JUPYTERLAB - SUPERCOMPUTING IN YOUR BROWSER

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

2022-05-16 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

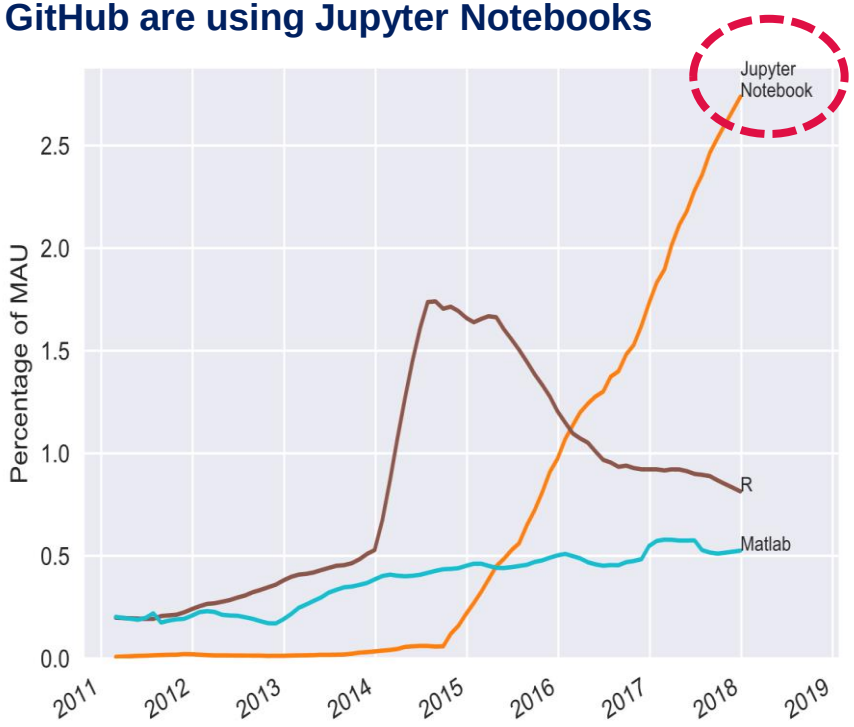
https://jupyter.org

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 …

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

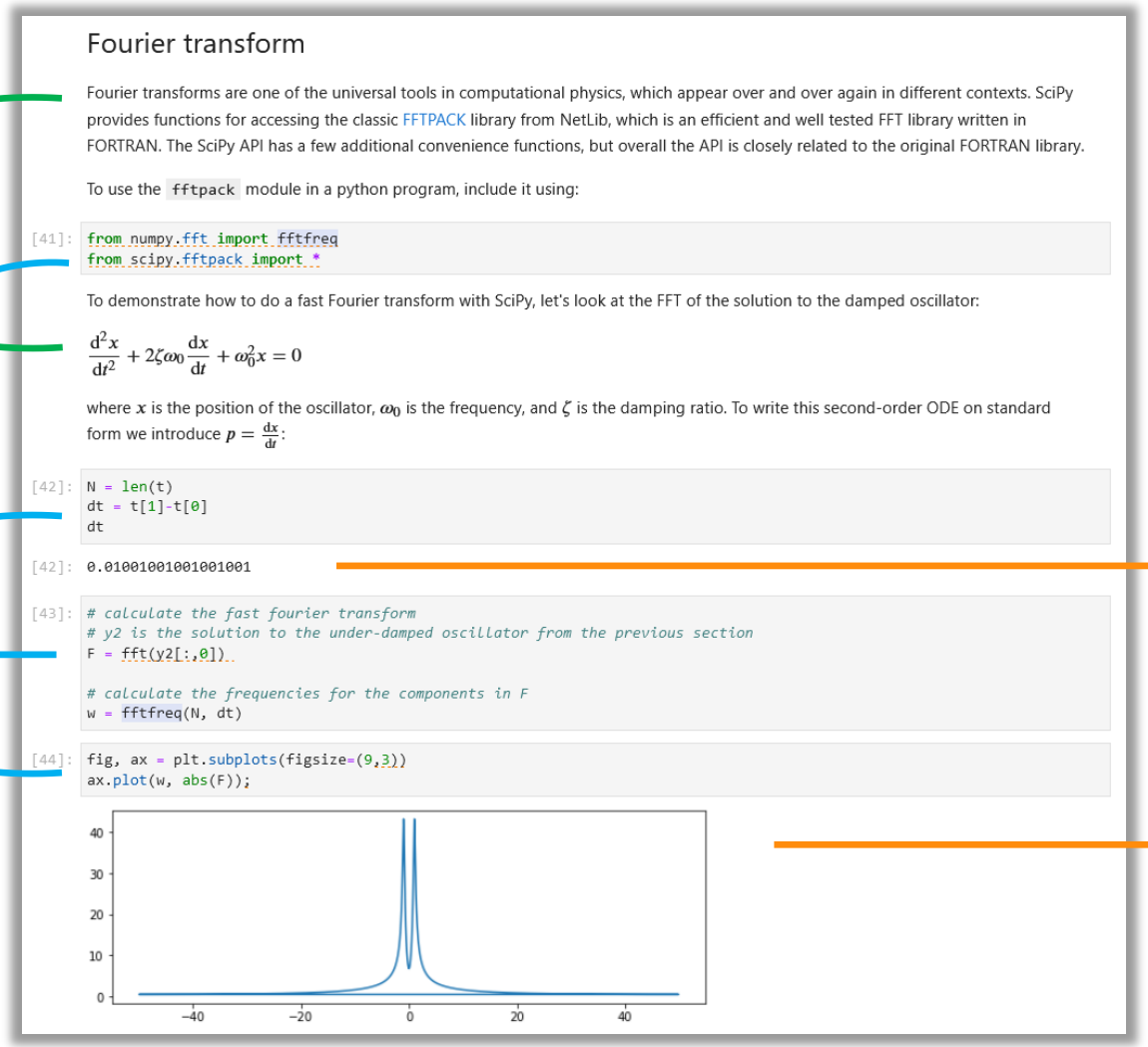
creating reproducible computational narratives

Markdown Cells

Code Cells

Output

Output



HISTORY OF JUPYTERLAB AT JSC

Initial Basis

JupyterLab module & installation
Authentication via Unity/IdM
Authorization via UNICORE
Orchestration with 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

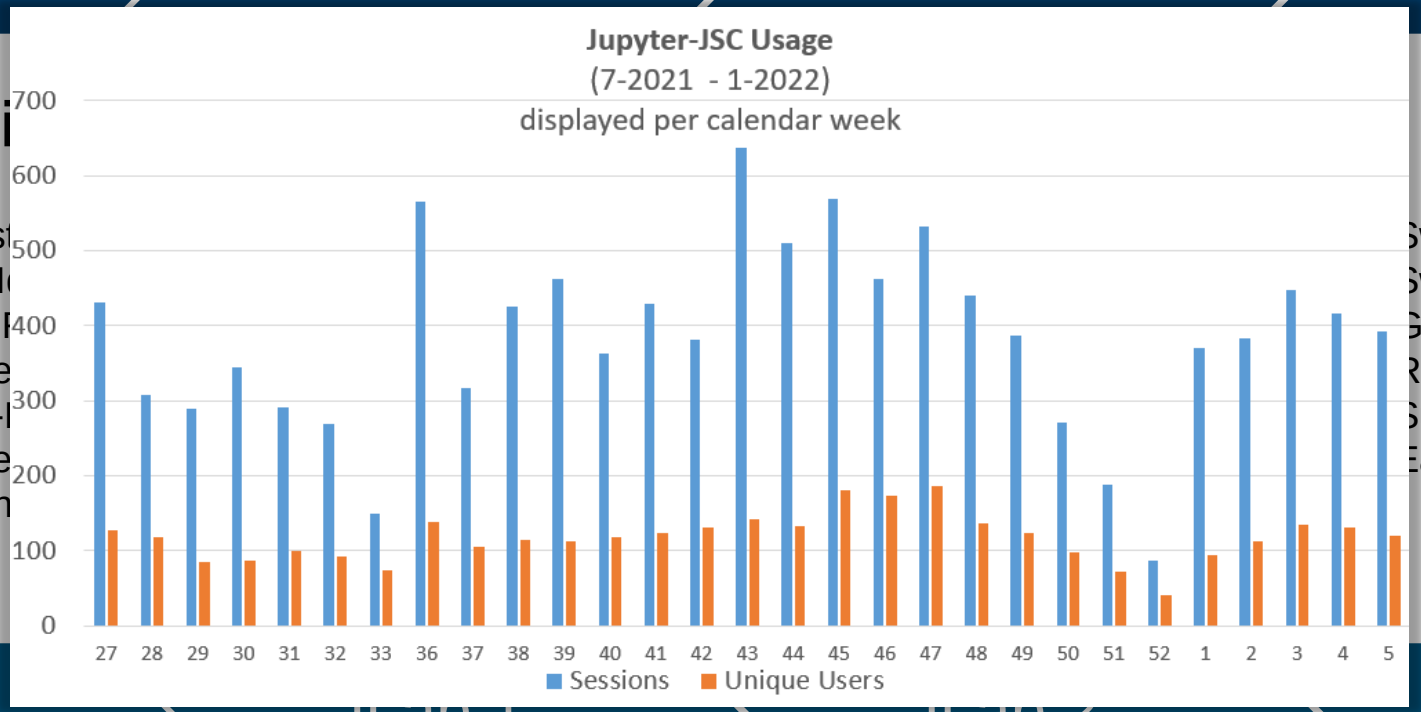
Switch to Kubernetes
Switch to JupyterLab 3
GPFS through UFTP
Redesign Management
Support for User Extensions
Easybuild Modularization

HISTORY OF JUPYTERLAB AT JSC

2018 | 2019 | 2020 | 2021

Initial Basi

JupyterLab module & inst
Authentication via Unity/I
Authorization via UNICOR
Orchestration with Docke
Synchronization of User-I
Basic Data Protection Re
Fulfill Safety Requiremen

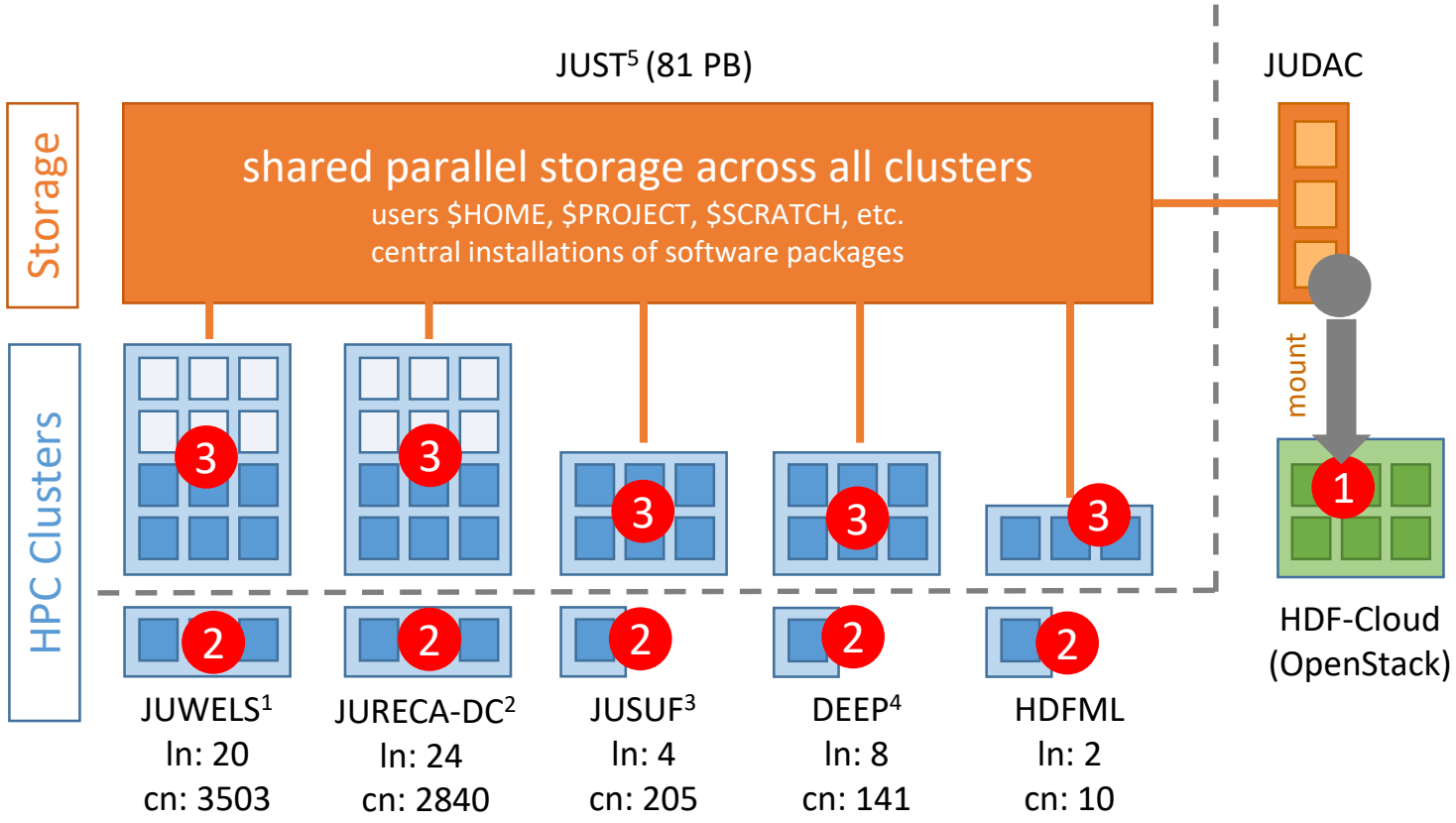


Redesign

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

JLab Beta | JLab 1 | JLab 2 | JLab 3

JUPYTERLAB EVERYWHERE



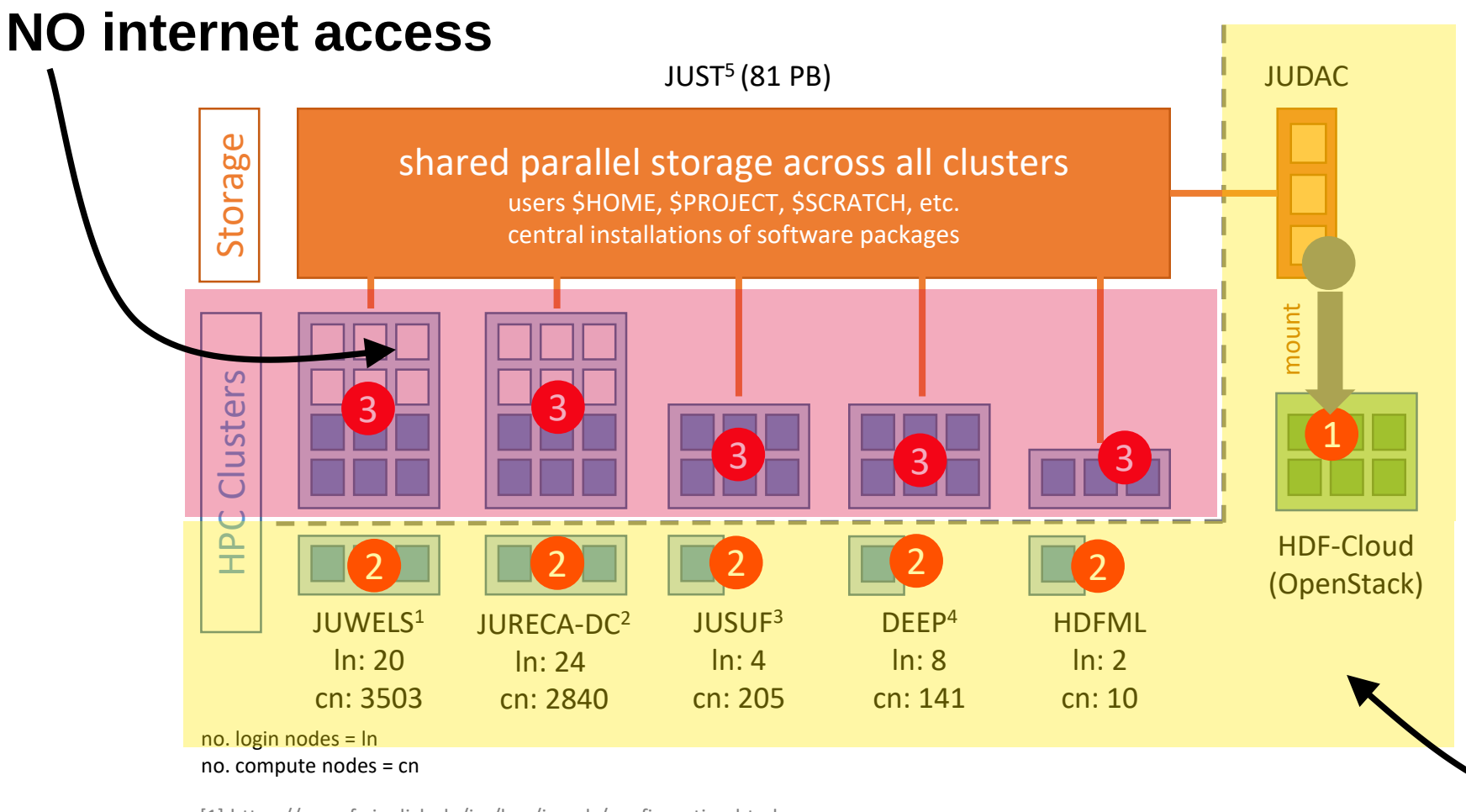
no. login nodes = ln
no. compute nodes = cn

JupyterLab everywhere

1. JupyterLab on HDF-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



JupyterLab everywhere

1. JupyterLab on HDF-Cloud
2. JupyterLab on login nodes
3. JupyterLab on compute nodes

internet access

[1] https://apps.fz-juelich.de/jsc/hps/juwels/configuration.html
[2] https://apps.fz-juelich.de/jsc/hps/jureca/configuration.html
[3] https://apps.fz-juelich.de/jsc/hps/jusuf/cluster/configuration.html
[4] https://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/DEEP-EST/_node.html
[5] https://www.fz-juelich.de/ias/jsc/EN/Expertise/Datamanagement/OnlineStorage/JUST/Configuration/Configuration_node.html

TERMINOLOGY

What is JupyterLab

JupyterLab

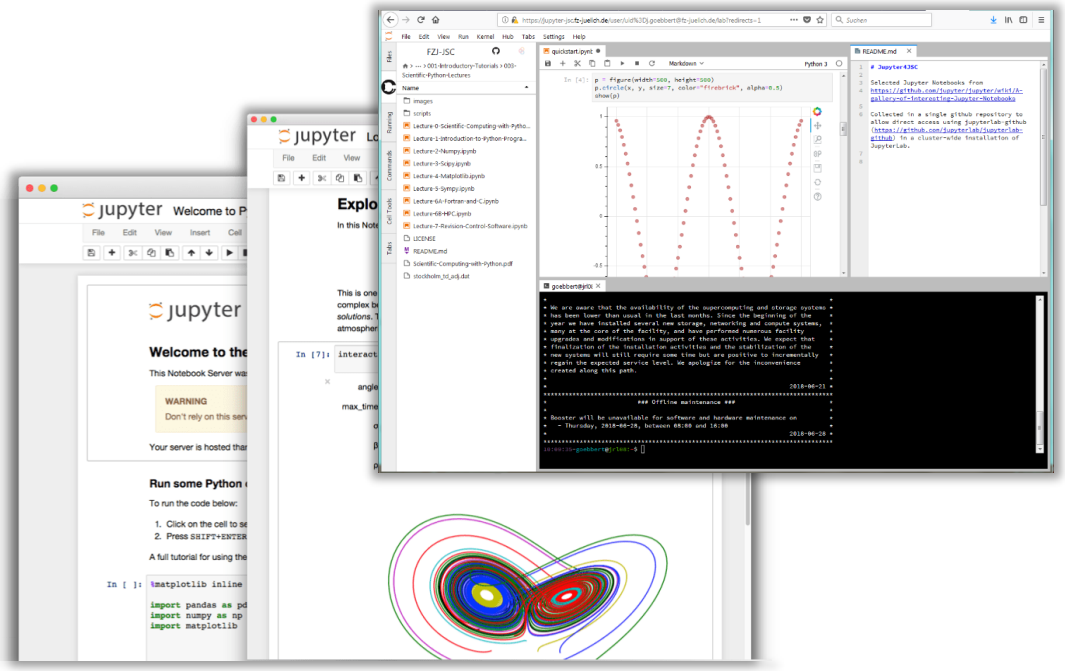
- **Interactive** working environment in the web browser
- For the creation of **reproducible** computer-aided narratives
- Very **popular** with researchers from all fields
- Jupyter = Julia + Python + R

Multi-purpose working environment

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

Document-Centered Computing (“*notebooks*”)

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



https://jupyterlab.readthedocs.io

TERMINOLOGY

What is a Jupyter Notebook?

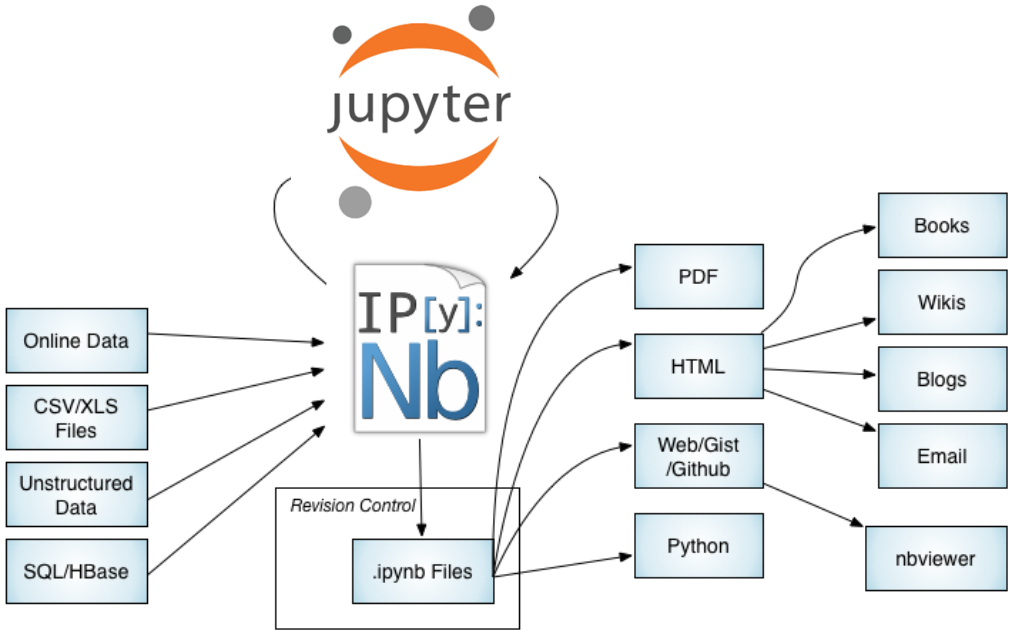
Jupyter Notebook

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

- It is a file, which stores your work in JSON format
- Based on a set of open standards for interactive computing

- Allows development of custom applications with embedded interactive computing.
- Can be extended by third parties

- Directly convertible to PDF, HTML, LateX ...
- Supported by many applications
  such as GitHub, GitLab, etc..



https://jupyter-notebook.readthedocs.io/
https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks

TERMINOLOGY

What is a Jupyter Kernel?

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

Jupyter Kernel

- **run code** in different programming languages and environments.
- can be **connected to** a notebook (one at a time).
- **communicates** via ZeroMQ with the JupyterLab.

- Multiple **preinstalled** Jupyter Kernels can be found on our clusters
  - Python, R, Julia, Bash, C++, Ruby, JavaScript
  - Specialized kernels for 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

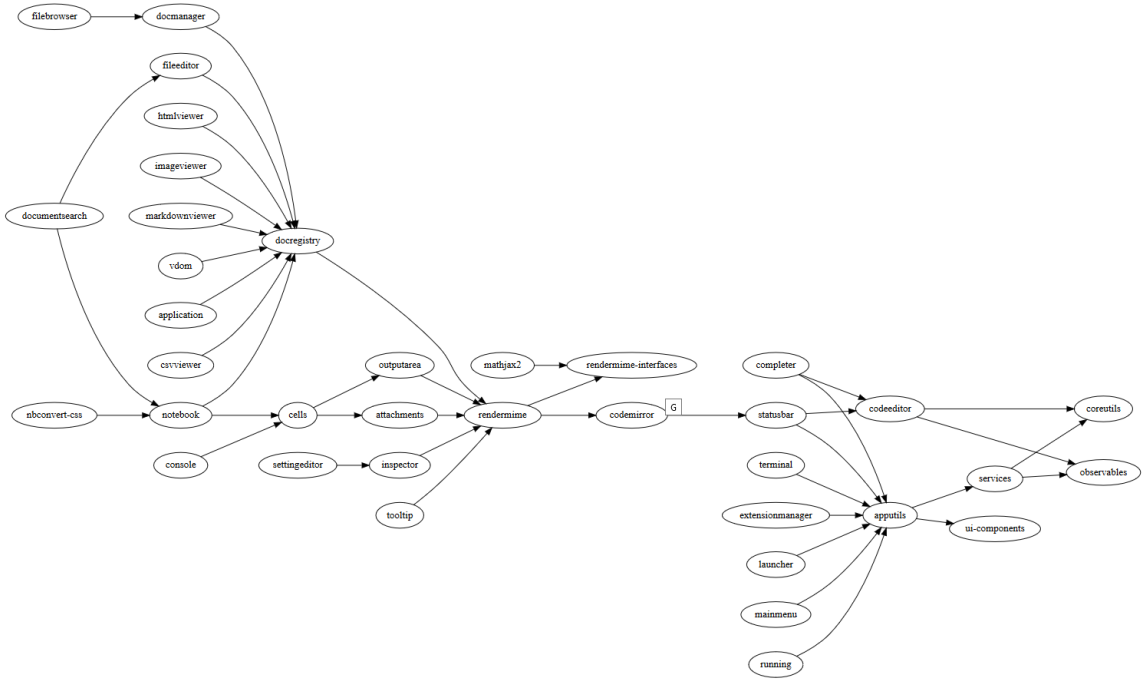
JupyterLab extensions can customize or enhance any part of JupyterLab.

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.



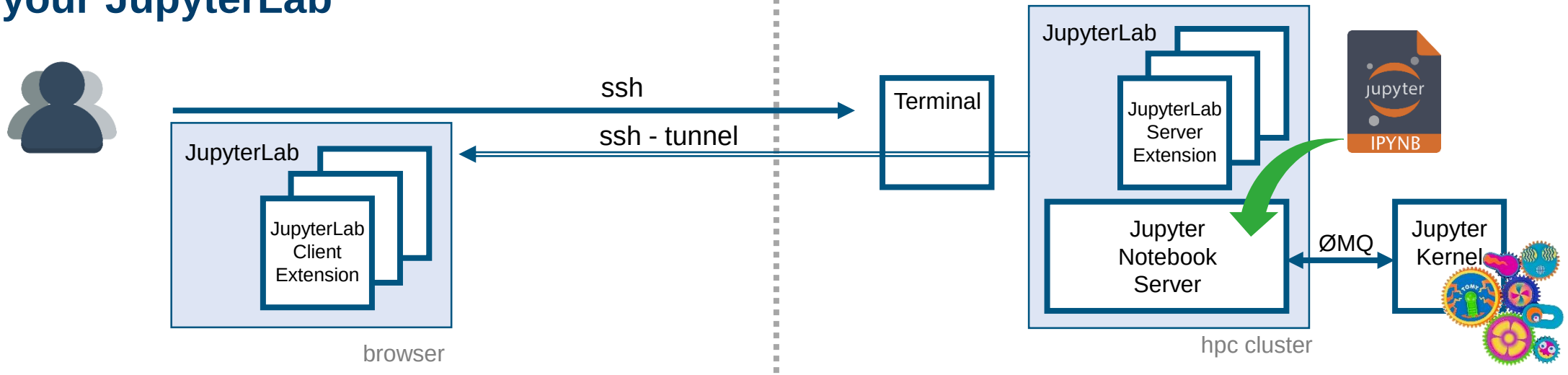
https://jupyterlab.readthedocs.io/en/stable/user/extensions.html
https://github.com/topics/jupyterlab-extension

With JupyterLab 3 prebuild extensions were introduced.

You can now (technically) extend a compiled JupyterLab 3+ with **your own extensions**.

JUPYTER-JSC WEBSERVICE

Start your JupyterLab



JUPYTERLAB - WHEREVER YOU PREFER

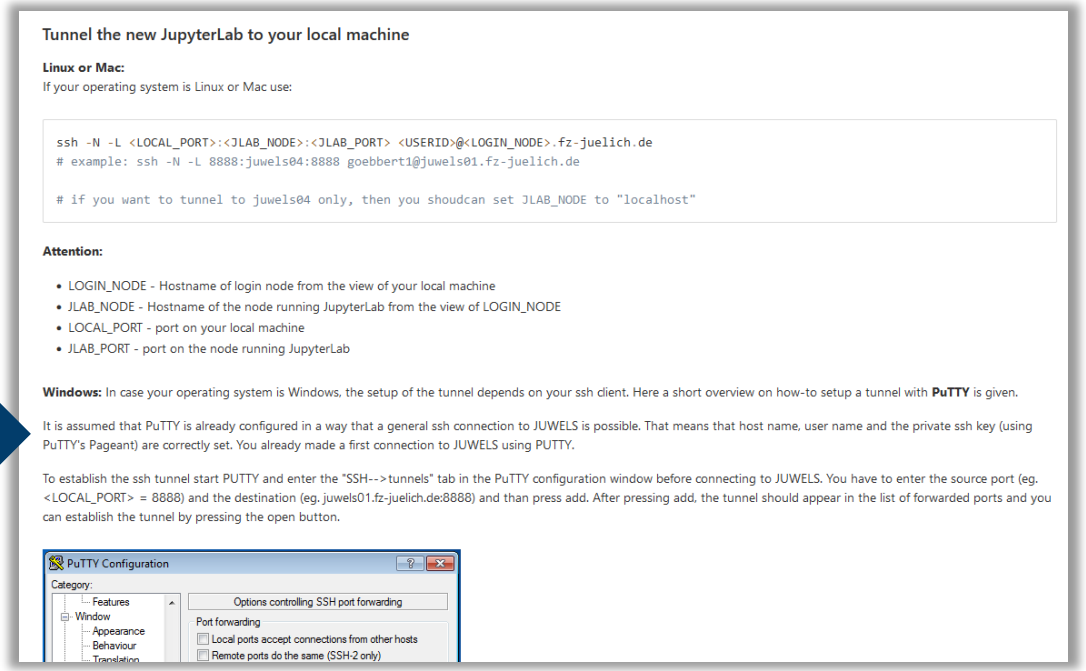
Local, Remote, Browser-only

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 in $HOME (e.g. using pip or miniconda)
- **JupyterLab** installed system-wide (e.g. with Easybuild, Spark)



Browser-only installation (alpha!):

- **JupyterLab** (local in your browser – JupyterLite = just for testing!)
  https://blog.jupyter.org/jupyter-everywhere-f8151c2cc6e8
  https://jupyter.org/try-jupyter/lab

JUPYTERLAB - WHEREVER YOU PREFER

Local, Remote, Browser-only

Local installation:

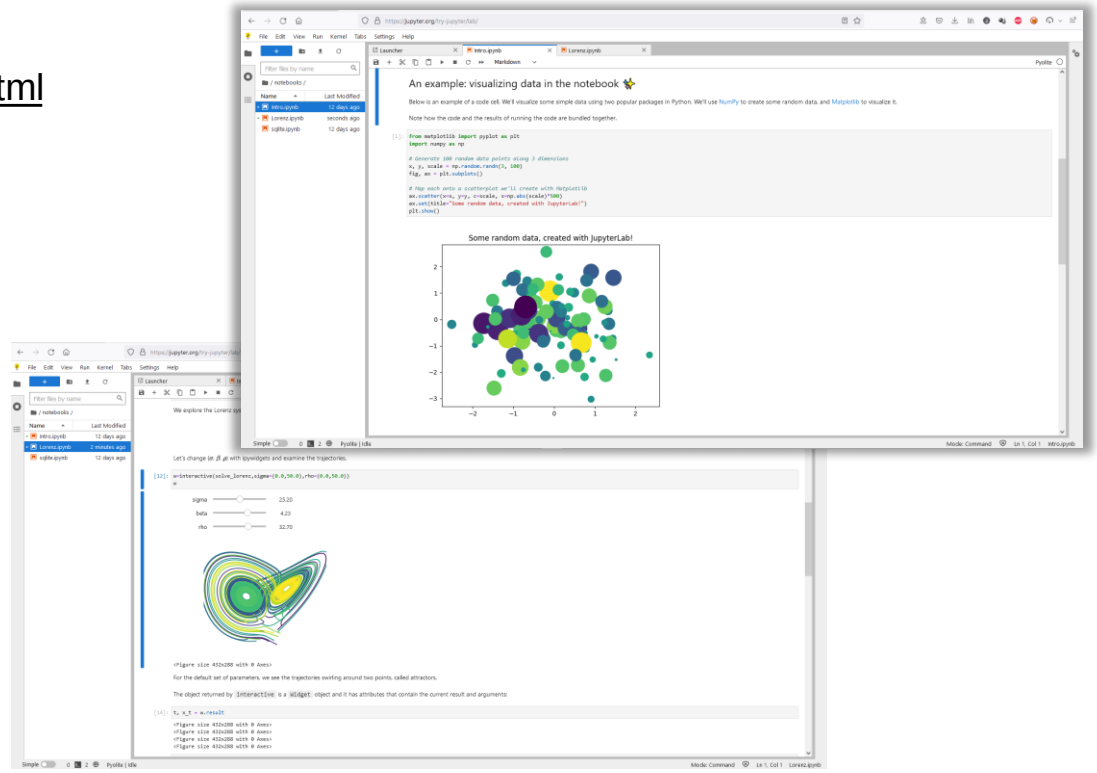
- **JupyterLab** installed using conda, mamba, pip, pipenv or docker.
  https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html

Remote (cluster) installation:

- **JupyterLab** installed in $HOME (e.g. using pip or miniconda)
- **JupyterLab** installed system-wide (e.g. with Easybuild, Spark)

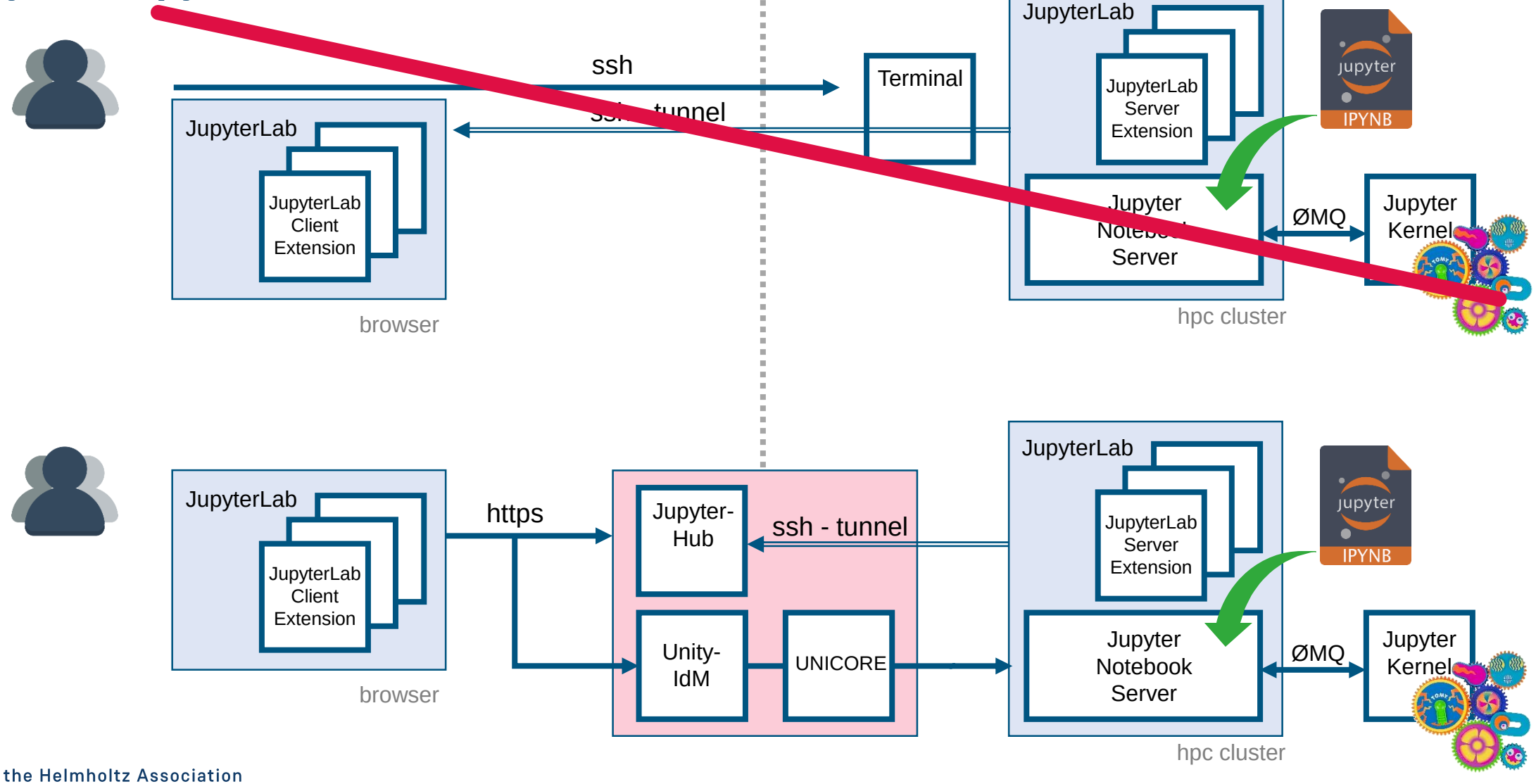
Browser-only installation (alpha!):

- **JupyterLab** (local in your browser – JupyterLite = just for testing!)
  https://blog.jupyter.org/jupyter-everywhere-f8151c2cc6e8
  Try it: https://jupyter.org/try-jupyter/lab



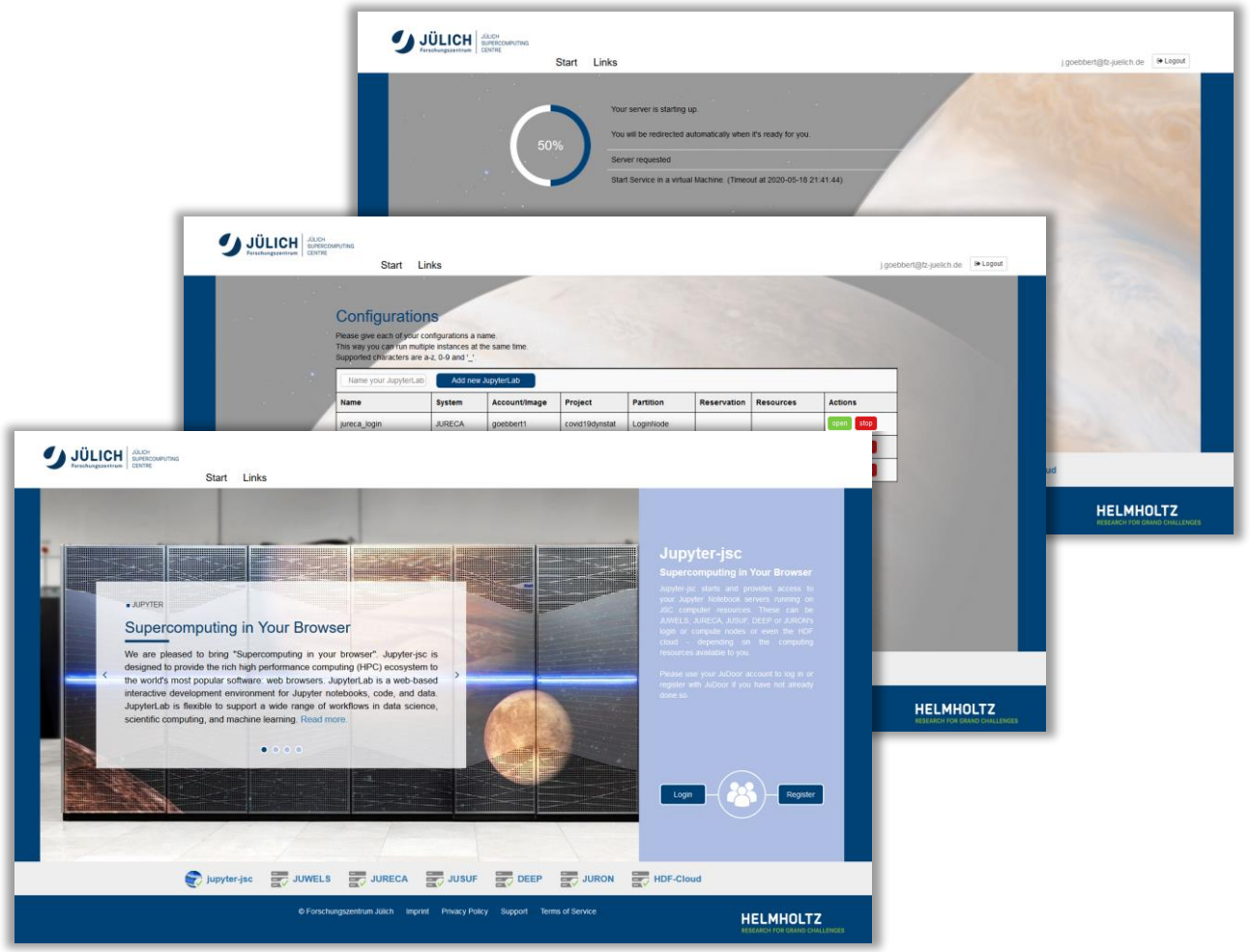
JUPYTER-JSC WEBSERVICE

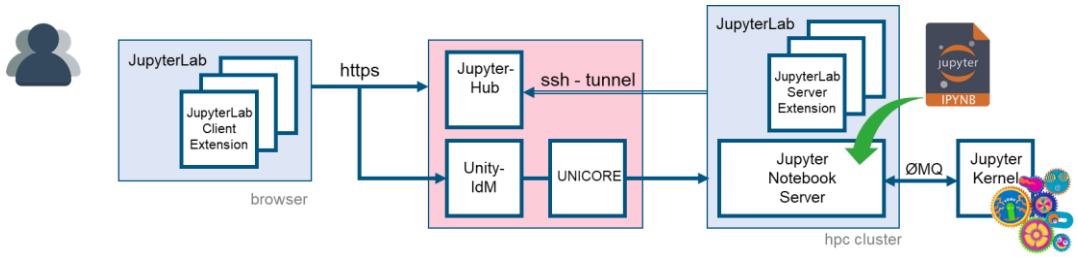
Start your JupyterLab

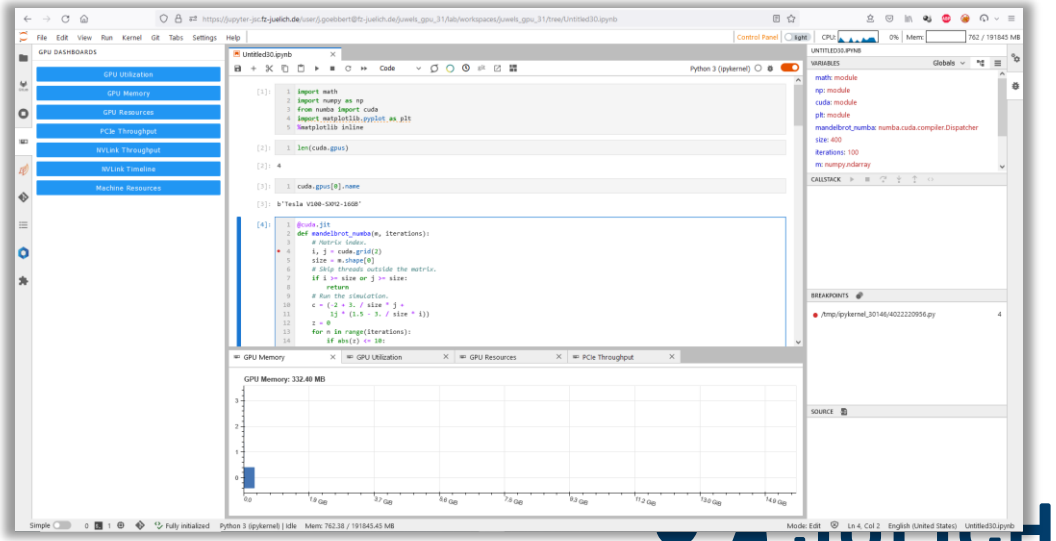


JUPYTER-JSC WEBSERVICE

Start your JupyterLab

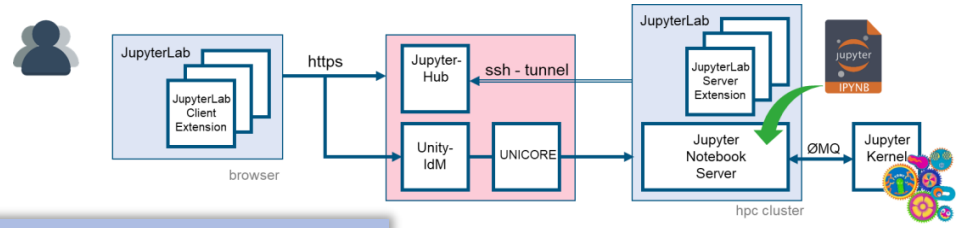






JUPYTER-JSC WEBSERVICE

First time login

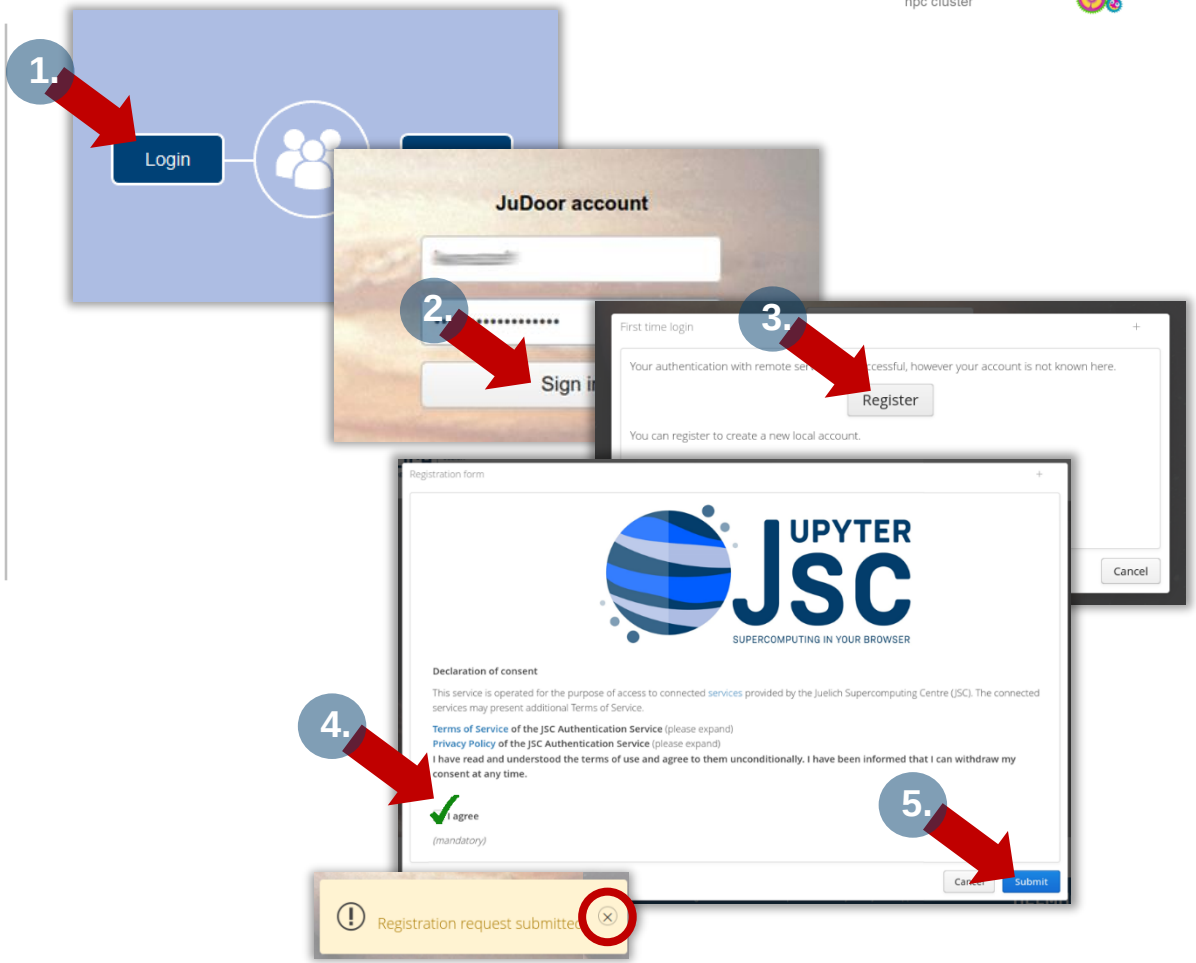


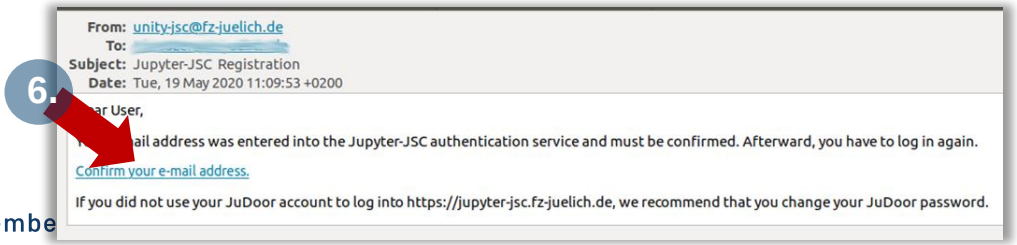
=> https://jupyter-jsc.fz-juelich.de

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

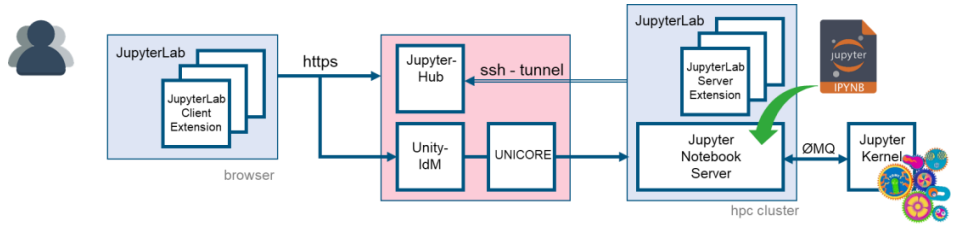




JÜLICH Forschungszentrum

JUPYTER-JSC WEBSERVICE

Control Panel



A. Jupyter-JSC – Add new JupyterLab

- Name your new JupyterLab configuration
  - Unique Jupyter workspace in `~/.jupyter`
- => the **JupyterLab Options** page will open

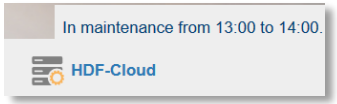
B. Jupyter-JSC – Actions

If a configuration has been added

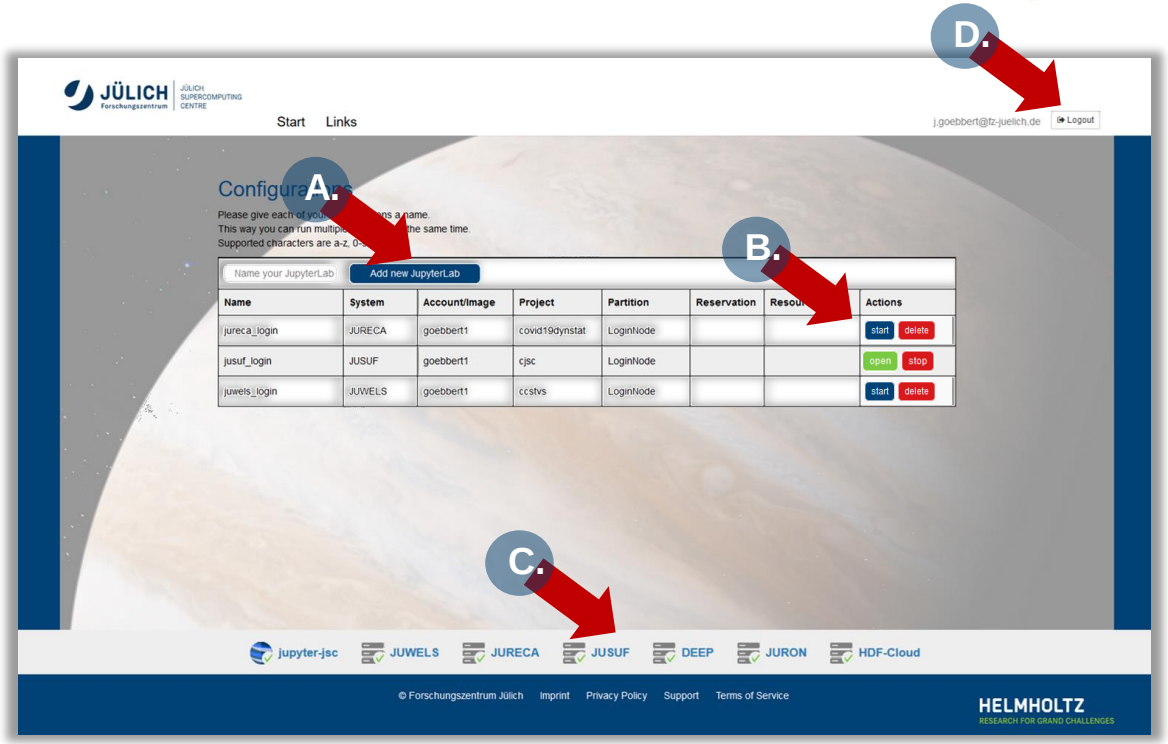
- Start/delete the named configuration (workspace will not be deleted)
- Open/stop a **running** JupyterLab

C. Jupyter-JSC -- Statusbar

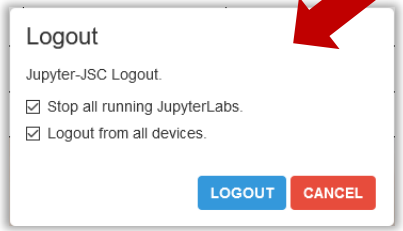
- Upcoming maintenance (mouse hover for details)



- System offline



B. Jupyter-JSC – Logout

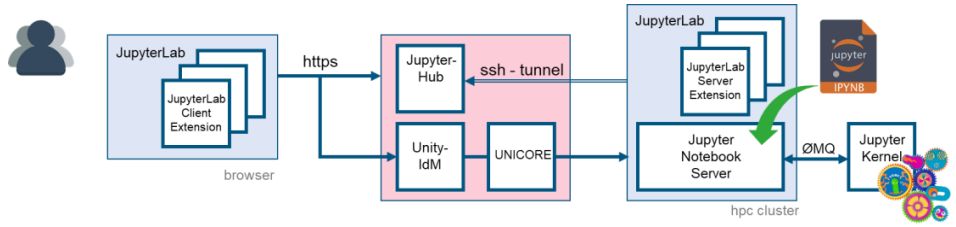


Logout will ask what you want to do with the running JupyterLabs – be careful what you answer!

JÜLICH Forschungszentrum

JUPYTER-JSC WEBSERVICE

JupyterLab Options



Jupyter-JSC – Options

Available options **depend on**

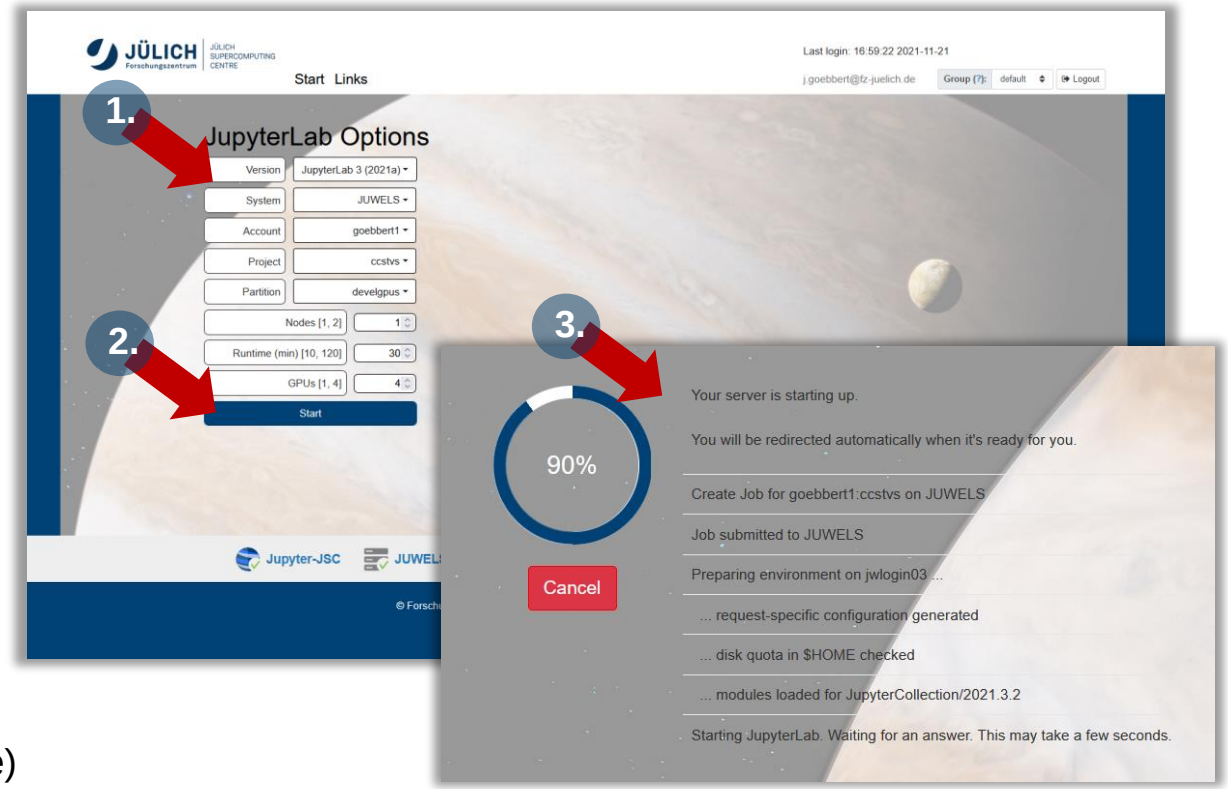
- user account settings visible in **judoor.fz-juelich.de**
- currently available systems in all of your projects
  - system specific usage agreement on JuDoor is signed

Basic options

- Version:
  JupyterLab 2 and JupyterLab 3 (default) is 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, …



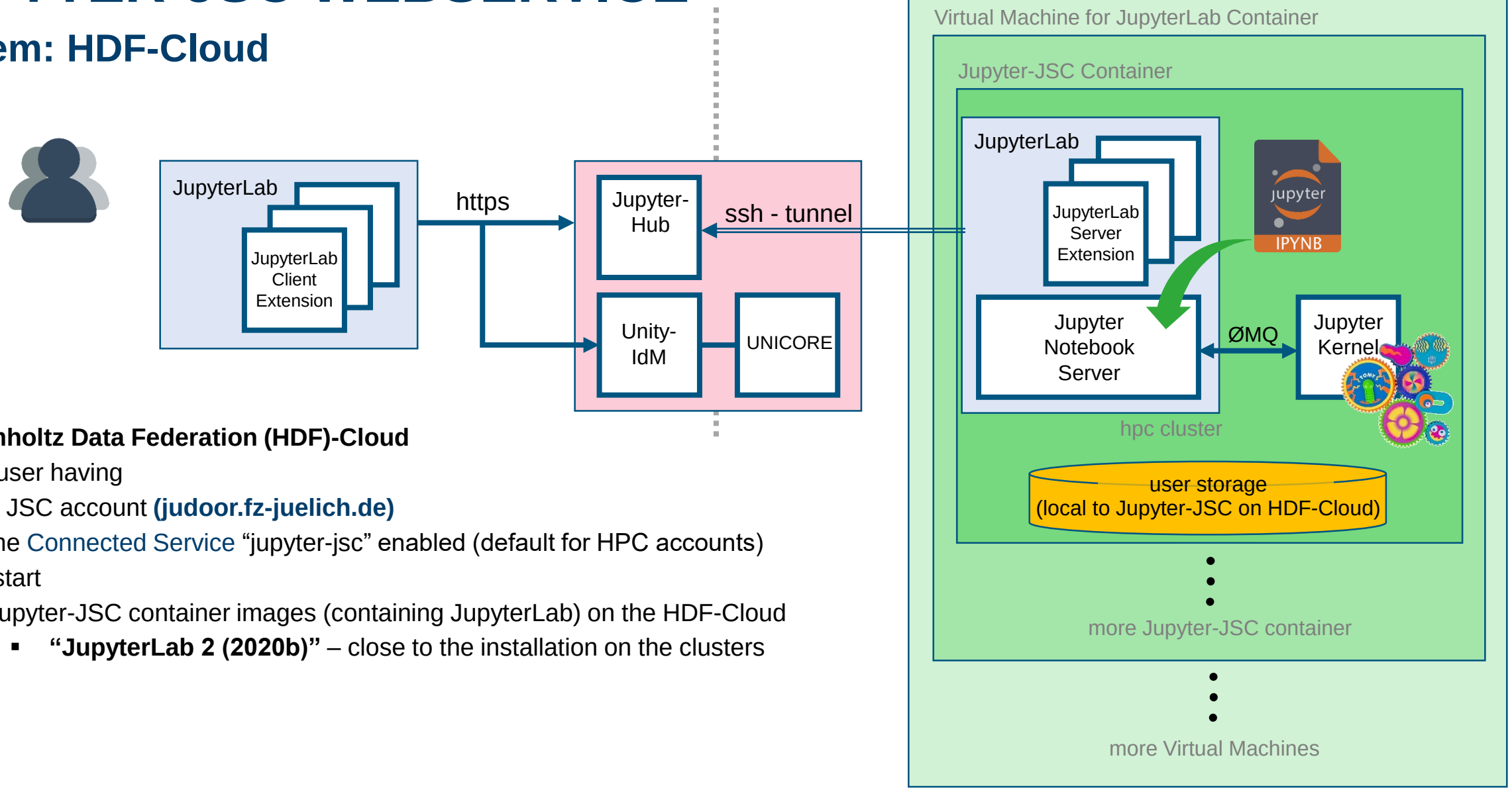
NEW: Version

- **Choose the version of JupyterLab 2 or 3 (default)**

https://docs.jupyter-jsc.fz-juelich.de/github/FZJ-JSC/jupyter-jsc-notebooks/blob/master/Announcement-2021-11_JupyterLab3_at_Jupyter-JSC.ipynb

JUPYTER-JSC WEBSERVICE

System: HDF-Cloud



Helmholtz Data Federation (HDF)-Cloud

Any user having

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

can start

- Jupyter-JSC container images (containing JupyterLab) on the HDF-Cloud
  - **“JupyterLab 2 (2020b)”** – 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

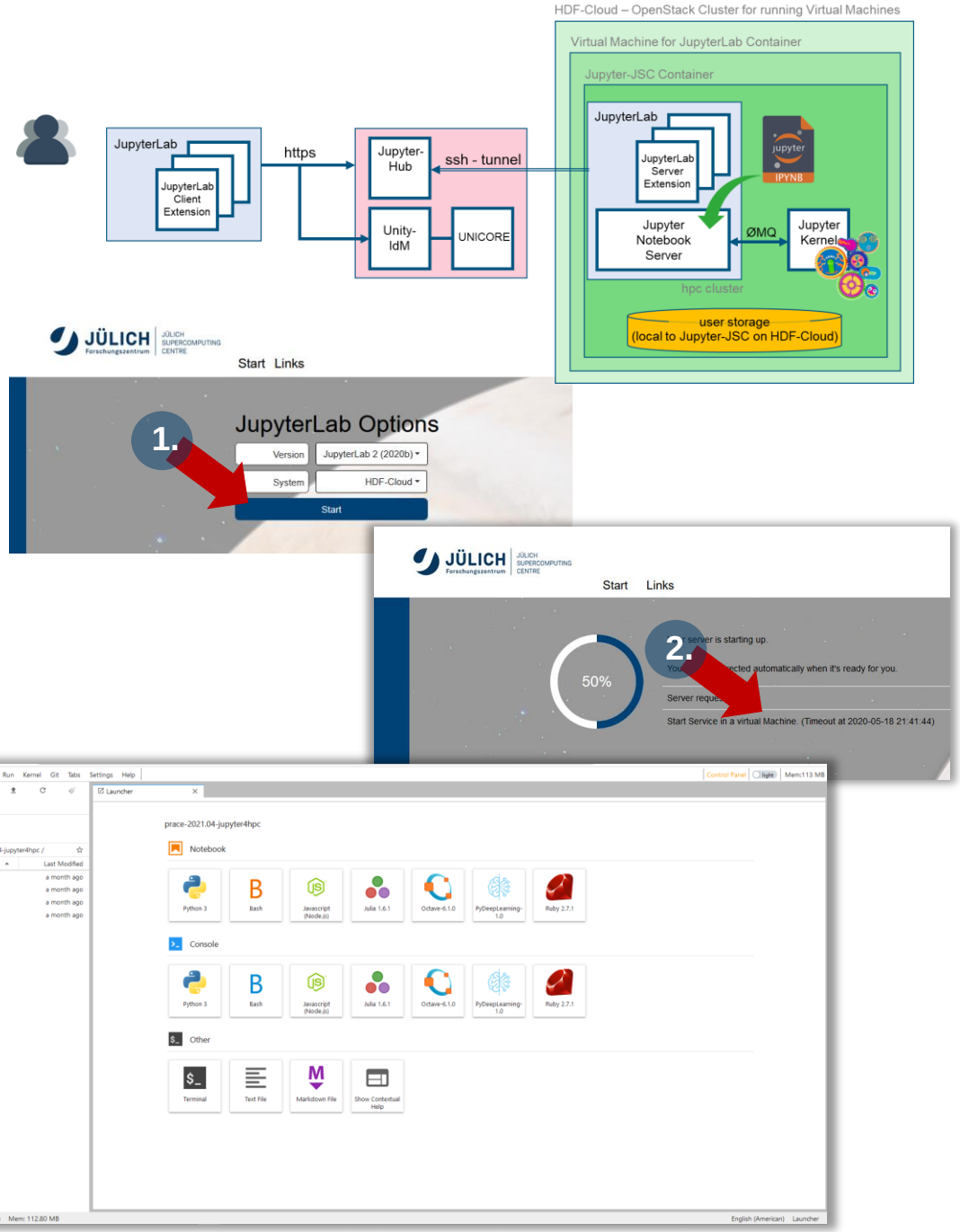
Start JupyterLab on HDF-Cloud

- 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

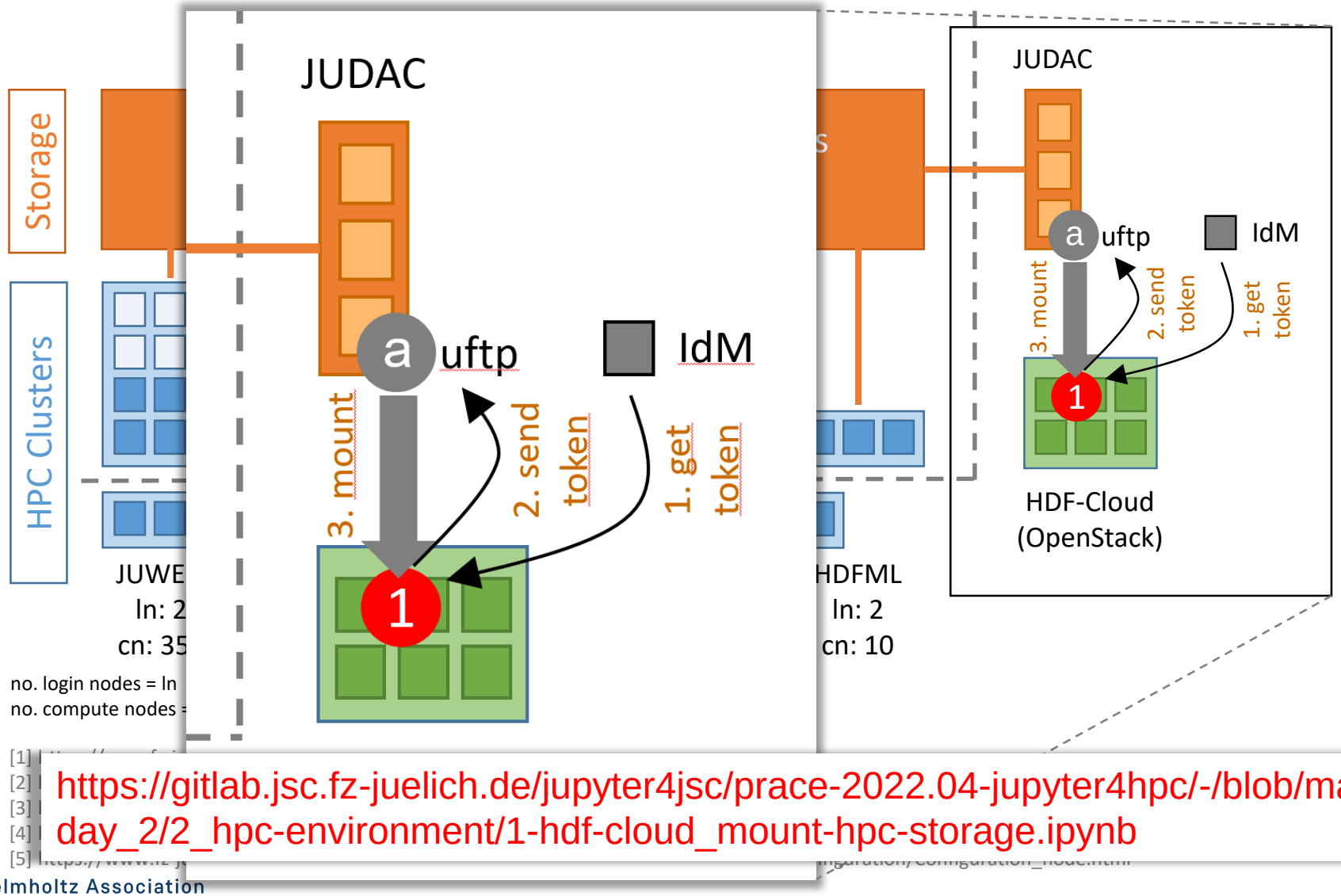
- Start a JupyterLab:
  - Version == “JupyterLab 2 (2020b)”
  - System == “HDF-Cloud”

Limitations on JupyterLab on HDF-Cloud

- max. **4 GB** memory
  - ATTENTION: the container automatically stops, when this limit is reached.
- Installed Jupyter Kernel limited
- Storage in Jupyter-JSC container
  - **is local** to the HDF-Cloud
  - only accessible from a Jupyter-JSC container
- HDF-Cloud has **no GPUs**



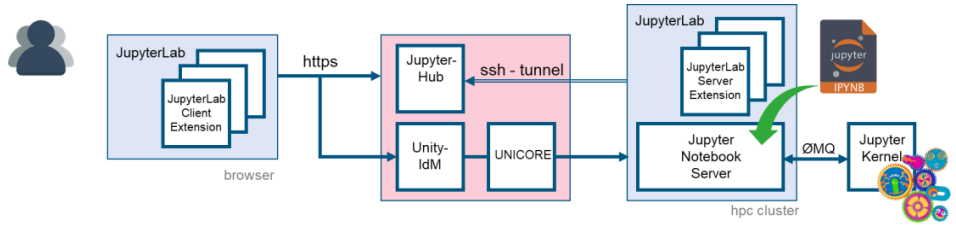
HOW TO MOUNT GPFS ON HDF-CLOUD



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 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

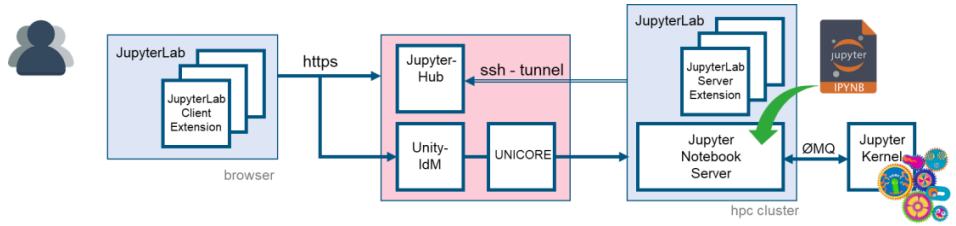
notebook cell

indicates active notebook cell

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

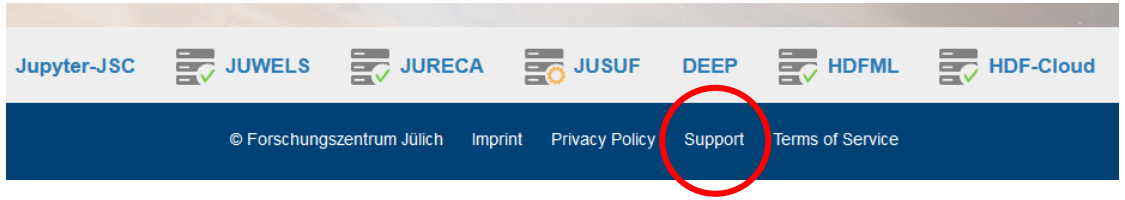
JUPYTER-JSC SECRETS

Very important to know



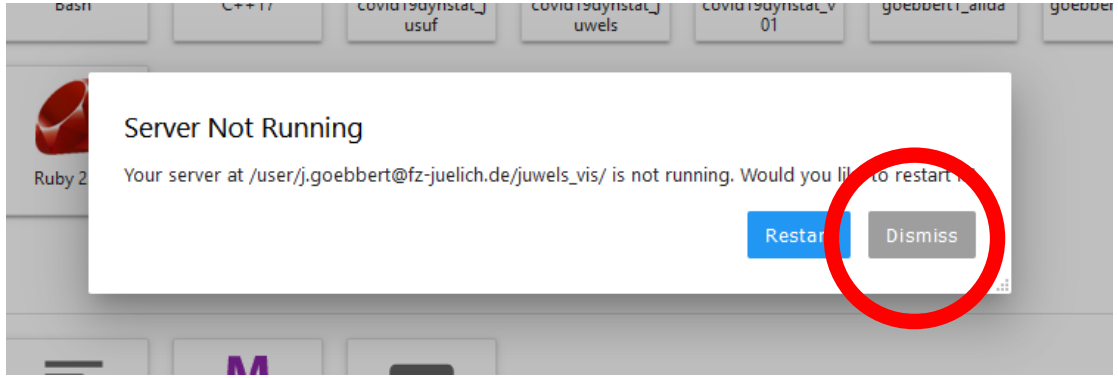
Secret 1: Support button

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

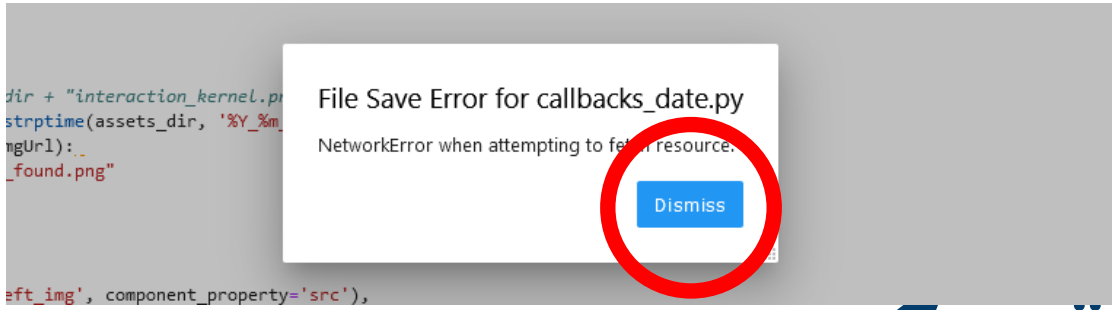


Secret 2: Reload on connection loss

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



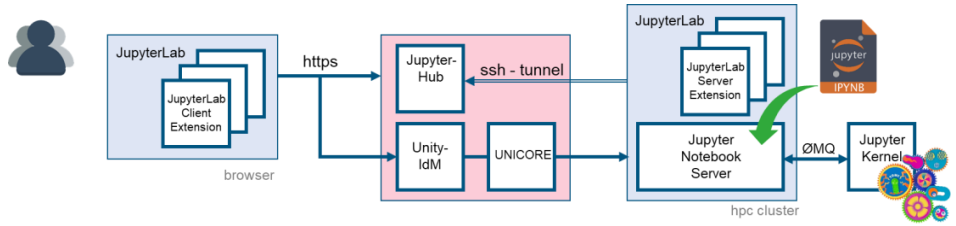
- "File Save Error for <…>"
  means, that your browser just lost connection
  => **Just hit "Dismiss" !!!**
  (as soon as you are online again)



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 use $OTHERSTAGES
module load Stages/2020
module load GCCcore/.10.3.0
module load JupyterCollection/2021.3.2
```

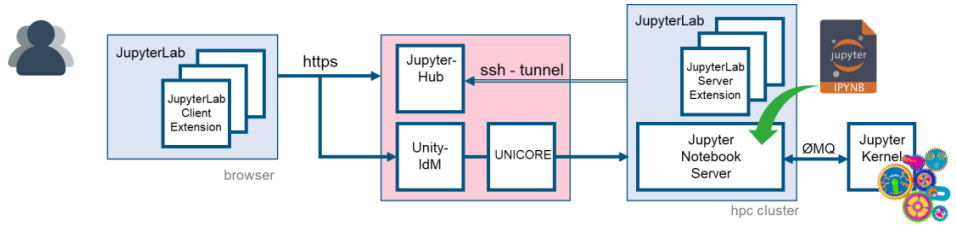
Switch to a different JupyterLab with
`$HOME/.jupyter/start_jupyter-jsc.sh`

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

JUPYTERLAB EXTENSIONS

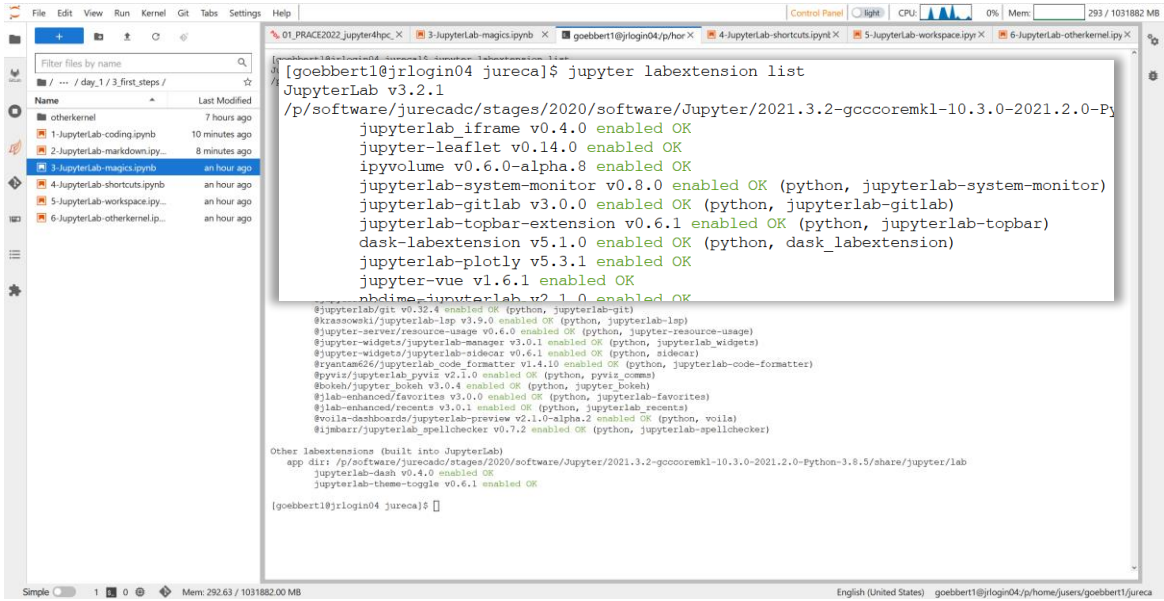
JUPYTER EXTENSIONS

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

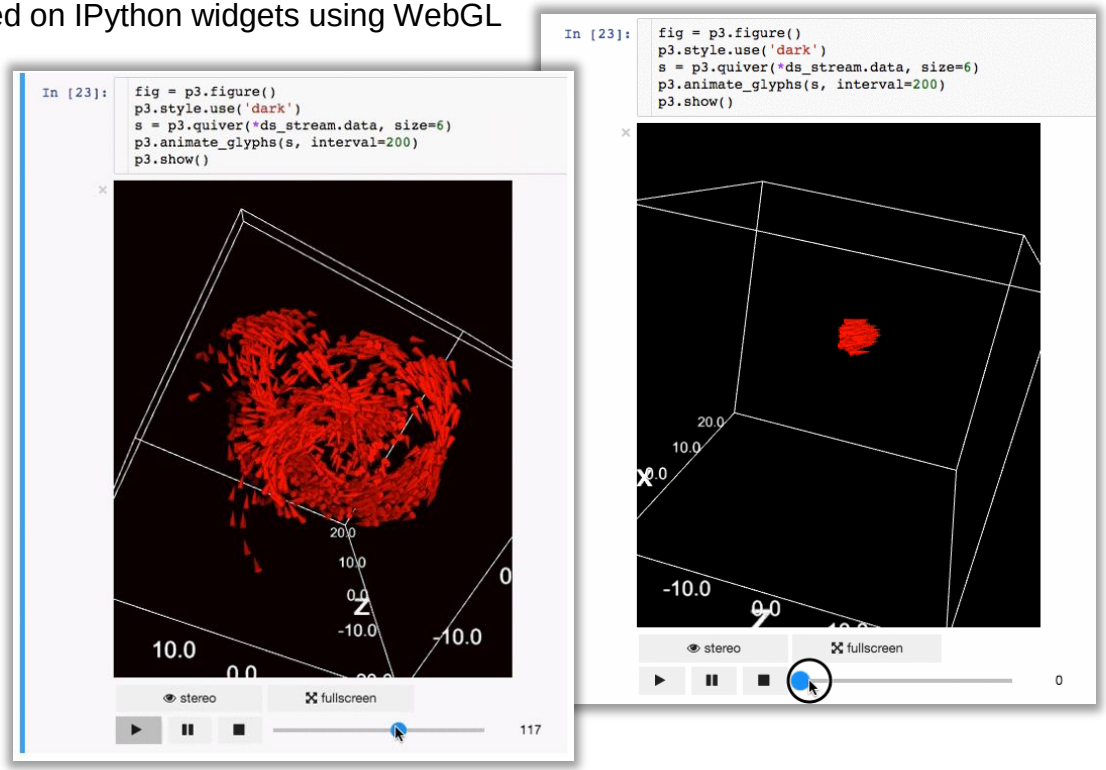
- stores any information that JupyterLab persists
  - including settings and built assets of extensions
- default location is `<sys-prefix>/share/jupyter/lab`
- can be relocated by setting $JUPYTERLAB_DIR
  - contains the JupyterLab static assets
    - (e.g. `static/index.html`)
  - **JupyterLab < 3:**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

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

JUPYTER-JSC EXTENSIONS

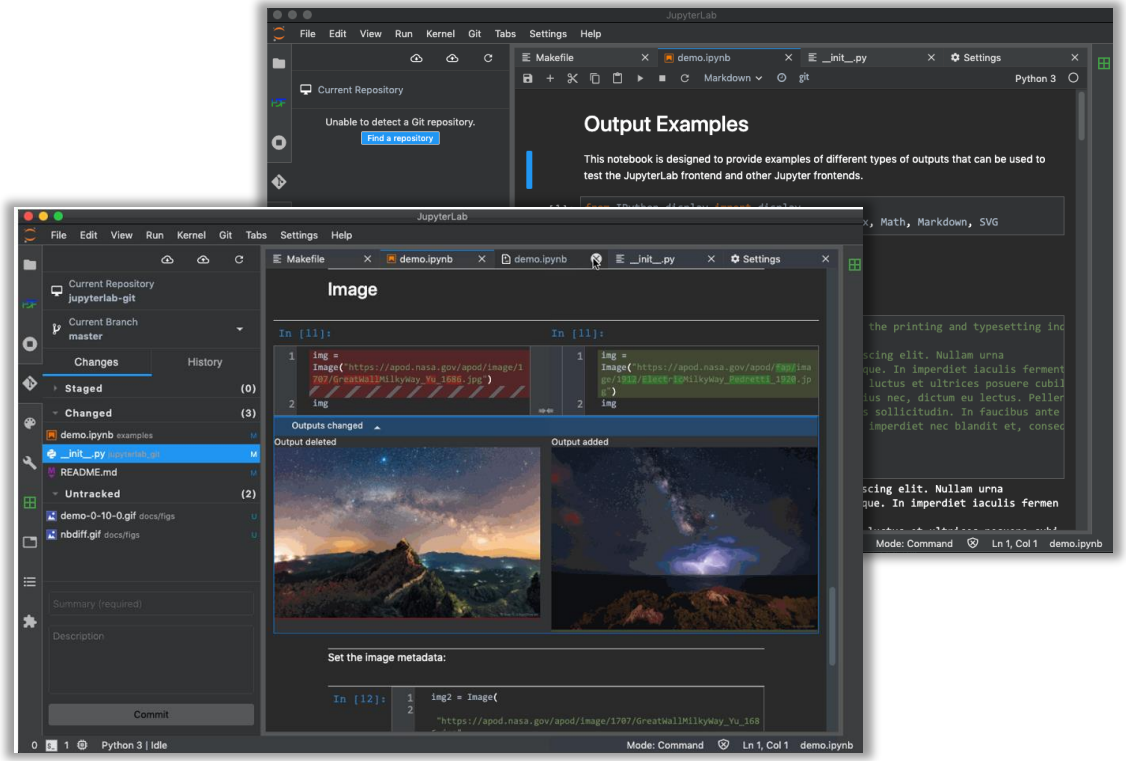
Installed by default

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



https://github.com/maartenbreddels/ipyvolume

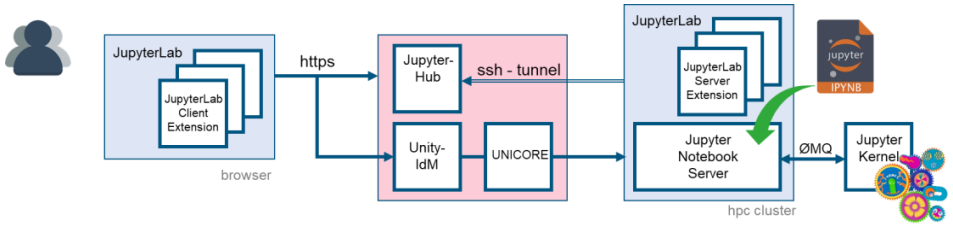
JupyterLab-Git
JupyterLab extension for version control using Git



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

JUPYTER-JSC EXTENSIONS

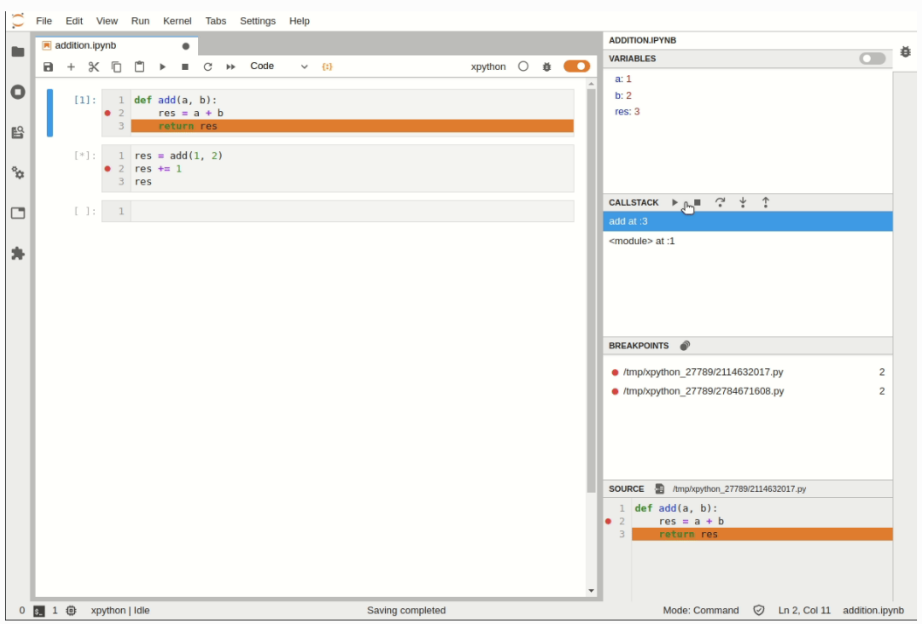
Installed by default



JupyterLab - Visual Debugger

JupyterLab 3.0 now ships with a Debugger front-end by default.

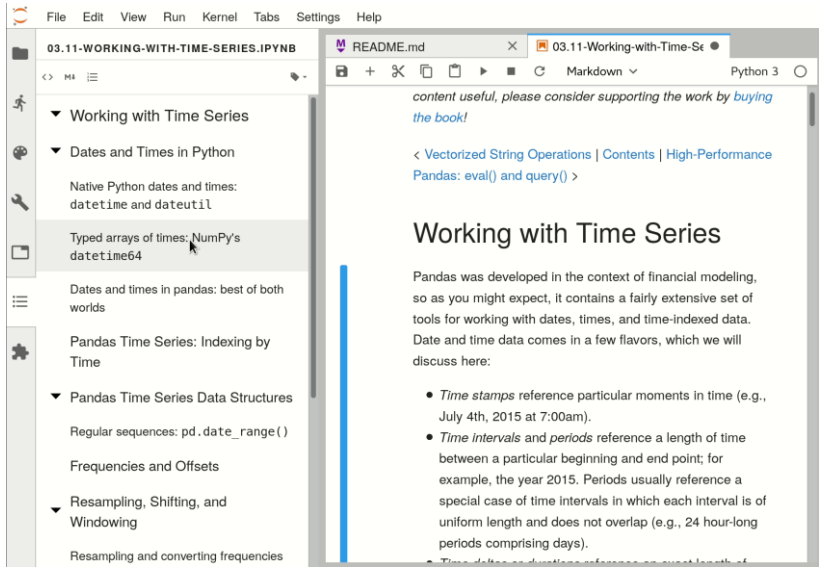
This means that notebooks, code consoles and files can now be debugged from JupyterLab directly! For the debugger to be enabled and visible, a kernel with support for debugging is required.



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

JupyterLab-toc

A Table of Contents extension for JupyterLab.
This auto-generates a table of contents in the left area when you have a notebook or markdown document open.The entries are clickable, and scroll the document to the heading in question.

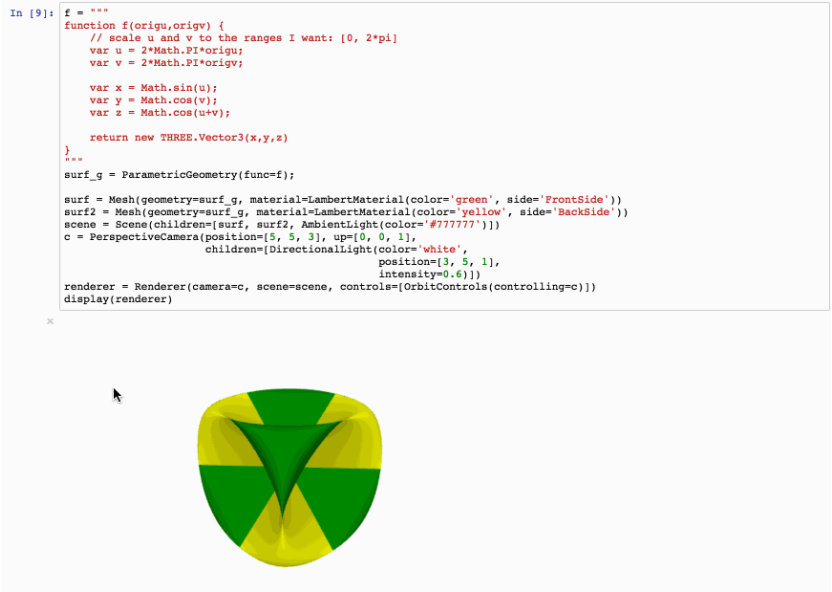


https://github.com/jupyterlab/jupyterlab-toc

JUPYTER-JSC EXTENSIONS

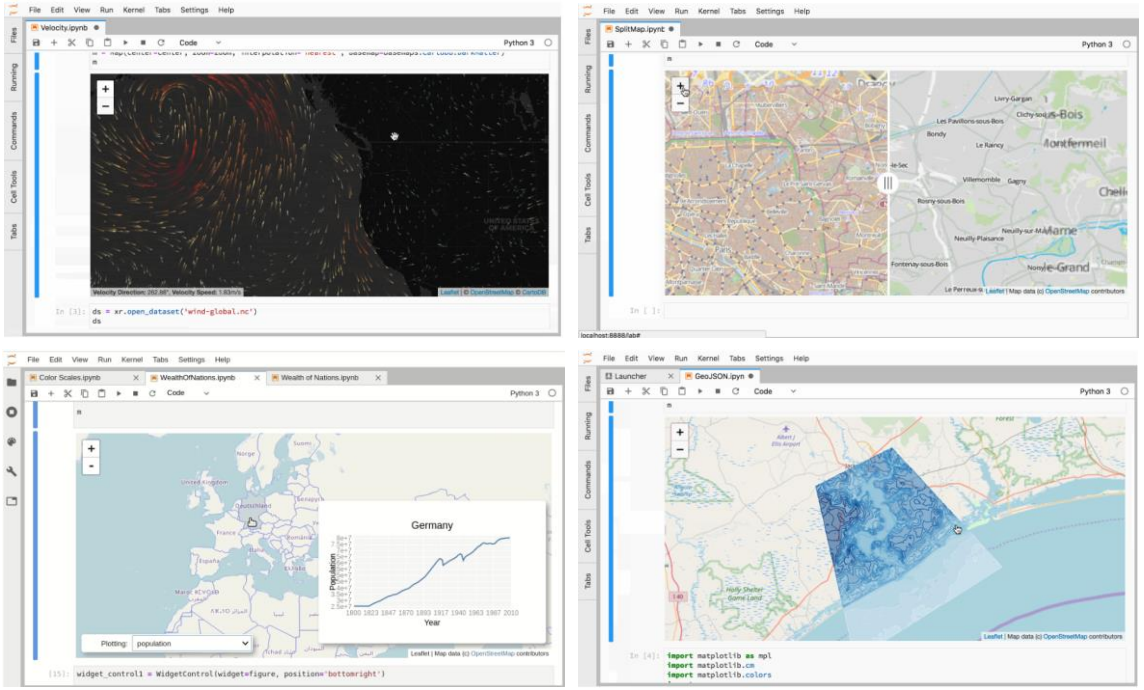
Installed by default

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



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

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

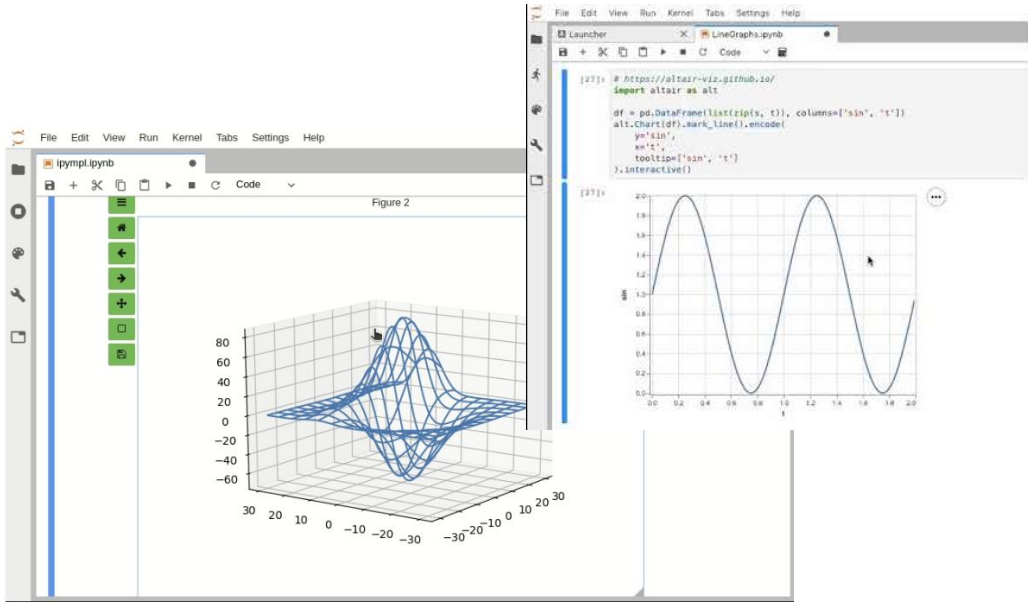


https://github.com/jupyter-widgets/ipyleaflet

JUPYTER-JSC EXTENSIONS

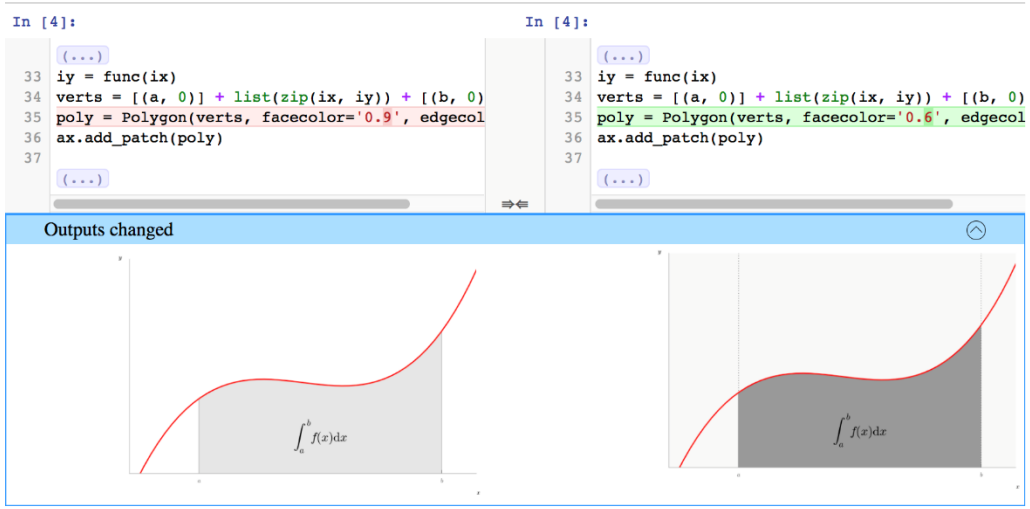
Installed by default

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



https://github.com/matplotlib/ipympl

NBDime
Tools for diffing and merging of Jupyter notebooks.



https://github.com/jupyter/nbdime

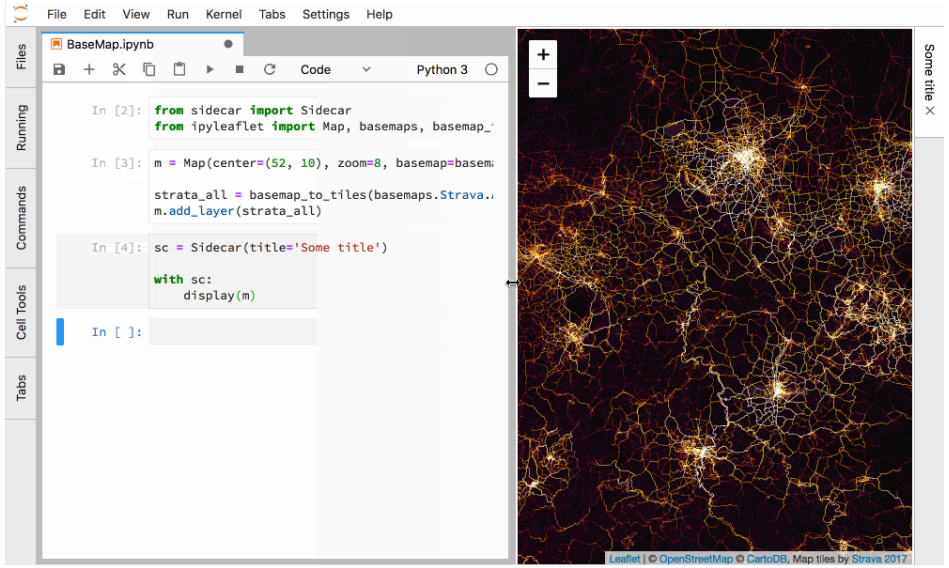
JUPYTER-JSC EXTENSIONS

Installed by default

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

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

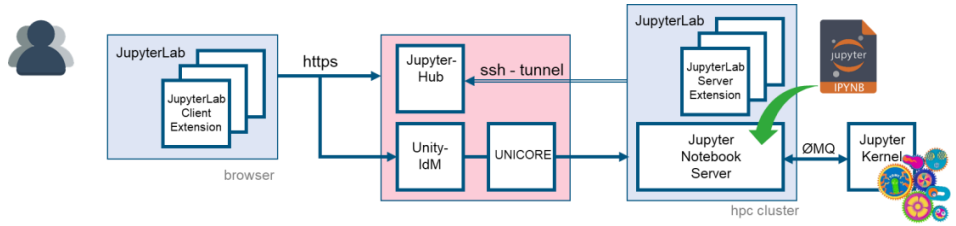
JupyterLab-Sidecar
A sidecar output widget for JupyterLab.



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

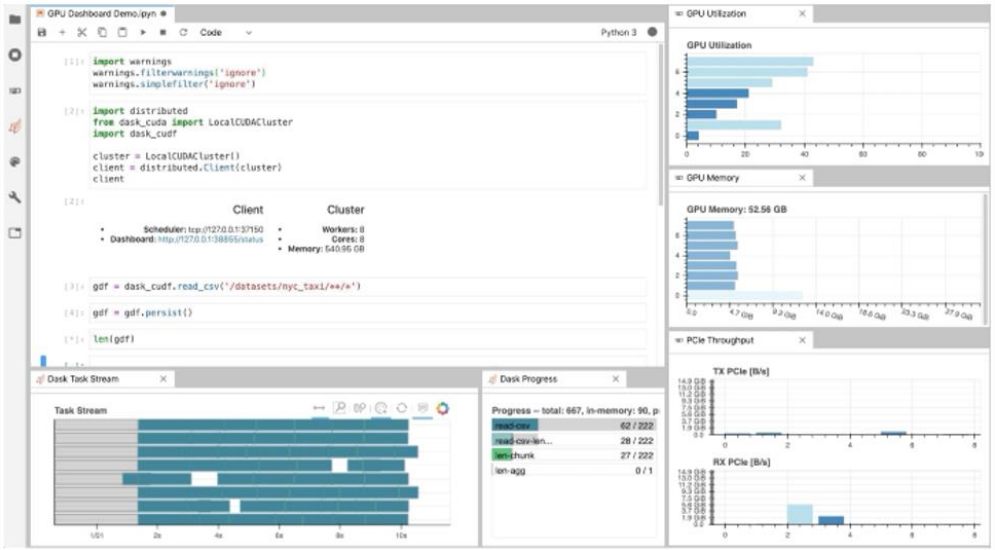
JUPYTER-JSC EXTENSIONS

Installed by default



NVDashboard

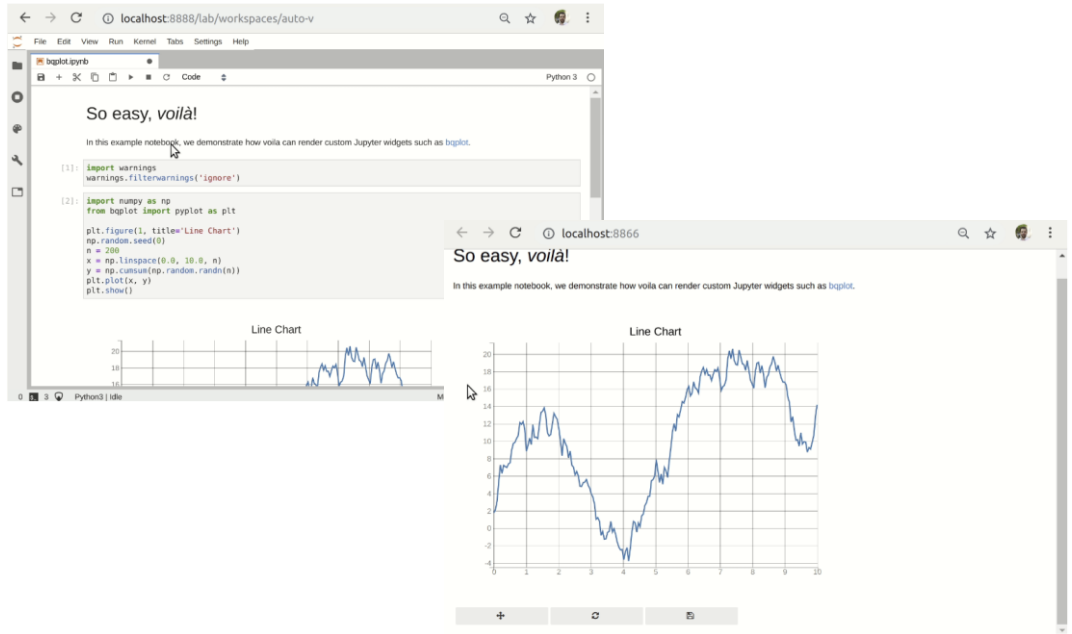
NVDashboard is an open-source package for the real-time visualization of NVIDIA GPU metrics in interactive Jupyter Lab environments.



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

Voilà

Voilà turns Jupyter notebooks into standalone web applications.

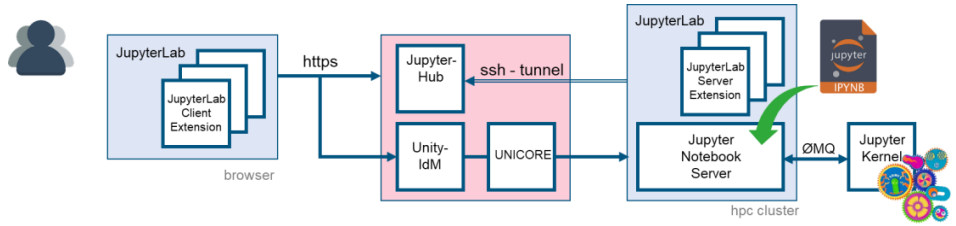


https://github.com/voila-dashboards/voila

JUPYTER-JSC EXTENSIONS

Installed by default

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



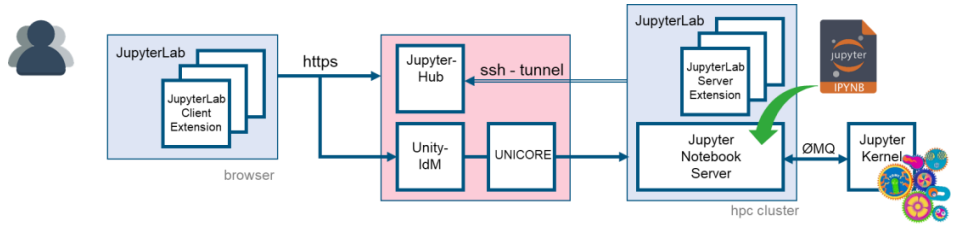
Data Visualization			
jupyter-matplotlib	v0.7.4	v0.9.0	prebuild
@bokeh/jupyter_bokeh	v2.0.4	v3.0.4	prebuild
jupyterlab-plotly	v4.14.3	v5.3.1	
bqplot	v0.5.22	v0.5.32	prebuild
@pyviz/jupyterlab_pyviz	v1.0.4	v2.1.0	prebuild
jupyter-leaflet	v0.13.3	v0.14.0	prebuild
ipyvolume	v0.6.0-alpha.5	v0.6.0-alpha.8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

https://docs.jupyter-jsc.fz-juelich.de/github/FZJ-JSC/jupyter-jsc-notebooks/blob/master/Announcement-2021-11_JupyterLab3_at_Jupyter-JSC.ipynb

JUPYTER KERNEL

JUPYTER KERNEL

How to create your own Juypter Kernel

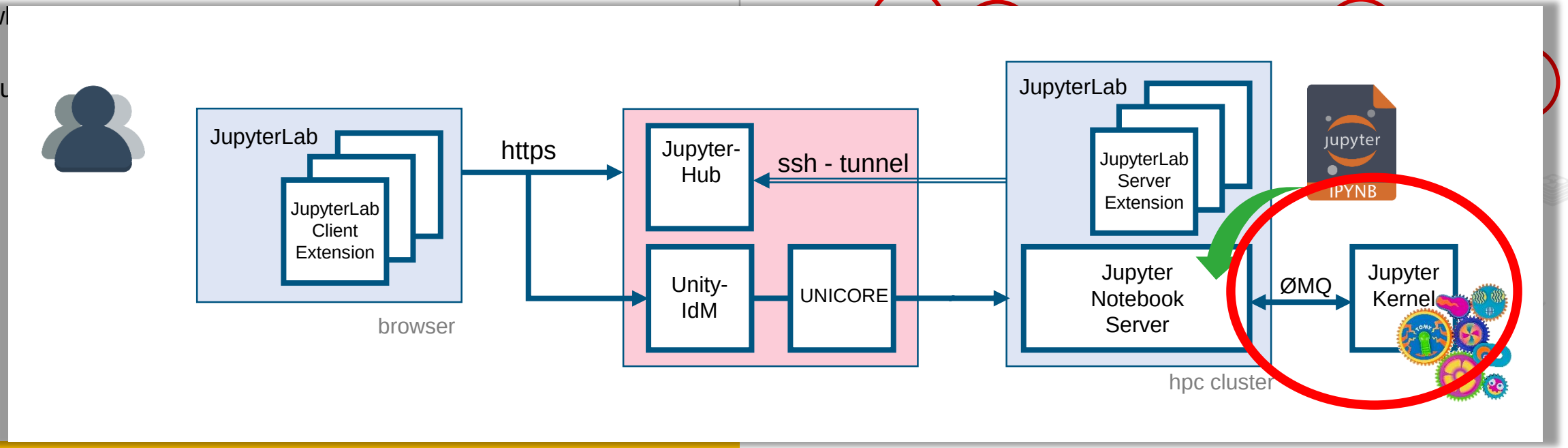


Jupyter Kernel

A “kernel” refers to the separate process wh

Ju

-
-
-
-

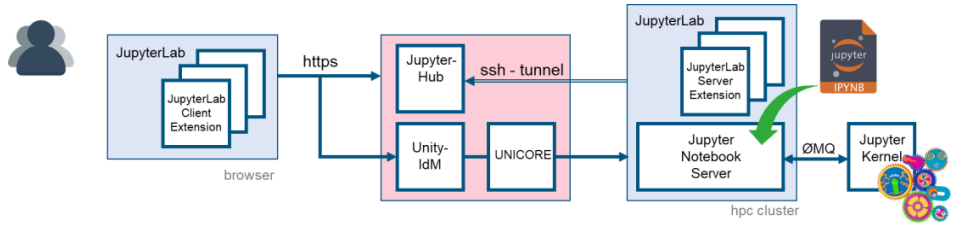


You can easily **create your own kernel** which for example runs your specialized virtual Python environment.

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

JUPYTER KERNEL

How to create your own Juypter Kernel



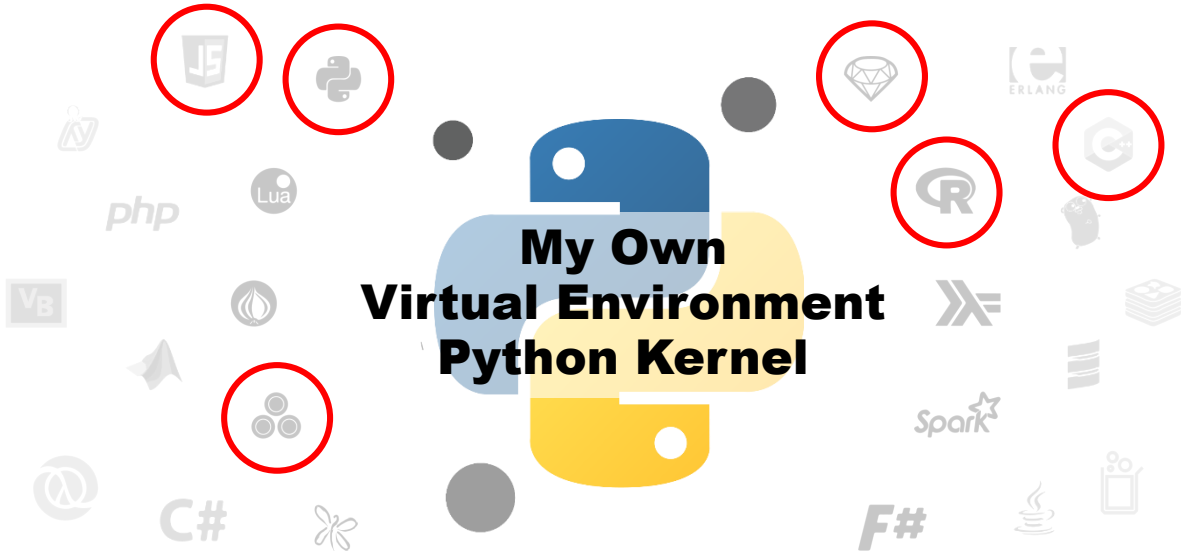
Jupyter Kernel

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, quantumcomputing

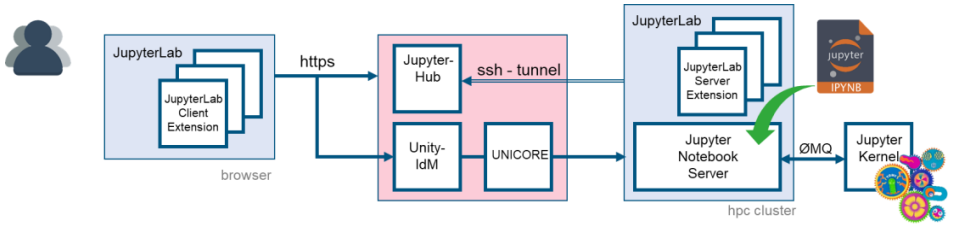


You can easily **create your own kernel** which for example runs your specialized virtual Python environment.

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, quantumcomputing

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

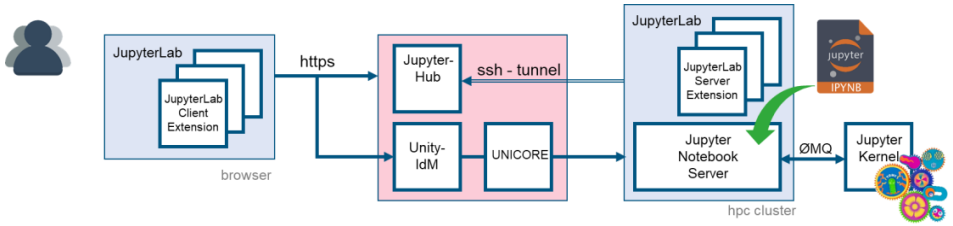
Building your own Jupyter kernel is a three step process

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 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**.
- clusters
  - Python, R, Julia, Bash, C++, Ruby, JavaScript
  - Specialized kernels for visualization, quantumcomputing

Building your own Jupyter kernel is a three step process

1. Create/Pimp new **virtual Python environment**
   `venv`
   `kernel.json`

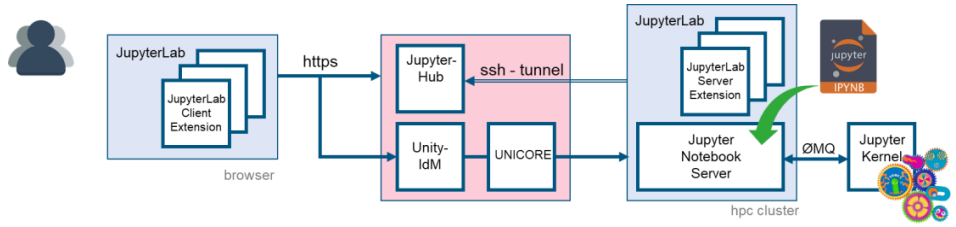
https://gitlab.version.fz-juelich.de/jupyter4jsc/j4j_notebooks/-/blob/master/001-Jupyter/Create_JupyterKernel_general.ipynb

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

Run your Jupyter kernel configuration



Run your Jupyter Kernel

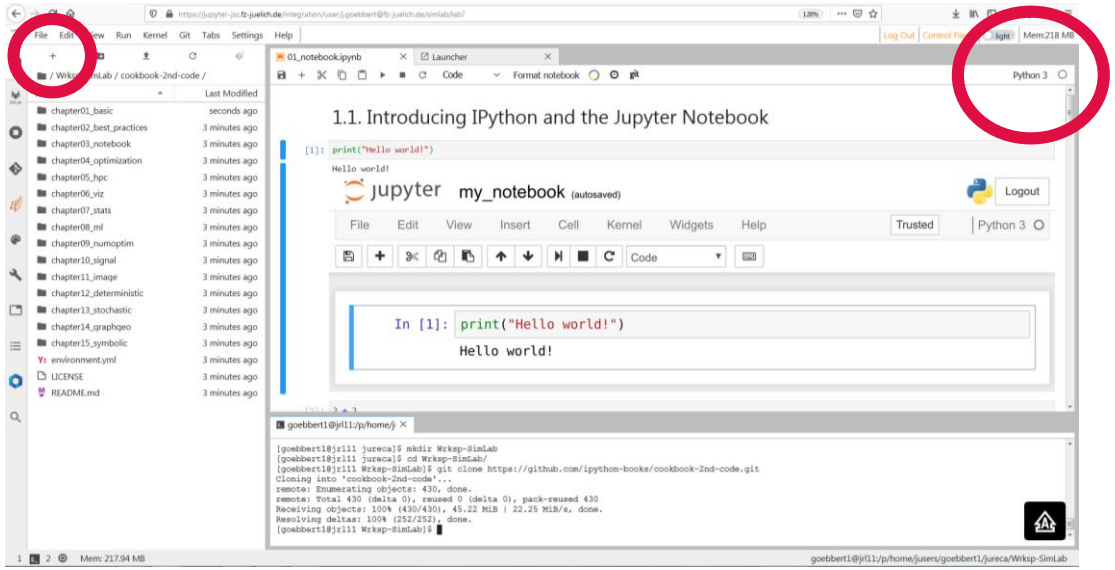
1. https://jupyter-jsc.fz-juelich.de
2. Choose system where your Jupyter kernel is installed in `~/.local/share/jupyter/kernels`
3. Select your kernel in the launch pad or click the kernel name.

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

Project kernel

On request Jupyter kernel can be made available to a whole project. They are installed then to
`$PROJECT/.local/share/jupyter/kernels`



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

JUPYTER CAN DO MORE

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.

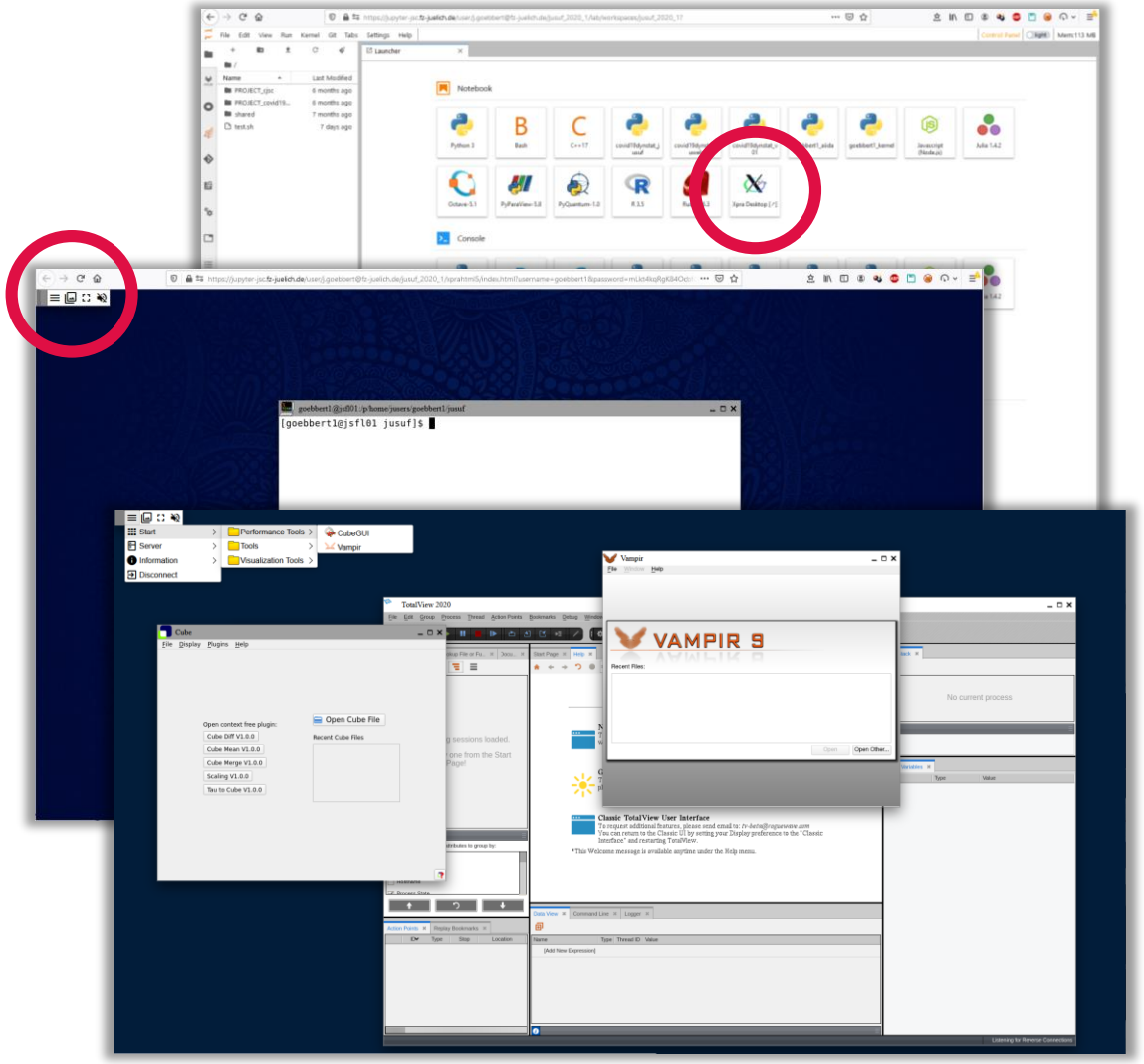
- Runs in a browser
- allows dis-/reconnection without disrupting the forwarded application
- https://xpra.org

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`



JÜLICH Forschungszentrum

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”

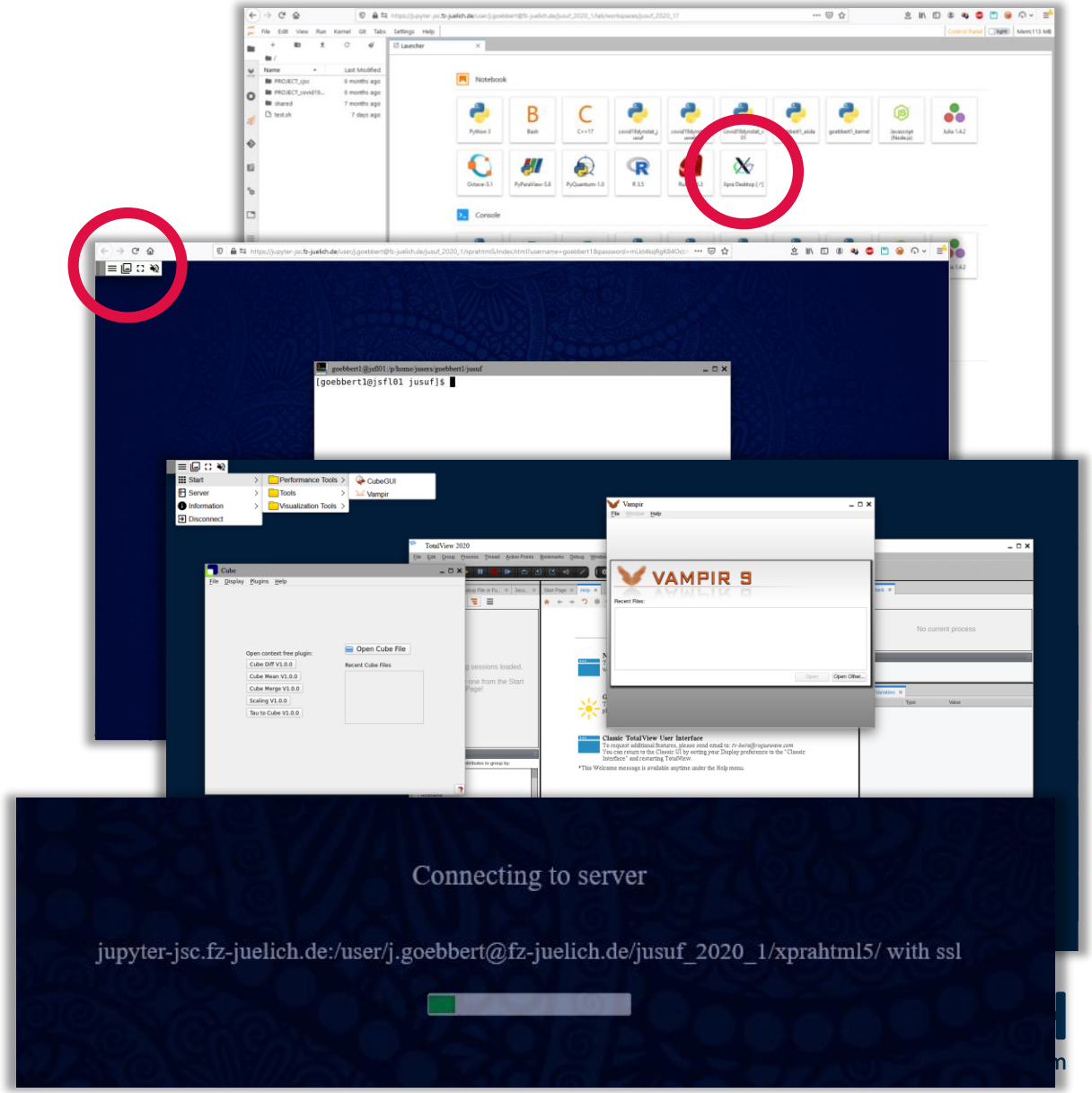
Xpra - X Persistent Remote Applications
is a tool which runs X clients on a remote host and directs their display to the local machine.

- 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.

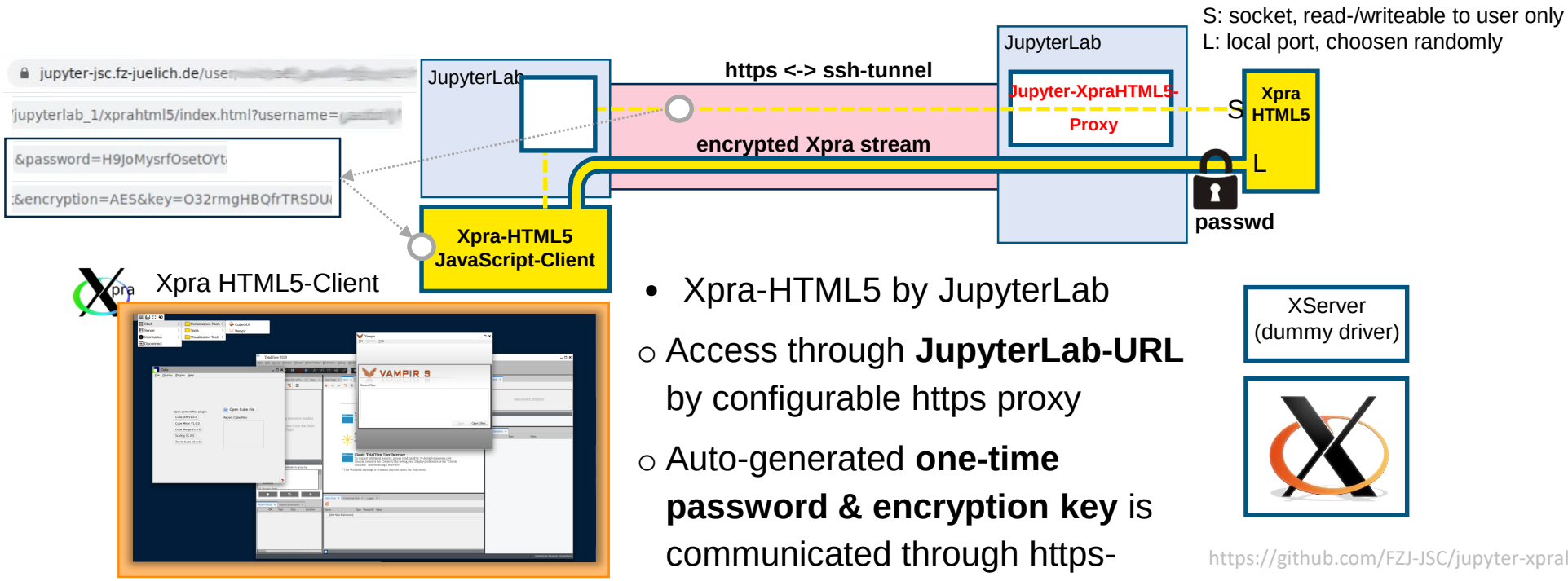
Hint:

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



JUPYTERLAB – REMOTE DESKTOP

Run your X11-Applications in the browser



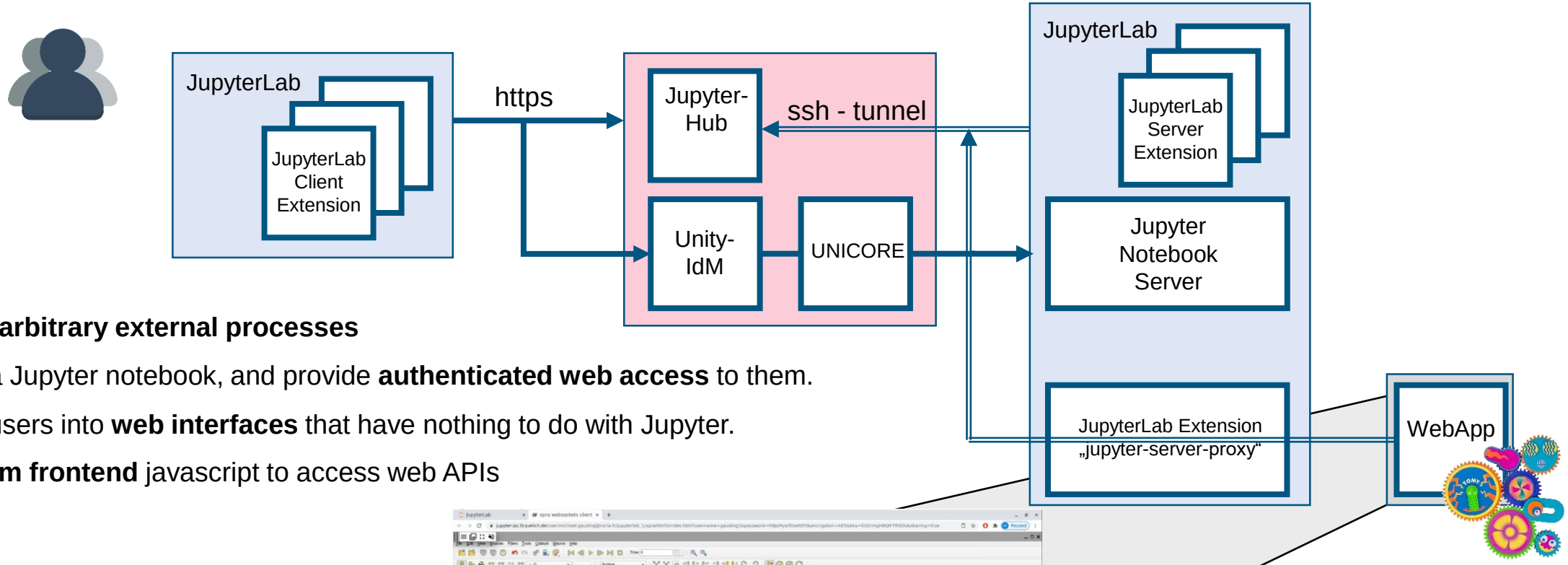
- Xpra-HTML5 by JupyterLab
  - Access through **JupyterLab-URL** by configurable https proxy
  - Auto-generated **one-time password & encryption key** is communicated through https-proxy

XServer (dummy driver)

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

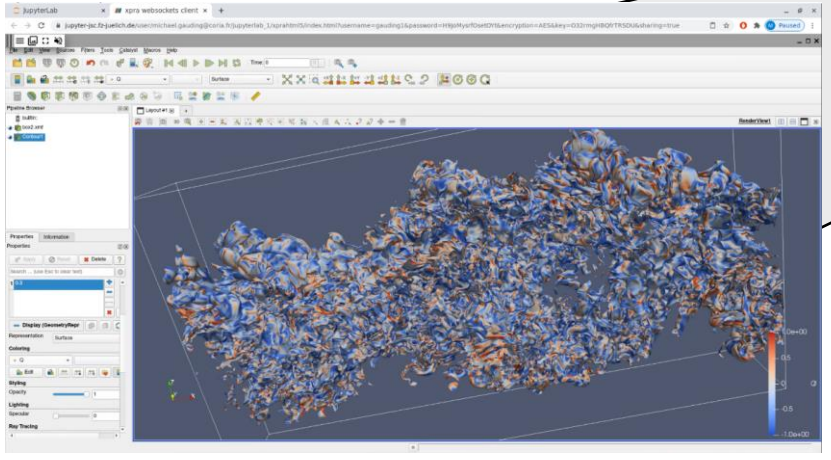
JUPYTERLAB – WEBSERVICE PROXY

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

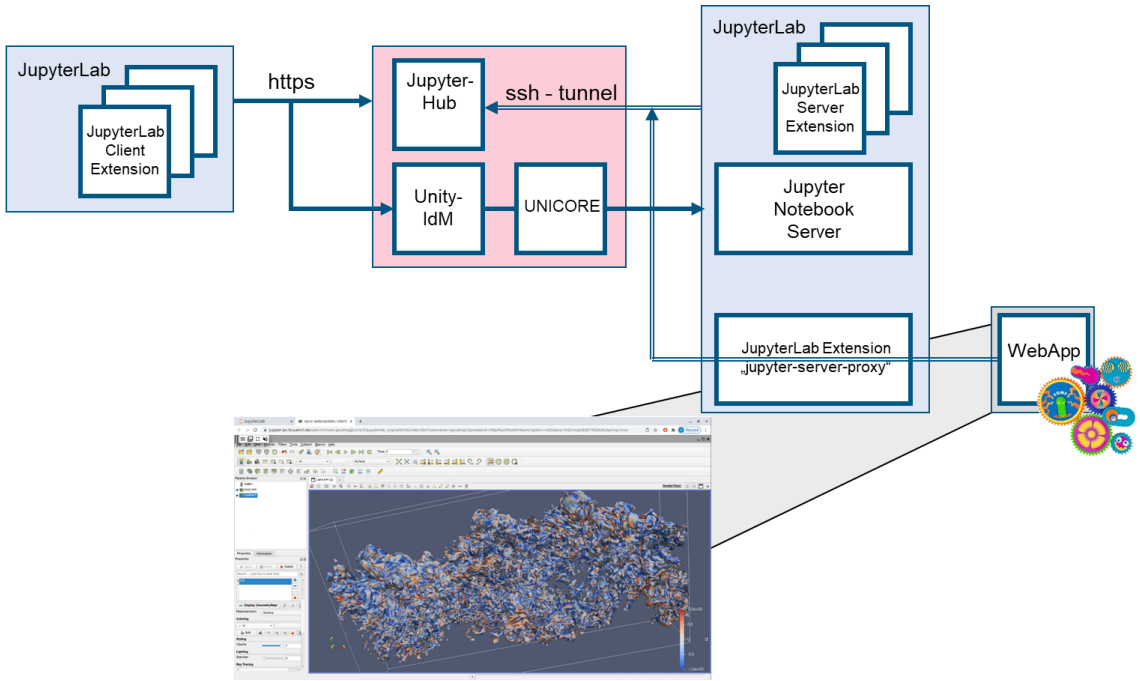
Extension: jupyter-server-proxy

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.

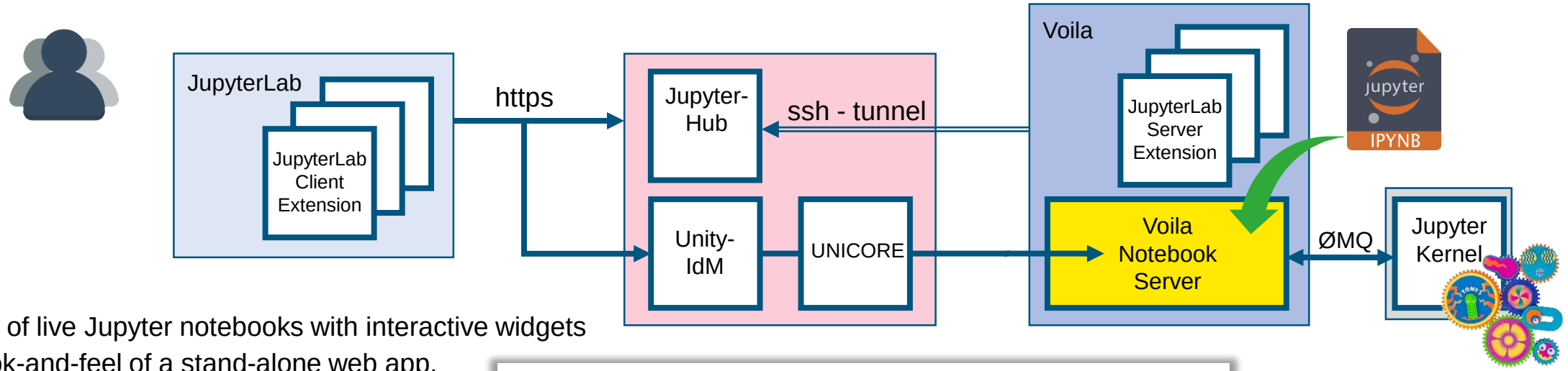


Example:
https://jupyter-jsc.fz-juelich.de/user/j.goebbert@fz-juelich.de/juwels_login/**proxy/12345**

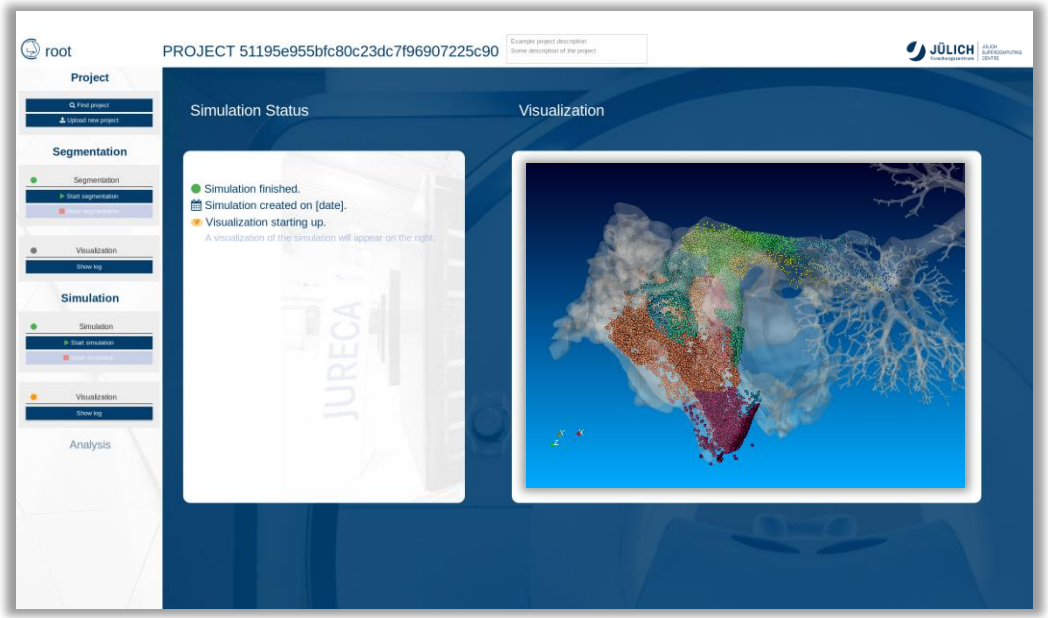
https://jupyter-server-proxy.readthedocs.io/en/latest/arbitrary-ports-hosts.html

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.



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

JÜLICH Forschungszentrum

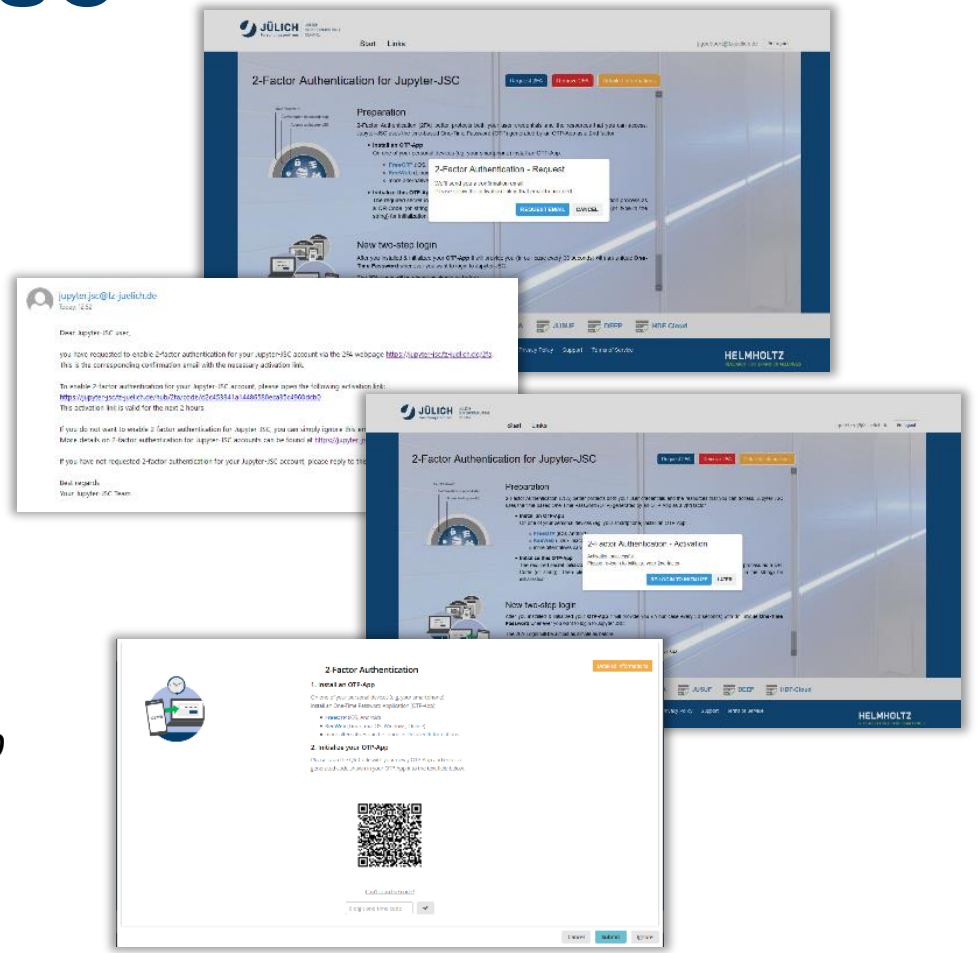
ENABLE 2FA FOR JUPYTER-JSC

To get ready to use 2-Factor Authentication (2FA)
for Jupyter-JSC you have to **prepare** it ONCE:

(1) **request 2FA** for Jupyter-JSC,
- (a) login to Jupyter-JSC
- (b) visit https://jupyter-jsc.fz-juelich.de/2fa and request 2FA
- (c) wait for a *confirmation emails* and
  click the provided *activation link*

(2) **activate 2FA** for Juypter-JSC,
- (a) install an **OTP-App**, which supports the TOTP algorithm
- (b) communicate the **secret initialization code** to this *OTP-App*
- (c) test a first **one-time password** generated.

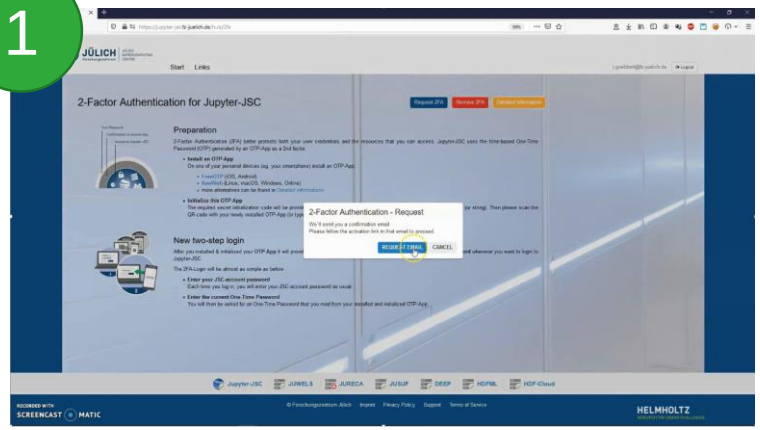
... and then 2FA is ready to be used next time you log in.



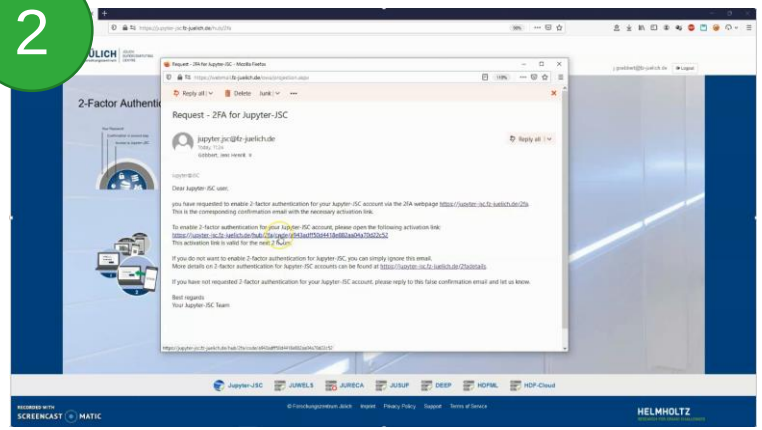
https://jupyter-jsc.fz-juelich.de/hub/2FA

More details on https://docs.jupyter-jsc.fz-juelich.de/github/FZJ-JSC/jupyter-jsc-notebooks/blob/master/001-Jupyter/Activate_JupyterJSC_2-factor-authentication.ipynb

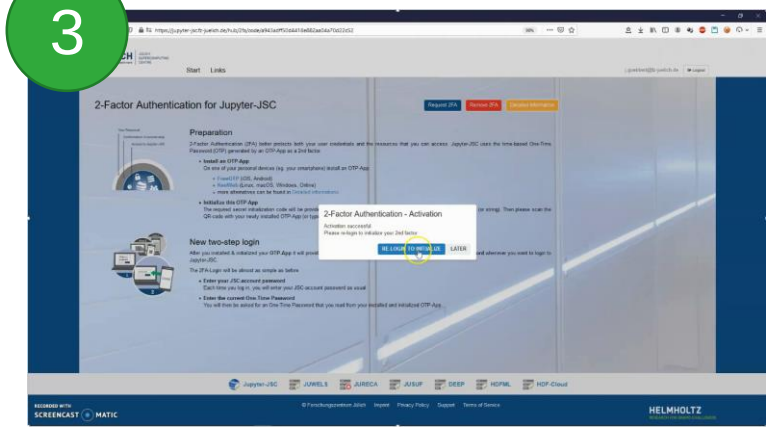
1


2-Factor Authentication for Jupyter-JSC
Preparation
New two-step login
2-Factor Authentication - Request

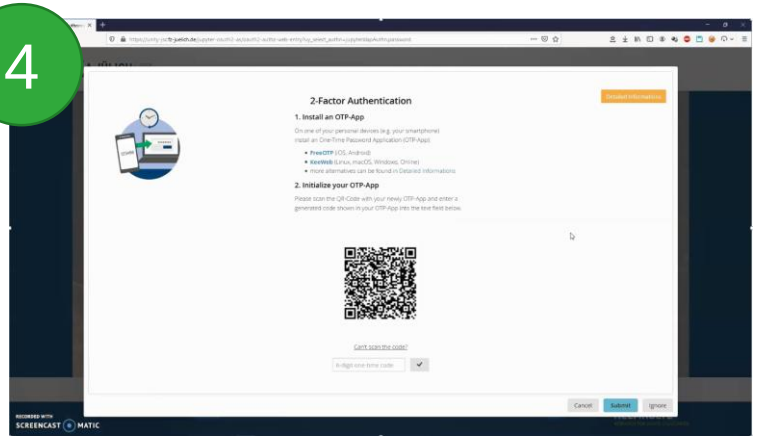
2


Request - 2FA for Jupyter-JSC

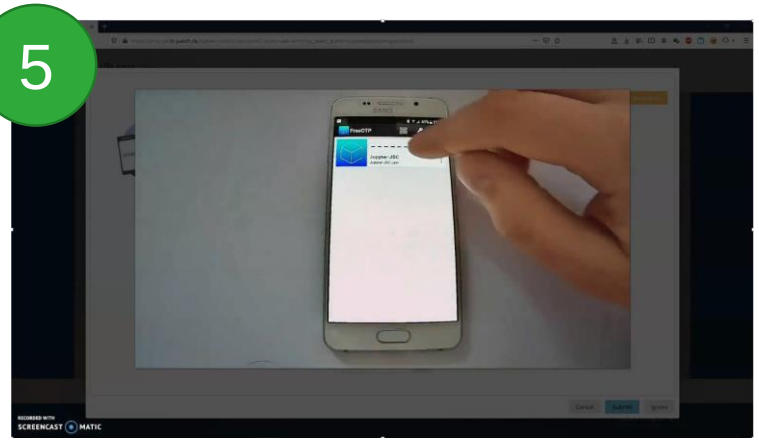
3


2-Factor Authentication for Jupyter-JSC
Preparation
New two-step login
2-Factor Authentication - Activation

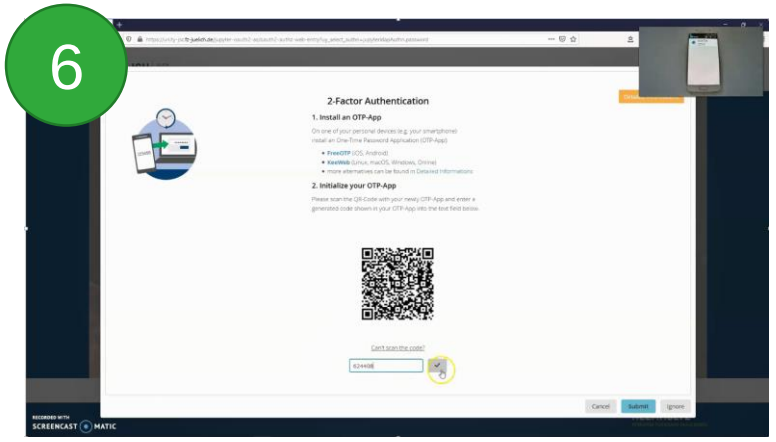
4


2-Factor Authentication
1. Install an OTP-App
2. Initialize your OTP-App

5



6

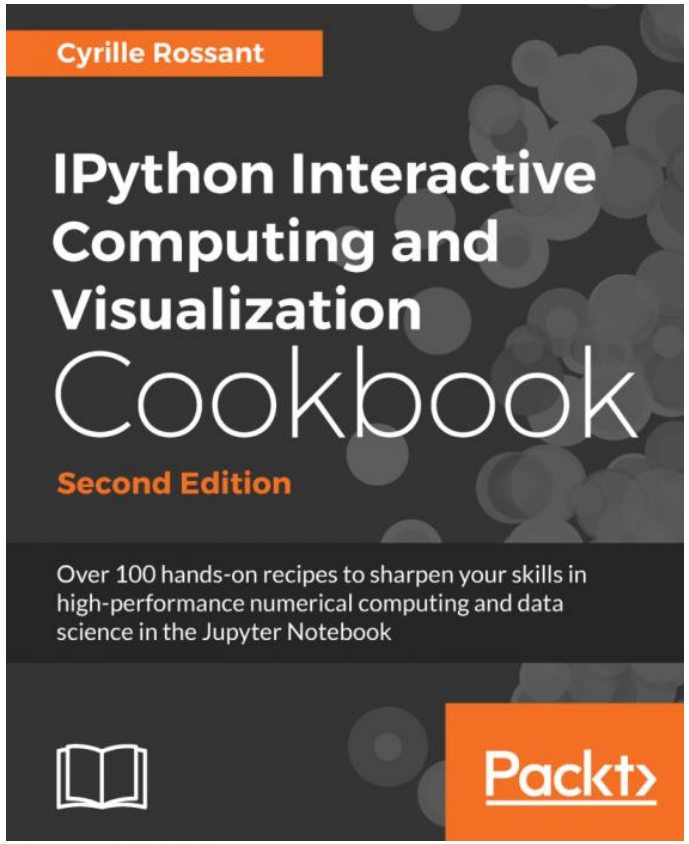

2-Factor Authentication
1. Install an OTP-App
2. Initialize your OTP-App

TUTORIALS

Get started with Jupyter

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

- 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



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

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.
- Platform and language **independent** because of its representation in JSON format.

QUESTIONS?

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