INTERACTIVE HIGH-PERFORMANCE COMPUTING WITH JUPYTERLAB

Webinar #2 – RISC2 Webinar Series

2022-09-22 I JENS H. GÖBBERT (J.GOEBBERT@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

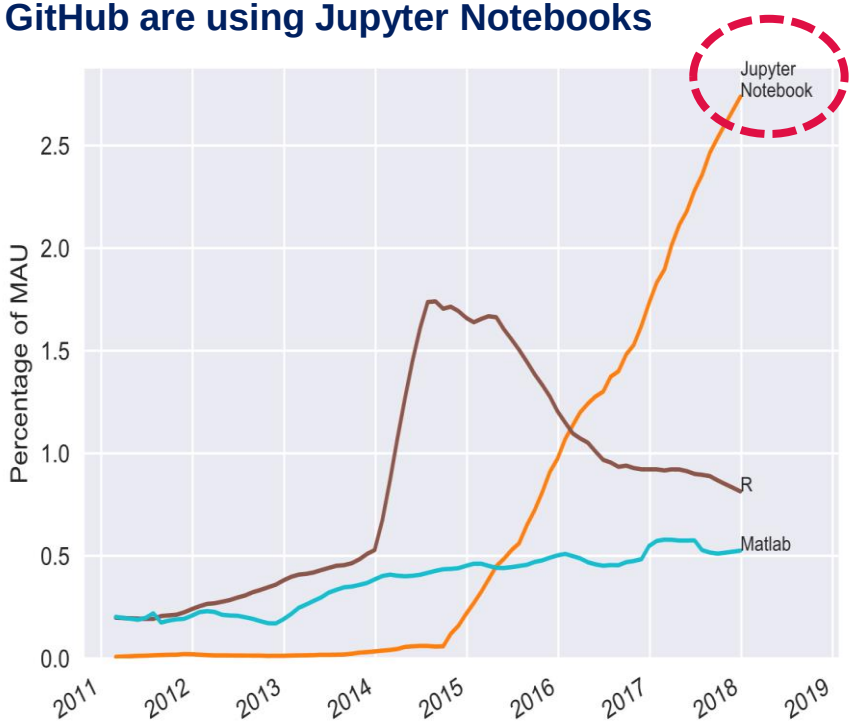
https://jupyter.org

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

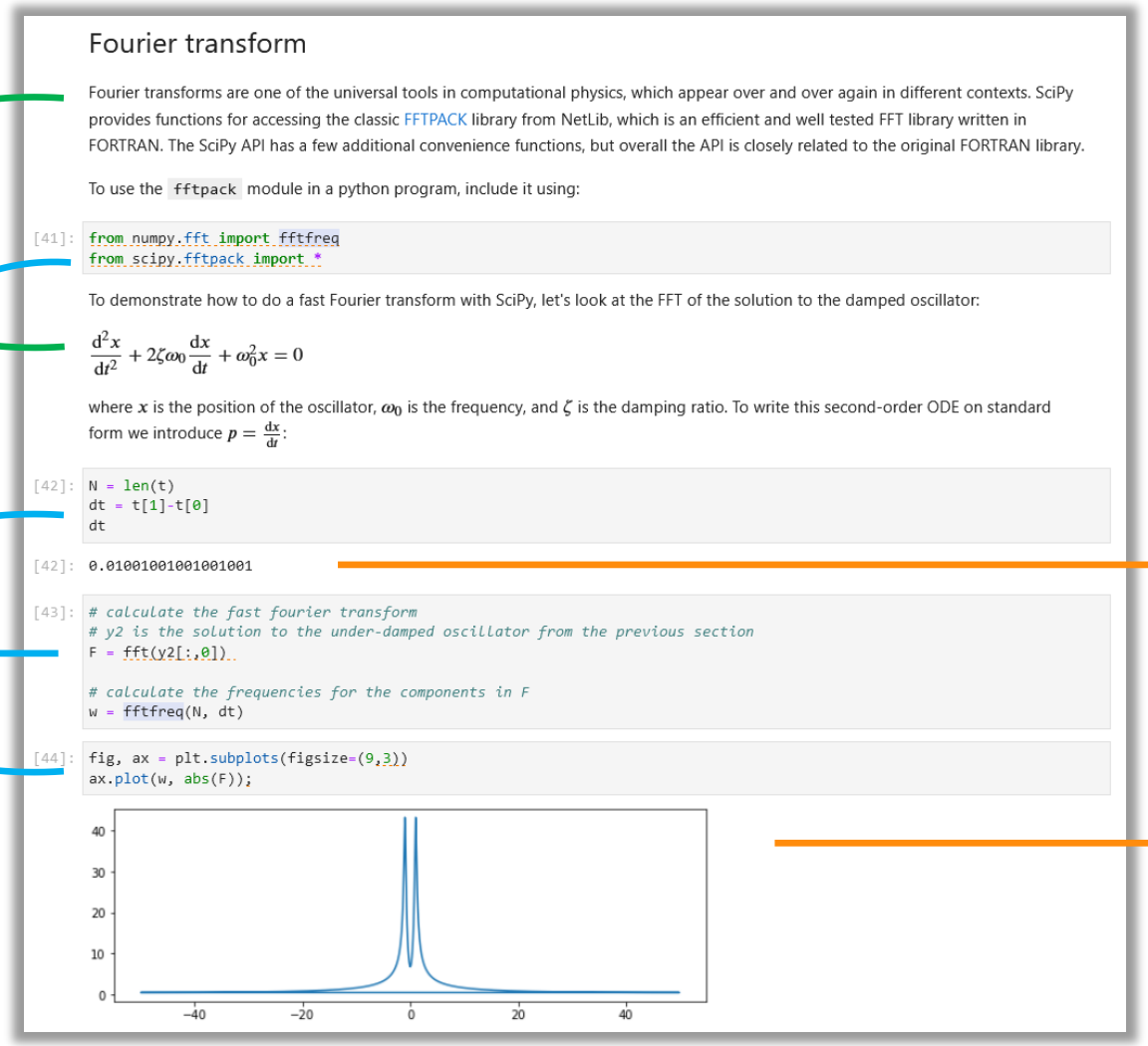
creating reproducible computational narratives

Markdown Cells

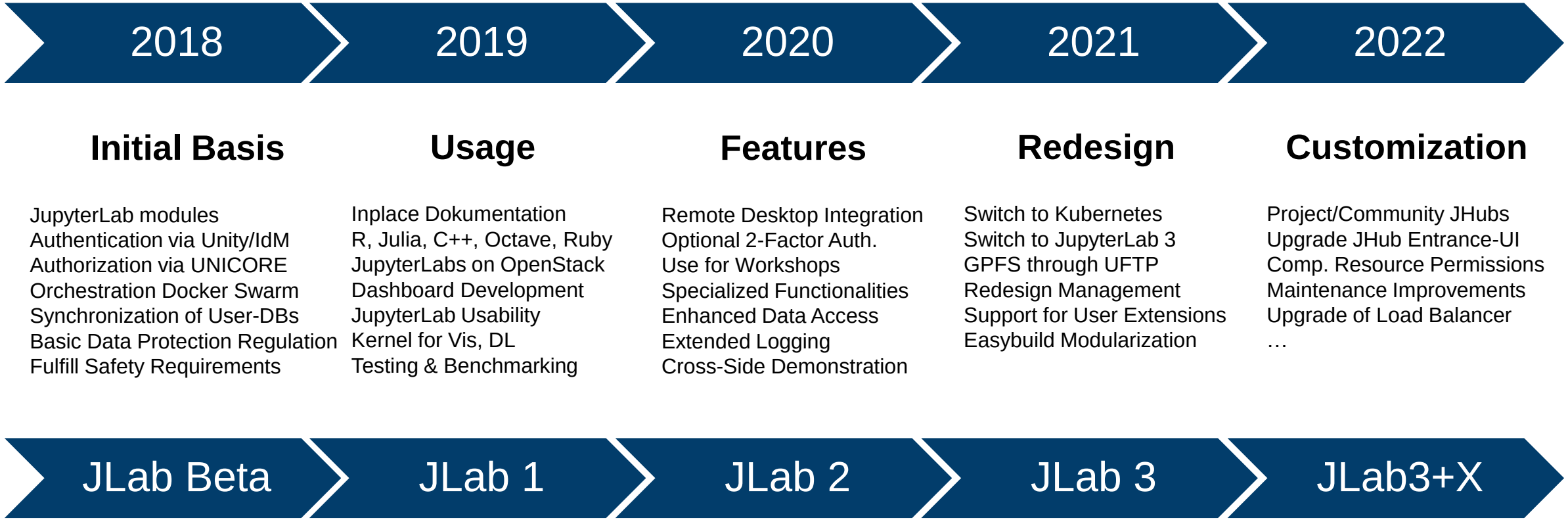
Code Cells

Output

Output



HISTORY OF JUPYTERLAB AT JSC

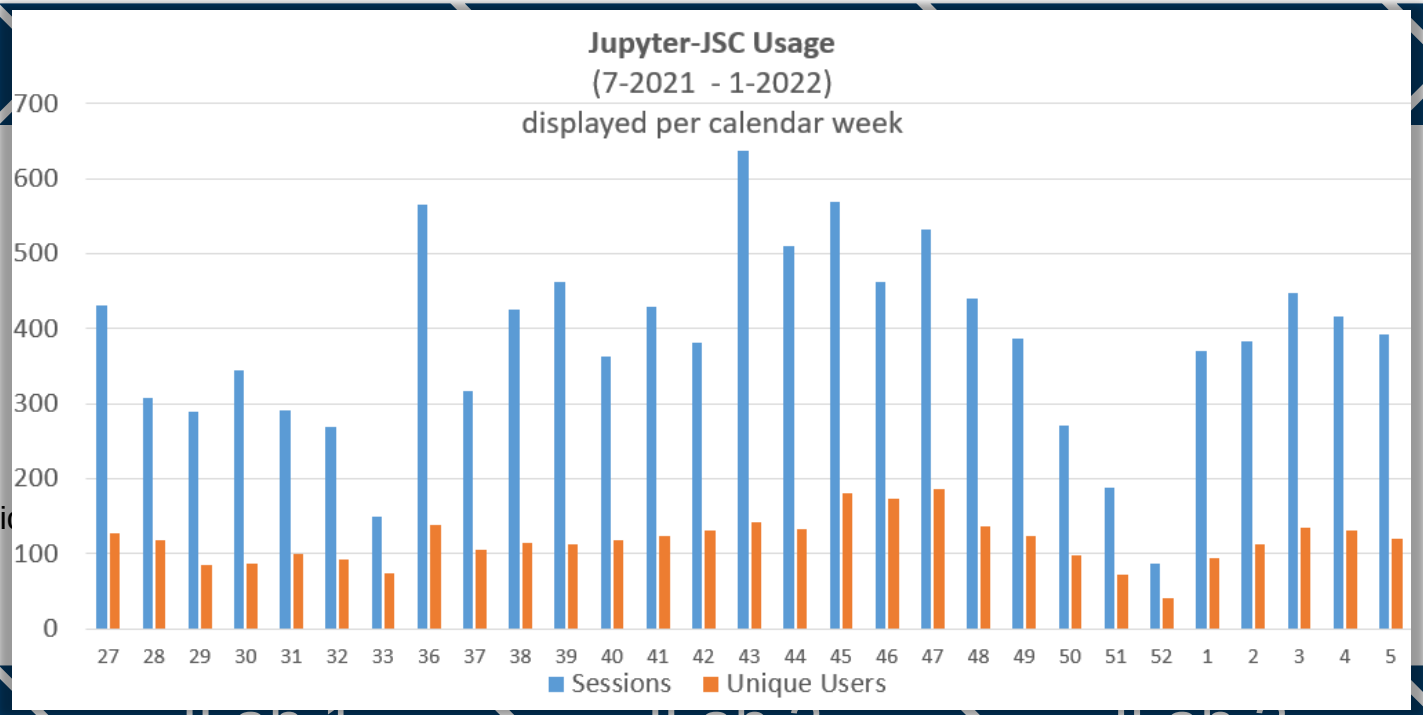


HISTORY OF JUPYTERLAB AT JSC

2018

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



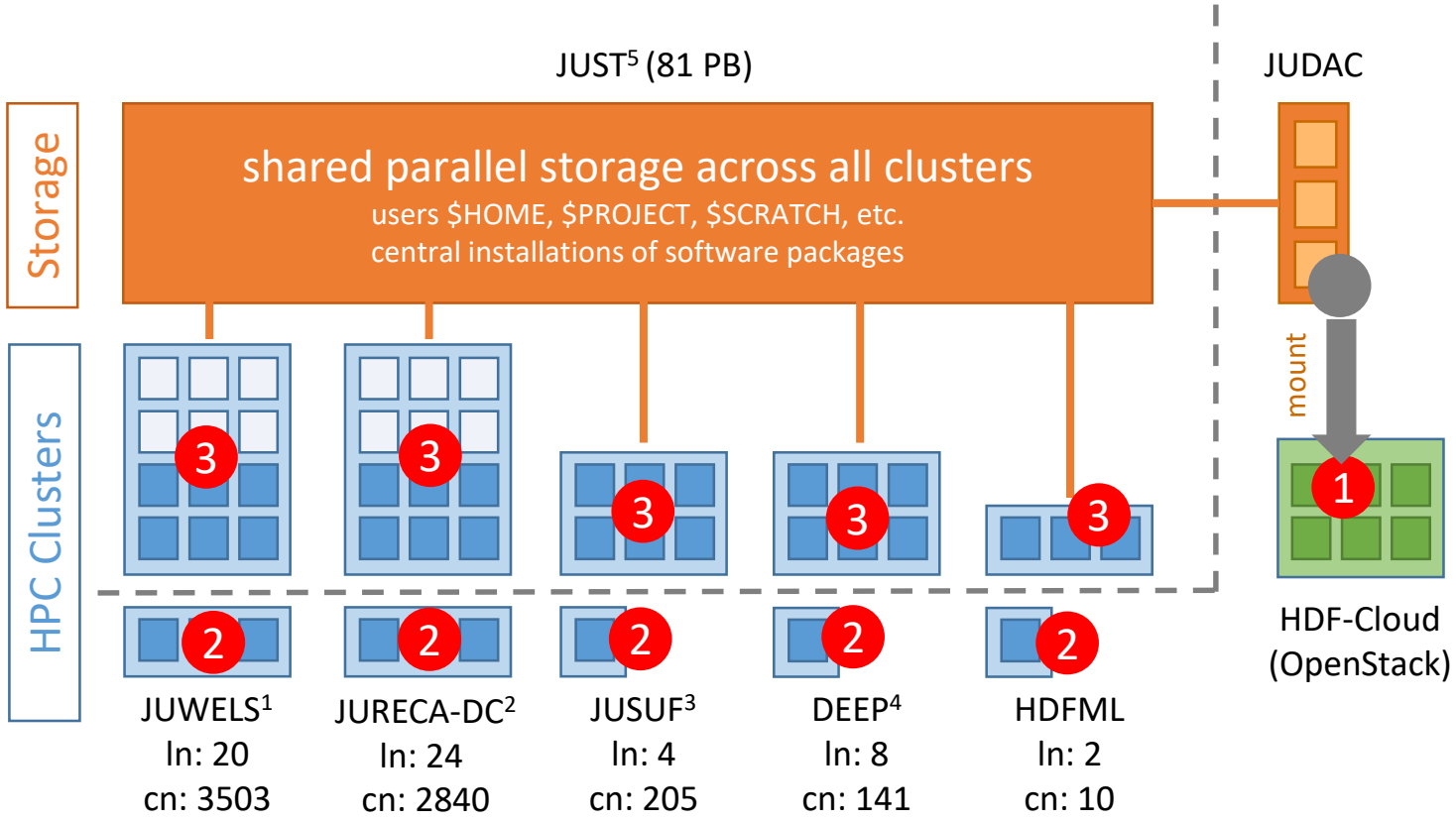
2022

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



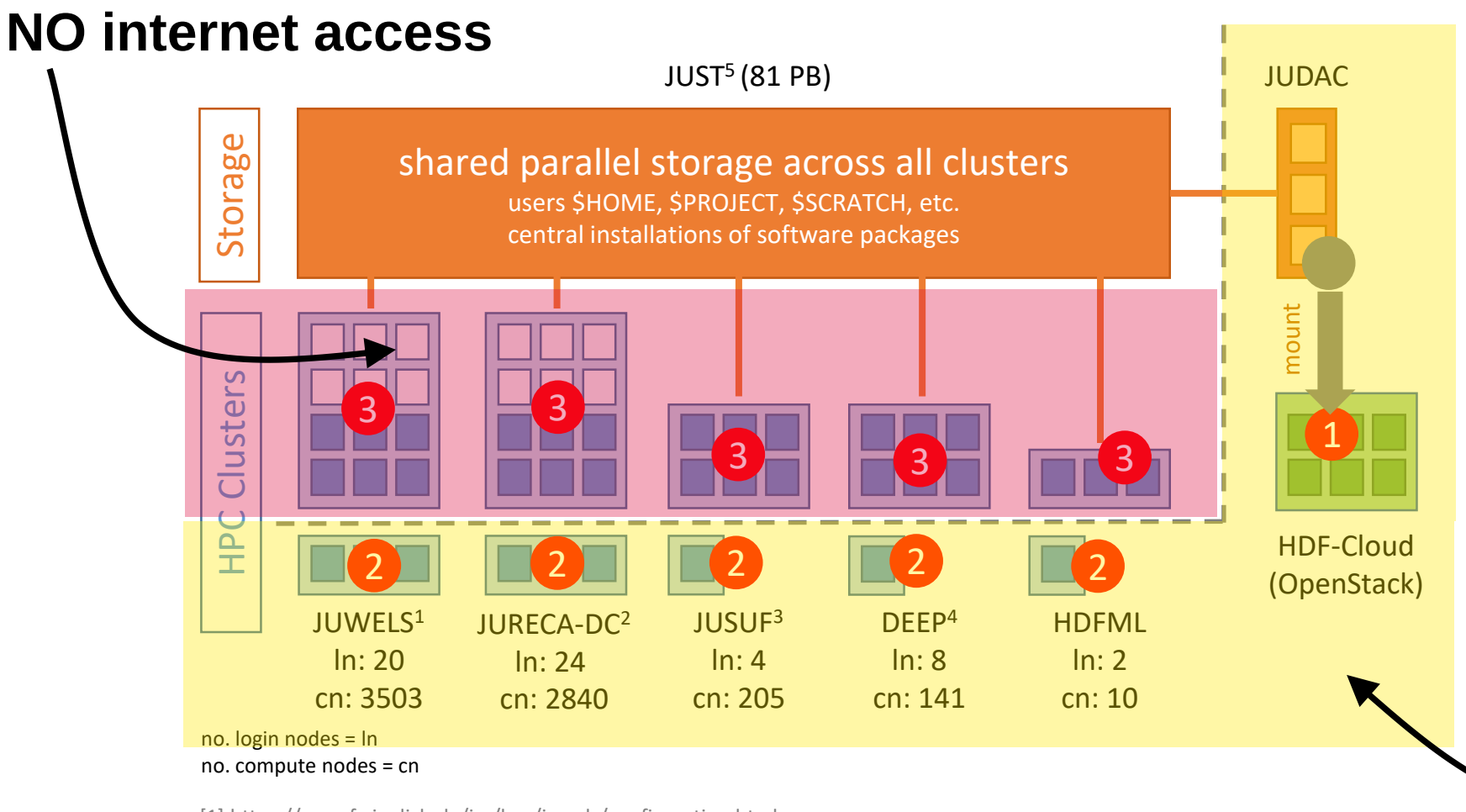
no. login nodes = ln
no. compute nodes = cn

JupyterLab everywhere

1. JupyterLab on 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

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

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

TERMINOLOGY

What is JupyterLab

JupyterLab

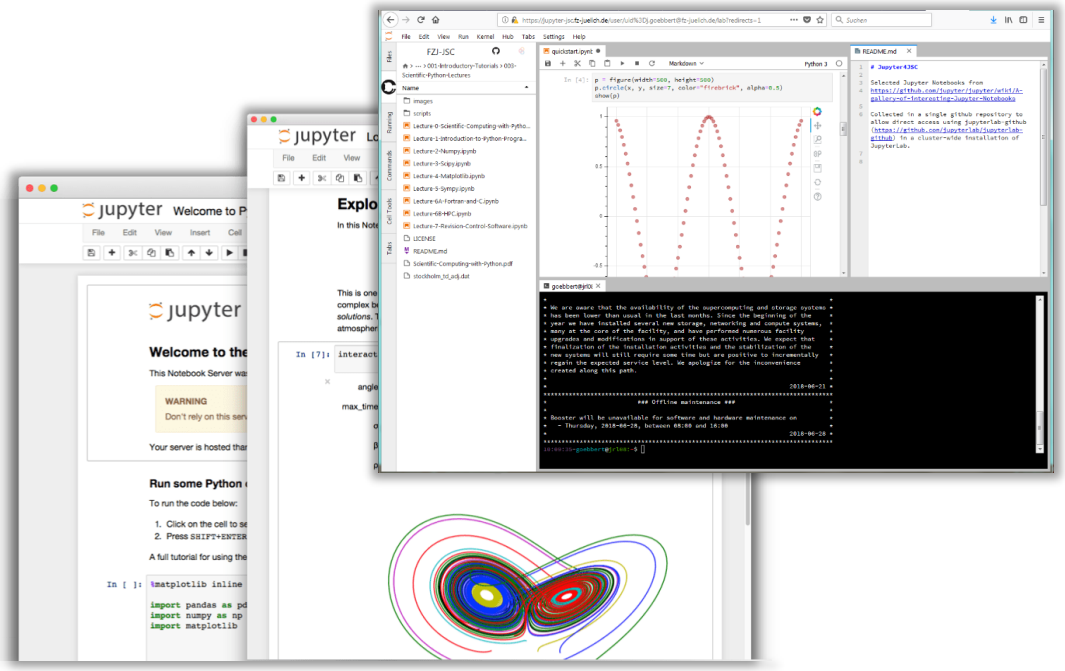
- **Interactive** working environment in the web browser
- For the creation of **reproducible** computer-aided narratives
- Very **popular** with researchers from all fields
- Jupyter = Julia + Python + R

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

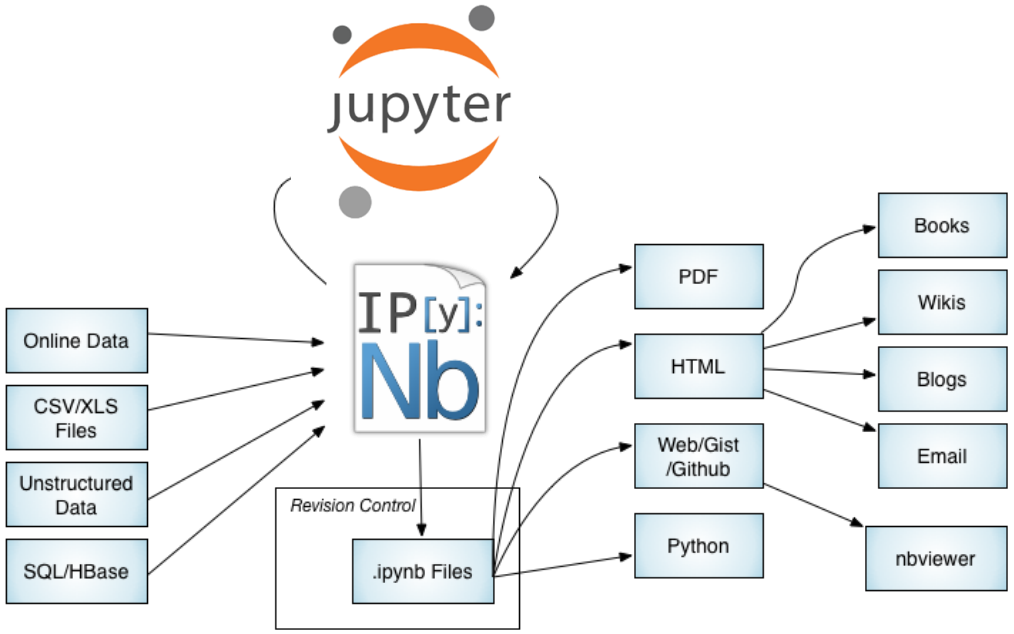
What is a Jupyter Notebook?

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?

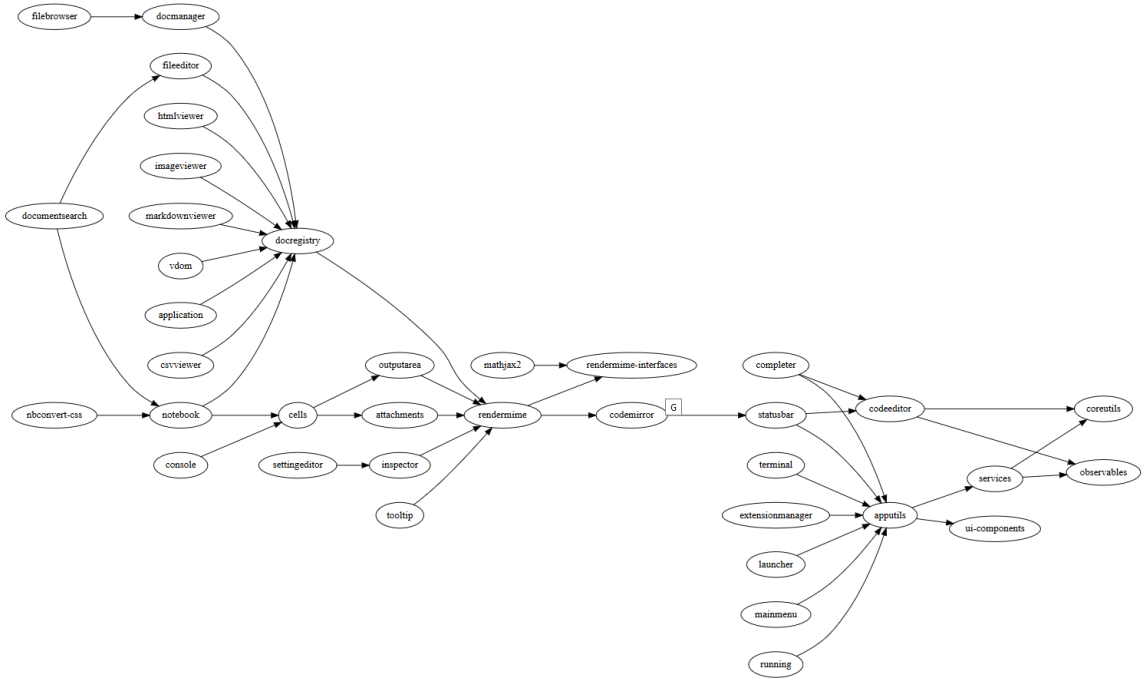
JupyterLab Extension

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**.

JUPYTERLAB - WHEREVER YOU PREFER

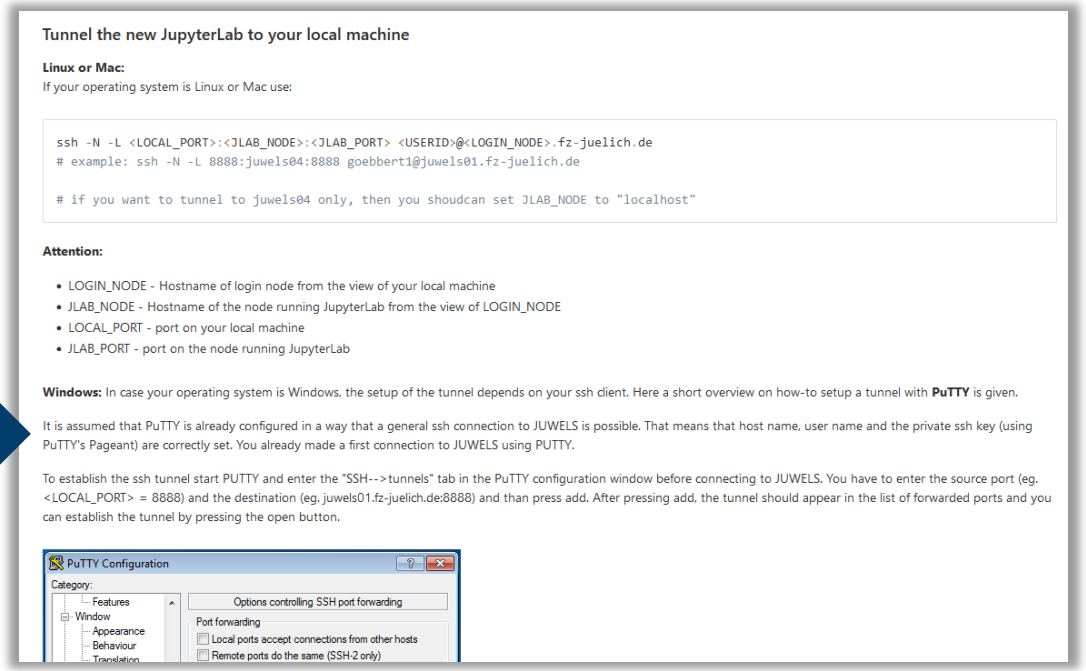
Local, Remote, Browser-only

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:

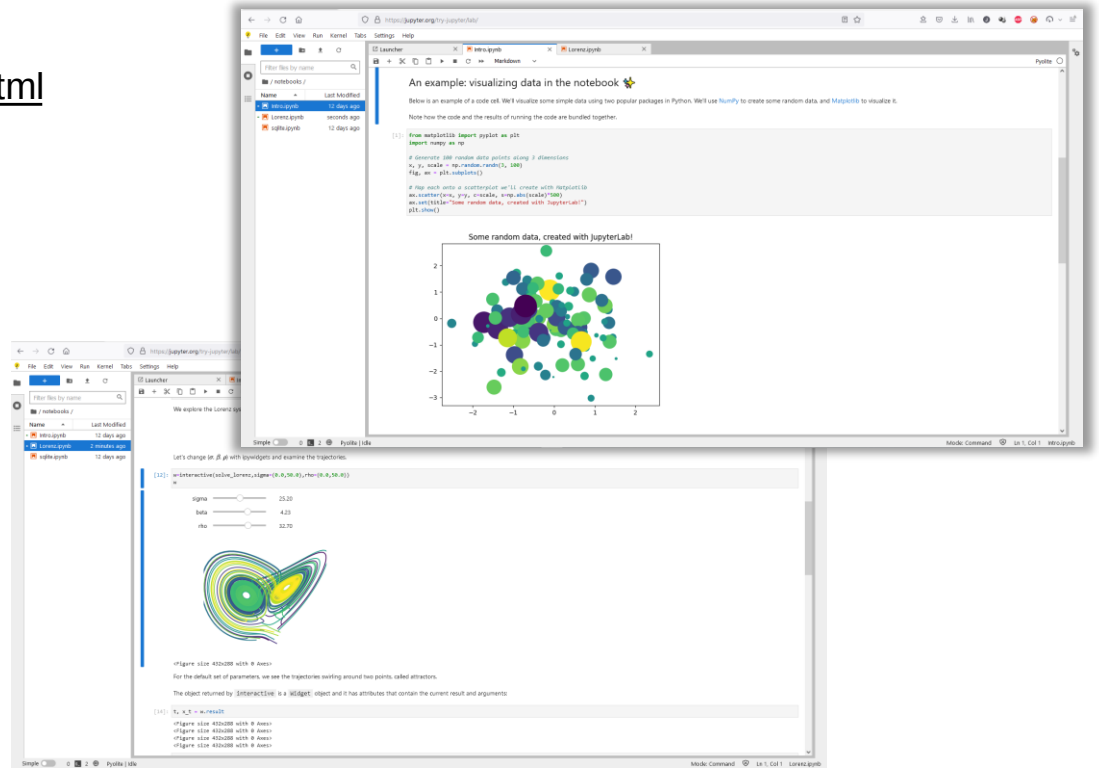
- **JupyterLab** installed using conda, mamba, pip, pipenv or docker.
  https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html

Remote (cluster) installation:

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

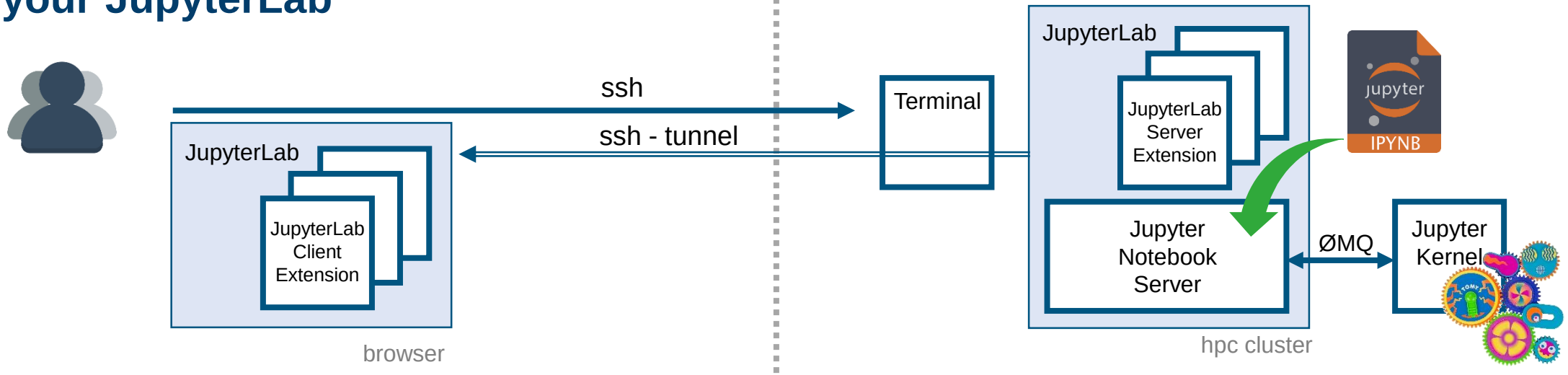
Browser-only installation (alpha!):

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



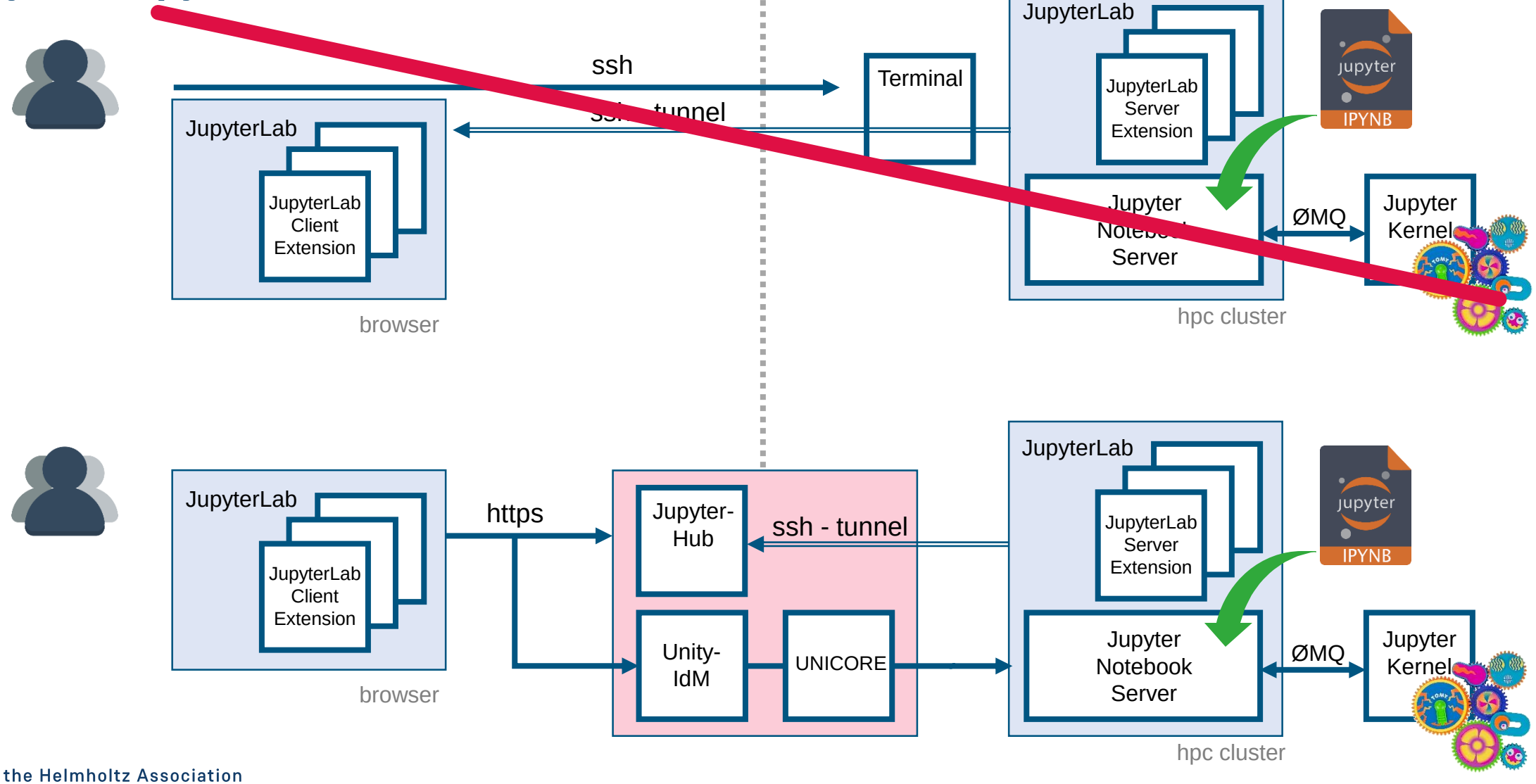
JUPYTER-JSC WEBSERVICE

Start your JupyterLab



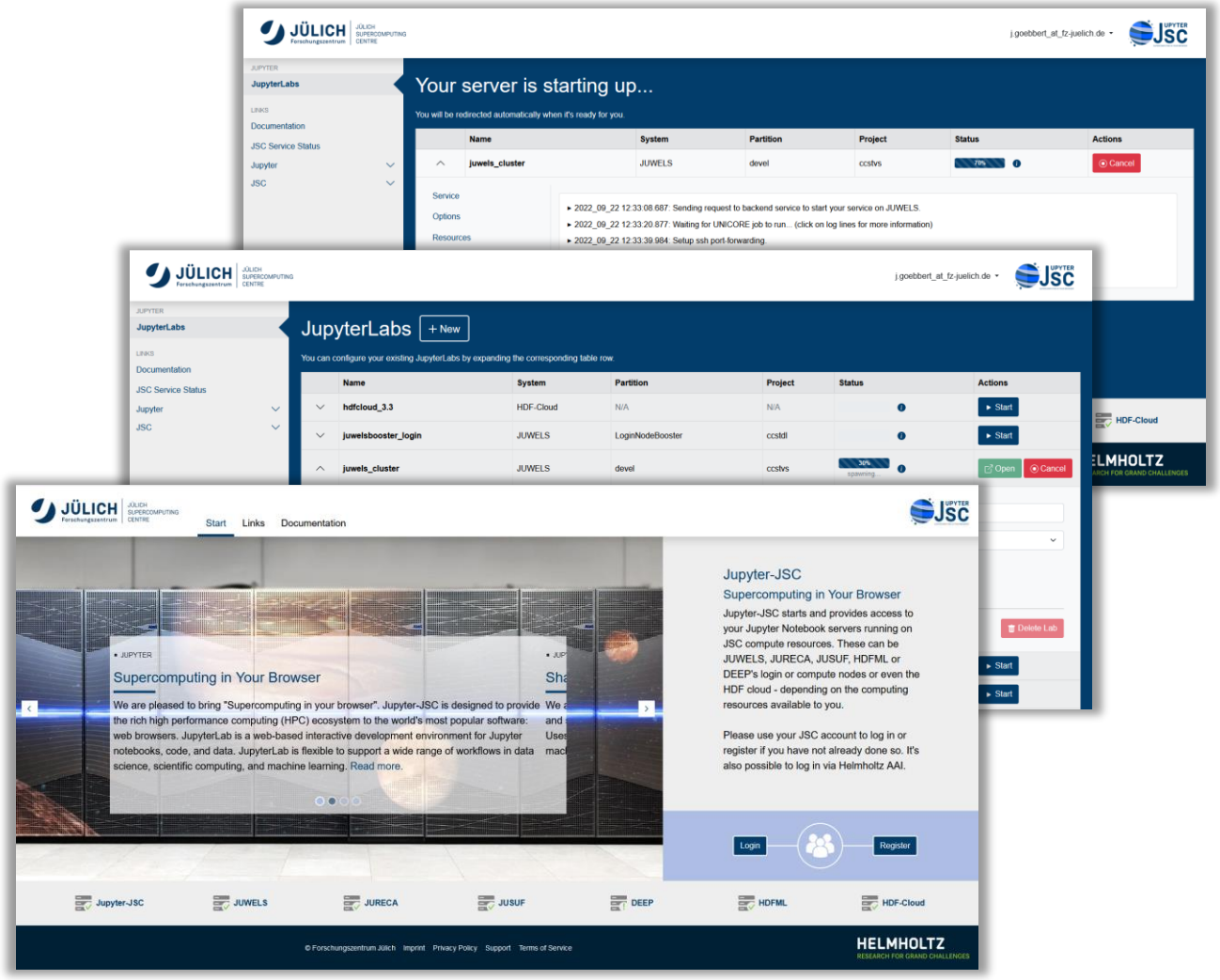
JUPYTER-JSC WEBSERVICE

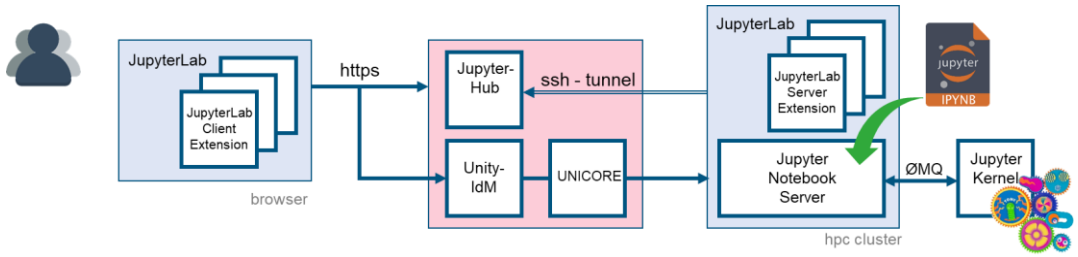
Start your JupyterLab

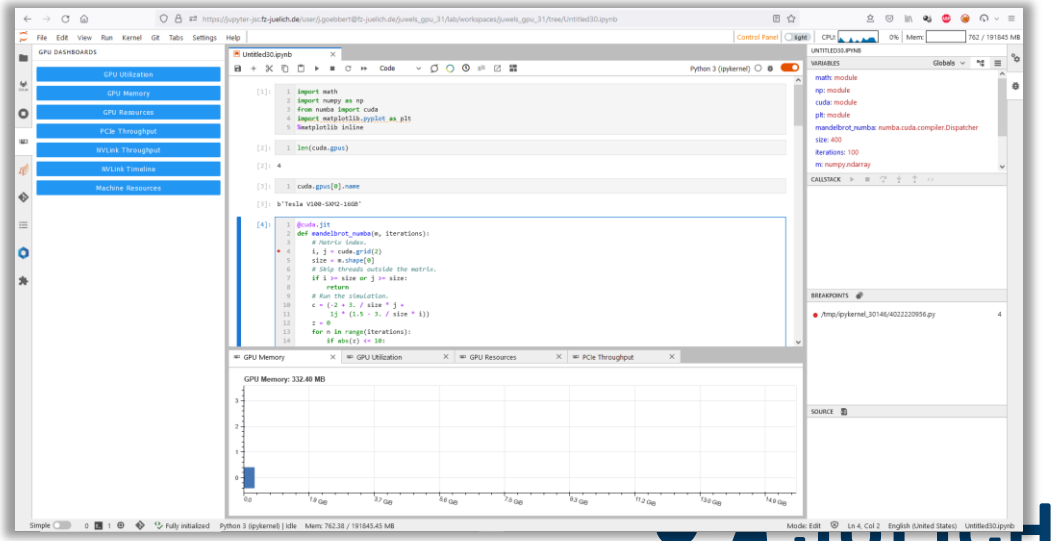


JUPYTER-JSC WEBSERVICE

Start your JupyterLab

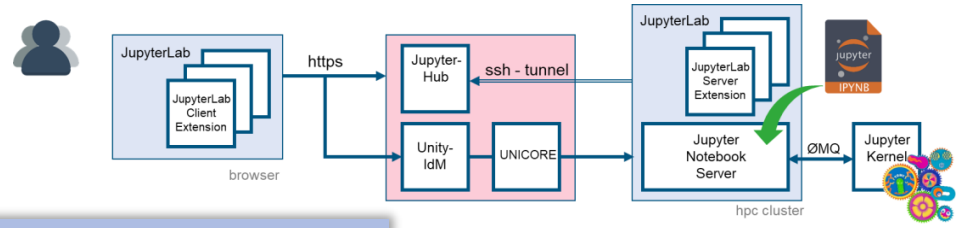






JUPYTER-JSC WEBSERVICE

First time login

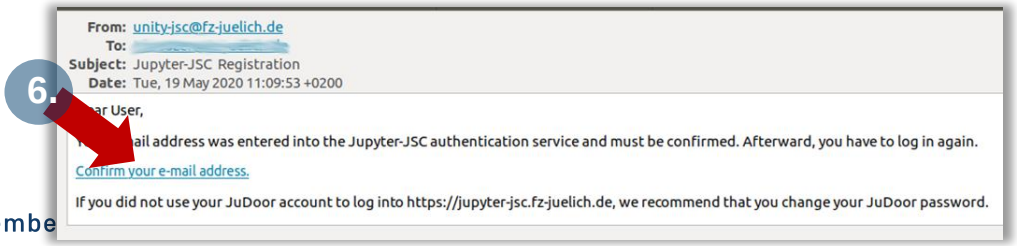


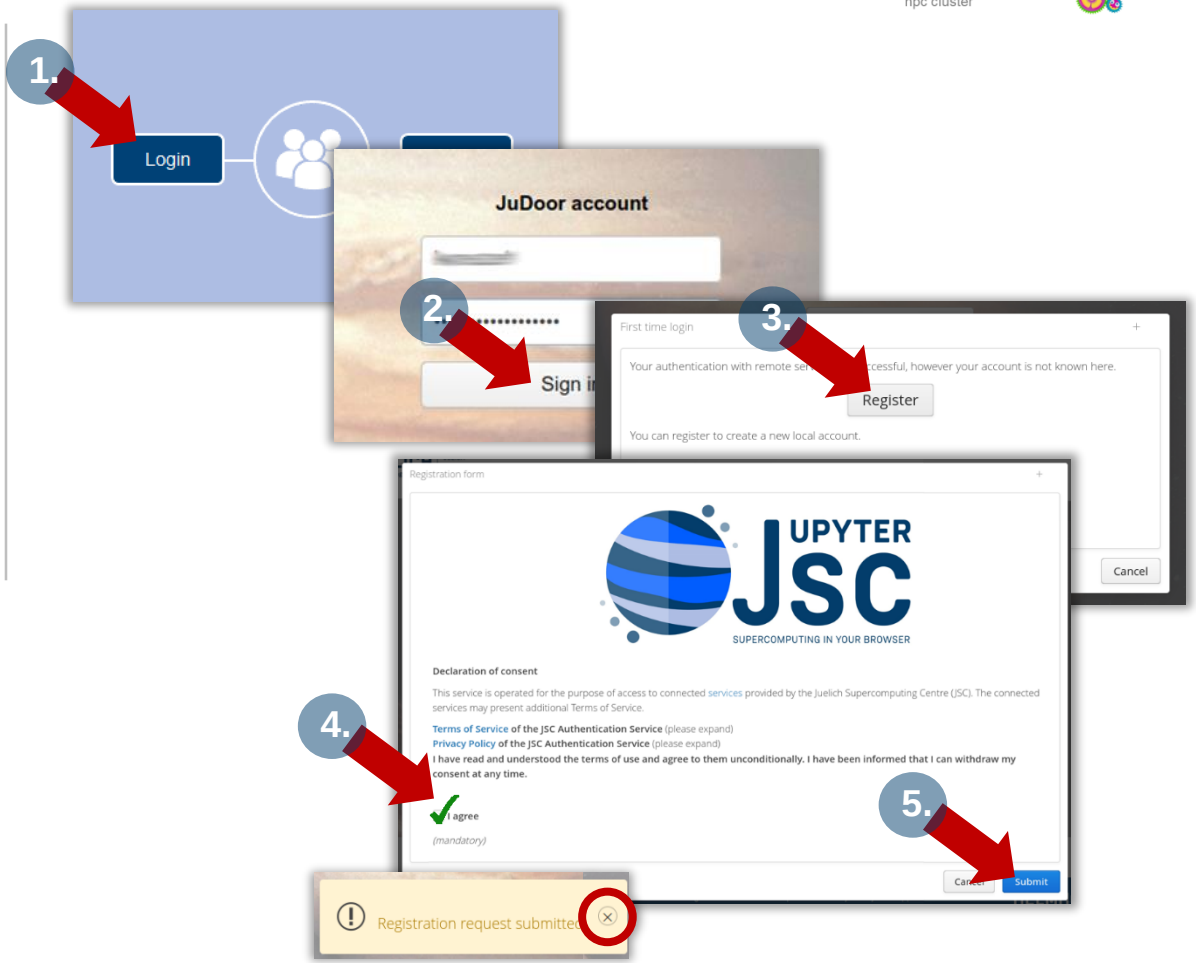
=> https://jupyter-jsc.fz-juelich.de

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

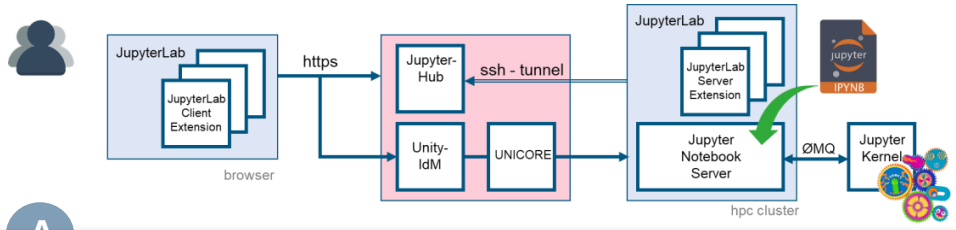




JÜLICH Forschungszentrum

JUPYTER-JSC WEBSERVICE

Control Panel



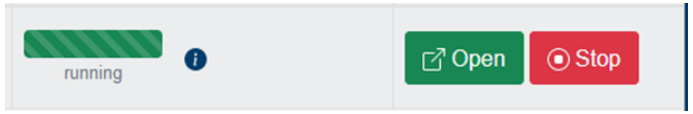
A. Jupyter-JSC – Add new JupyterLab

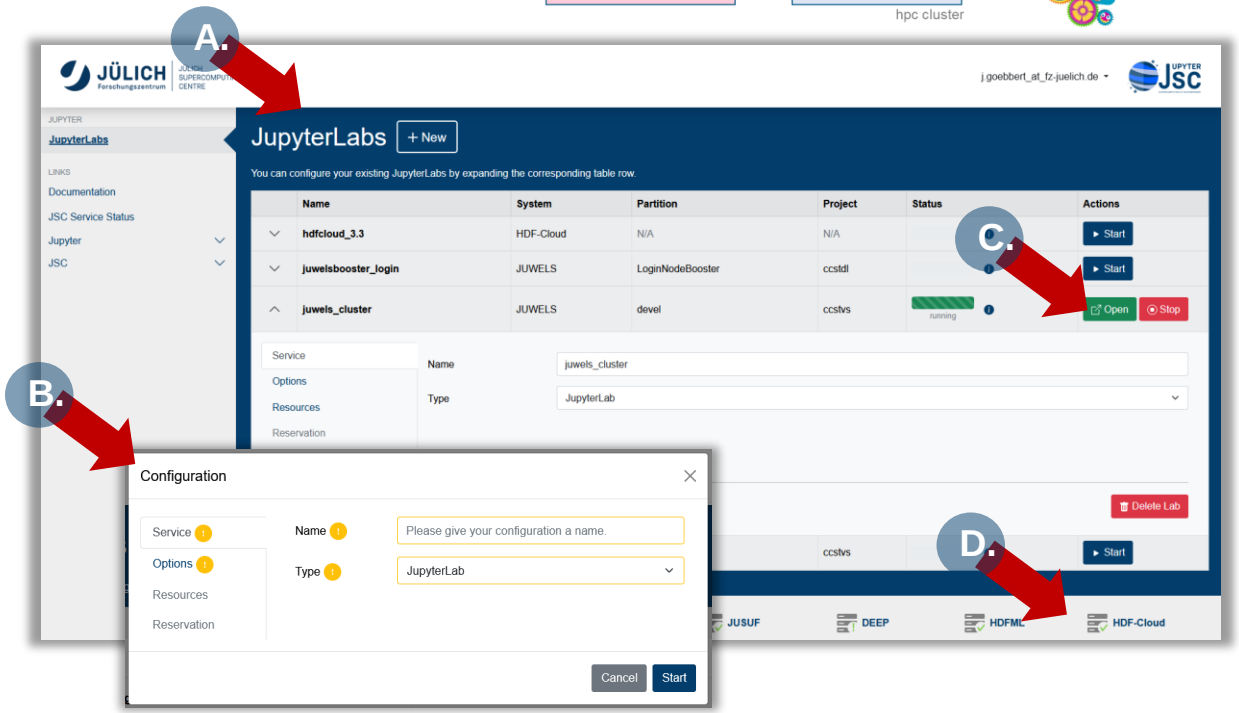
B. Configuration Dialog

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

C. Jupyter-JSC – Actions

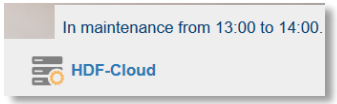
- Open/Stop a running JupyterLab
- Change/Delete **configuration**





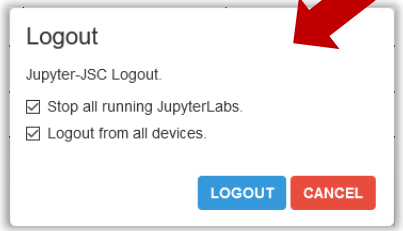
D. Jupyter-JSC -- Statusbar

- Upcoming maintenance (mouse hover for details)



- System offline

E. Jupyter-JSC – Logout

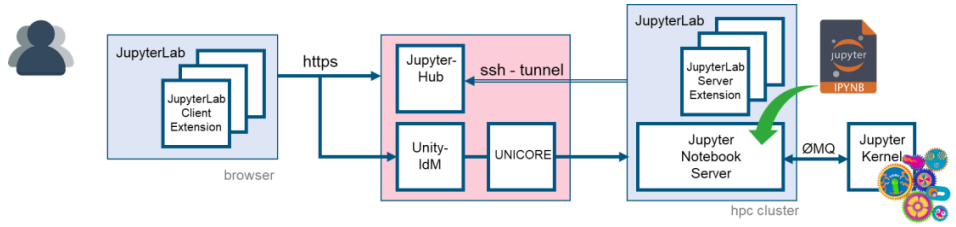


Logout will ask what you want to do with the running JupyterLabs – be careful what you answer!

JÜLICH Forschungszentrum

JUPYTER-JSC WEBSERVICE

JupyterLab Configuration



Jupyter-JSC – Configuration

Available options **depend on**

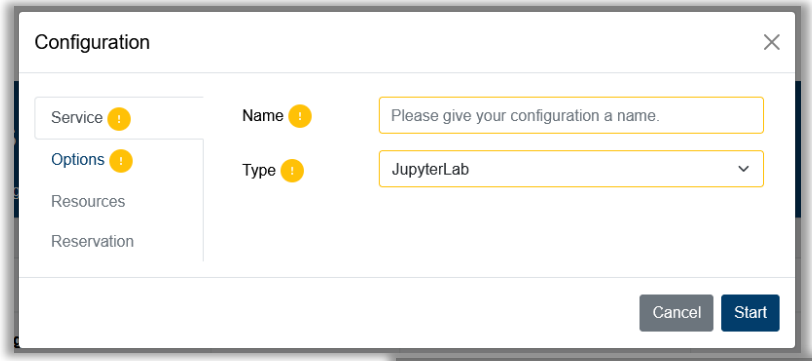
- user account settings visible in **judoor.fz-juelich.de**
- currently available systems in all of your projects
  - system specific usage agreement on JuDoor is signed

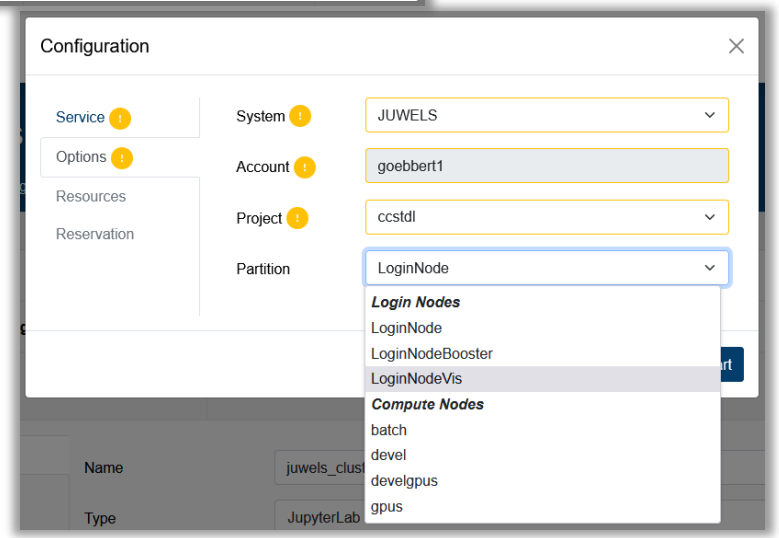
Basic options

- Type:
  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

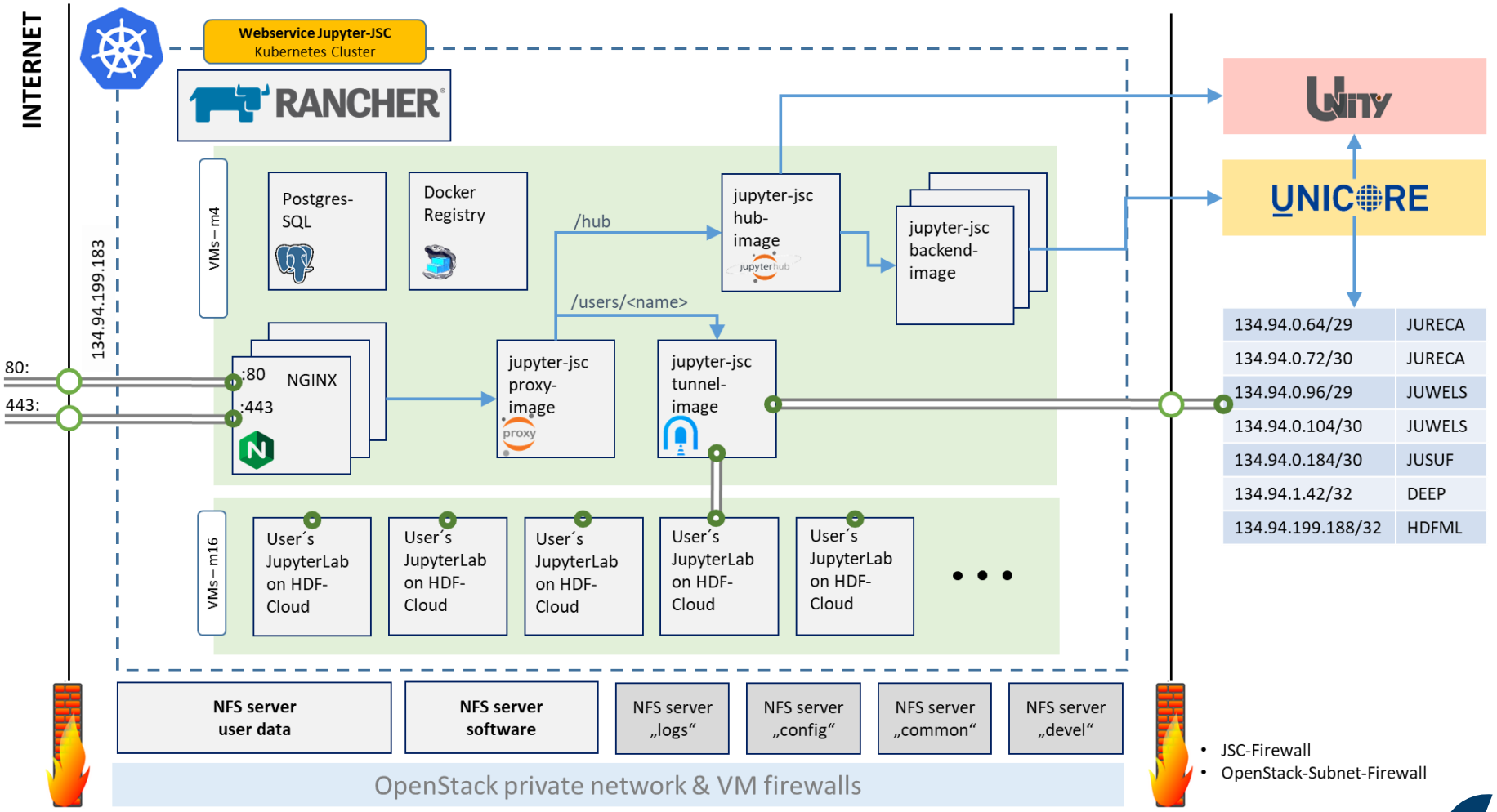
- Partition == compute Nodes, Runtime, GPUs, …





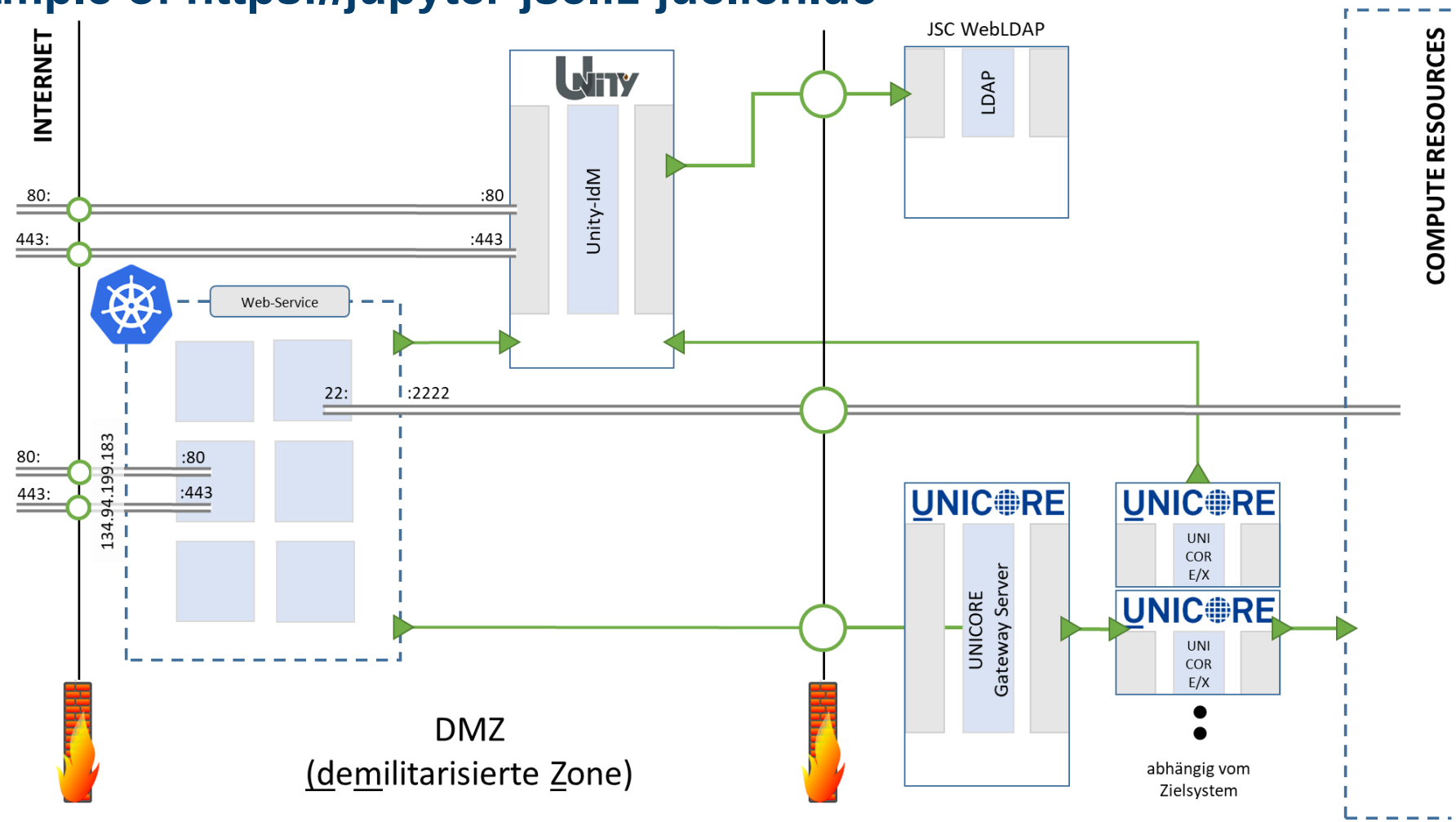
JUPYTER HUB WEBSERVICE

On the example of https://jupyter-jsc.fz-juelich.de



JUPYTER HUB WEBSERVICE

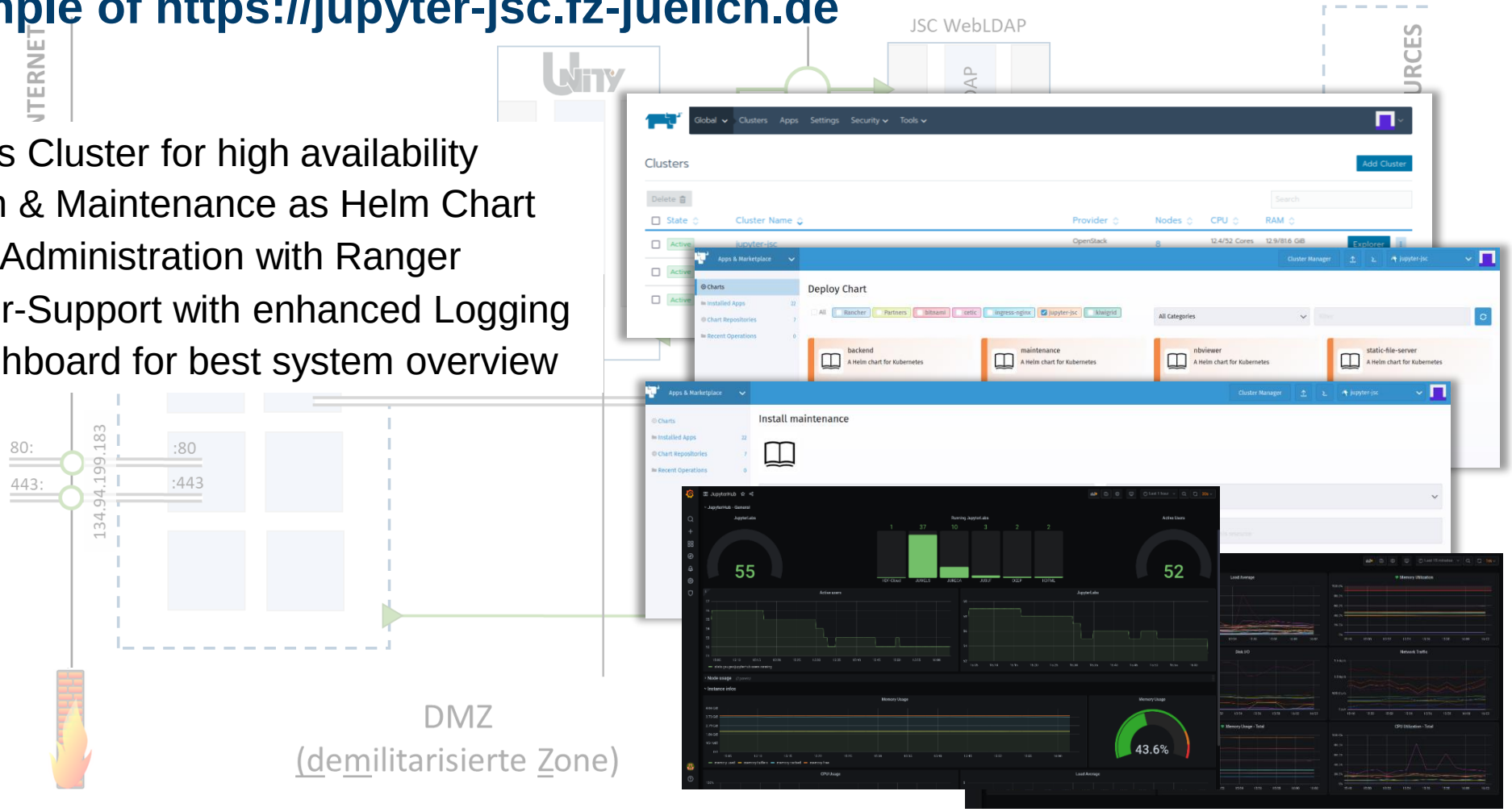
On the example of https://jupyter-jsc.fz-juelich.de



JUPYTER HUB WEBSERVICE

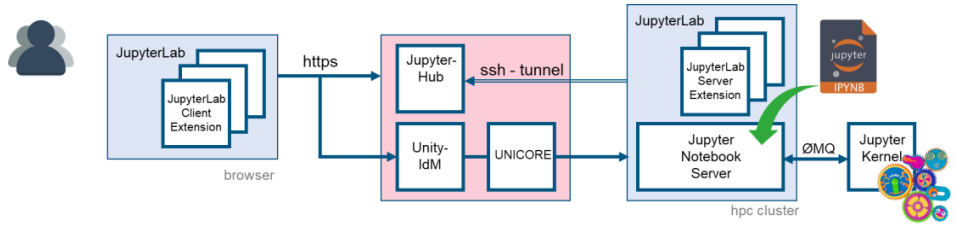
On the example of https://jupyter-jsc.fz-juelich.de

- Kubernetes Cluster for high availability
- Application & Maintenance as Helm Chart
- Advanced Administration with Ranger
- Better User-Support with enhanced Logging
- Admin dashboard for best system overview



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

tutorials & examples

memory consuption (keep an eye on that!)

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

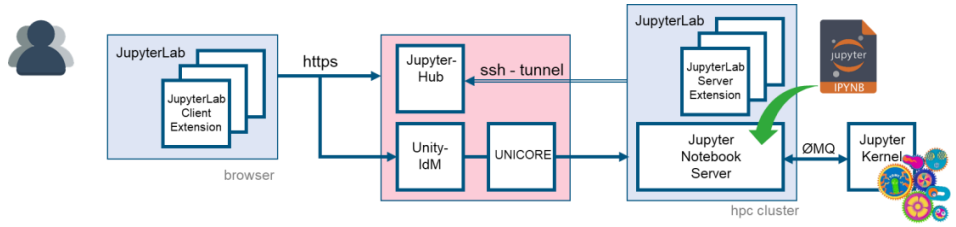
indicates active notebook cell

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

JUPYTERLAB EXTENSIONS

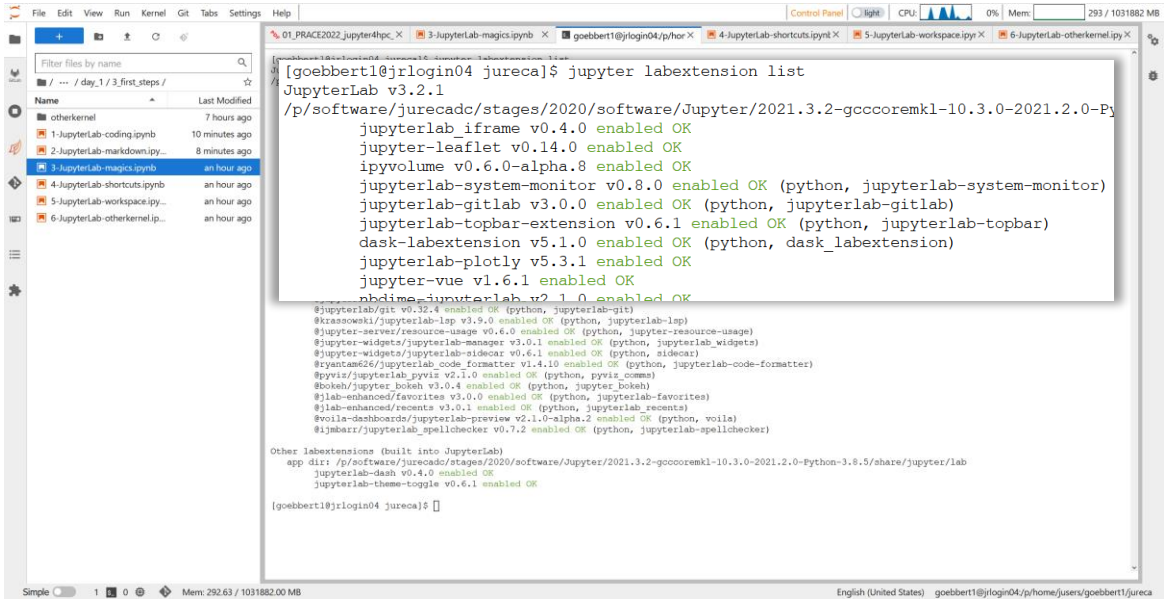
JUPYTER EXTENSIONS

Some general information



List the installed JupyterLab extensions

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



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

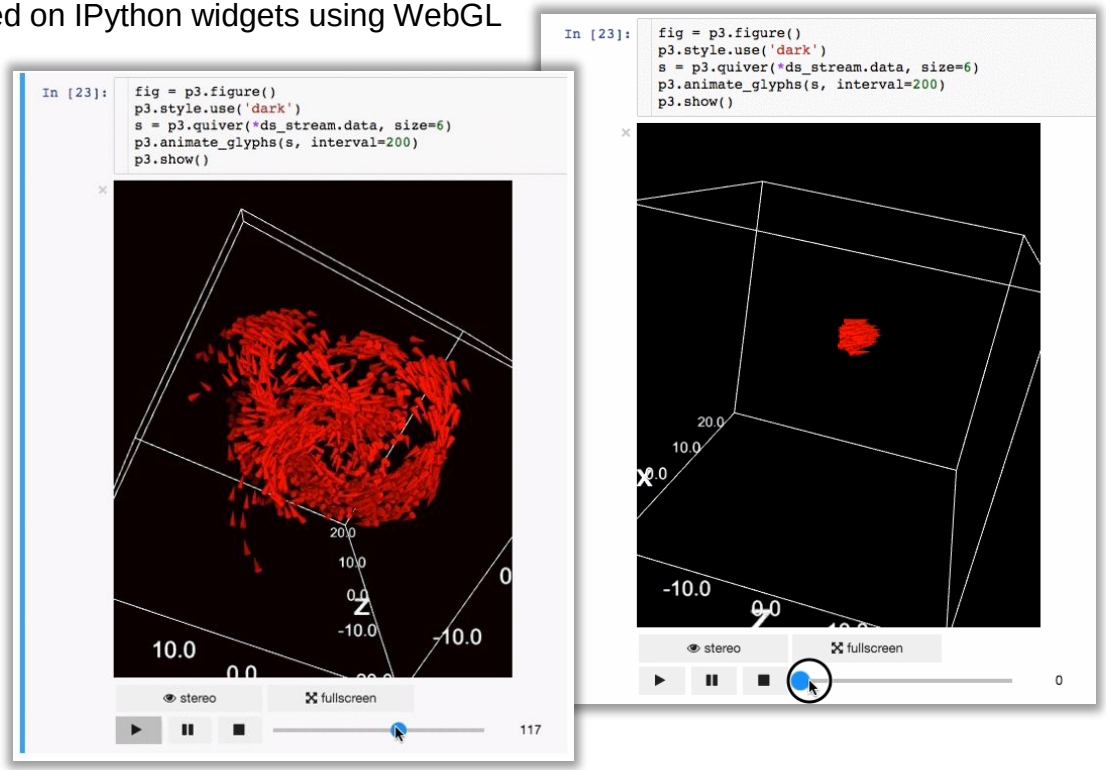
Extensions are installed in JupyterLab´s Application Directory, which

- 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

JUPYTER-JSC EXTENSIONS

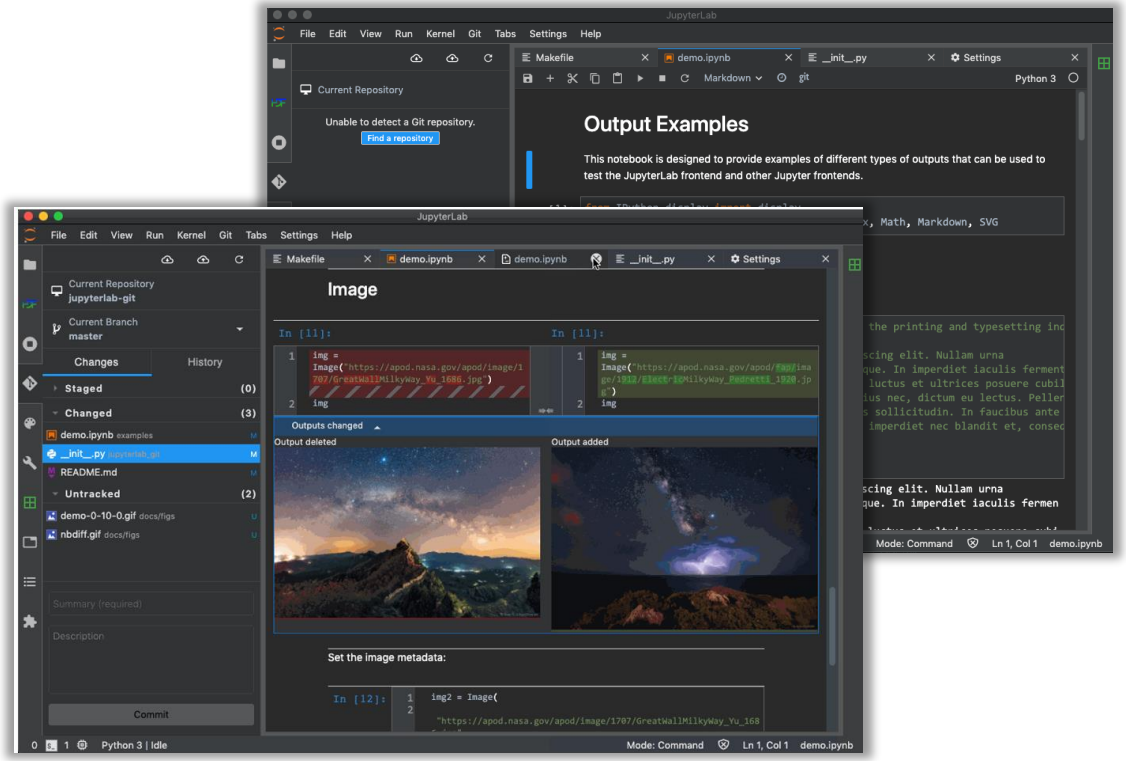
Installed by default

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



https://github.com/maartenbreddels/ipyvolume

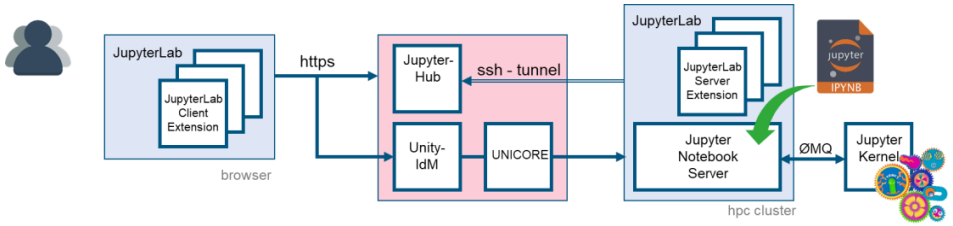
JupyterLab-Git
JupyterLab extension for version control using Git



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

JUPYTER-JSC EXTENSIONS

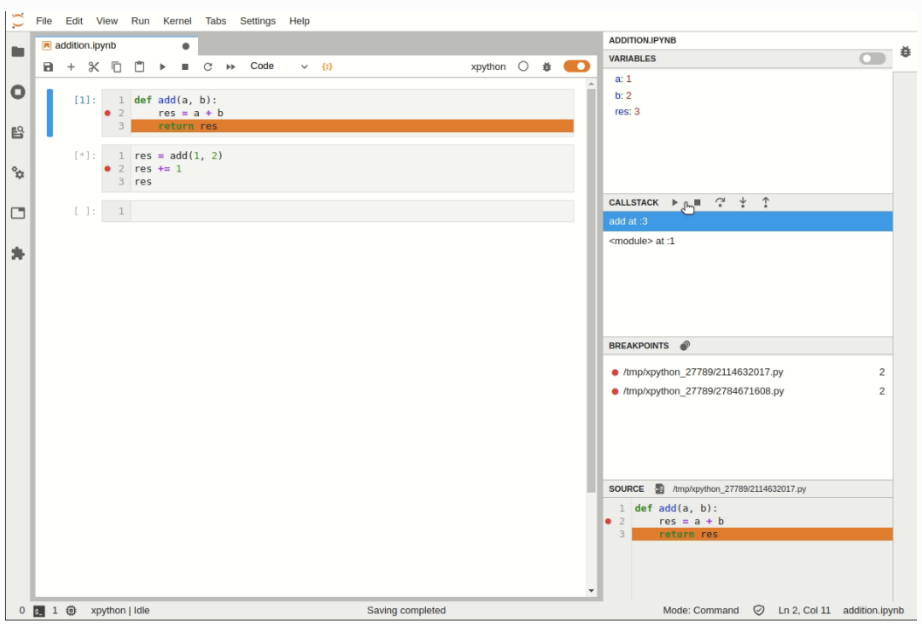
Installed by default



JupyterLab - Visual Debugger

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

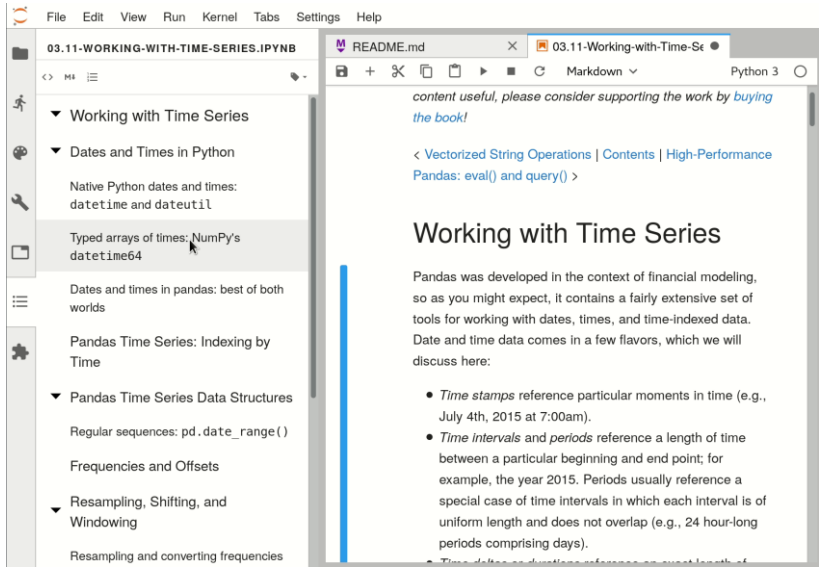
This means that notebooks, code consoles and files can now be debugged from JupyterLab directly! For the debugger to be enabled and visible, a kernel with support for debugging is required.



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.

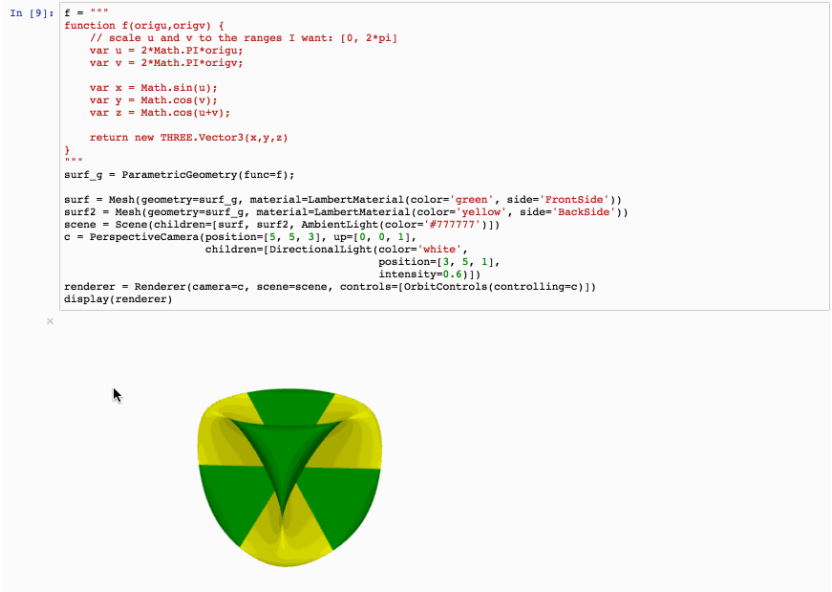


https://github.com/jupyterlab/jupyterlab-toc

JUPYTER-JSC EXTENSIONS

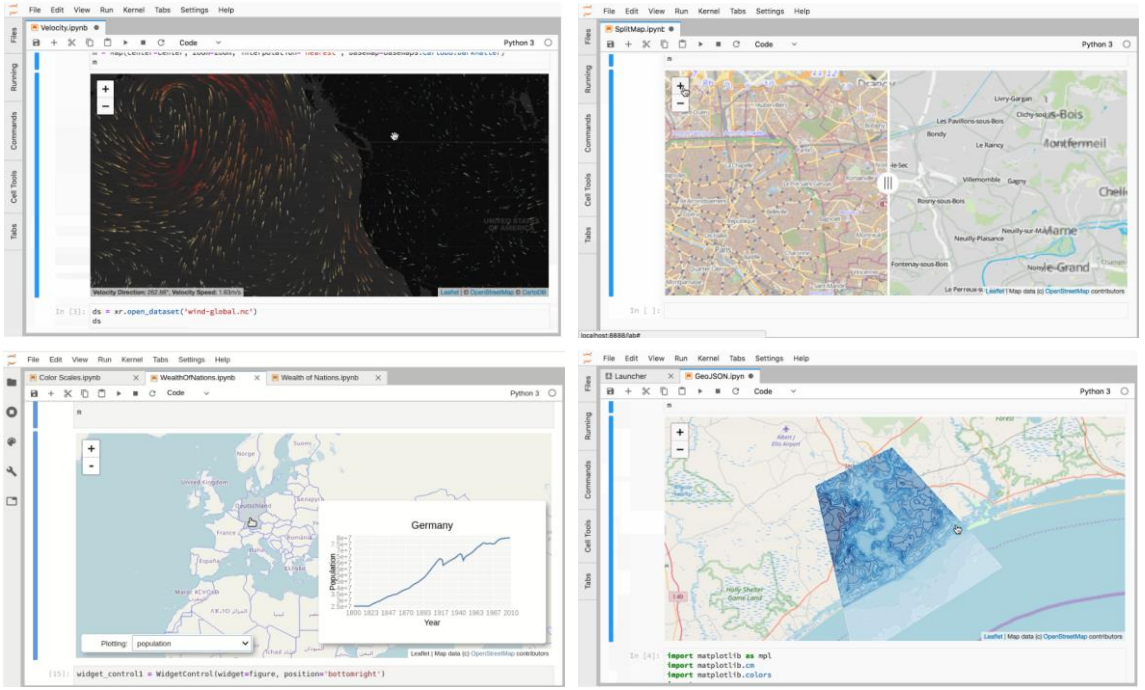
Installed by default

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.

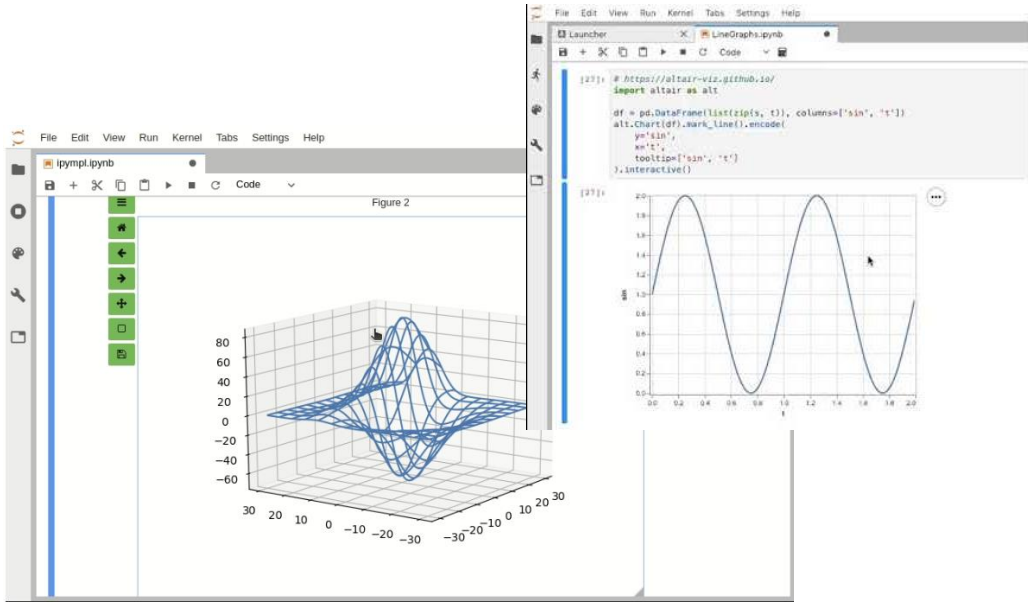


https://github.com/jupyter-widgets/ipyleaflet

JUPYTER-JSC EXTENSIONS

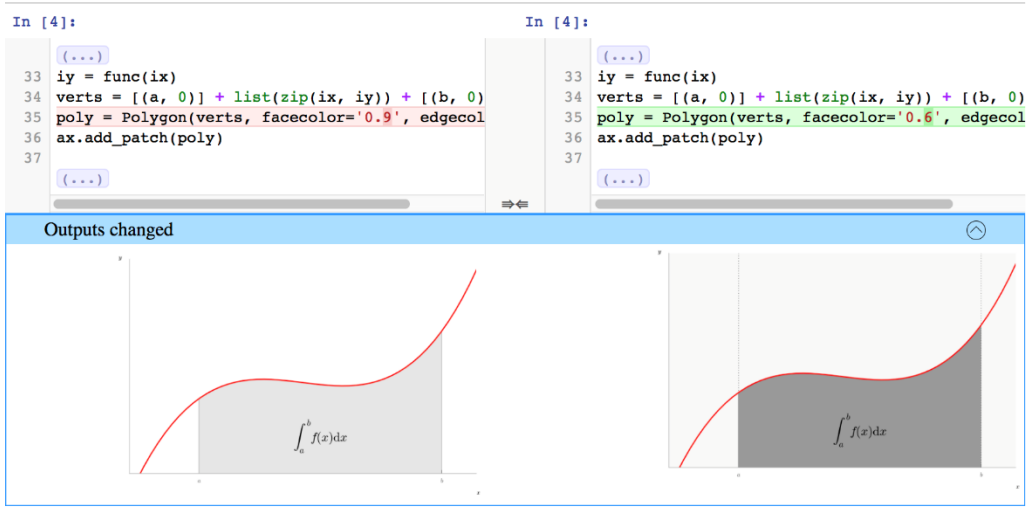
Installed by default

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

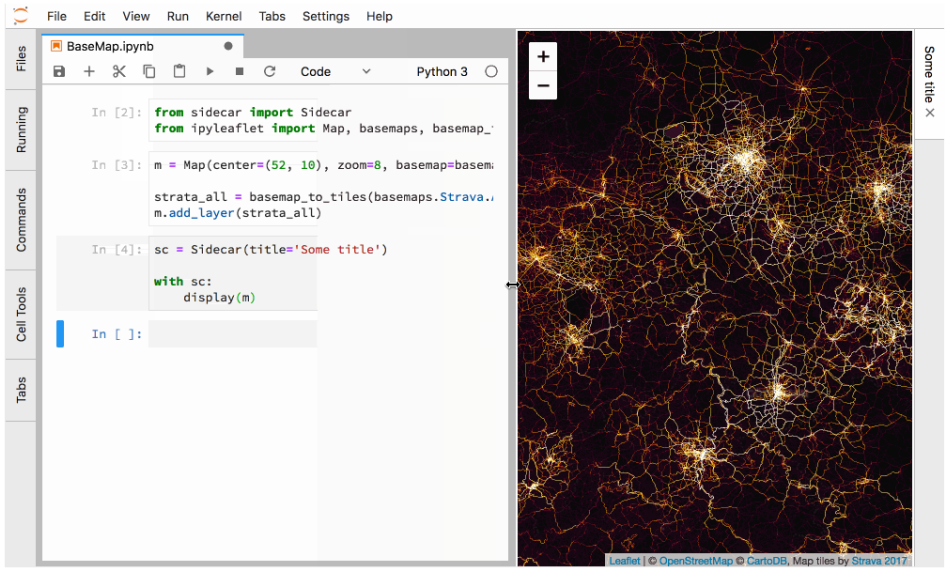
JUPYTER-JSC EXTENSIONS

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

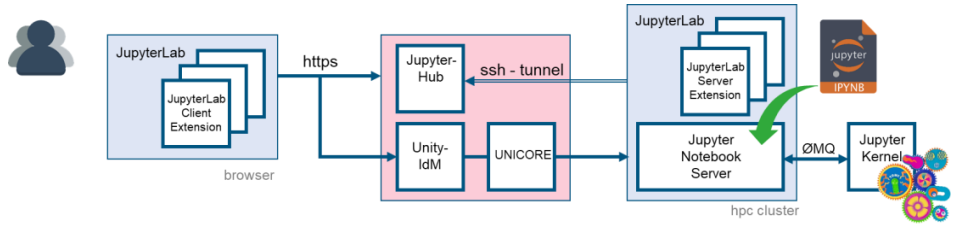
JupyterLab-Sidecar
A sidecar output widget for JupyterLab.



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

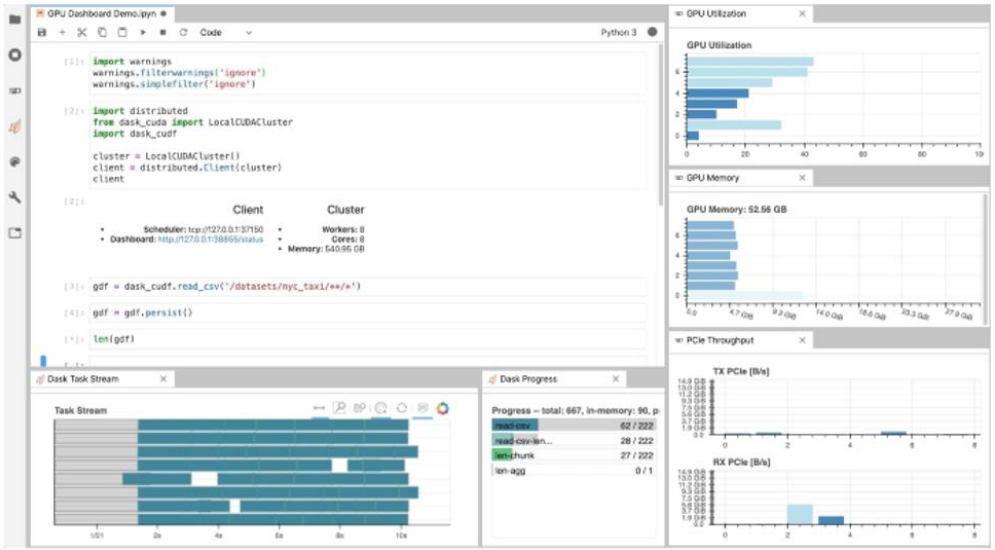
JUPYTER-JSC EXTENSIONS

Installed by default



NVDashboard

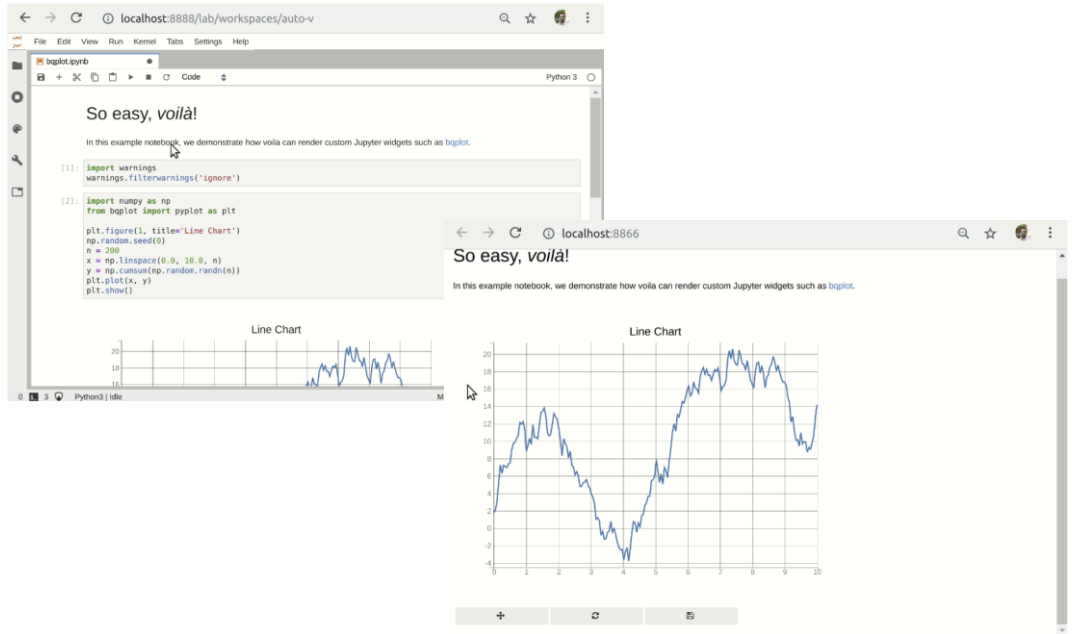
NVDashboard is an open-source package for the real-time visualization of NVIDIA GPU metrics in interactive Jupyter Lab environments.



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.

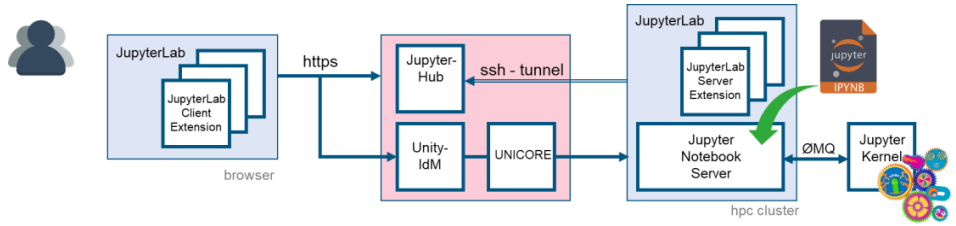


https://github.com/voila-dashboards/voila

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
@jimbarr/jupyterlab_spellchecker	v0.2.0	v0.7.2	prebuild
jupyterlab-nvdashboard		v0.6.0	prebuild



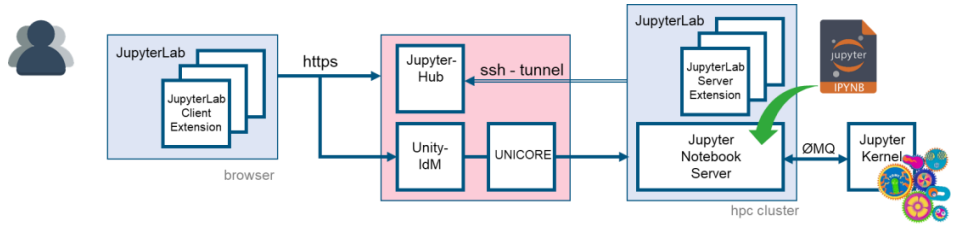
Data Visualization			
jupyter-matplotlib	v0.7.4	v0.9.0	prebuild
@bokeh/jupyter_bokeh	v2.0.4	v3.0.4	prebuild
jupyterlab-plotly	v4.14.3	v5.3.1	
bqplot	v0.5.22	v0.5.32	prebuild
@pyviz/jupyterlab_pyviz	v1.0.4	v2.1.0	prebuild
jupyter-leaflet	v0.13.3	v0.14.0	prebuild
ipyvolume	v0.6.0-alpha.5	v0.6.0-alpha.8	prebuild
jupyter-threejs	v2.2.0	v2.3.0	prebuild
@jupyter-widgets/jupyterlab-sidecar	v0.5.0	v0.6.1	prebuild
Framework Integrations			
dask-labextension	v3.0.0	v5.1.0	prebuild
@jupyterlab/latex	v2.0.1	v3.1.0	prebuild
jupyter-webrtc	v0.5.0	v0.6.0	prebuild
Dashboard 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

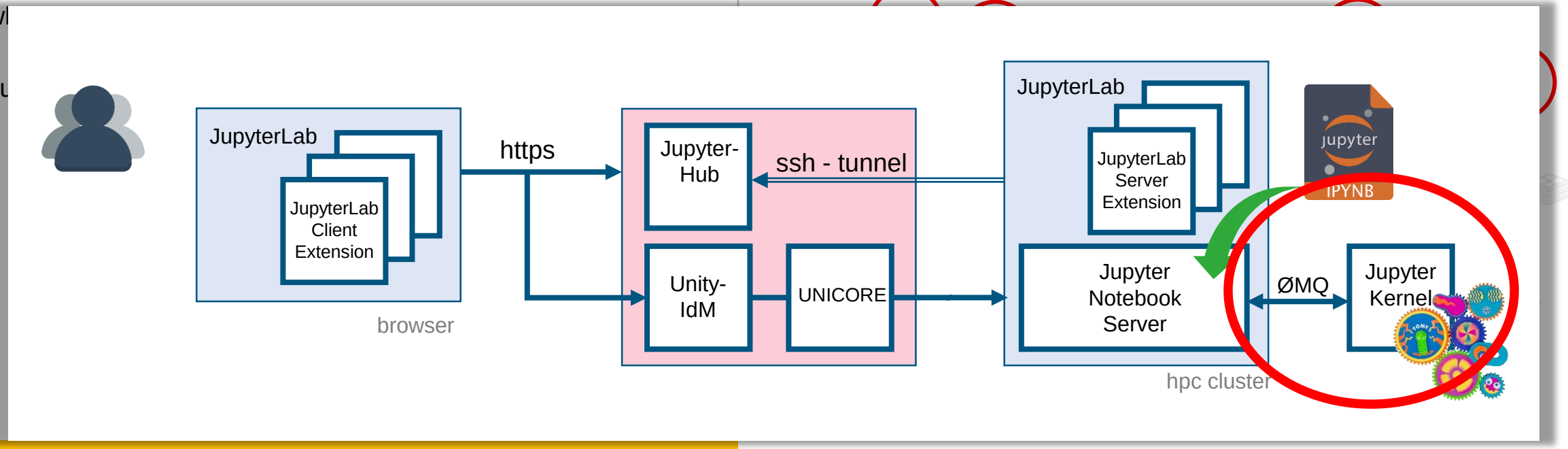


Jupyter Kernel

A “kernel” refers to the separate process wh

Ju

-
-
-
-

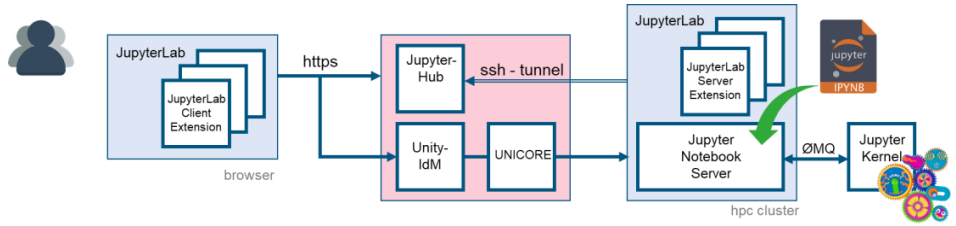


You can easily **create your own kernel** which for example runs your specialized virtual Python environment.

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

JUPYTER KERNEL

How to create your own Juypter Kernel



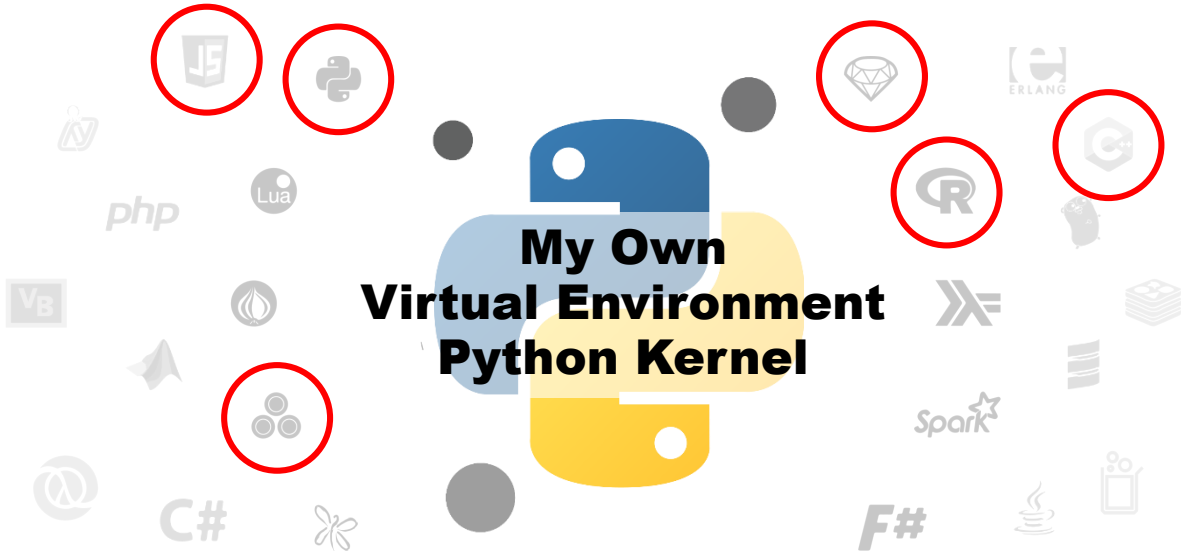
Jupyter Kernel

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

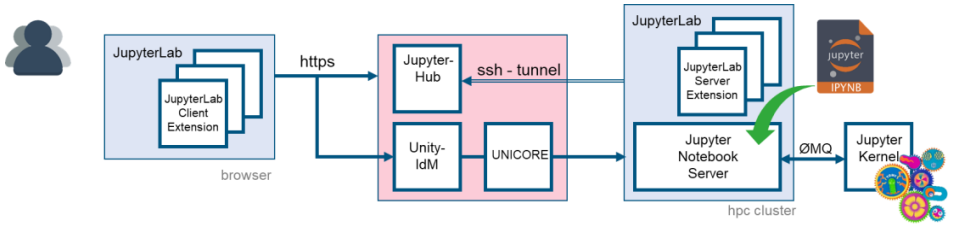


You can easily **create your own kernel** which for example runs your specialized virtual Python environment.

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.

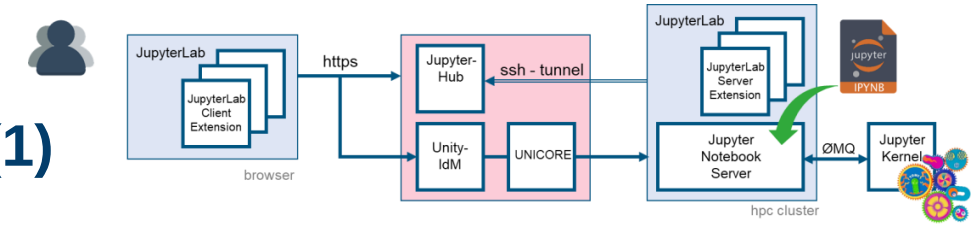
Building your own Jupyter kernel is a three step process

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

1. Create/Pimp new virtual Python environment (1)



1. **Login to JupyterLab and open terminal**

2. **Load required modules**

```
Lnode:> module purge
Lnode:> module use $OTHERSTAGES
Lnode:> module load Stages/2020
Lnode:> module load GCCcore/.10.3.0
Lnode:> module load Python/3.8.5
```

3. **Load extra modules you need for your kernel**

```
Lnode:> module load <module you need>
```

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`

1. **Create a virtual environment named <venv_name> at a path of your choice:**

```
Lnode:> python -m venv --system-site-packages <your_path>/<venv_name>
```

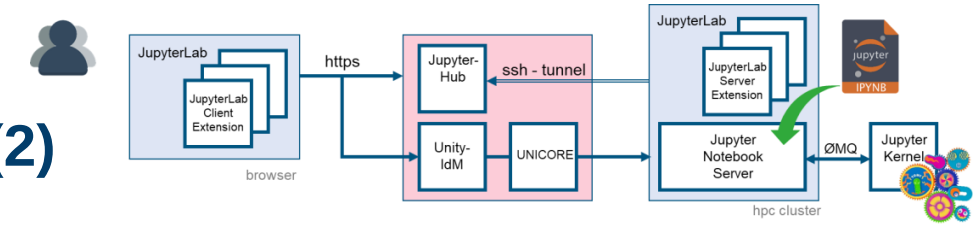
2. **Activate your environment**

```
Lnode:> source <your_path>/<venv_name>/bin/activate
```

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

JUPYTER KERNEL

1. Create/Pimp new virtual Python environment (2)



1. **Ensure python packages installed in the virtual environment are always prefered**

```
(<venv_name>) Lnode:> export PYTHONPATH=\
${VIRTUAL_ENV}/lib/python3.8/site-packages:${PYTHONPATH}
```

2. **Install Python libraries required for communication with Jupyter**

```
(<venv_name>) Lnode:>
        pip install --ignore-installed ipykernel
```

3. **Install whatever else you need in your Python virtual environment (using pip)**

```
(<venv_name>) Lnode:>
        pip install <python-package you need>
```

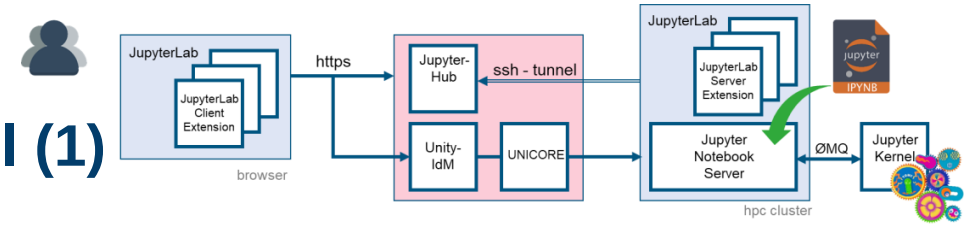
Building your own Jupyter kernel is a three step process

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

JUPYTER KERNEL

2. Create/Edit launch script for the Jupyter kernel (1)



1. **Create launch script, which loads your Python virtual environment and starts the ipykernel process inside:**

```
(<venv_name>) Lnode:> touch ${VIRTUAL_ENV}/kernel.sh
```

2. **Make launch script executable**

```
(<venv_name>) Lnode:> chmod +x ${VIRTUAL_ENV}/kernel.sh
```

3. **Edit the launch script for your new Jupyter kernel**

```
(<venv_name>) Lnode:> vi ${VIRTUAL_ENV}/kernel.sh
```

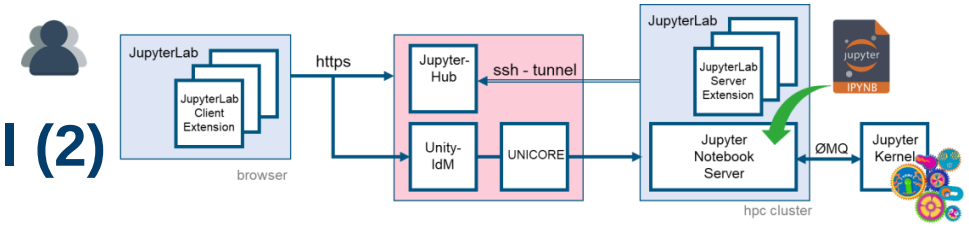
Building your own Jupyter kernel is a three step process

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

JUPYTER KERNEL

2. Create/Edit launch script for the Jupyter kernel (2)




```
#!/bin/bash

# Load required modules
module purge
module load $OTHERSTAGES
module load Stages/2020
module load GCCcore/.10.3.0
module load Python/3.8.5

# Load extra modules you need for your kernel
#module load <module you need>

# Activate your Python virtual environment
source <your_path>/<venv_name>/bin/activate

# Ensure python packages installed in the virtual environment are always prefered
export PYTHONPATH=${VIRTUAL_ENV}/lib/python3.8/site-packages:${PYTHONPATH}

exec python -m ipykernel $@
```

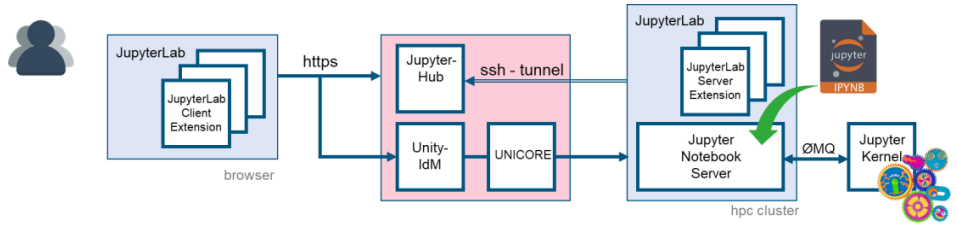
Building your own Jupyter kernel is a three step process

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

JUPYTER KERNEL

3. Create/Edit Jupyter kernel configuration (1)



1. **Create your Jupyter kernel configuration files**

```
(<venv_name>) Lnode:>
python -m ipykernel install --user --name=<my-kernel-name>
```

2. **Update your kernel file to use the lauch script**

```
(<venv_name>) Lnode:>
vi ~/.local/share/jupyter/kernels/<my-kernel-name>/kernel.json
{
 "argv": [
  "<your_path>/<venv_name>/kernel.sh",
  "-m",
  "ipykernel_launcher",
  "-f",
  "{connection_file}"
 ],
 "display_name": "<my-kernel-name>",
 "language": "python"
}
```

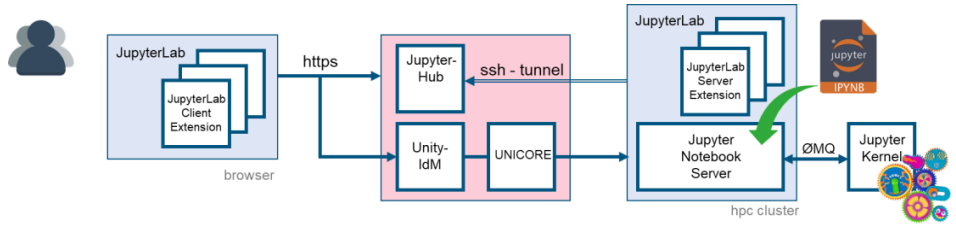
Building your own Jupyter kernel is a three step process

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

JUPYTER KERNEL

Run your Jupyter kernel configuration

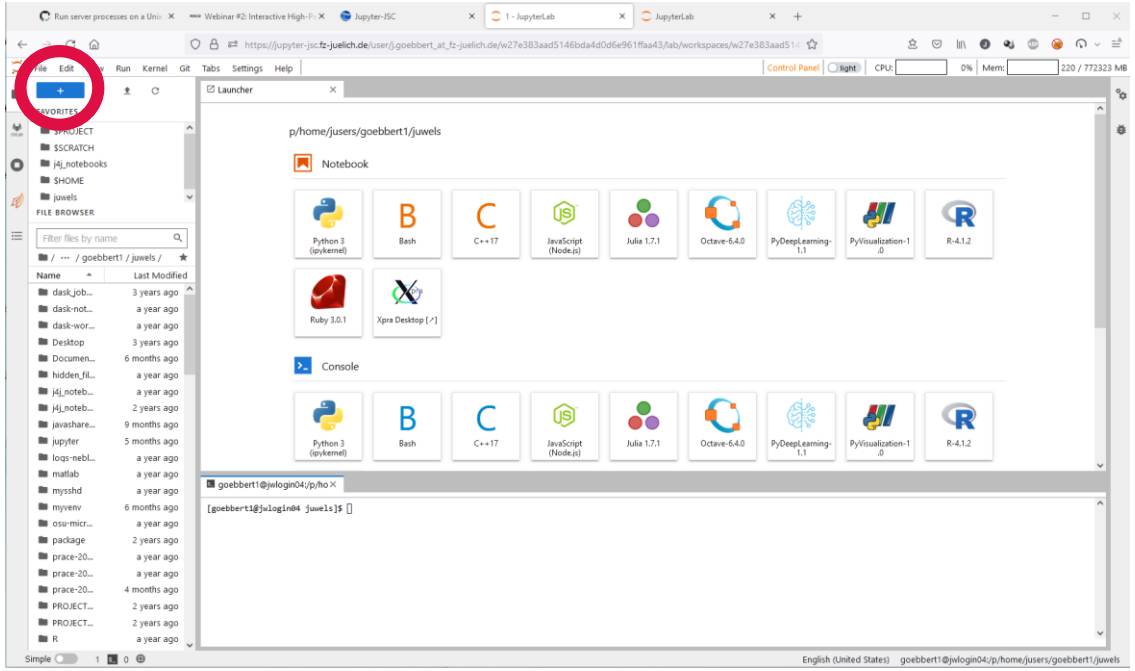


Run your Jupyter Kernel

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

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

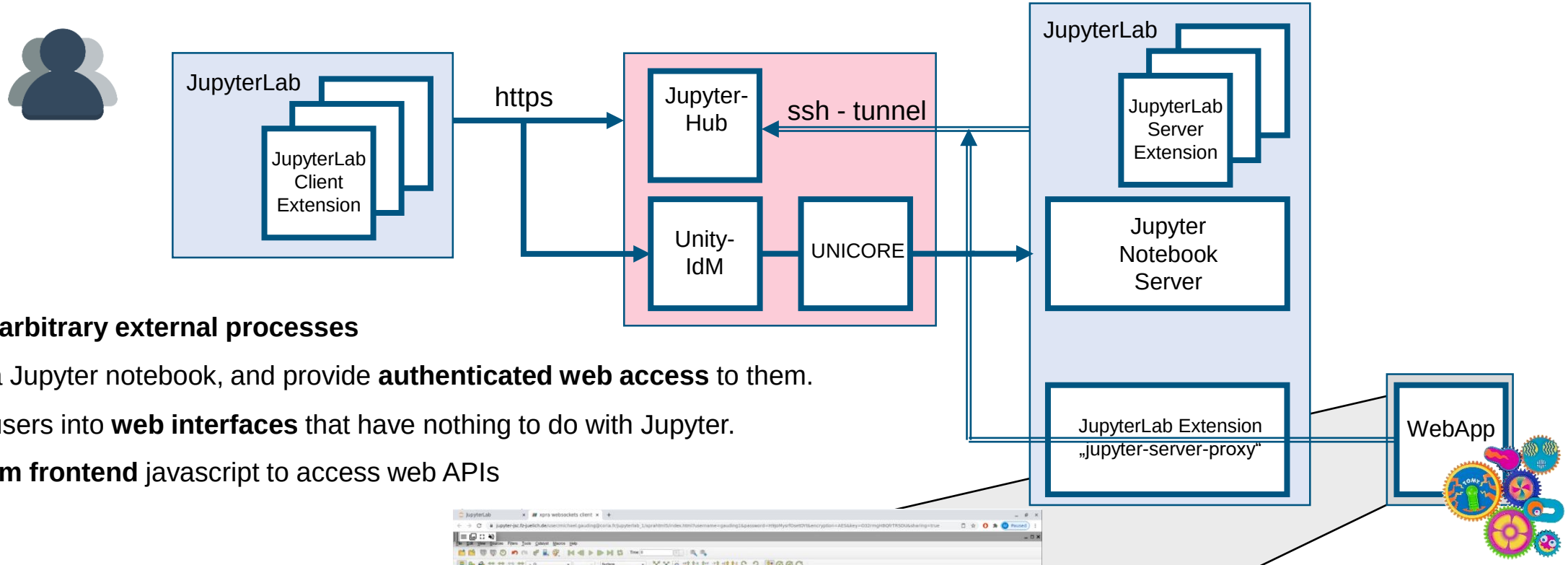


https://gitlab.version.fz-juelich.de/jupyter4jsc/j4j_notebooks/-/blob/master/001-Jupyter/Create_JupyterKernel_general.ipynb

JUPYTER CAN DO MORE

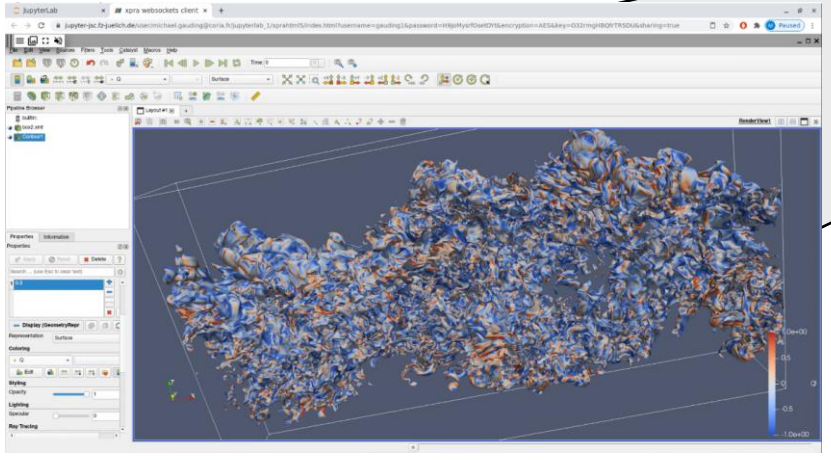
JUPYTERLAB – WEBSERVICE PROXY

Extension: jupyter-server-proxy



Allows to run **arbitrary external processes**

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

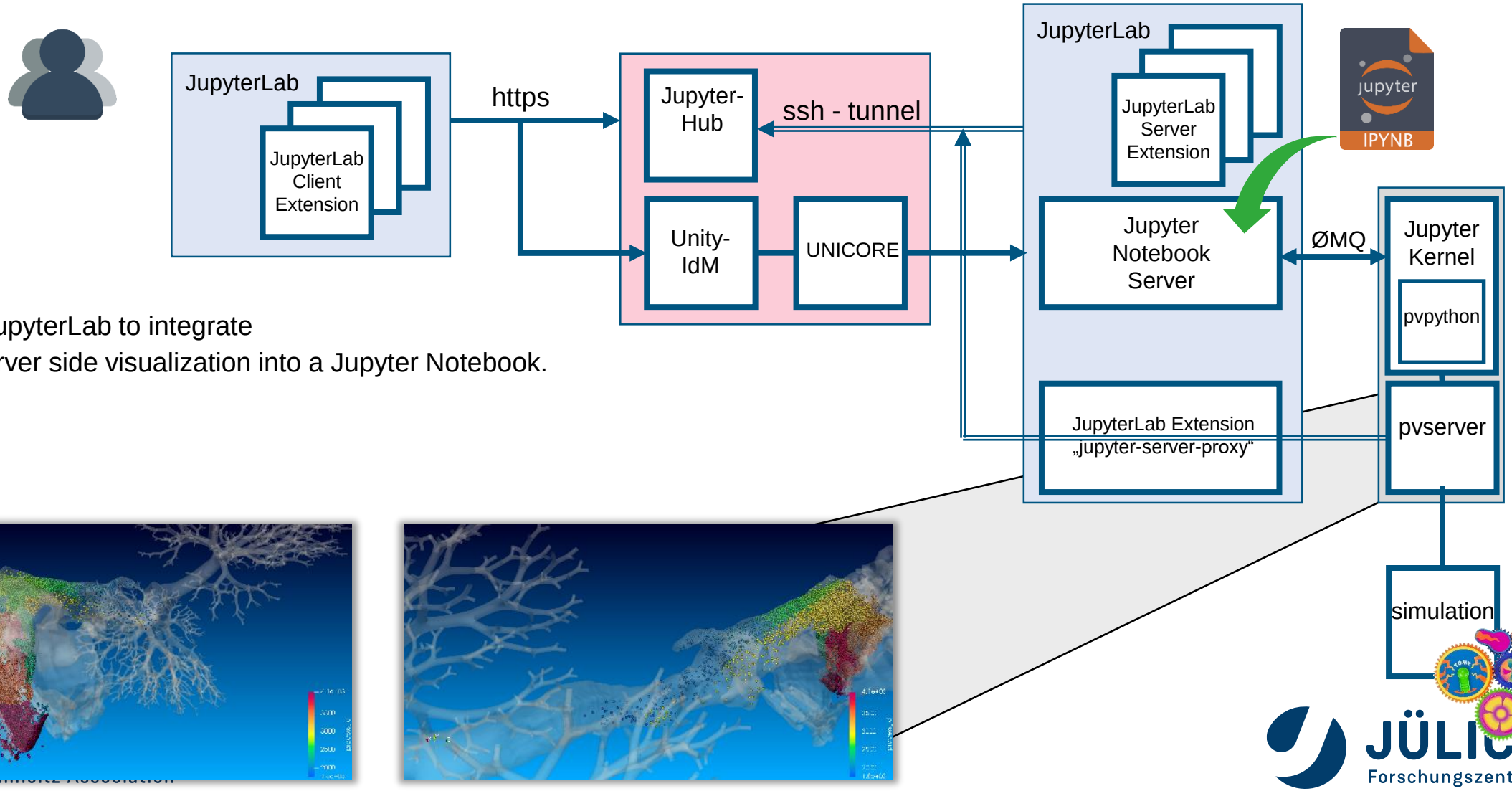


Turbulent mixing with variable density,
subset of 1939x600x3584 grid points, Michael Gauding, CORIA

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

JUPYTERLAB – WEBSERVICE PROXY

Extension: jupyter-server-proxy



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

JÜLICH Forschungszentrum

JUPYTERLAB – WEBSERVICE PROXY

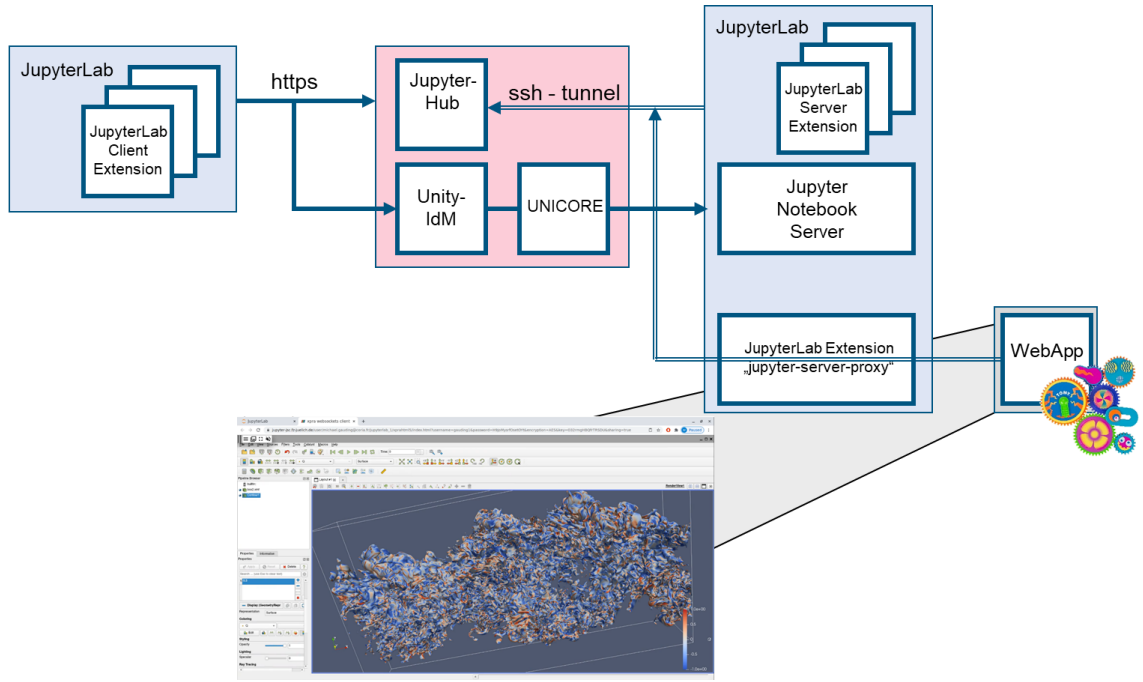
Extension: jupyter-server-proxy

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

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.

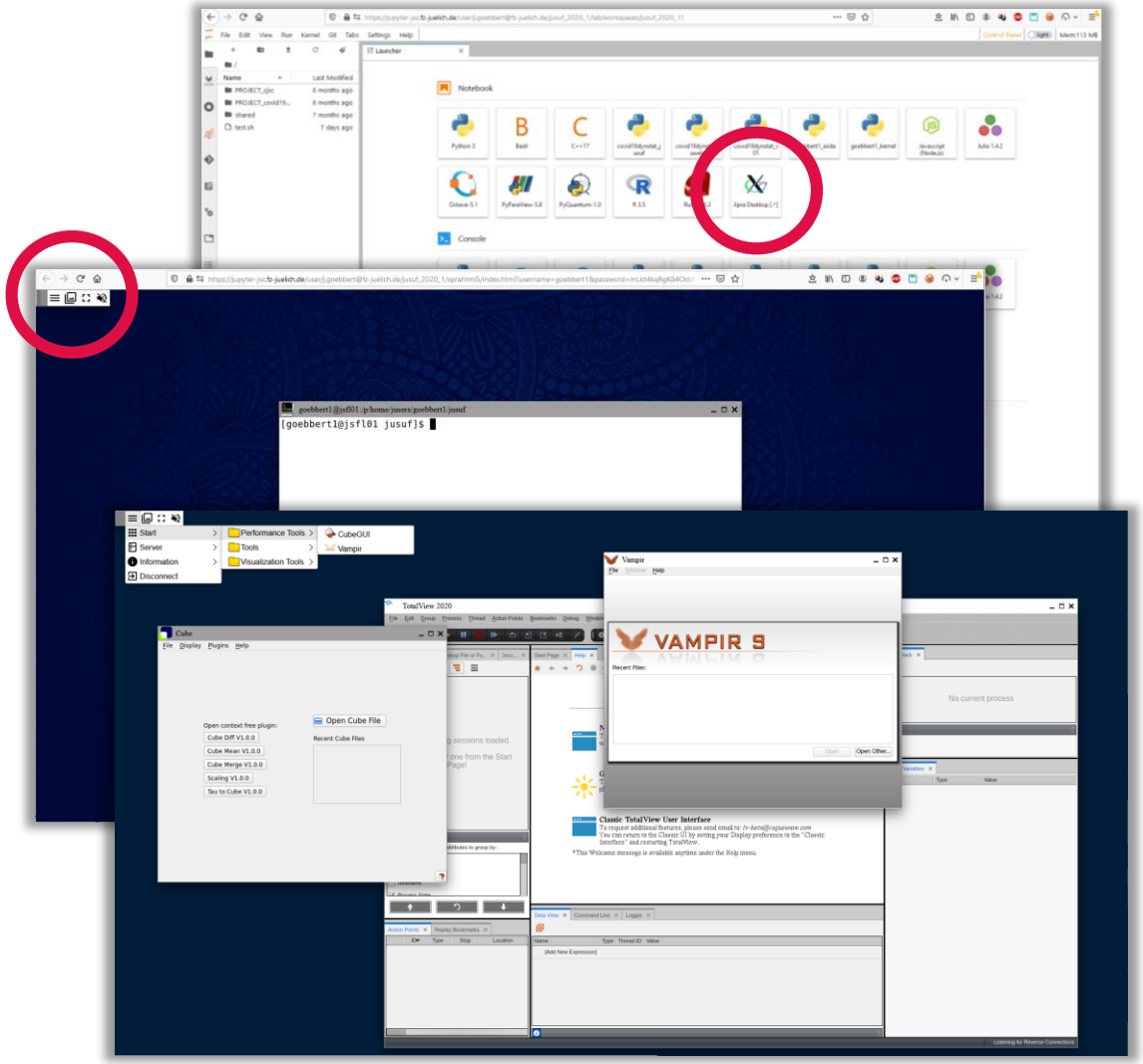
- Runs in a browser
- allows dis-/reconnection without disrupting the forwarded application
- https://xpra.org

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:

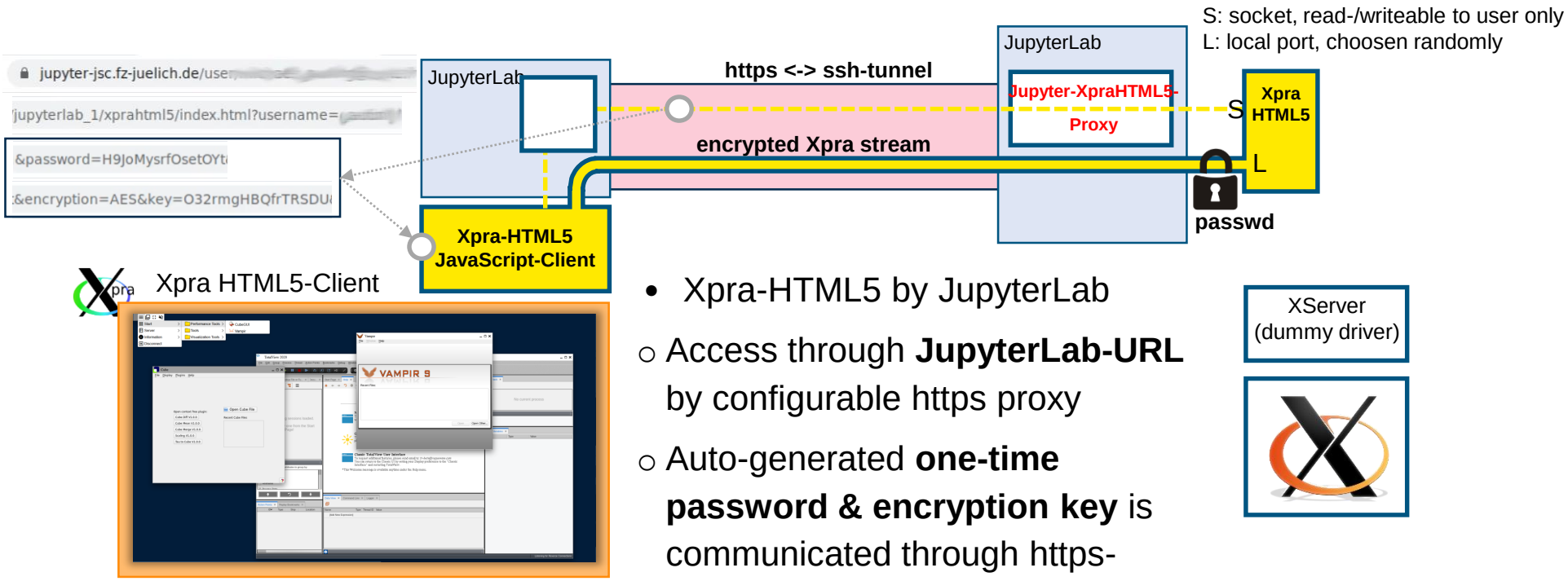
- `CTRL + C -> CTRL + Insert`
- `CTRL + V -> SHIFT + Insert`



JÜLICH Forschungszentrum

JUPYTERLAB – REMOTE DESKTOP

Run your X11-Applications in the browser



- 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

BENEFITS

Why Jupyter is so popular among Data Scientists

Some of the reasons …

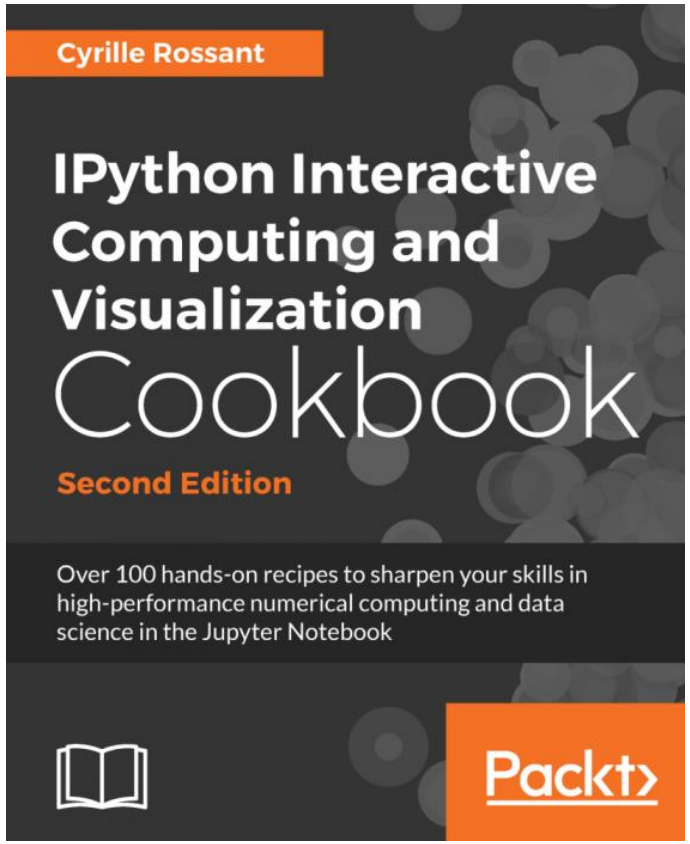
- Jupyter allows to **view the results of the code in-line** without the dependency of other parts of the code.
- Jupyter mixes easy for users who extend their code **line-by-line with feedback** attached all along the way
- Jupyter Notebooks support visualization and include rendering data in **live-graphics 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.

TUTORIALS

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

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

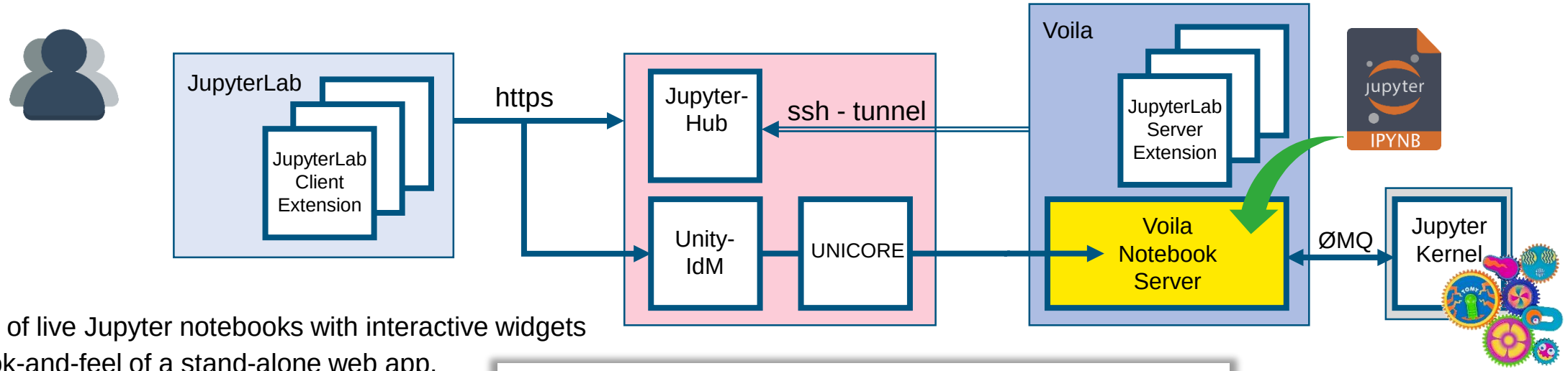


https://ipython-books.github.io
https://github.com/ipython-books/cookbook-2nd

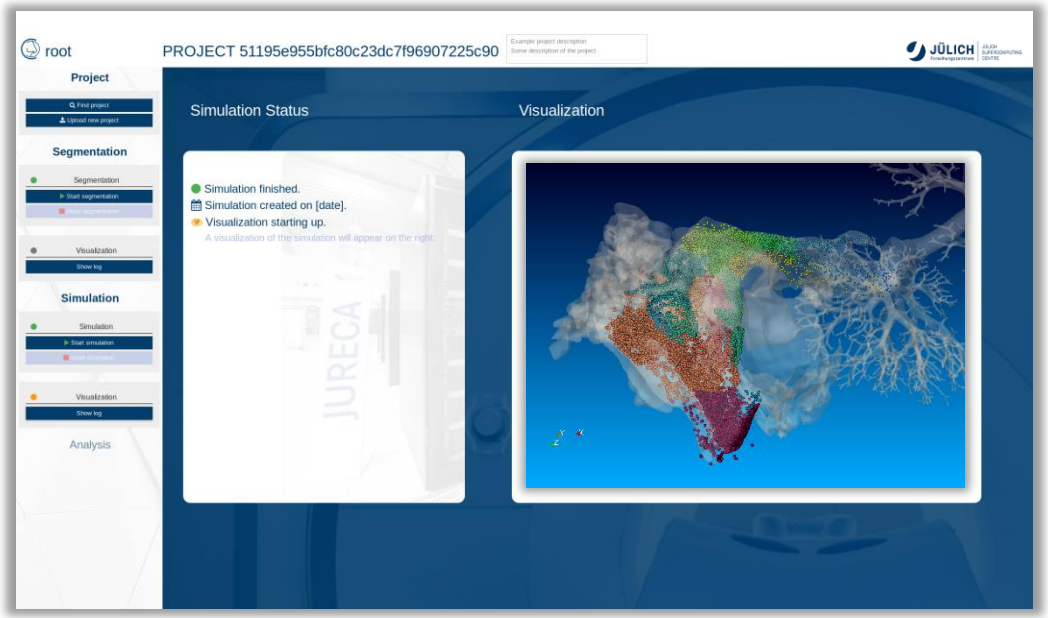
QUESTIONS?

DASHBOARDS WITH JUPYTER/VOILA

Voilà turns Jupyter notebooks into standalone web applications



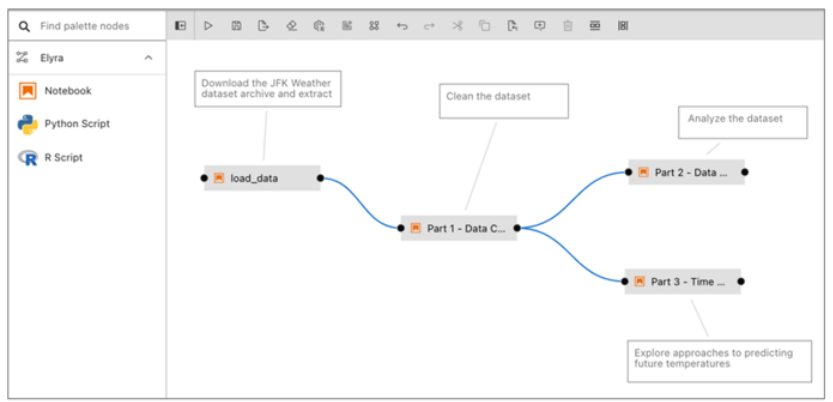
- **Rendering** of live Jupyter notebooks with interactive widgets with the look-and-feel of a stand-alone web app.
- Voilà disallows execute requests from the front-end, **preventing** execution of arbitrary code.
- **Enables** HPC users to develop easily web applications from their Jupyter notebooks.

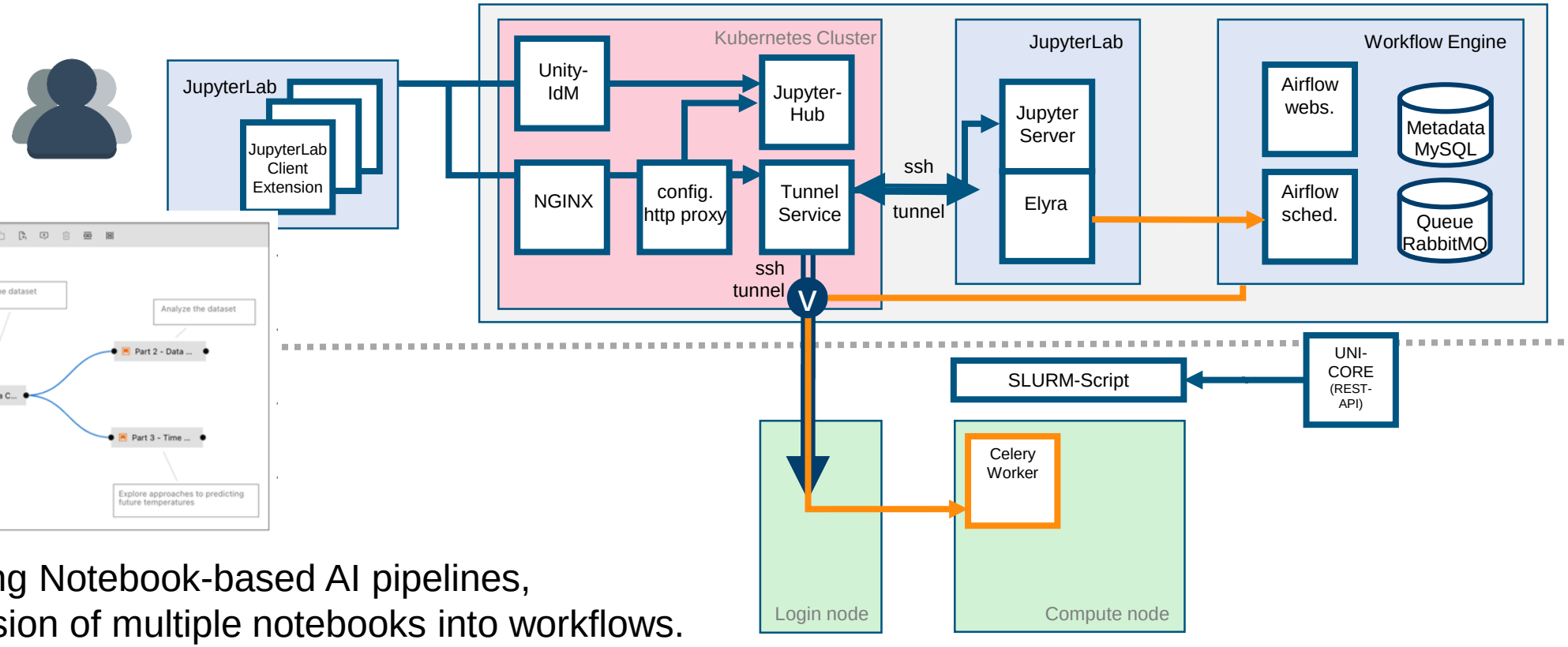


https://github.com/voila-dashboards/voila
https://voila-gallery.org

JÜLICH Forschungszentrum

ELYRA – WORKFLOW UI FOR JUPYTERLAB





Visual editor for building Notebook-based AI pipelines, simplifying the conversion of multiple notebooks into workflows. With custom components new functionality can be added [3]

[1] https://elyra.readthedocs.io
[2] https://developer.ibm.com/blogs/open-source-elyra-ai-toolkit-simplifies-data-model-development
[3] https://medium.com/ibm-data-ai/elyra-3-3-pipelines-custom-components-and-catalogs-74cf198fdf48
[4] https://github.com/IBM/claimed