https://blog.jupyter.org/jupyter-everywhere-f8151c2cc6e8
  https://jupyter.org/try-jupyter/lab



An example: visualizing data in the notebook ✨

Below is an example of a code cell. We'll visualize some simple data using two popular packages in Python. We'll use NumPy to create some random data, and Matplotlib to visualize it.

Note how the code and the results of running the code are bundled together.

```
[1]: from matplotlib import pyplot as plt
     import numpy as np

     # Generate 100 random data points along 3 dimensions
     x, y, scale = np.random.randn(3, 100)
     fig, ax = plt.subplots()

     # Map each onto a scatterplot we'll create with Matplotlib
     ax.scatter(x=x, y=y, c=scale, s=np.abs(scale)*500)
     ax.set(title="Some random data, created with JupyterLab!")
     plt.show()
```

We explore the Lorenz sys

Let's change (σ, β, ρ) with ipywidgets and examine the trajectories.

```
[12]: w=interactive(solve_lorenz,sigma=(0.0,50.0),rho=(0.0,50.0))
      w
```

```
<Figure size 432x288 with 0 Axes>
```

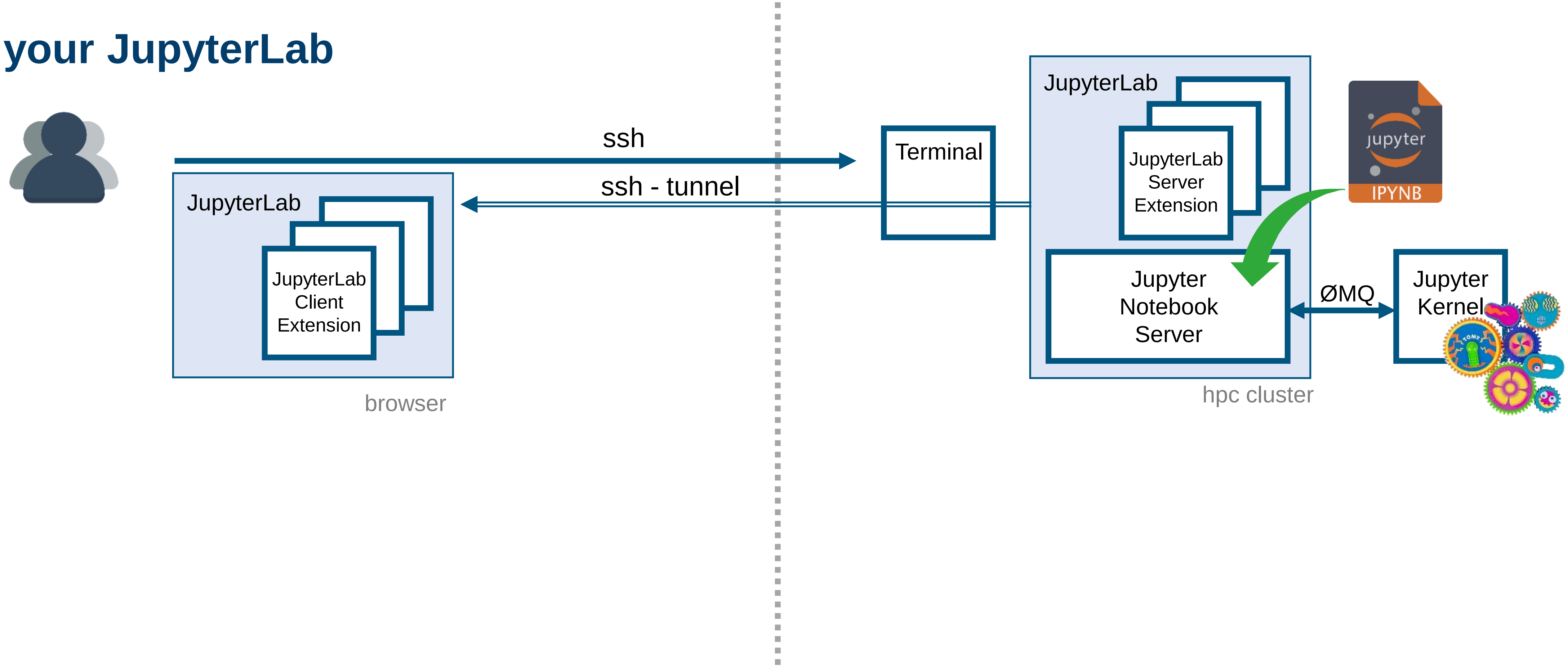
For the default set of parameters, we see the trajectories swirling around two points, called attractors.

The object returned by `interactive` is a `Widget` object and it has attributes that contain the current result and arguments:

```
[14]: t, x_t = w.result
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
```

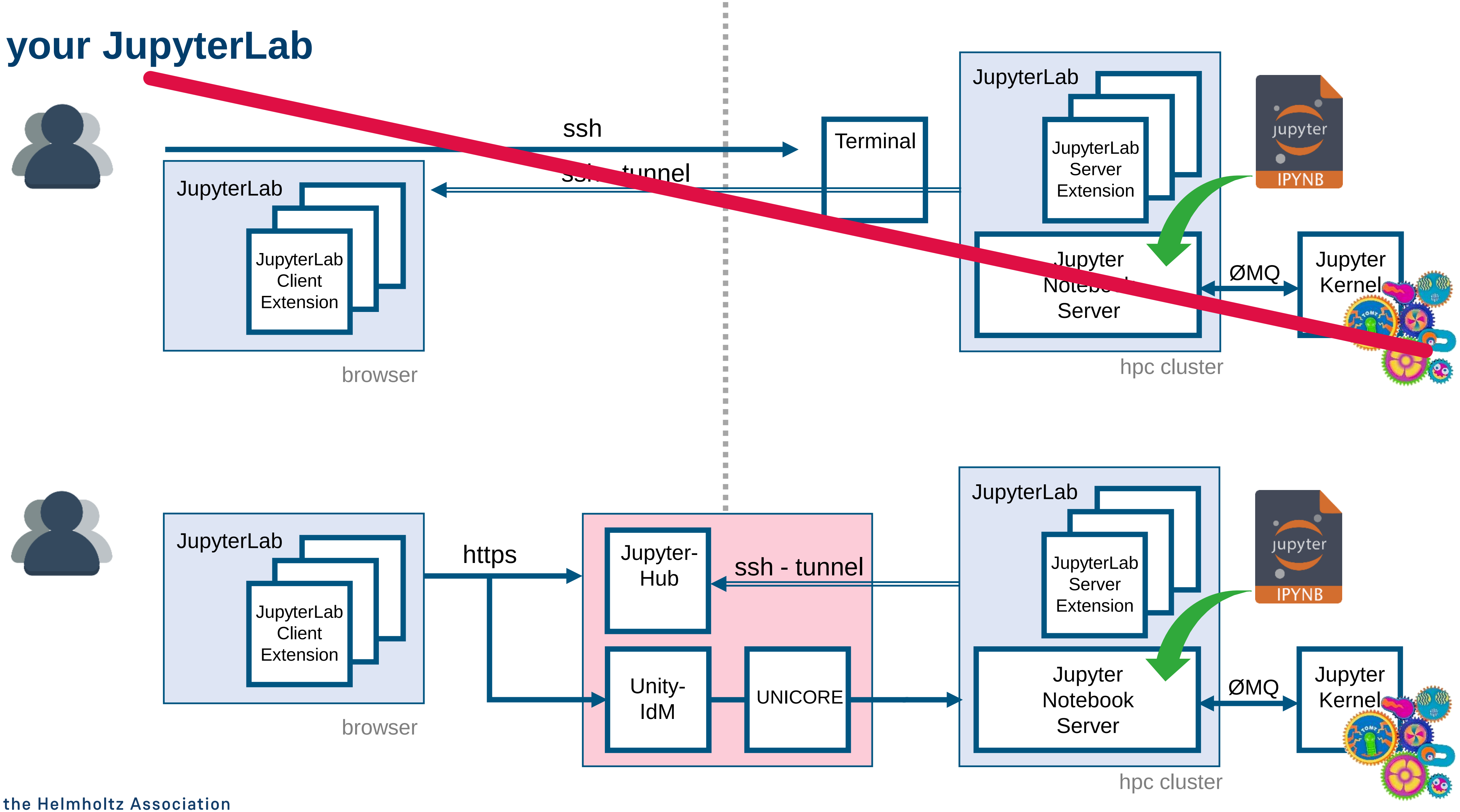
JUPYTERLAB ON HPC

Start your JupyterLab



JUPYTERLAB ON HPC

Start your JupyterLab



JUPYTER-JSC WEBSERVICE

Prepare

1) Register & **Login**

- ✓ https://judoor.fz-juelich.de

2) Join a project

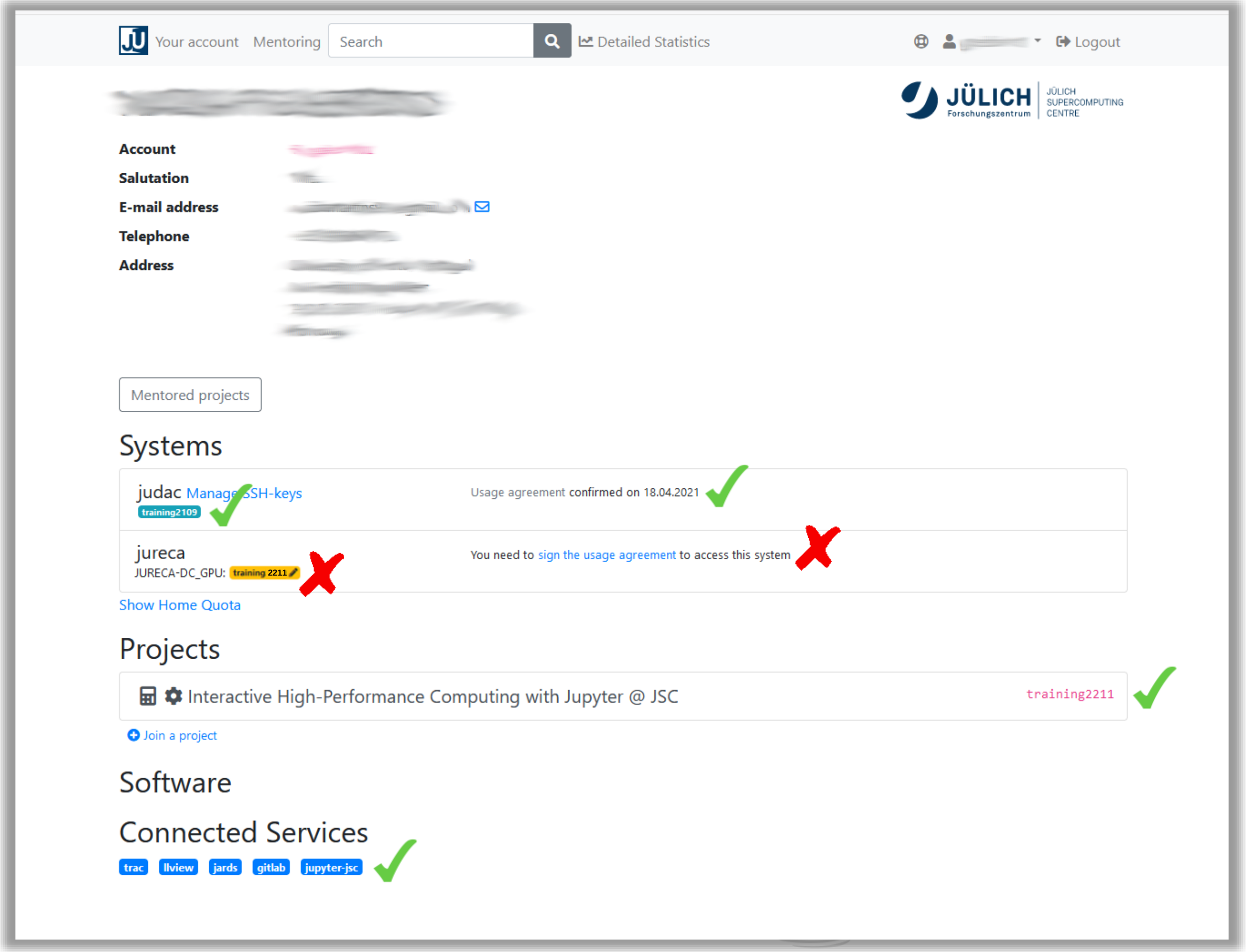
- ✓ Wait to get joined by the project PI

3) Sign usage agreement

- ✓ Wait for creation of HPC accounts

4) Check Connected Services:

- ✓ jupyter-jsc



[redacted]

Account [redacted]
Salutation [redacted]
E-mail address [redacted]
Telephone [redacted]
Address [redacted]

Systems

judac Manage SSH-keys ✓
training2109
Usage agreement confirmed on 18.04.2021 ✓

jureca
JURECA-DC_GPU: training 2211 ✗
You need to sign the usage agreement to access this system ✗

Show Home Quota

Projects

Interactive High-Performance Computing with Jupyter @ JSC training2211 ✓

Join a project

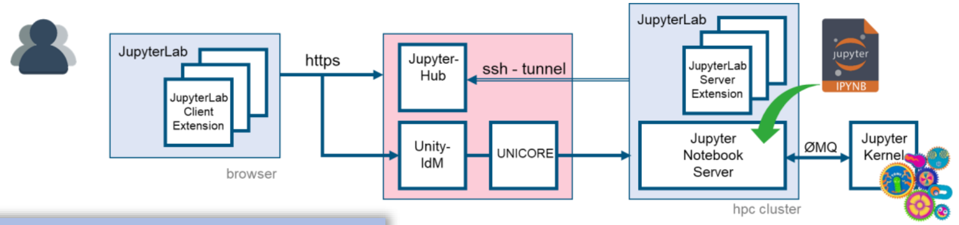
Software

Connected Services

trac llview jards gitlab jupyter-jsc ✓

JUPYTER-JSC WEBSERVICE

First time login

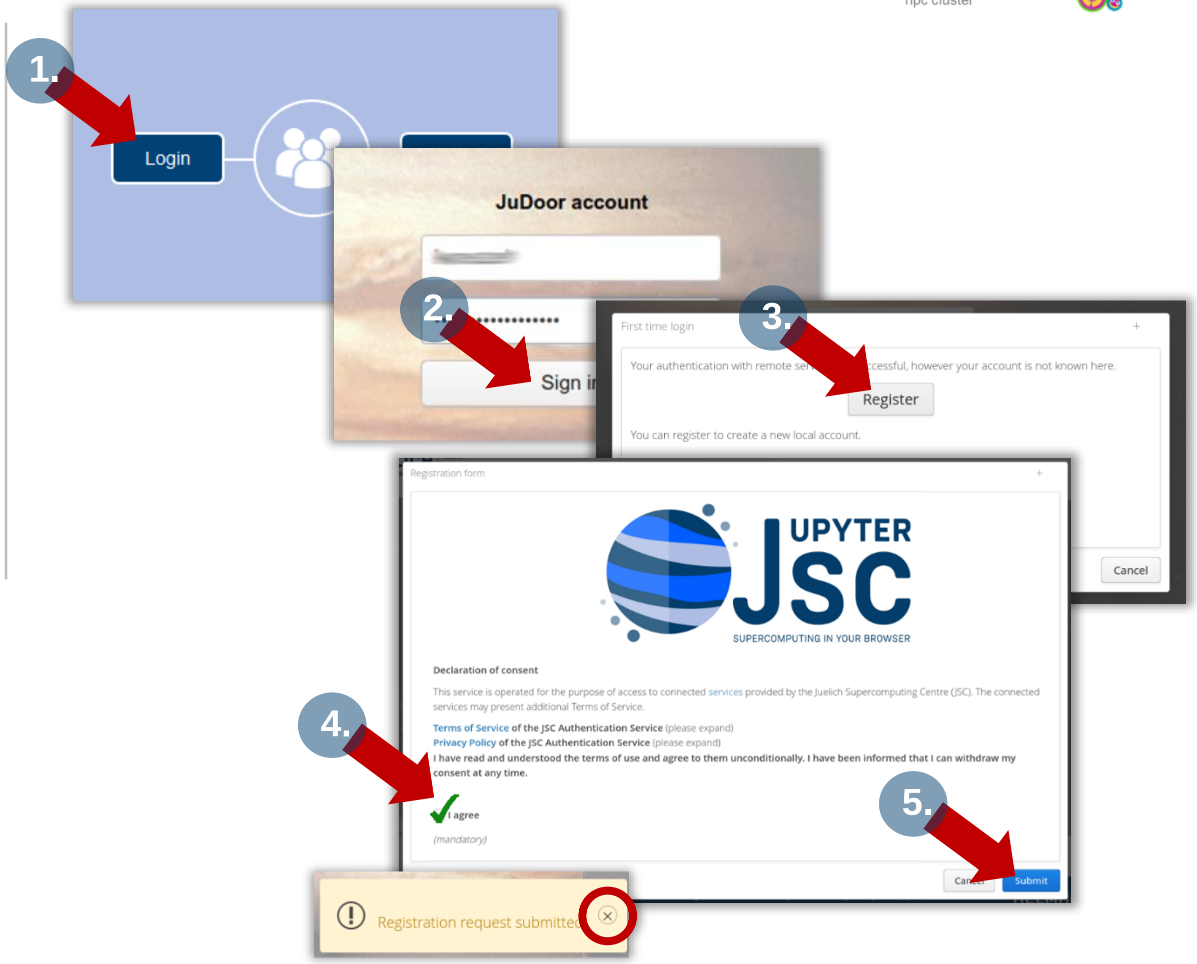


=> https://jupyter-jsc.fz-juelich.de

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



1. Login

2. JuDoor account

[redacted]

•••••••••••••

Sign in

3. First time login

Your authentication with remote ser… …ccessful, however your account is not known here.

Register

You can register to create a new local account.

Cancel

Registration form

UPYTER JSC
SUPERCOMPUTING IN YOUR BROWSER

Declaration of consent

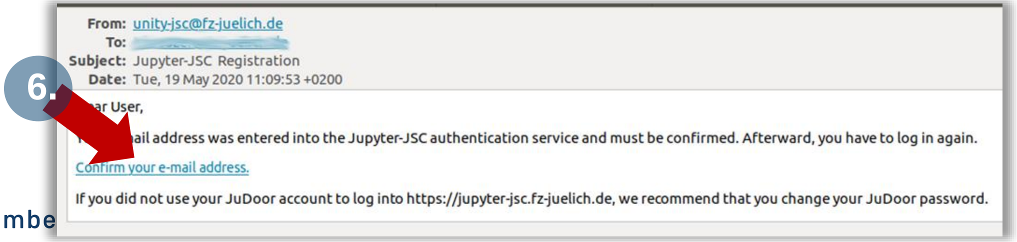
This service is operated for the purpose of access to connected services provided by the Juelich Supercomputing Centre (JSC). The connected services may present additional Terms of Service.

Terms of Service of the JSC Authentication Service (please expand)
Privacy Policy of the JSC Authentication Service (please expand)
I have read and understood the terms of use and agree to them unconditionally. I have been informed that I can withdraw my consent at any time.

4. ✓ I agree
(mandatory)

5. Cancel Submit

Registration request submitted ⊗



6.

From: unity-jsc@fz-juelich.de
To: [redacted]
Subject: Jupyter-JSC Registration
Date: Tue, 19 May 2020 11:09:53 +0200

…ar User,

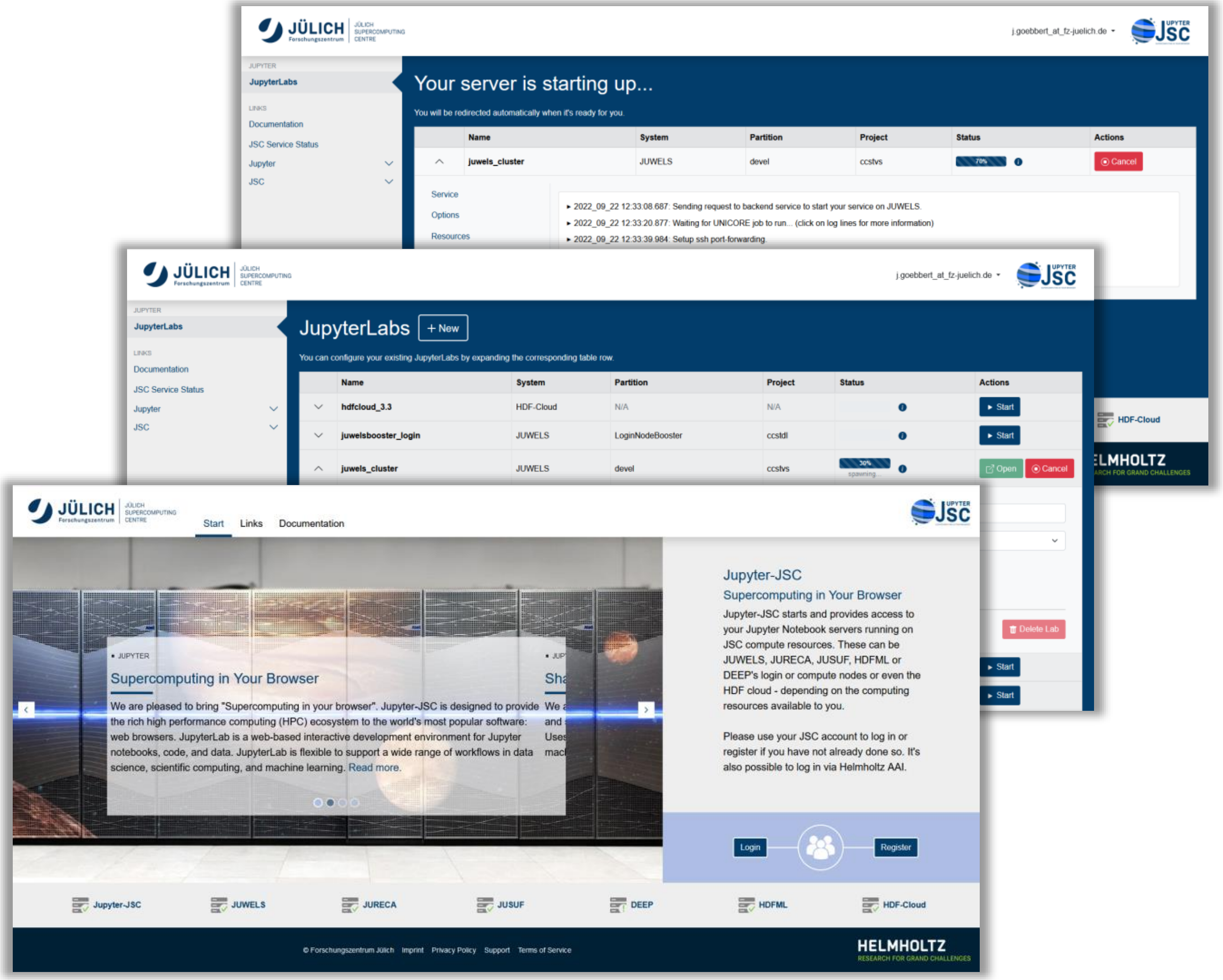
…ail address was entered into the Jupyter-JSC authentication service and must be confirmed. Afterward, you have to log in again.

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.

JUPYTER-JSC WEBSERVICE

Start your JupyterLab



j.goebbert_at_fz-juelich.de

JUPYTER
JupyterLabs
LINKS
Documentation
JSC Service Status
Jupyter
JSC

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	ccstvs	70%	Cancel

Service
Options
Resources

```
▸ 2022_09_22 12:33:08.687: Sending request to backend service to start your service on JUWELS.
▸ 2022_09_22 12:33:20.877: Waiting for UNICORE job to run... (click on log lines for more information)
▸ 2022_09_22 12:33:39.984: Setup ssh port-forwarding.
```

JupyterLabs + New

You can configure your existing JupyterLabs 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	ccstdl		▸ Start
juwels_cluster	JUWELS	devel	ccstvs	30% spawning...	Open Cancel

Start Links Documentation

JUPYTER

Supercomputing in Your Browser

We are pleased to bring "Supercomputing in your browser". Jupyter-JSC is designed to provide 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.

Jupyter-JSC

Supercomputing in Your Browser

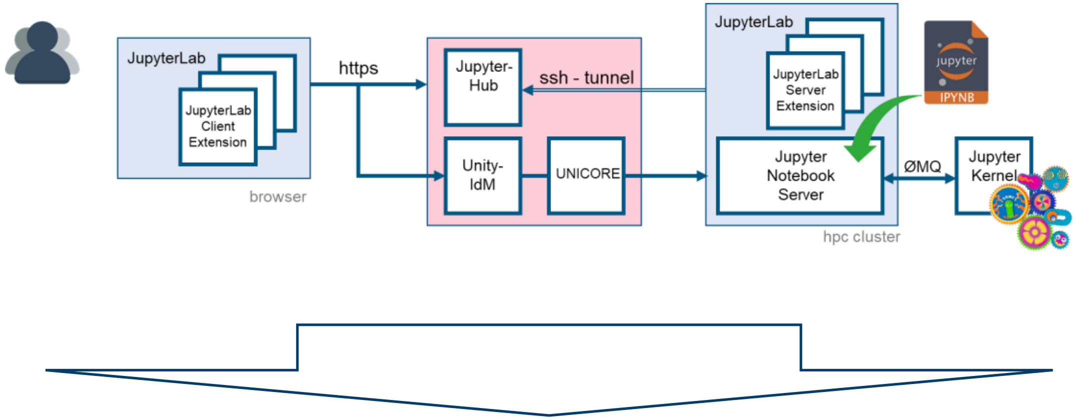
Jupyter-JSC starts and provides access to your Jupyter Notebook servers running on JSC compute resources. These can be JUWELS, JURECA, JUSUF, HDFML or DEEP's login or compute nodes or even the HDF cloud - depending on the computing resources available to you.

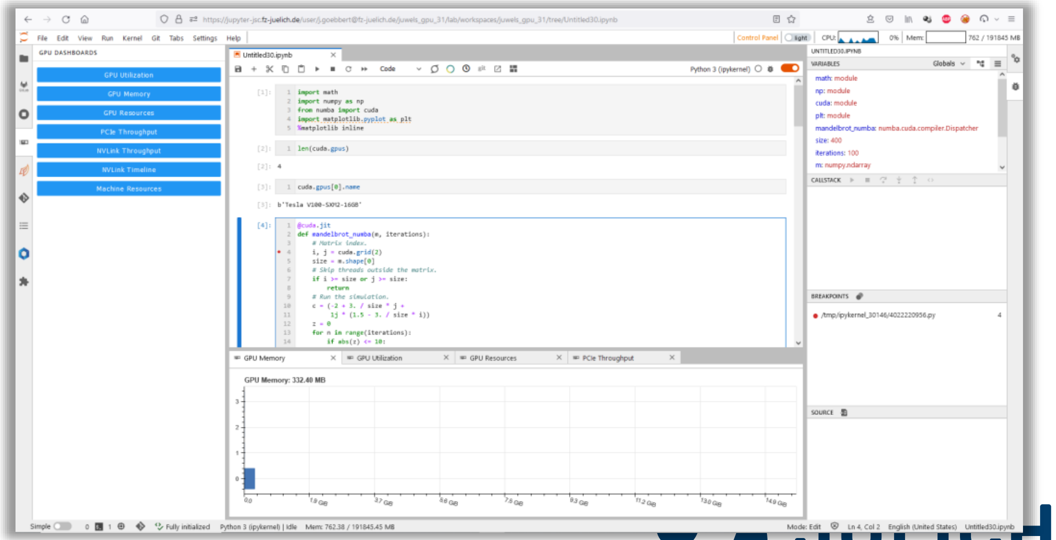
Please use your JSC account to log in or register if you have not already done so. It's also possible to log in via Helmholtz AAI.

Login Register

Jupyter-JSC JUWELS JURECA JUSUF DEEP HDFML HDF-Cloud

© Forschungszentrum Jülich Imprint Privacy Policy Support Terms of Service





https://jupyter-jsc.fz-juelich.de/user/j.goebbert@fz-juelich.de/juwels_gpu_31/lab/workspaces/juwels_gpu_31/tree/Untitled30.ipynb

GPU DASHBOARDS
GPU Utilization
GPU Memory
GPU Resources
PCIe Throughput
NVLink Throughput
NVLink Timeline
Machine Resources

Untitled30.ipynb — Python 3 (ipykernel)

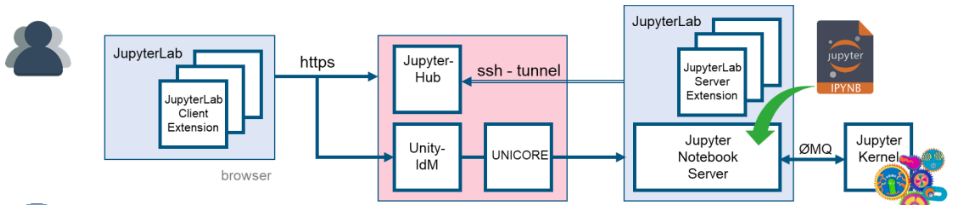
```
[1]: import math
     import numpy as np
     from numba import cuda
     import matplotlib.pyplot as plt
     %matplotlib inline

[2]: len(cuda.gpus)
[2]: 4

[3]: cuda.gpus[0].name
[3]: b'Tesla V100-SXM2-16GB'

[4]: @cuda.jit
     def mandelbrot_numba(m, iterations):
         # Matrix index.
         i, j = cuda.grid(2)
         size = m.shape[0]
         # Skip threads outside the matrix.
         if i >= size or j >= size:
             return
         # Run the simulation.
         c = (-2 + 3. / size * j +
              1j * (1.5 - 3. / size * i))
         z = 0
         for n in range(iterations):
             if abs(z) <= 10:
```

GPU Memory: 332.40 MB

VARIABLES
math: module
np: module
cuda: module
plt: module
mandelbrot_numba: numba.cuda.compiler.Dispatcher
size: 400
iterations: 100
m: numpy.ndarray

JUPYTER-JSC WEBSERVICE

Control Panel



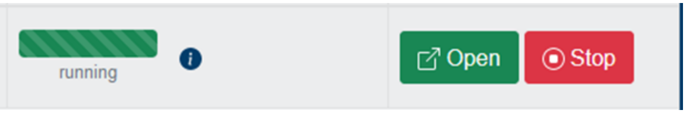
A. Jupyter-JSC – Add new JupyterLab

B. Configuration Dialog

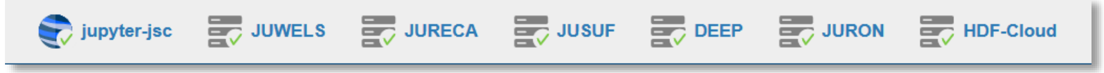
- set Name, Type, System, Account, Project, Partition

C. Jupyter-JSC – Actions

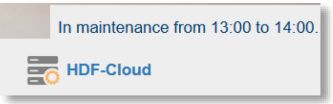
- Open/Stop a running JupyterLab
- Change/Delete **configuration**



D. Jupyter-JSC -- Statusbar

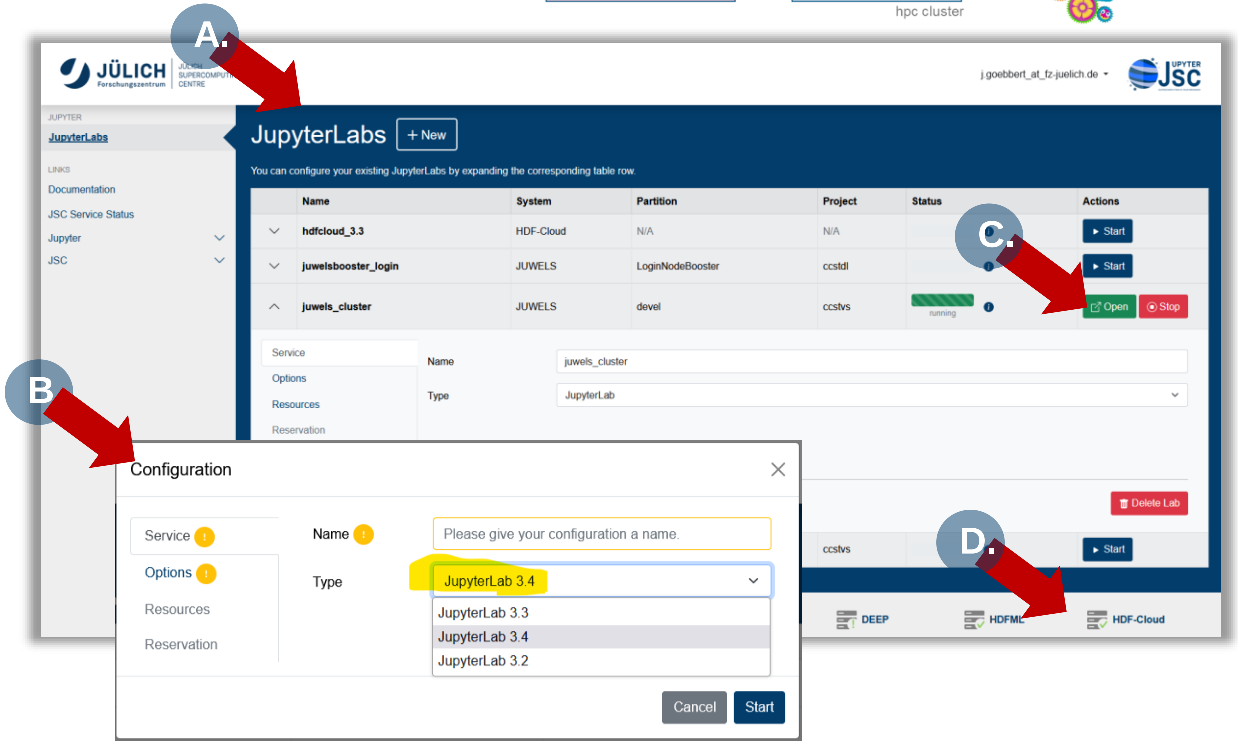


- Upcoming maintenance (mouse hover for details)



In maintenance from 13:00 to 14:00.

- System offline



You can configure your existing JupyterLabs 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	ccstdl		▸ Start
∧	juwels_cluster	JUWELS	devel	ccstvs	running	Open Stop
				ccstvs		▸ Start

Service
Options
Resources
Reservation

Name: juwels_cluster

Type: JupyterLab

Configuration

Service
Options
Resources
Reservation

Name: Please give your configuration a name.

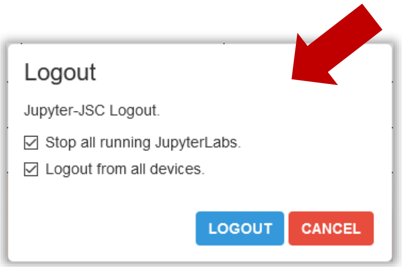
Type: JupyterLab 3.4

JupyterLab 3.3
JupyterLab 3.4
JupyterLab 3.2

Cancel Start

E. Jupyter-JSC – Logout

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



Logout

Jupyter-JSC Logout.

☑ Stop all running JupyterLabs.

☑ Logout from all devices.

LOGOUT CANCEL

JUPYTER-JSC WEBSERVICE

JupyterLab Configuration

Jupyter-JSC – Configuration

Available options **depend on**

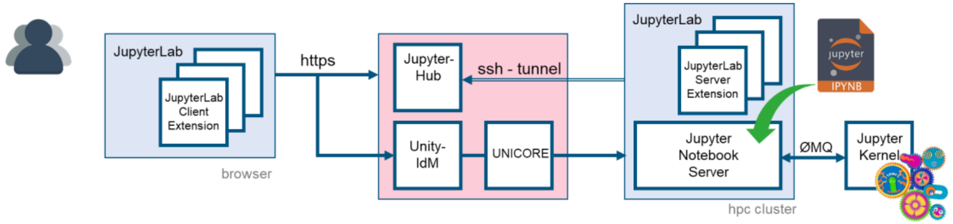
- user account settings visible in **judoor.fz-juelich.de**
- system specific usage agreement on JuDoor is signed (!!!)
- currently available systems in all of your projects

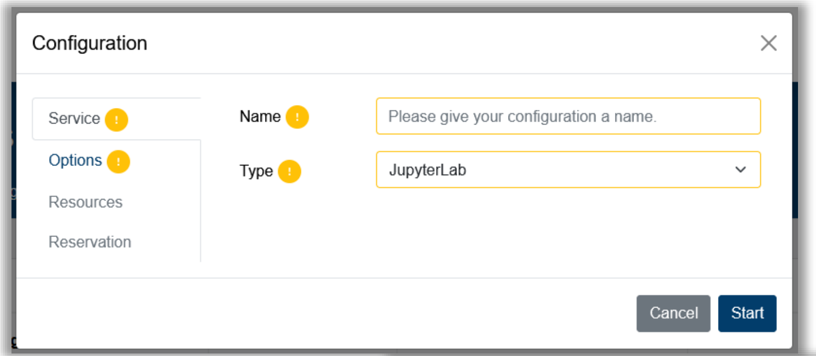
Basic options

- Type:
  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   Nodes, Runtime, GPUs, …



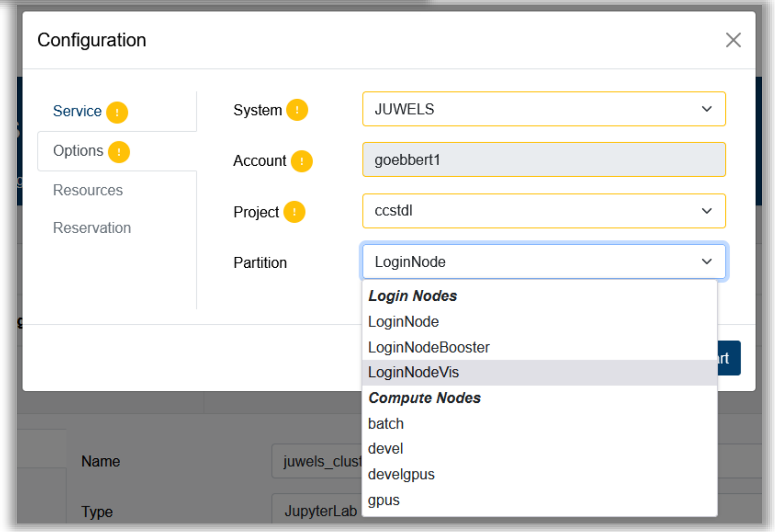

Configuration

Service | Options | Resources | Reservation

Name: Please give your configuration a name.

Type: JupyterLab

Cancel | Start


Configuration

Service | Options | Resources | Reservation

System: JUWELS

Account: goebbert1

Project: ccstdl

Partition: LoginNode

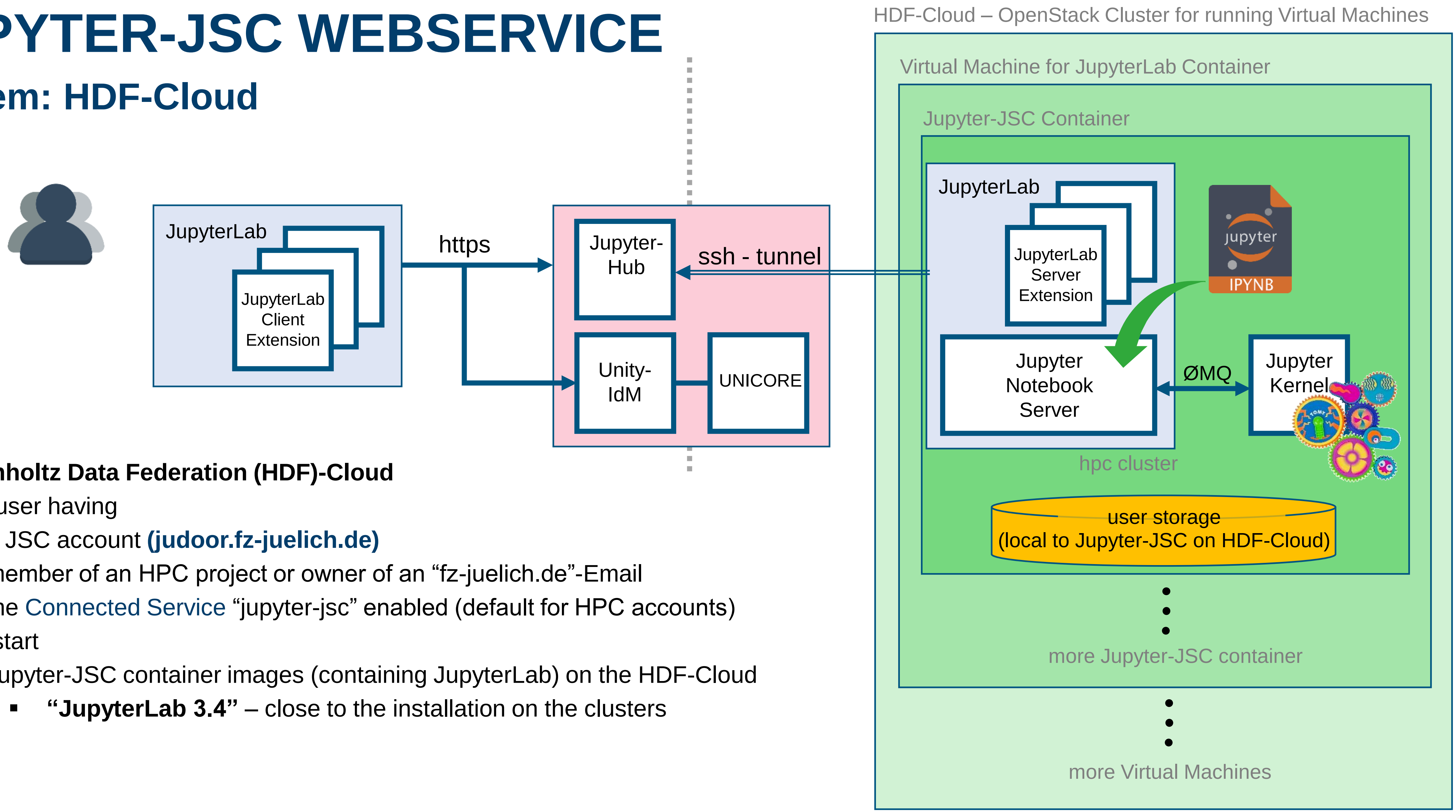
Login Nodes
LoginNode
LoginNodeBooster
LoginNodeVis
Compute Nodes
batch
devel
develgpus
gpus

Name: juwels_clust

Type: JupyterLab

JUPYTER-JSC WEBSERVICE

System: HDF-Cloud


HDF-Cloud – OpenStack Cluster for running Virtual Machines
Virtual Machine for JupyterLab Container
Jupyter-JSC Container
JupyterLab
JupyterLab Client Extension
https
Jupyter-Hub
ssh - tunnel
Unity-IdM
UNICORE
JupyterLab
JupyterLab Server Extension
jupyter
IPYNB
Jupyter Notebook Server
ØMQ
Jupyter Kernel
hpc cluster
user storage
(local to Jupyter-JSC on HDF-Cloud)
more Jupyter-JSC container
more Virtual Machines

Helmholtz Data Federation (HDF)-Cloud

Any user having

- a JSC account **(judoor.fz-juelich.de)**
- member of an HPC project or owner of an “fz-juelich.de”-Email
- the Connected Service “jupyter-jsc” enabled (default for HPC accounts)

can start

- Jupyter-JSC container images (containing JupyterLab) on the HDF-Cloud
  - **“JupyterLab 3.4”** – close to the installation on the clusters

https://www.fz-juelich.de/ias/jsc/EN/Expertise/SciCloudServices/HDFCloud/_node.html

JUPYTER-JSC WEBSERVICE

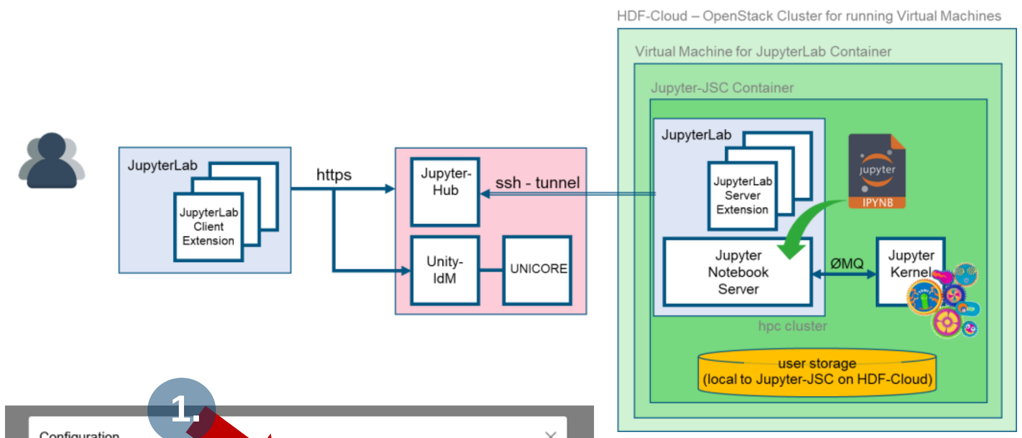
System: HDF-Cloud

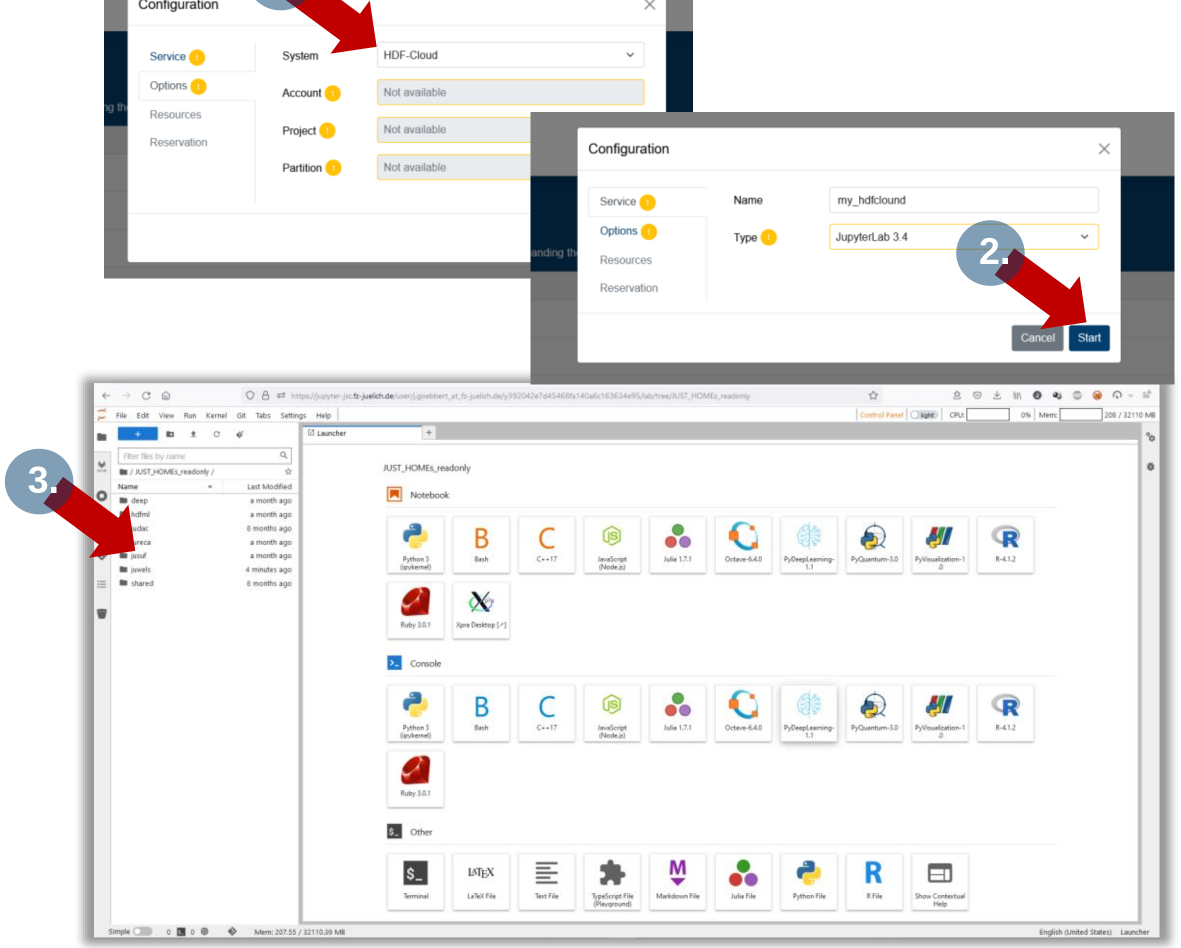
Start JupyterLab on HDF-Cloud

- Requirements:
  - Registered JSC account at judoor.fz-juelich.de
  - Logged in to Jupyter-JSC at jupyter-jsc.fz-juelich.de
  - Named a new JupyterLab configuration
- Start a JupyterLab:
  - Version == “JupyterLab 3.4”
  - System == “HDF-Cloud”

Limitations on JupyterLab on HDF-Cloud

- max. **4 GB** memory
  - ATTENTION: the container automatically stops, when this limit is reached.
- Storage in Jupyter-JSC container
  - **is local** to the HDF-Cloud
  - HPC $HOMEs are mounted read-only
  - only accessible from a Jupyter-JSC container
- HDF-Cloud has **no GPUs**


HDF-Cloud – OpenStack Cluster for running Virtual Machines
Virtual Machine for JupyterLab Container
Jupyter-JSC Container
JupyterLab
JupyterLab Client Extension
https
Jupyter-Hub
ssh - tunnel
Unity-IdM
UNICORE
JupyterLab
JupyterLab Server Extension
jupyter
IPYNB
Jupyter Notebook Server
ØMQ
Jupyter Kernel
hpc cluster
user storage (local to Jupyter-JSC on HDF-Cloud)

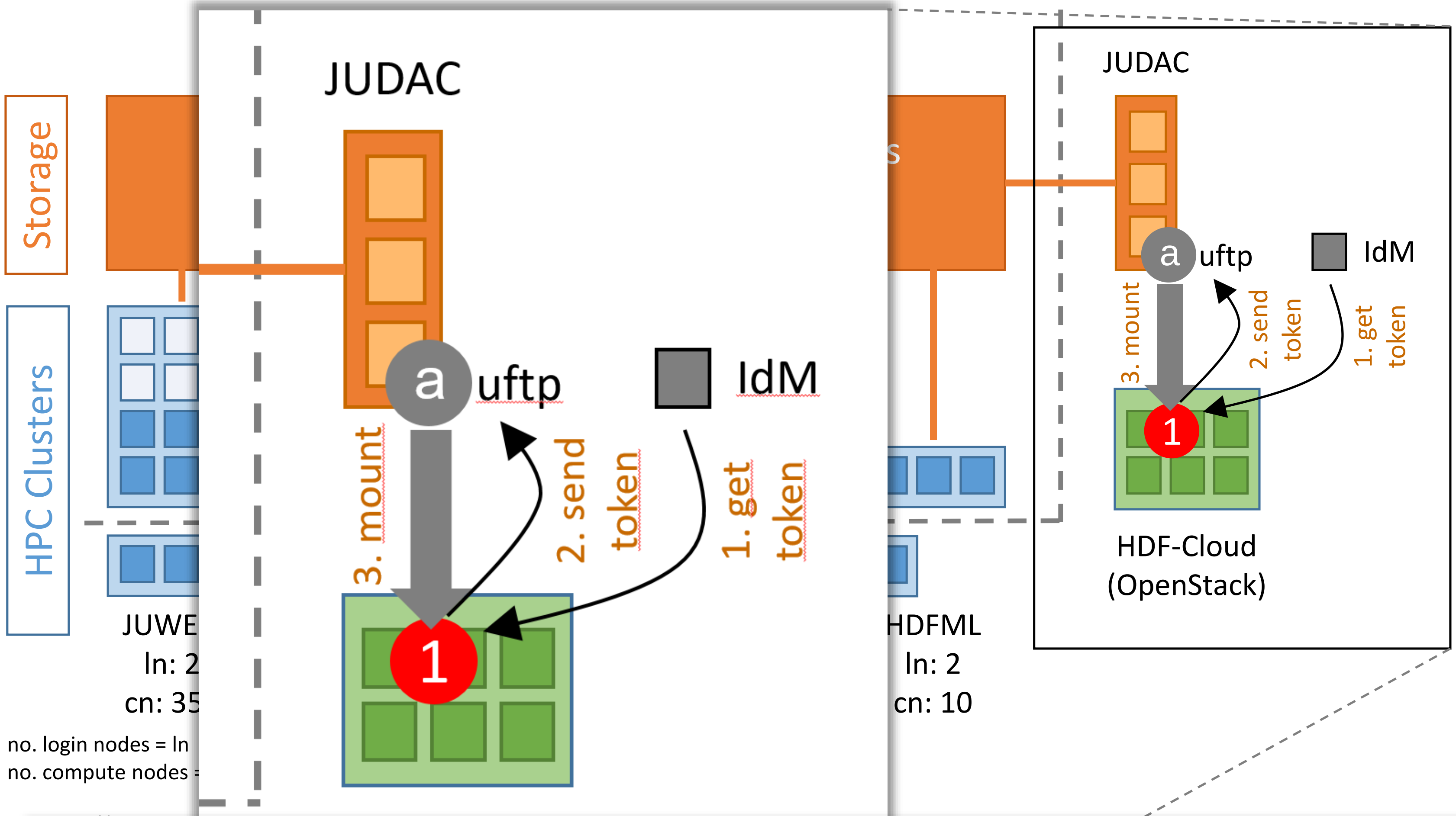


1. Configuration — Service: System: HDF-Cloud; Account: Not available; Project: Not available; Partition: Not available

2. Configuration — Options: Name: my_hdfclound; Type: JupyterLab 3.4; Cancel, Start

3. Launcher — JUST_HOMEs_readonly

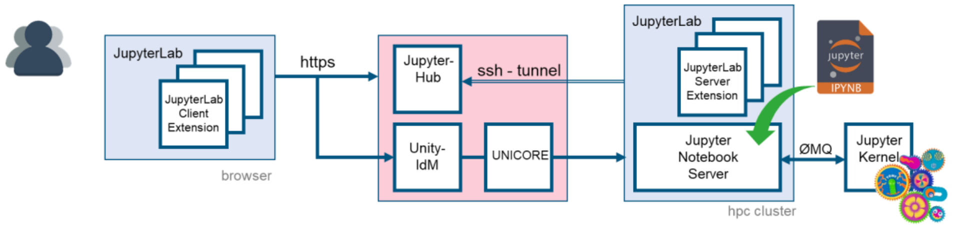
HOW TO MOUNT GPFS ON HDF-CLOUD



https://gitlab.jsc.fz-juelich.de/jupyter4jsc/prace-2022.04-jupyter4hpc/-/blob/main/day_2/2_hpc-environment/1-hdf-cloud_mount-hpc-storage.ipynb

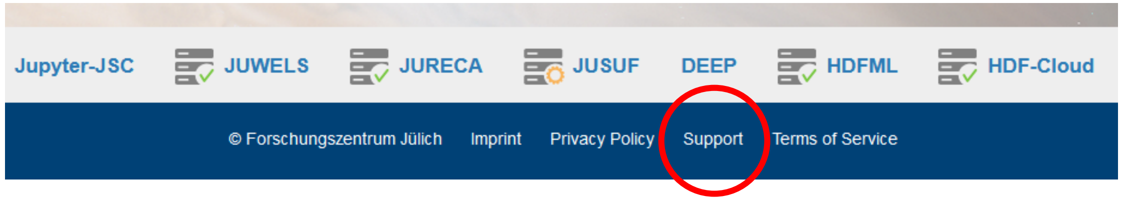
JUPYTER-JSC SECRETS

Very important to know



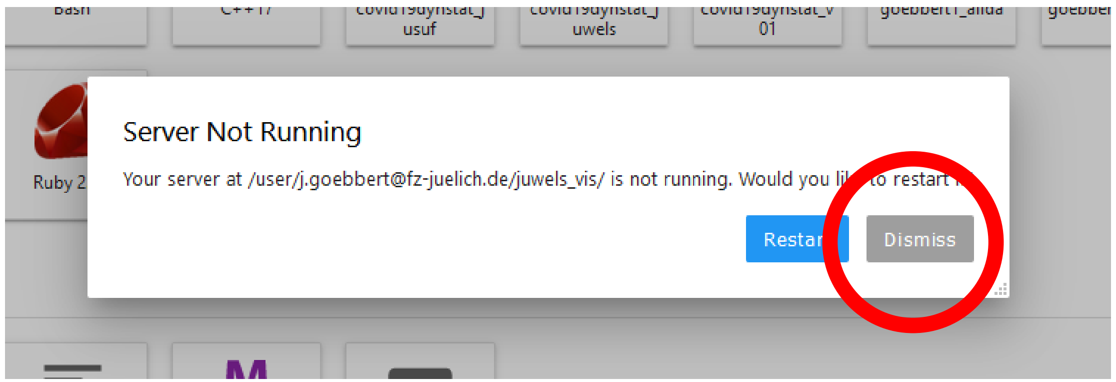
Secret 1: Support button

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



Secret 2: Reload on connection loss

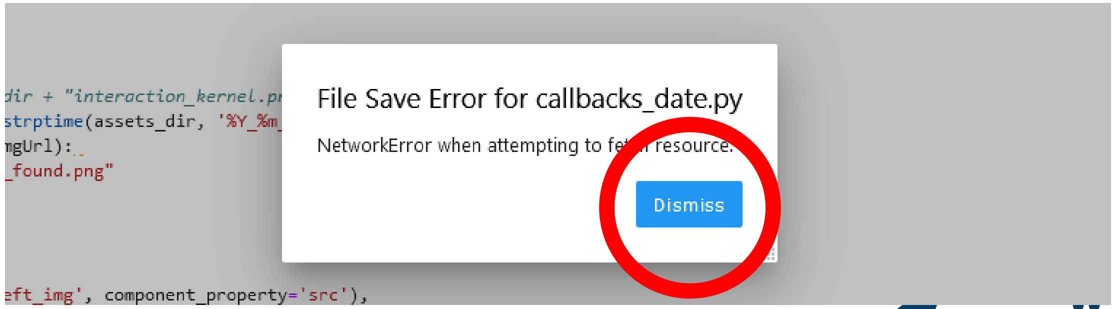
- “Server Not Running”
  means, that your browser just lost connection
  => **Just hit “Dismiss” !!!**
  (as soon as you are online again)



Server Not Running

Your server at /user/j.goebbert@fz-juelich.de/juwels_vis/ is not running. Would you li to restart

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



```
dir + "interaction_kernel.p
strptime(assets_dir, '%Y_%m
ngUrl):
_found.png"
eft_img', component_property='src'),
```

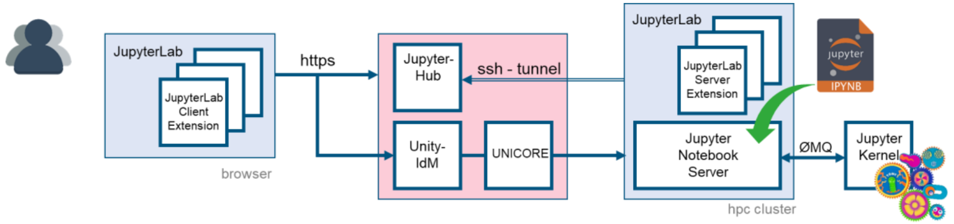
File Save Error for callbacks_date.py

NetworkError when attempting to fe resource.

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.

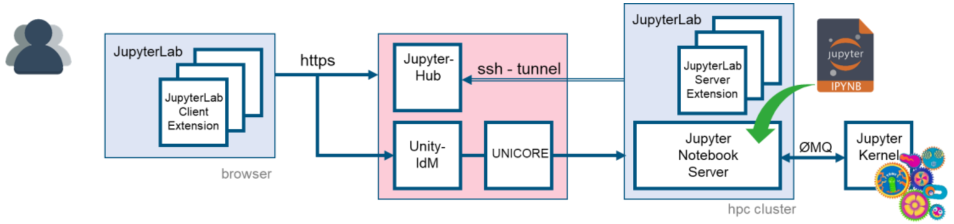
```
#!/bin/bash

module purge
module load Stages/2022
module load GCCcore/.11.2.0
module load JupyterCollection/2022.3.4
```

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.

JUPYTER-JSC WEBSERVICE

Some comments about the UI



open filebrowser

open launcher

type of active notebook cell

no close, but go back to Jupyter-JSC´s controll panel

memory consuption (keep an eye on that!)

tutorials & examples

Type of Jupyter kernel this notebook is connected to (click to change)

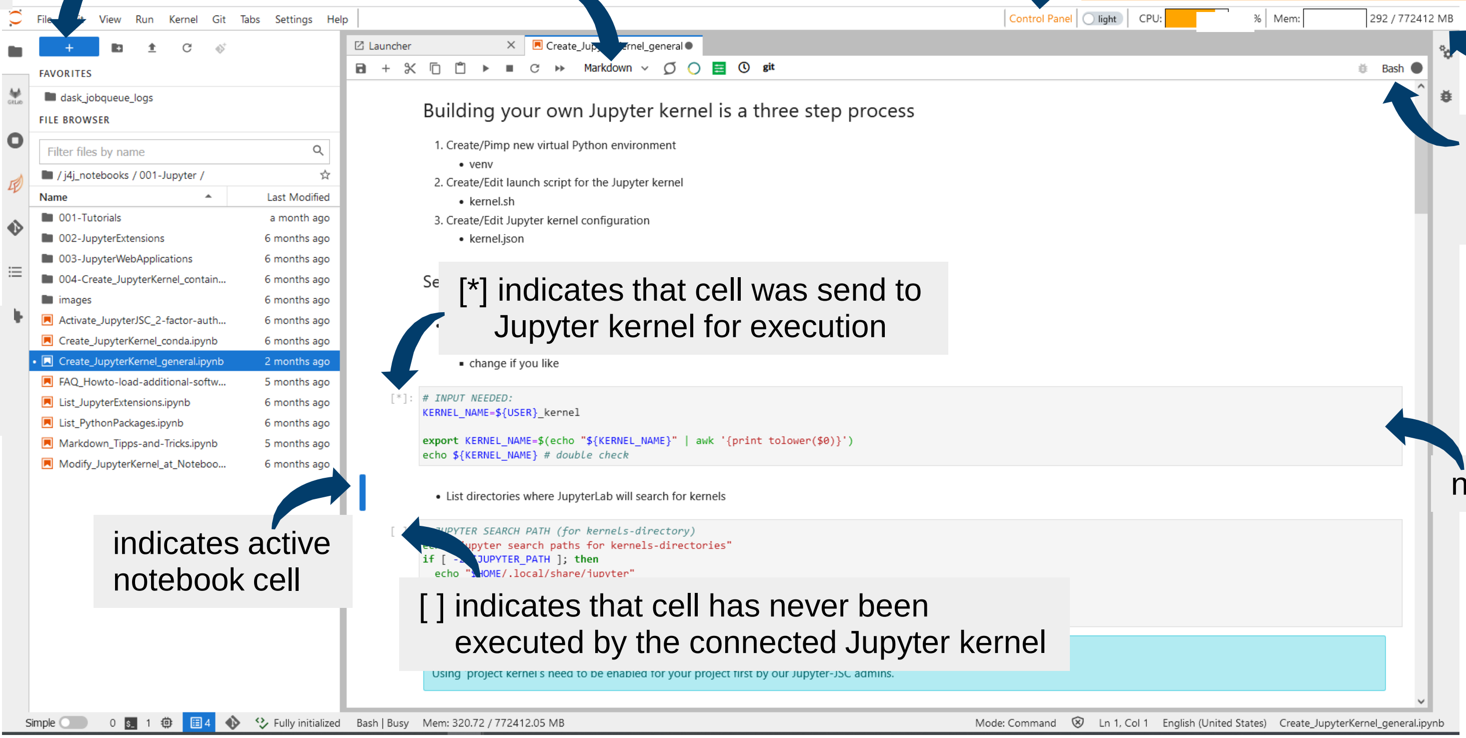
sidebar with core and extentions features

[*] indicates that cell was send to Jupyter kernel for execution

notebook cell

indicates active notebook cell

[ ] indicates that cell has never been executed by the connected Jupyter kernel



dask_jobqueue_logs

/ j4j_notebooks / 001-Jupyter /

Name	Last Modified
001-Tutorials	a month ago
002-JupyterExtensions	6 months ago
003-JupyterWebApplications	6 months ago
004-Create_JupyterKernel_contain...	6 months ago
images	6 months ago
Activate_JupyterJSC_2-factor-auth...	6 months ago
Create_JupyterKernel_conda.ipynb	6 months ago
Create_JupyterKernel_general.ipynb	2 months ago
FAQ_Howto-load-additional-softw...	5 months ago
List_JupyterExtensions.ipynb	6 months ago
List_PythonPackages.ipynb	6 months ago
Markdown_Tipps-and-Tricks.ipynb	5 months ago
Modify_JupyterKernel_at_Noteboo...	6 months ago

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

Se

- change if you like

```
[*]: # INPUT NEEDED:
     KERNEL_NAME=${USER}_kernel

     export KERNEL_NAME=$(echo "${KERNEL_NAME}" | awk '{print tolower($0)}')
     echo ${KERNEL_NAME} # double check
```

- List directories where JupyterLab will search for kernels

```
[    UPYTER SEARCH PATH (for kernels-directory)
     ec   upyter search paths for kernels-directories"
     if [ -  JUPYTER_PATH ]; then
       echo "  HOME/.local/share/jupyter"
```

Using "project kernel's need to be enabled for your project first by our Jupyter-JSC admins.

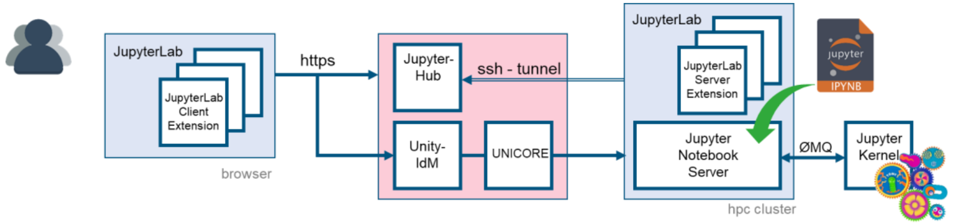
Member of the Helmholtz Association

JÜLICH Forschungszentrum

JUPYTERLAB EXTENSIONS

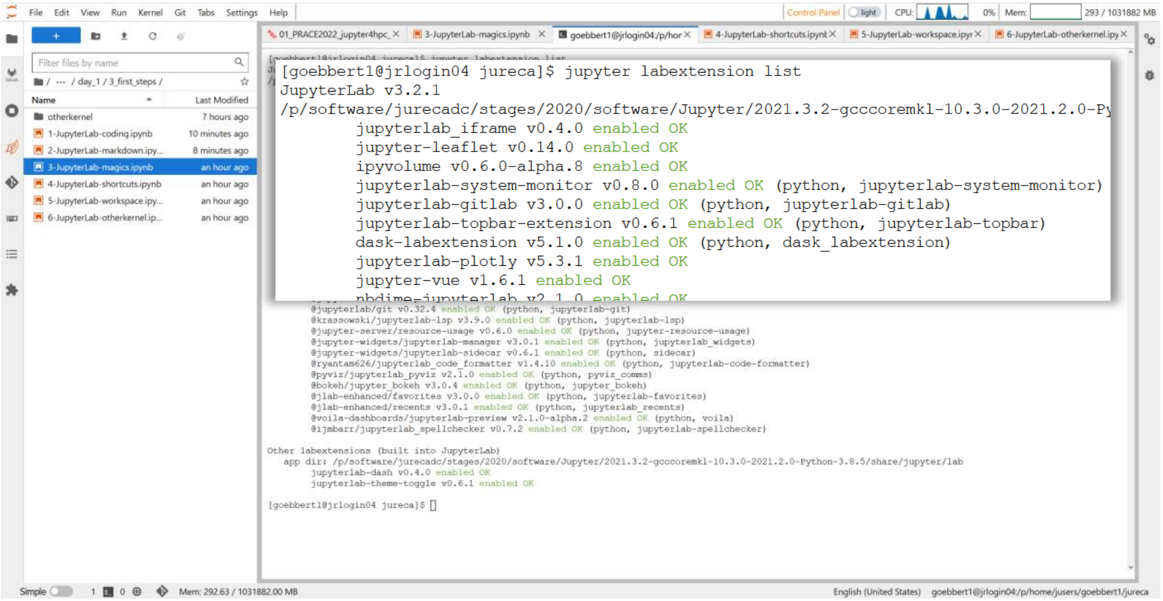
JUPYTERLAB EXTENSIONS

Some general information



List the installed JupyterLab extensions

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


File Edit View Run Kernel Git Tabs Settings Help
Control Panel
light
CPU:
0%
Mem:
293 / 1031882 MB
01_PRACE2022_jupyter4hpc_
3-JupyterLab-magics.ipynb
goebbert1@jrlogin04:/p/hor
4-JupyterLab-shortcuts.ipynb
5-JupyterLab-workspace.ipyr
6-JupyterLab-otherkernel.ipy
Filter files by name
/ ··· / day_1 / 3_first_steps /
Name
Last Modified
otherkernel
7 hours ago
1-JupyterLab-coding.ipynb
10 minutes ago
2-JupyterLab-markdown.ipy...
8 minutes ago
3-JupyterLab-magics.ipynb
an hour ago
4-JupyterLab-shortcuts.ipynb
an hour ago
5-JupyterLab-workspace.ipy...
an hour ago
6-JupyterLab-otherkernel.ip...
an hour ago
Simple
1
0
Mem: 292.63 / 1031882.00 MB
English (United States)
goebbert1@jrlogin04:/p/home/jusers/goebbert1/jureca

```
[goebbert1@jrlogin04 jureca]$ jupyter labextension list
JupyterLab v3.2.1
/p/software/jurecadc/stages/2020/software/Jupyter/2021.3.2-gcccoremkl-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
        nbdime-jupyterlab v2.1.0 enabled OK
        @jupyterlab/git v0.32.4 enabled OK (python, jupyterlab-git)
        @krassowski/jupyterlab-lsp v3.9.0 enabled OK (python, jupyterlab-lsp)
        @jupyter-server/resource-usage v0.6.0 enabled OK (python, jupyter-resource-usage)
        @jupyter-widgets/jupyterlab-manager v3.0.1 enabled OK (python, jupyterlab_widgets)
        @jupyter-widgets/jupyterlab-sidecar v0.6.1 enabled OK (python, sidecar)
        @ryantam626/jupyterlab_code_formatter v1.4.10 enabled OK (python, jupyterlab-code-formatter)
        @pyviz/jupyterlab_pyviz v2.1.0 enabled OK (python, pyviz_comms)
        @bokeh/jupyter_bokeh v3.0.4 enabled OK (python, jupyter_bokeh)
        @jlab-enhanced/favorites v3.0.0 enabled OK (python, jupyterlab-favorites)
        @jlab-enhanced/recents v3.0.1 enabled OK (python, jupyterlab_recents)
        @voila-dashboards/jupyterlab-preview v2.1.0-alpha.2 enabled OK (python, voila)
        @ijmbarr/jupyterlab_spellchecker v0.7.2 enabled OK (python, jupyterlab-spellchecker)

Other labextensions (built into JupyterLab)
   app dir: /p/software/jurecadc/stages/2020/software/Jupyter/2021.3.2-gcccoremkl-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@jrlogin04 jureca]$ 
```

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

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

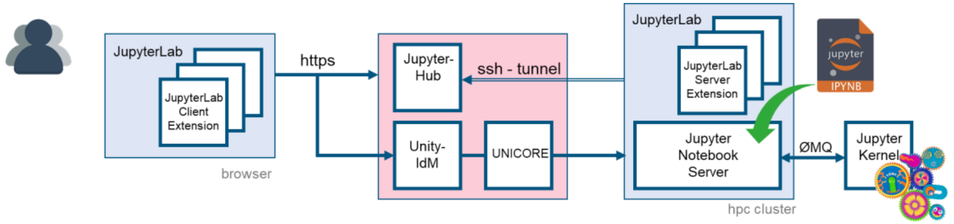
Hint: JupyterLab Playground

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

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

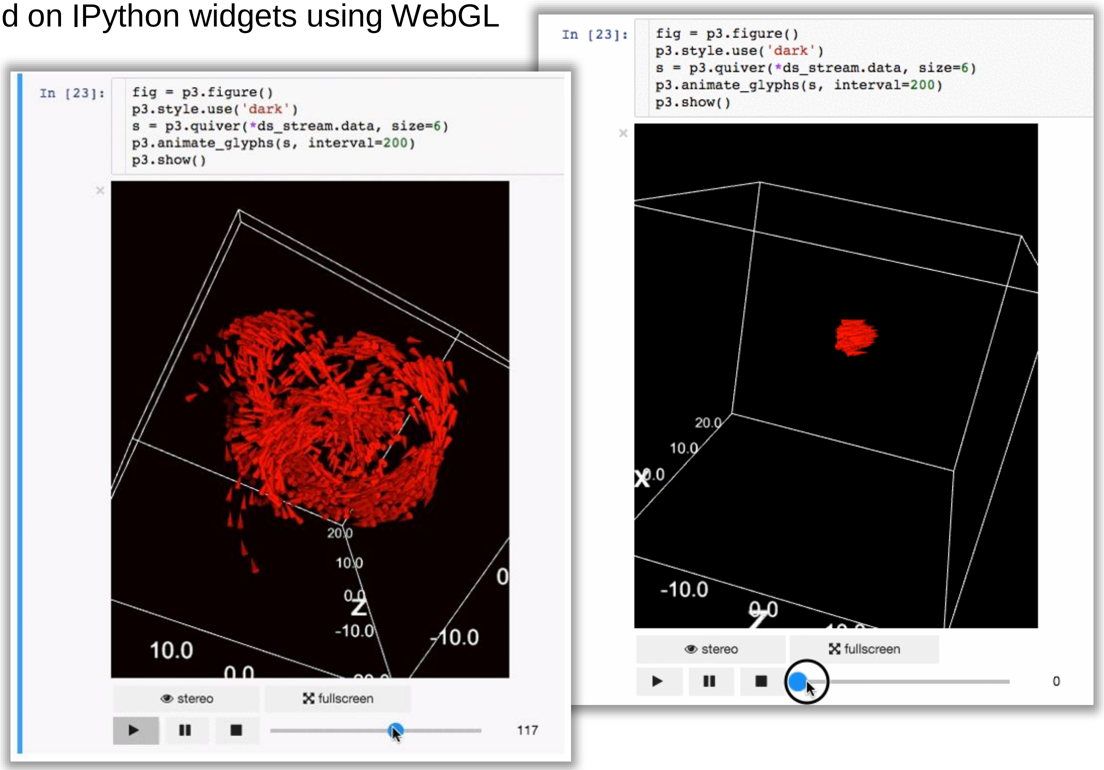
JUPYTERLAB EXTENSIONS

Installed by default at Jupyter-JSC



IPyVolume

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

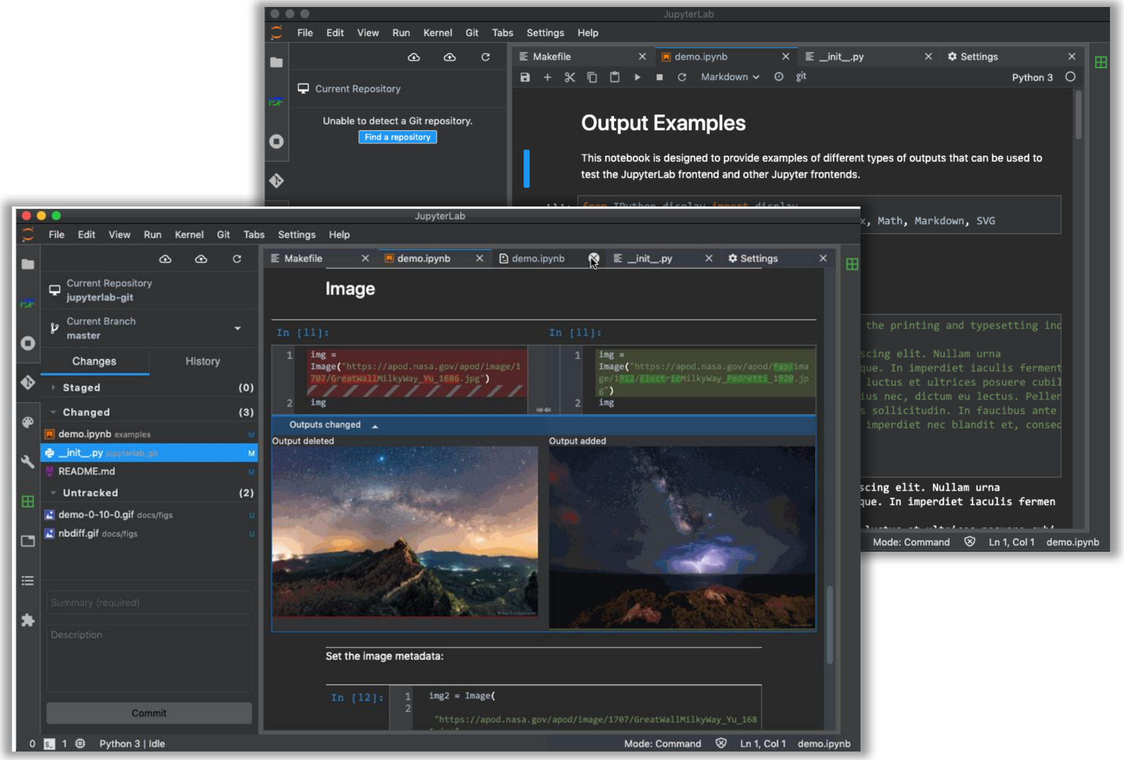


```
In [23]:  fig = p3.figure()
          p3.style.use('dark')
          s = p3.quiver(*ds_stream.data, size=6)
          p3.animate_glyphs(s, interval=200)
          p3.show()
```

https://github.com/maartenbreddels/ipyvolume

JupyterLab-Git

JupyterLab extension for version control using Git



Current Repository

Unable to detect a Git repository.

Output Examples

This notebook is designed to provide examples of different types of outputs that can be used to test the JupyterLab frontend and other Jupyter frontends.

Current Repository
jupyterlab-git

Current Branch
master

Staged (0)

Changed (3)
- demo.ipynb examples M
- __init__.py jupyterlab_git M
- README.md M

Untracked (2)
- demo-0-10-0.gif docs/figs U
- nbdiff.gif docs/figs U

Image

```
In [11]:
img =
Image("https://apod.nasa.gov/apod/image/1707/GreatWallMilkyWay_Yu_1686.jpg")
img
```

```
In [11]:
img =
Image("https://apod.nasa.gov/apod/fap/image/1912/ElectricMilkyWay_Pedretti_1920.jpg")
img
```

Outputs changed

Output deleted

Output added

Set the image metadata:

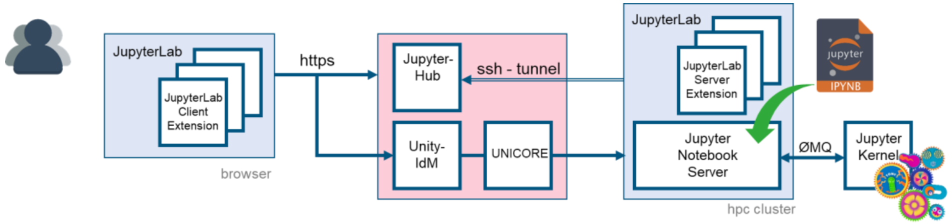
```
In [12]:  img2 = Image(
              "https://apod.nasa.gov/apod/image/1707/GreatWallMilkyWay_Yu_168
```

0 1 Python 3 | Idle   Mode: Command   Ln 1, Col 1   demo.ipynb

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

JUPYTERLAB EXTENSIONS

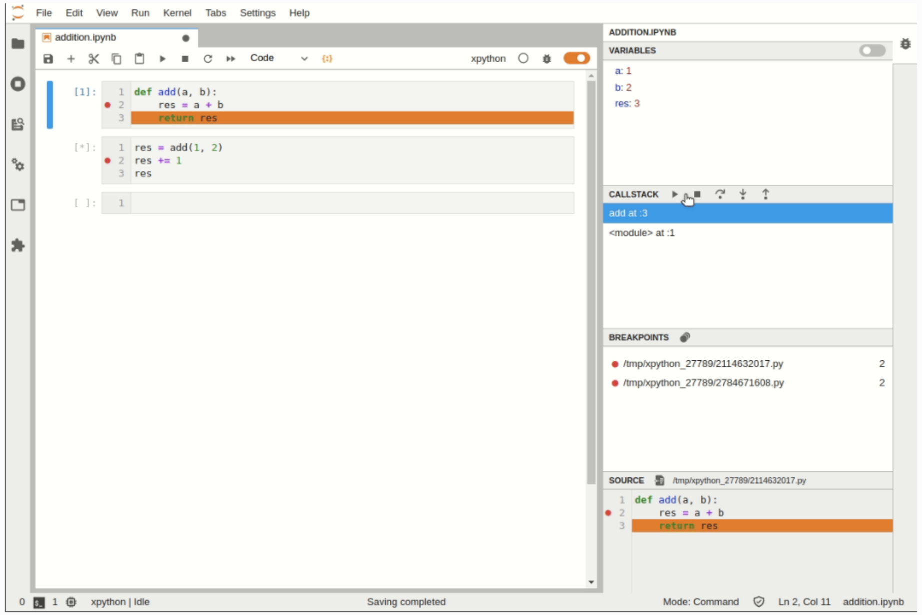
Installed by default at Jupyter-JSC



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.



```
[1]: def add(a, b):
         res = a + b
         return res
```

```
[*]: res = add(1, 2)
     res += 1
     res
```

```
[ ]:
```

a: 1
b: 2
res: 3

add at :3
<module> at :1

/tmp/xpython_27789/2114632017.py 2
/tmp/xpython_27789/2784671608.py 2

SOURCE /tmp/xpython_27789/2114632017.py

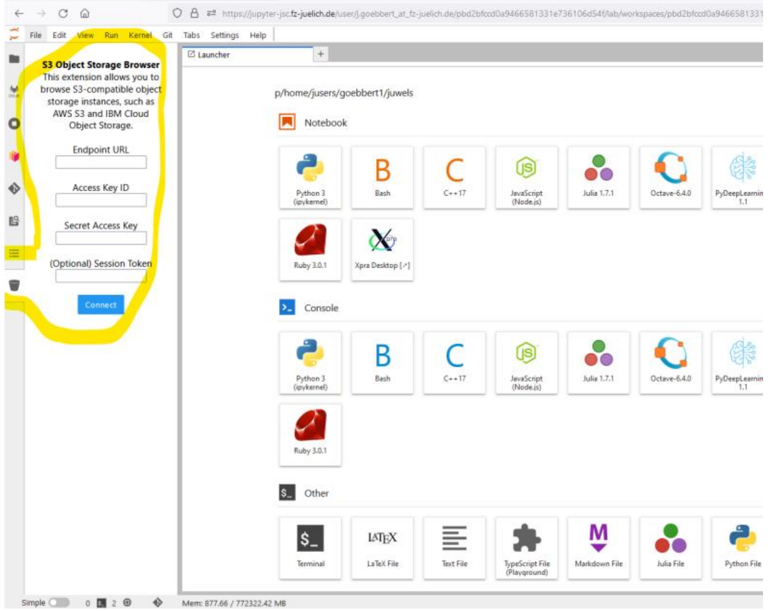
```
def add(a, b):
    res = a + b
    return res
```

0 1 xpython | Idle Saving completed Mode: Command Ln 2, Col 11 addition.ipynb

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

JupyterLab-S3-browser

A JupyterLab extension for browsing S3-compatible object storage



S3 Object Storage Browser

This extension allows you to browse S3-compatible object storage instances, such as AWS S3 and IBM Cloud Object Storage.

Endpoint URL

Access Key ID

Secret Access Key

(Optional) Session Token

Connect

p/home/jusers/goebbert1/juwels

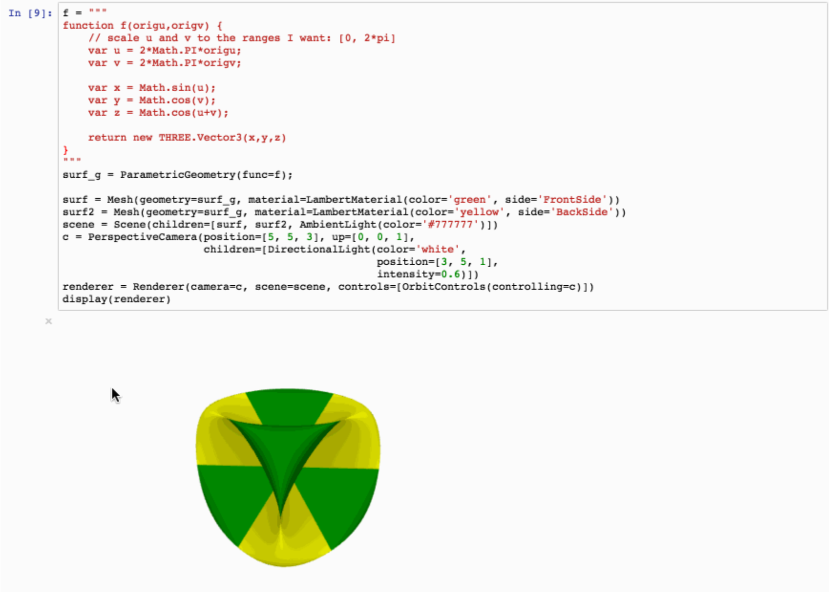
Mem: 877.66 / 772322.42 MB

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.

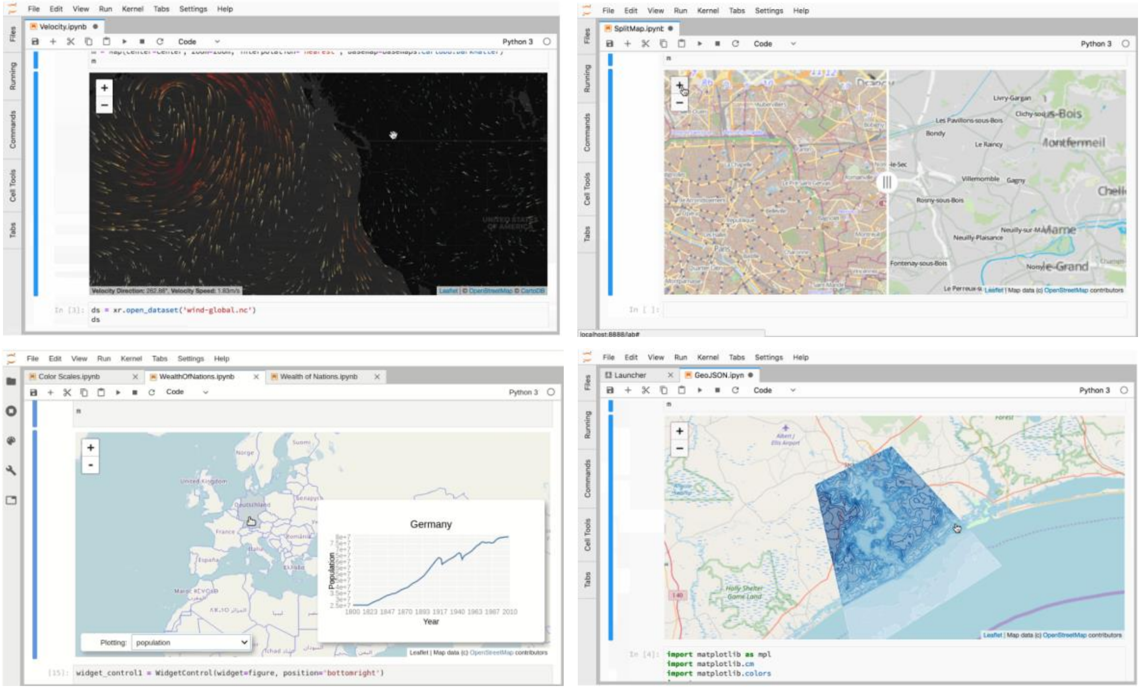


```
In [9]: f = """
function f(origu,origv) {
    // scale u and v to the ranges I want: [0, 2*pi]
    var u = 2*Math.PI*origu;
    var v = 2*Math.PI*origv;

    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.


File Edit View Run Kernel Tabs Settings Help
Velocity.ipynb
SplitMap.ipynb
Color Scales.ipynb
WealthOfNations.ipynb
Wealth of Nations.ipynb
Launcher
GeoJSON.ipynb
Code
Python 3
Velocity Direction: 262.88°, Velocity Speed: 1.83m/s
Leaflet | © OpenStreetMap © CartoDB
Germany
Population
Year
Plotting: population
Leaflet | Map data (c) OpenStreetMap contributors

```
In [3]: ds = xr.open_dataset('wind-global.nc')
        ds
```

```
[15]: widget_control1 = WidgetControl(widget=figure, position='bottomright')
```

```
In [4]: import matplotlib as mpl
        import matplotlib.cm
        import matplotlib.colors
```

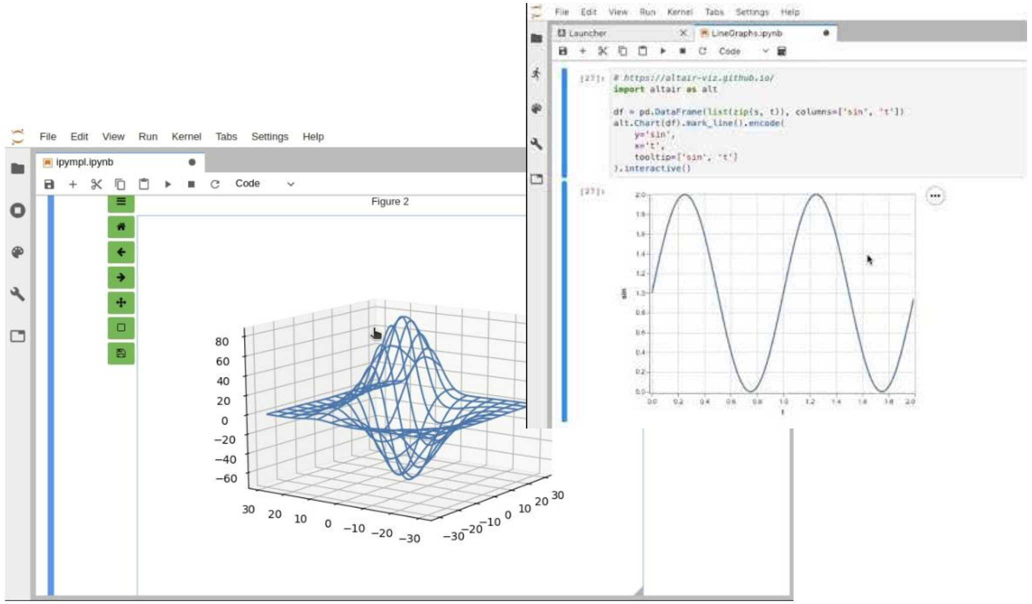
https://github.com/jupyter-widgets/ipyleaflet

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.

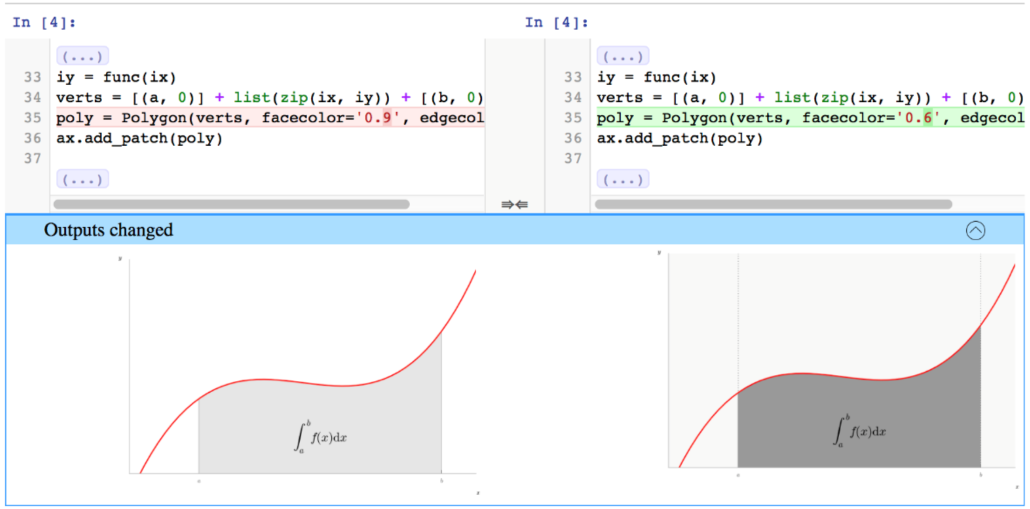


```
# https://altair-viz.github.io/
import altair as alt

df = pd.DataFrame(list(zip(s, t)), columns=['sin', 't'])
alt.Chart(df).mark_line().encode(
    y='sin',
    x='t',
    tooltip=['sin', 't']
).interactive()
```

https://github.com/matplotlib/ipympl

NBDime

Tools for diffing and merging of Jupyter notebooks.



```
iy = func(ix)
verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)
poly = Polygon(verts, facecolor='0.9', edgecol
ax.add_patch(poly)
```

```
iy = func(ix)
verts = [(a, 0)] + list(zip(ix, iy)) + [(b, 0)
poly = Polygon(verts, facecolor='0.6', edgecol
ax.add_patch(poly)
```

https://github.com/jupyter/nbdime

JUPYTERLAB EXTENSIONS

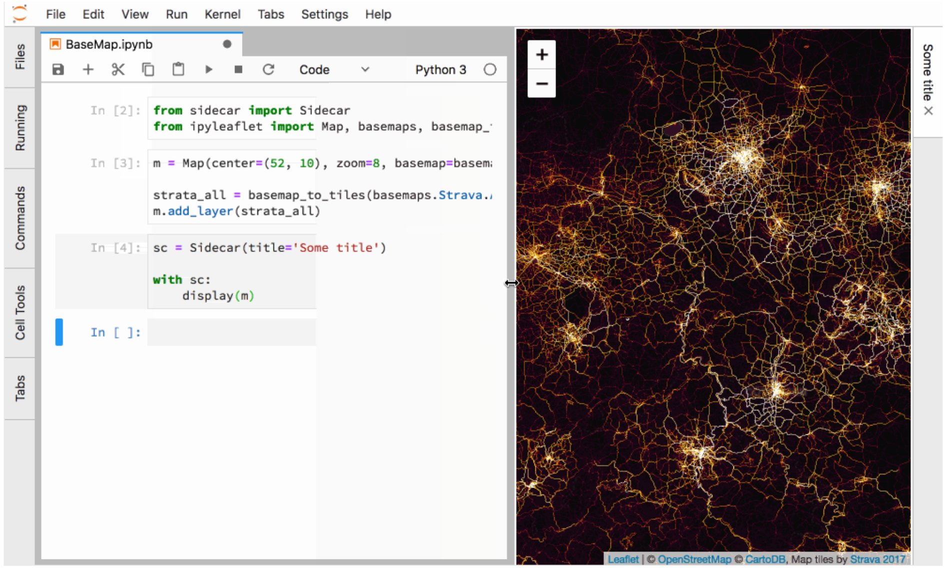
Installed by default at Jupyter-JSC

Plotly
JupyterLab extension for the interactive and browser-based graphing library Plotly.
https://plotly.com/python/

(−0.4845348, 0.3549026)
plotly
Co
Cobalt
Atomic Mass: 58.933195
As
Rh Pd Ag Cd In Sn Sb
Ir Pt Au Hg Tl Pb Bi
Mt Ds Rg Cn Uut Fl Uup
Eu Gd Tb Dy Ho Er Tm
Am Cm Bk Cf Es Fm Md

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

JupyterLab-Sidecar
A sidecar output widget for JupyterLab.

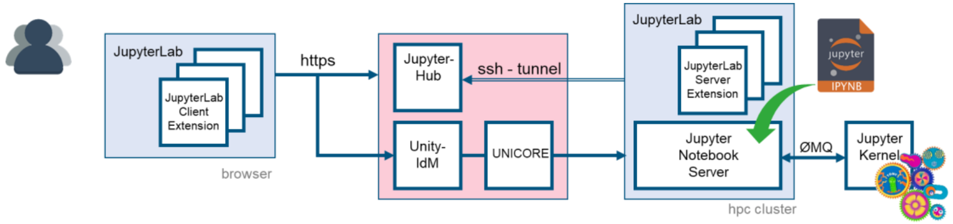


```
In [2]: from sidecar import Sidecar
        from ipyleaflet import Map, basemaps, basemap_

In [3]: m = Map(center=(52, 10), zoom=8, basemap=basema

        strata_all = basemap_to_tiles(basemaps.Strava.
        m.add_layer(strata_all)

In [4]: sc = Sidecar(title='Some title')

        with sc:
            display(m)

In [ ]:
```

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

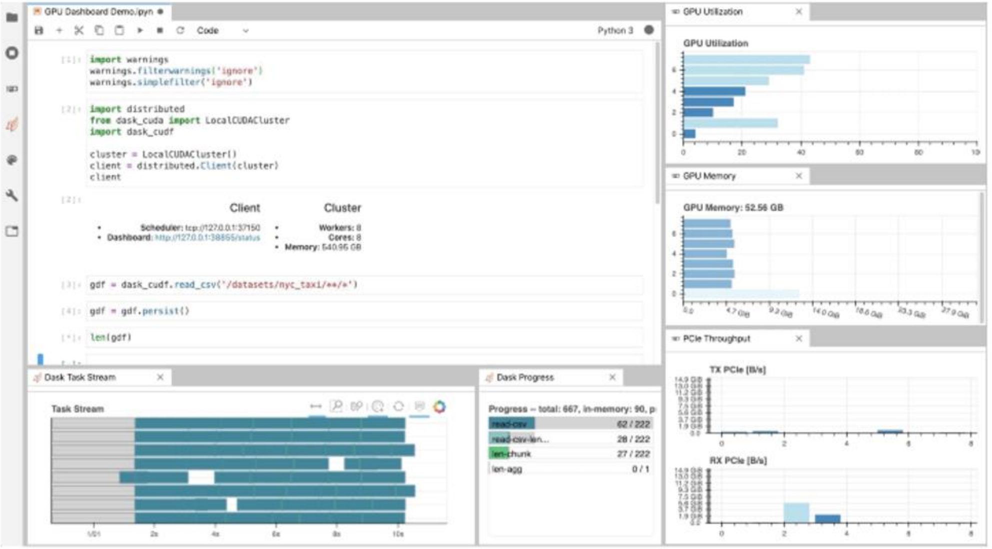
JUPYTERLAB EXTENSIONS

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.



```
[1]: import warnings
     warnings.filterwarnings('ignore')
     warnings.simplefilter('ignore')

[2]: import distributed
     from dask_cuda import LocalCUDACluster
     import dask_cudf

     cluster = LocalCUDACluster()
     client = distributed.Client(cluster)
     client
```

[2]:

Client
- Scheduler: tcp://127.0.0.1:37150
- Dashboard: http://127.0.0.1:38865/status

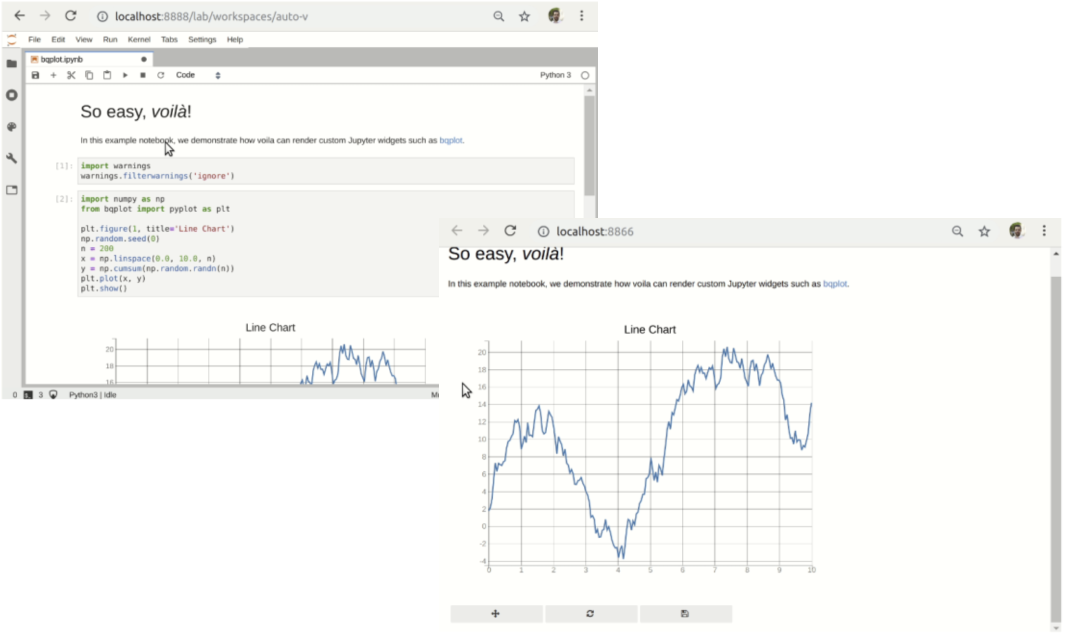
Cluster
- Workers: 8
- Cores: 8
- Memory: 540.96 GB

```
[3]: gdf = dask_cudf.read_csv('/datasets/nyc_taxi/**/*')

[4]: gdf = gdf.persist()

[*]: len(gdf)
```

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.



So easy, *voilà*!

In this example notebook, we demonstrate how voila can render custom Jupyter widgets such as bqplot.

```
[1]: import warnings
     warnings.filterwarnings('ignore')

[2]: import numpy as np
     from bqplot import pyplot as plt

     plt.figure(1, title='Line Chart')
     np.random.seed(0)
     n = 200
     x = np.linspace(0.0, 10.0, n)
     y = np.cumsum(np.random.randn(n))
     plt.plot(x, y)
     plt.show()
```

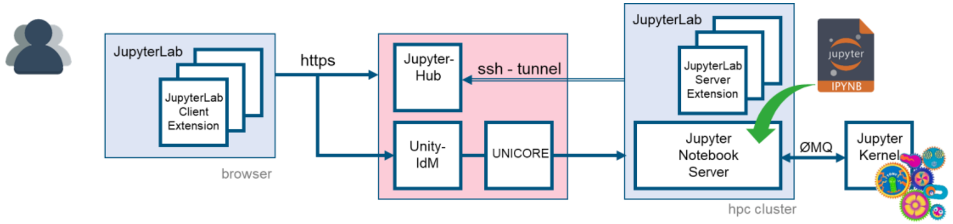
So easy, *voilà*!

In this example notebook, we demonstrate how voila can render custom Jupyter widgets such as bqplot.

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

JUPYTERLAB EXTENSIONS

Installed by default at Jupyter-JSC

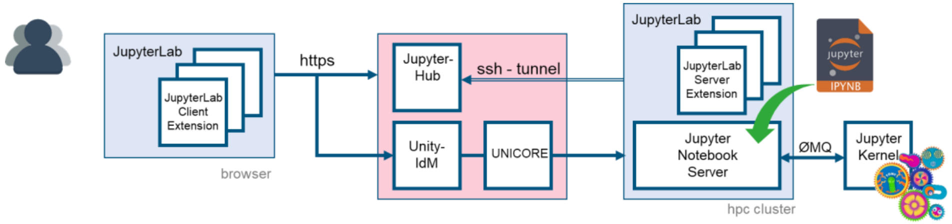


Extensions	old version	new version	type
Core			
@jupyterlab/server-proxy	v2.1.0	v3.1.0	prebuild
@jupyter-widgets/jupyterlab-manager	v2.0.0	v3.0.1	prebuild
jupyterlab-datawidgets	v6.3.0	v7.0.0	source
UI Enhancements			
@jlab-enhanced/recents		v3.0.1	prebuild
@jlab-enhanced/favorites	v2.0.0	v3.0.0	prebuild
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
@ijmbarr/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 Developement			
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

JUPYTER KERNEL

JUPYTER KERNEL

How to create your own Juypter Kernel

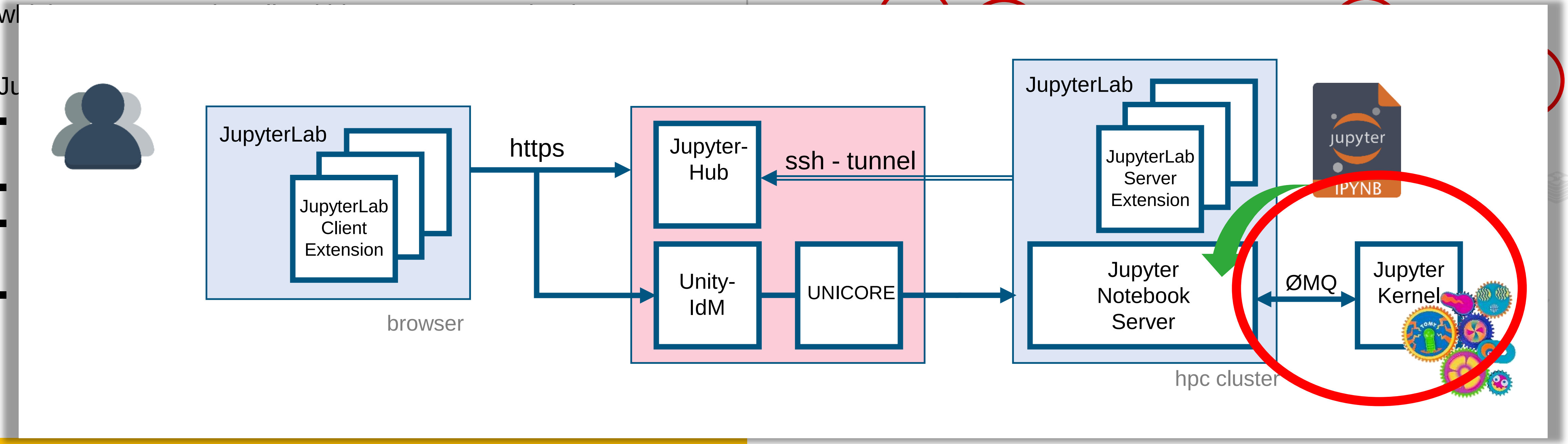


Jupyter Kernel

A “kernel” refers to the separate process

wl

Ju

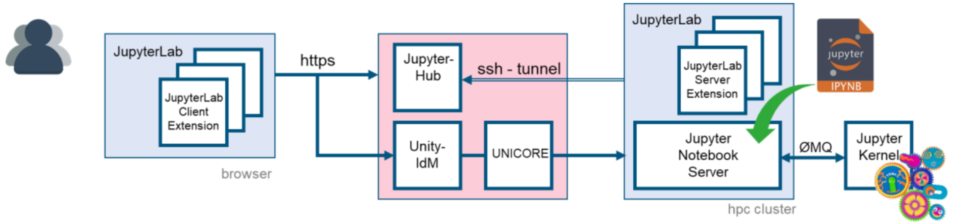


You can easily **create your own kernel** which for example runs your specialized virtual Python environment.

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

JUPYTER KERNEL

How to create your own Juypter Kernel

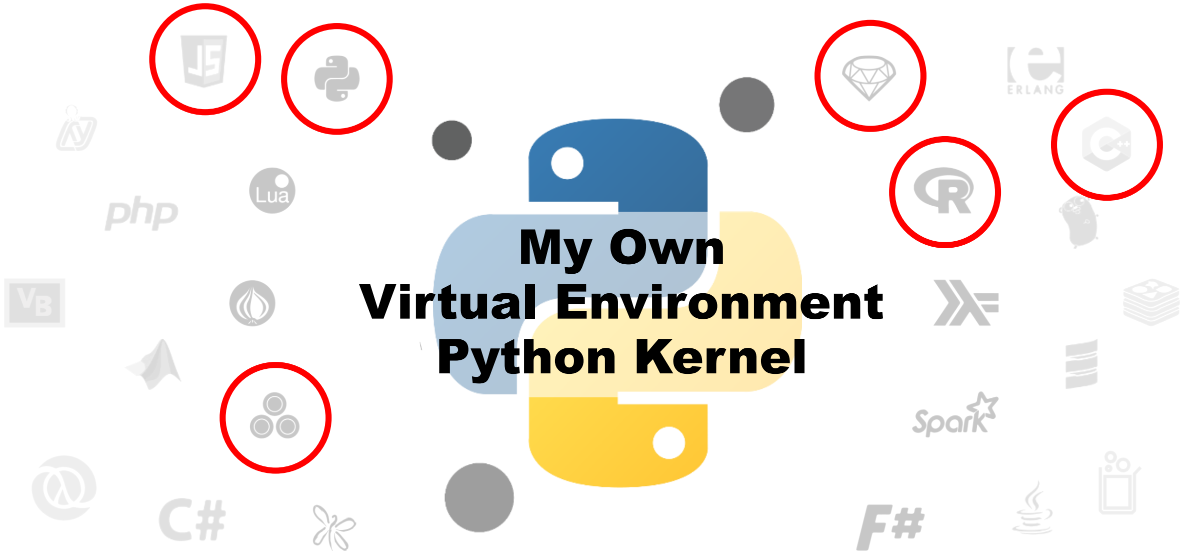


Jupyter Kernel
A “kernel” refers to the separate process
which executes code cells within a Jupyter notebook.

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



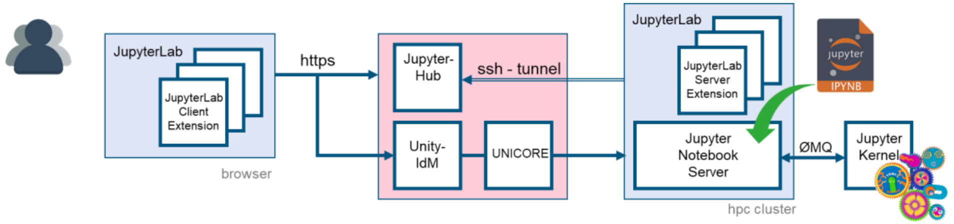
My Own
Virtual Environment
Python Kernel

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

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

JUPYTER KERNEL

How to create your own Juypter 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

Building your own Jupyter kernel
is a three step process

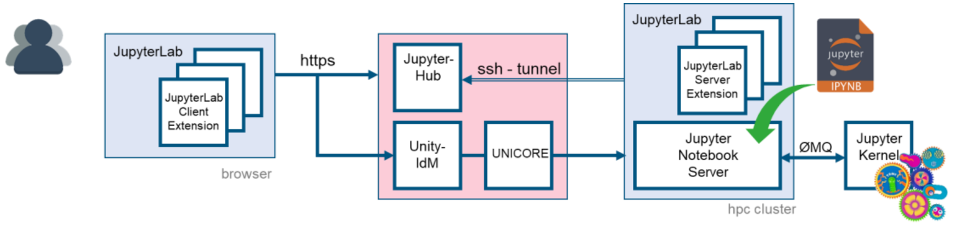
1. Create/Pimp new **virtual Python environment**
   `venv`
2. Create/Edit **launch script** for the Jupyter kernel
   `kernel.sh`
3. Create/Edit Jupyter **kernel configuration**
   `kernel.json`

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

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

JUPYTER KERNEL

How to create your own Juypter 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)

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

clusters

- Python, R, Julia, Bash, C++, Ruby, JavaScript
- Specialized kernels for visualization, quantum computing

Building your own Jupyter kernel is a three step process

1.Create/Pimp new **virtual Python environment**

```
venv
```

```
kernel.json
```

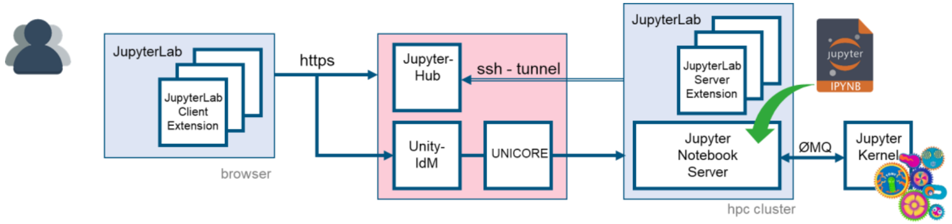
You can easily **create your own kernel** which for example runs your specialized virtual Python environment.

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

Member of the Helmholtz Association

JUPYTER KERNEL

Run your Jupyter kernel configuration

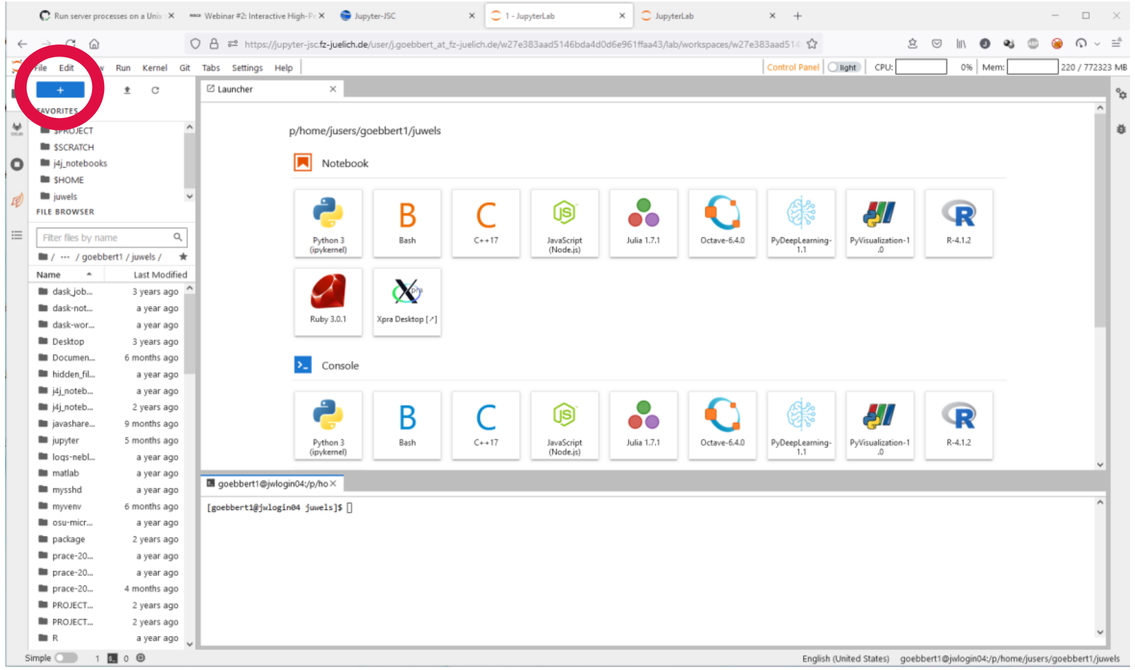


Run your Jupyter Kernel

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.

Conda

How to 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



Name	Last Modified
dask_job...	3 years ago
dask-not...	a year ago
dask-wor...	a year ago
Desktop	3 years ago
Documen...	6 months ago
hidden_fil...	a year ago
j4j_noteb...	a year ago
j4j_noteb...	2 years ago
javashare...	9 months ago
jupyter	5 months ago
logs-nebl...	a year ago
matlab	a year ago
mysshd	a year ago
myvenv	6 months ago
osu-micr...	a year ago
package	2 years ago
prace-20...	a year ago
prace-20...	a year ago
prace-20...	4 months ago
PROJECT...	2 years ago
PROJECT...	2 years ago
R	a year ago

```
[goebbert1@jwlogin04 juwels]$ 
```

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

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.

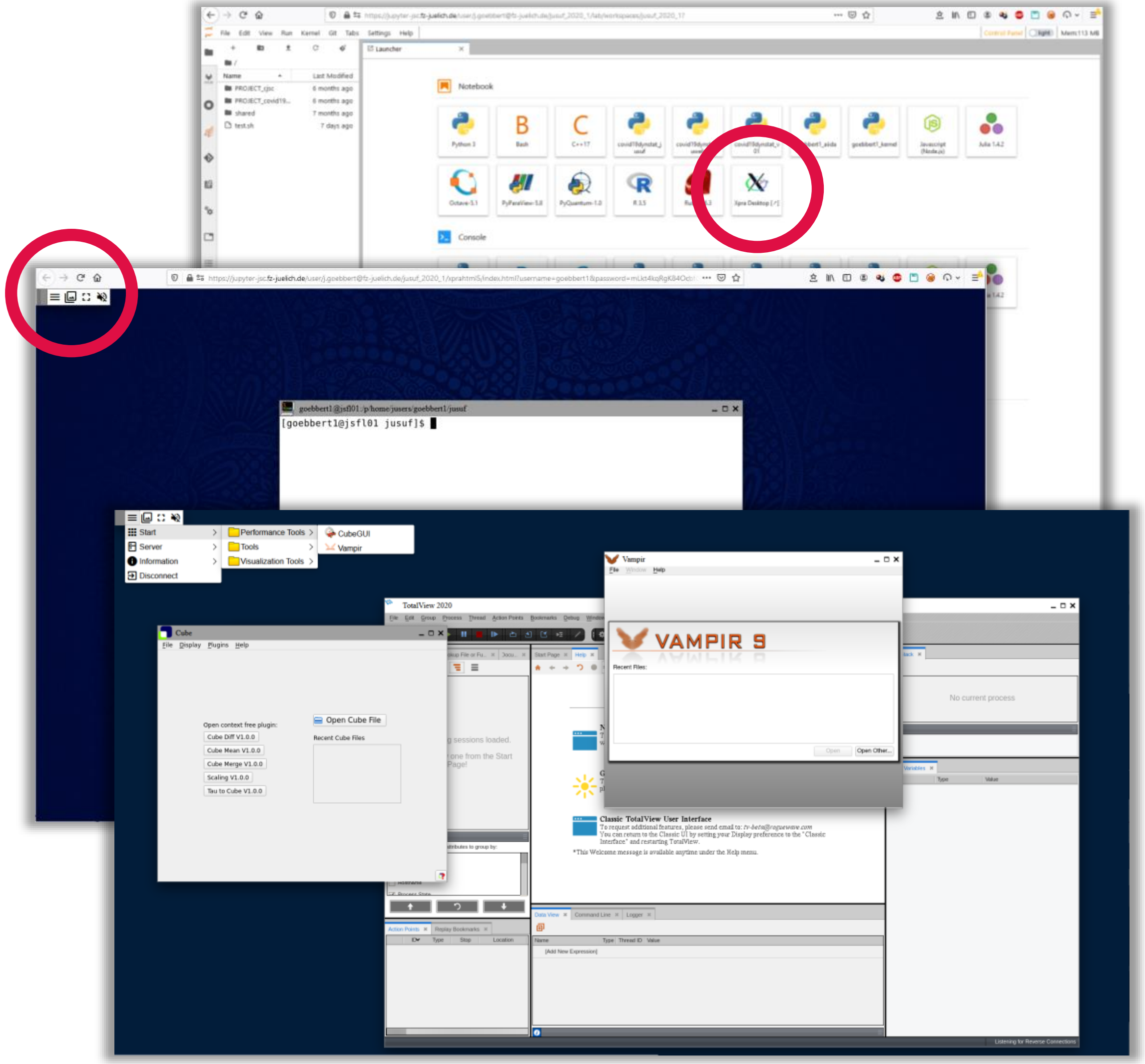
- Runs in a browser
- allows dis-/reconnection without disrupting the forwarded application
- https://xpra.org

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`



```
goebbert1@jsfl01:/p/home/jusers/goebbert1/jusuf
[goebbert1@jsfl01 jusuf]$
```

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”

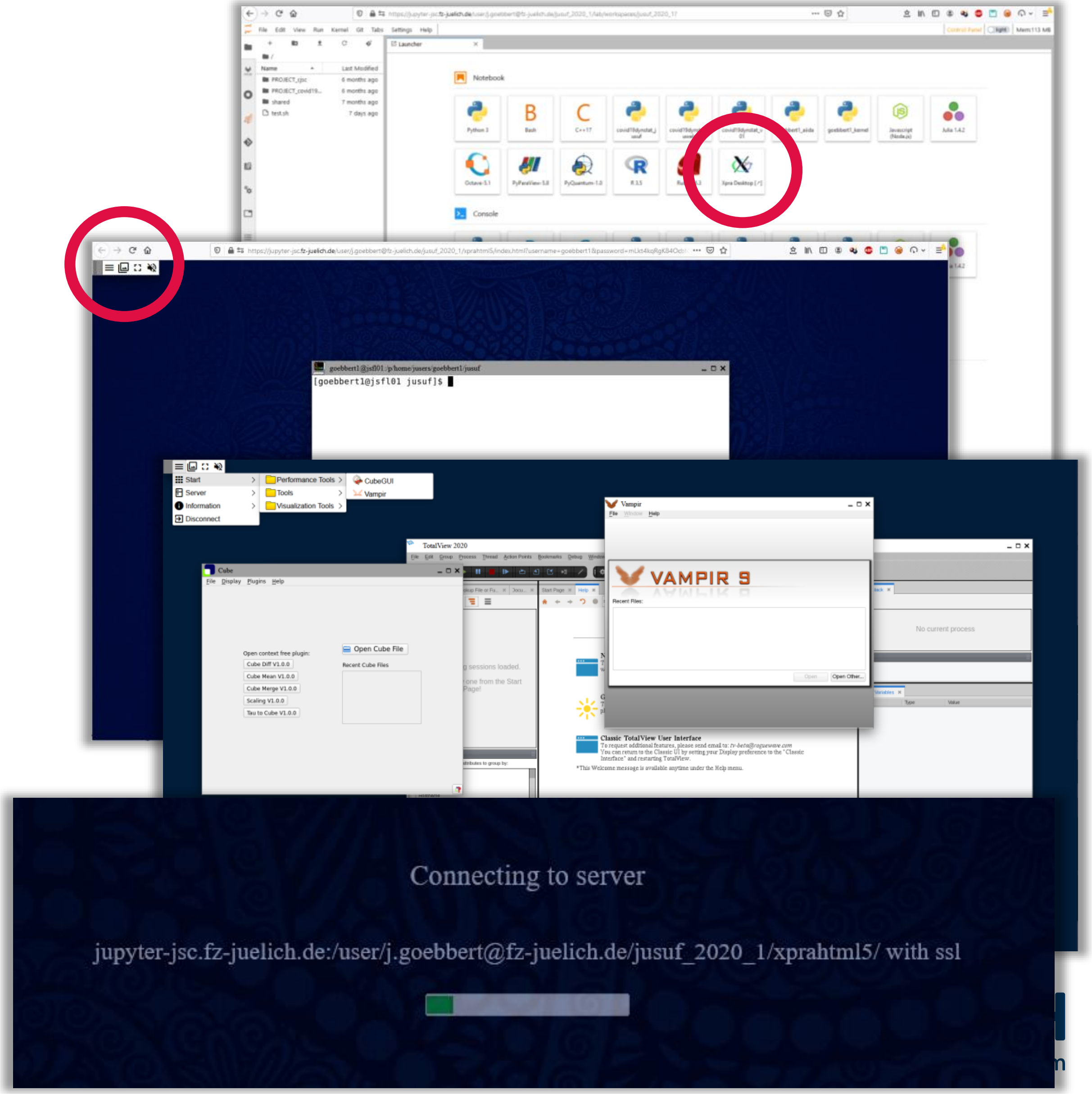
Xpra - X Persistent Remote Applications
is a tool which runs X clients on a remote host and directs their display to the local machine.

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

**If the connection got lost at some point,
just hit the “reload” button of your browser.**

Hint:

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



goebbert1@jsfl01:/p/home/jusers/goebbert1/jusuf

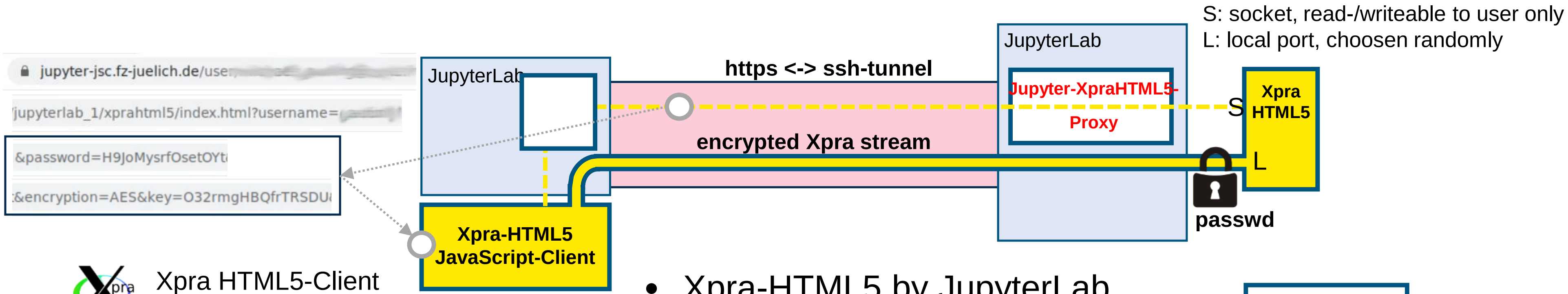
```
[goebbert1@jsfl01 jusuf]$
```

Connecting to server

jupyter-jsc.fz-juelich.de:/user/j.goebbert@fz-juelich.de/jusuf_2020_1/xprahtml5/ with ssl

JUPYTERLAB – REMOTE DESKTOP

Run your X11-Applications in the browser



jupyter-jsc.fz-juelich.de/user/[redacted]

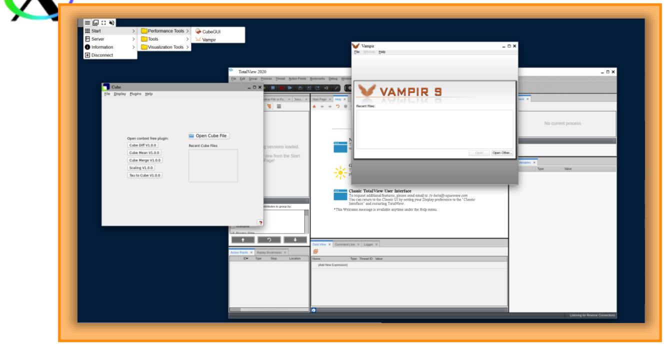
jupyterlab_1/xprahtml5/index.html?username=[redacted]

&password=H9JoMysrfOsetOYt

&encryption=AES&key=O32rmgHBQfrTRSDU

S: socket, read-/writeable to user only
L: local port, choosen randomly

Xpra HTML5-Client



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

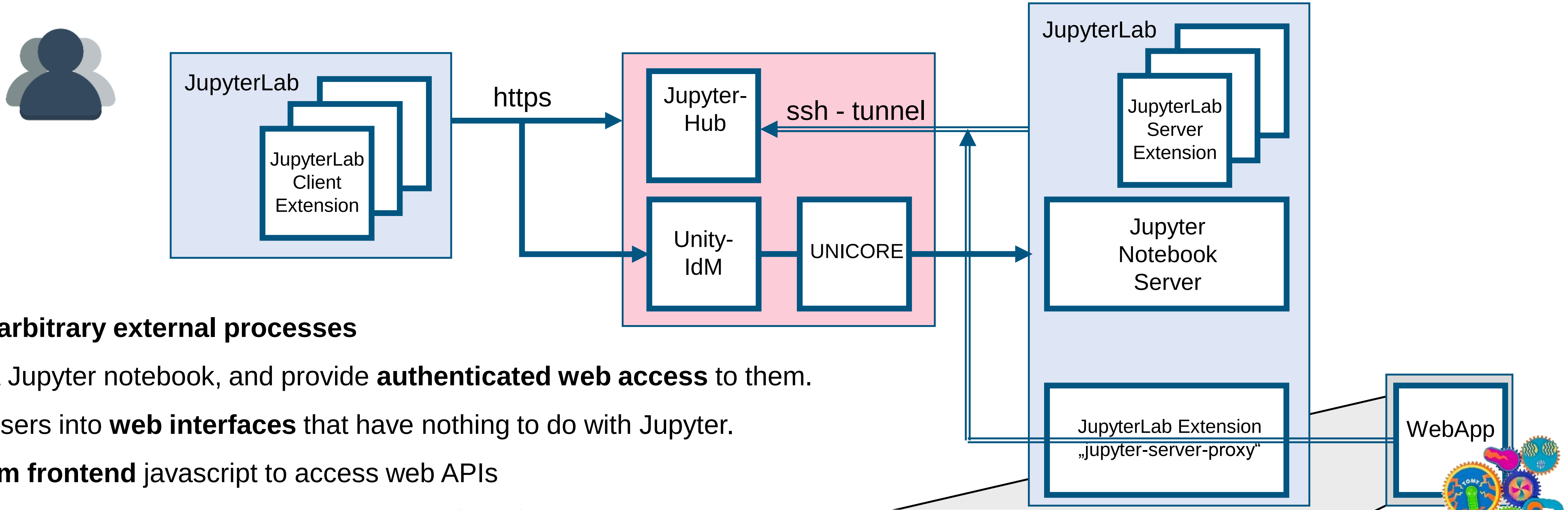
XServer
(dummy driver)

https://github.com/FZJ-JSC/jupyter-xprahtml5-proxy

JUPYTER CAN DO MORE

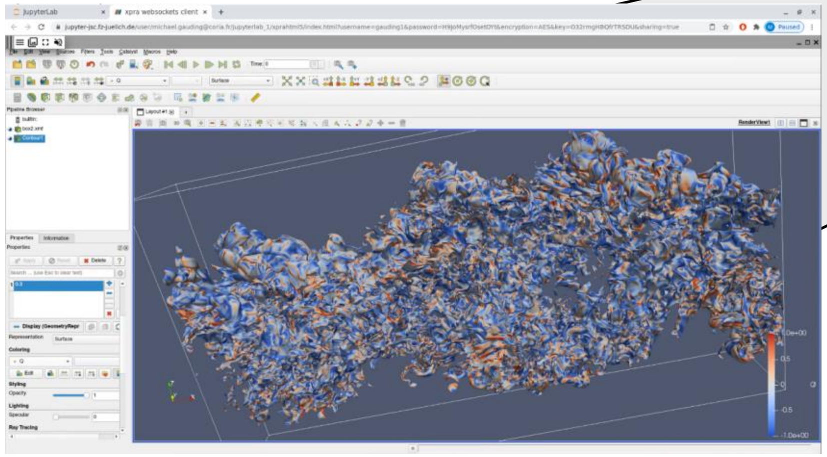
JUPYTERLAB – WEBSERVICE PROXY

Extension: jupyter-server-proxy


JupyterLab
JupyterLab Client Extension
https
Jupyter-Hub
ssh - tunnel
Unity-IdM
UNICORE
JupyterLab
JupyterLab Server Extension
Jupyter Notebook Server
JupyterLab Extension „jupyter-server-proxy“
WebApp

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



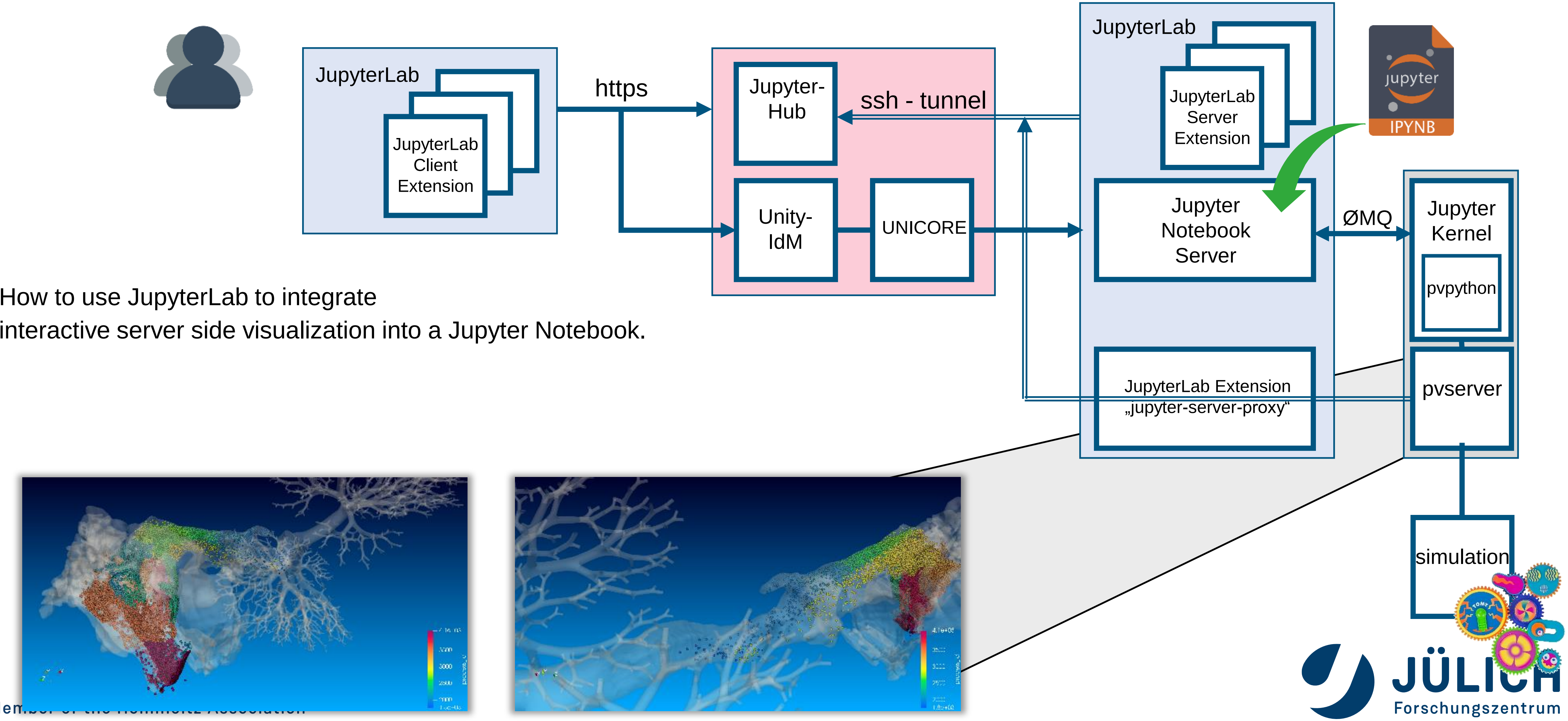
Turbulent mixing with variable density,
subset of 1939x600x3584 grid points, Michael Gauding, CORIA

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.



JUPYTERLAB – WEBSERVICE PROXY

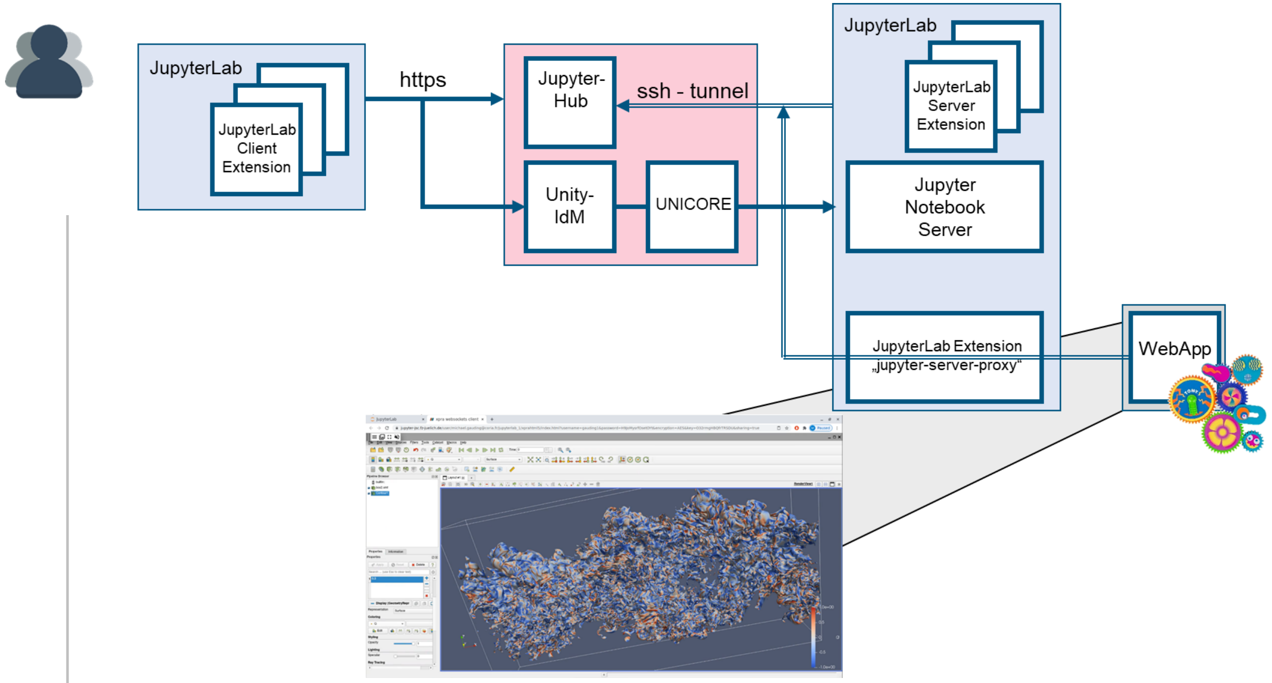
Extension: jupyter-server-proxy

Accessing Arbitrary Ports or Hosts

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.

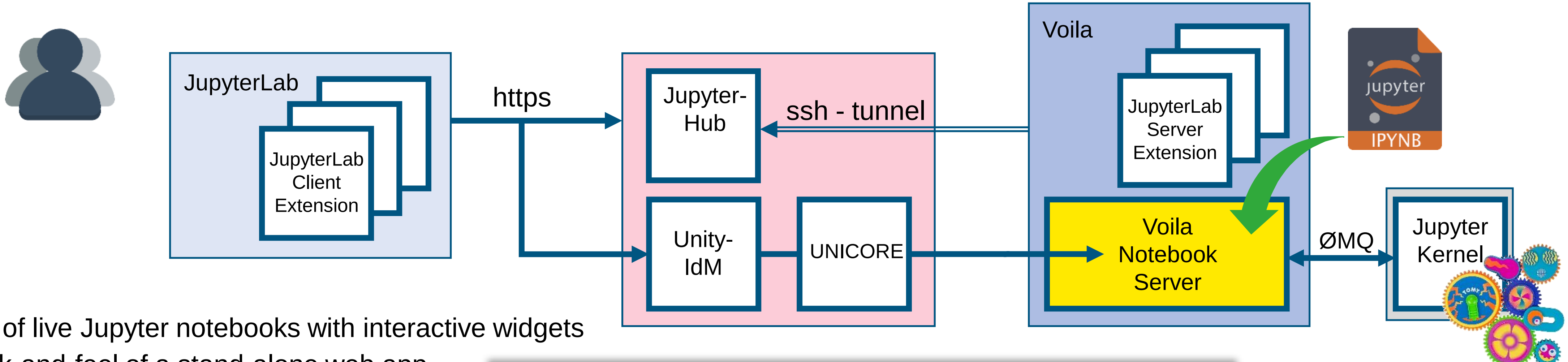

JupyterLab
JupyterLab Client Extension
https
Jupyter-Hub
ssh - tunnel
Unity-IdM
UNICORE
JupyterLab
JupyterLab Server Extension
Jupyter Notebook Server
JupyterLab Extension „jupyter-server-proxy“
WebApp

Example:
https://jupyter-jsc.fz-juelich.de/user/j.goebbert@fz-juelich.de/juwels_login/**proxy/12345**

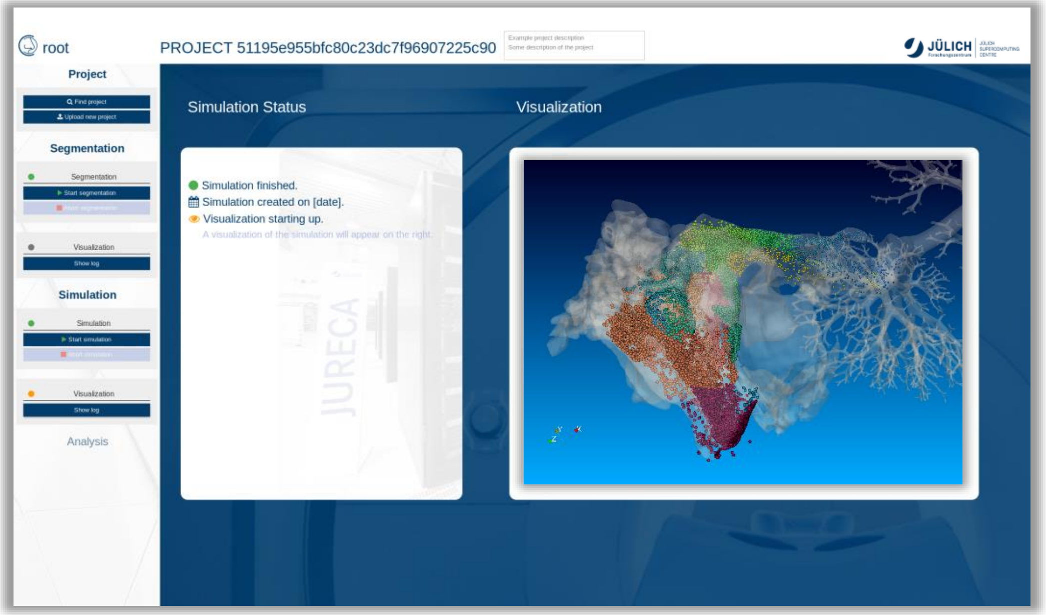
https://jupyter-server-proxy.readthedocs.io/en/latest/arbitrary-ports-hosts.html

DASHBOARDS WITH JUPYTER/VOILA

Voilà turns Jupyter notebooks into standalone web applications



- **Rendering** of live Jupyter notebooks with interactive widgets with the look-and-feel of a stand-alone web app.
- Voilà disallows execute requests from the front-end, **preventing** execution of arbitrary code.
- **Enables** HPC users to develop easily web applications from their Jupyter notebooks.



Simulation finished.
Simulation created on [date].
Visualization starting up.
A visualization of the simulation will appear on the right.

https://github.com/voila-dashboards/voila
https://voila-gallery.org

BENEFITS

Why Jupyter is so popular among Data Scientists

Some of the reasons …

- Jupyter allows to **view the results of the code in-line** without the dependency of other parts of the code.
- Jupyter mixes easy for users who extend their code **line-by-line with feedback** attached all along the way
- Jupyter Notebooks support visualization and include rendering data in **live-graphics and charts**.
- Jupyter is maintaining the **state of execution of each cell** automatically.
- Supports IPyWidget packages, which provide **standard user interface** for exploring code and data interactively.
- Jupyter is platform and language **independent** with a well known file format in JSON.

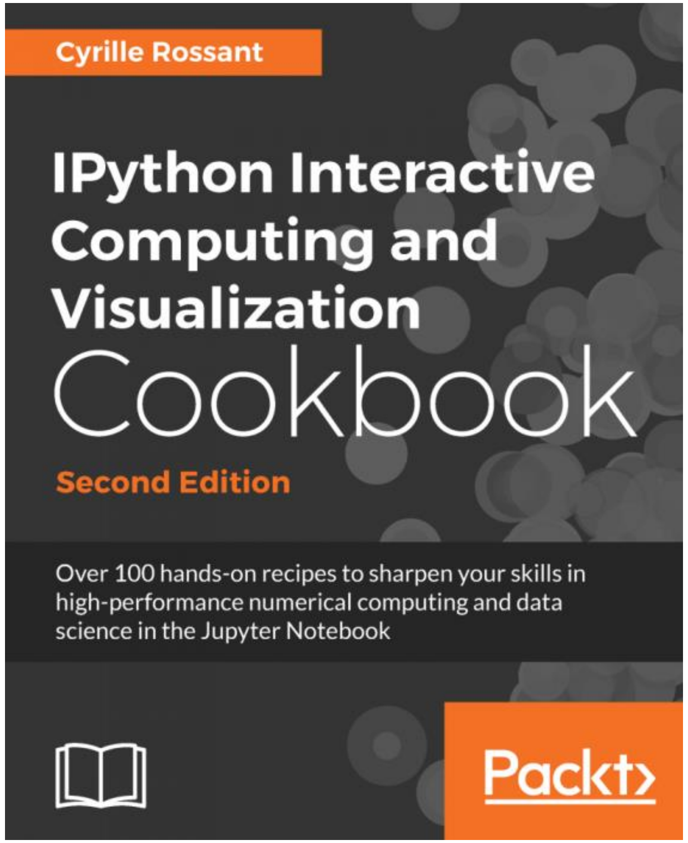
- JupyterLab is build to be extentended, has a hudge community and a long history and is therefore future-proof.

TUTORIALS

Get started with Jupyter/JupyterLab – published 2018 but still a great starting point

Possible start to enter the world of
interactive computing with IPython in Jupyter/JupyterLab:

- Leverage the Jupyter Notebook for interactive data science and visualization
- High-performance computing and visualization for data analysis and scientific modeling
- A comprehensive coverage of scientific computing through many hands-on, example-driven recipes with detailed, step-by-step explanations



Cyrille Rossant

IPython Interactive Computing and Visualization Cookbook

Second Edition

Over 100 hands-on recipes to sharpen your skills in high-performance numerical computing and data science in the Jupyter Notebook

Packt>

https://ipython-books.github.io
https://github.com/ipython-books/cookbook-2nd

QUESTIONS?