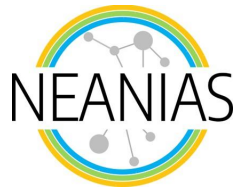


The advantages of Notebooks orbiting the Cloud



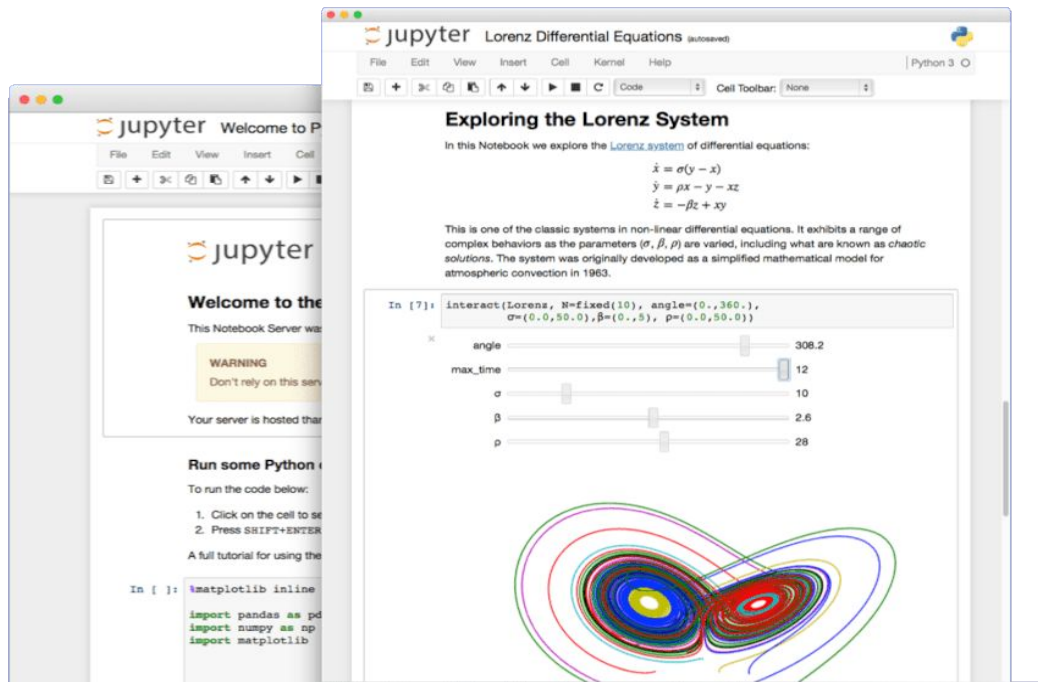
Mario Raciti

mario.raciti@inaf.it

INAF - Osservatorio Astrofisico di Catania

Project Jupyter

Open-source project to support interactive data science and scientific computing across multiple programming languages.



Jupyter Tools

JupyterLab



The latest web-based
interactive development
environment

Jupyter Notebook



The original web application
for creating and sharing
computational documents

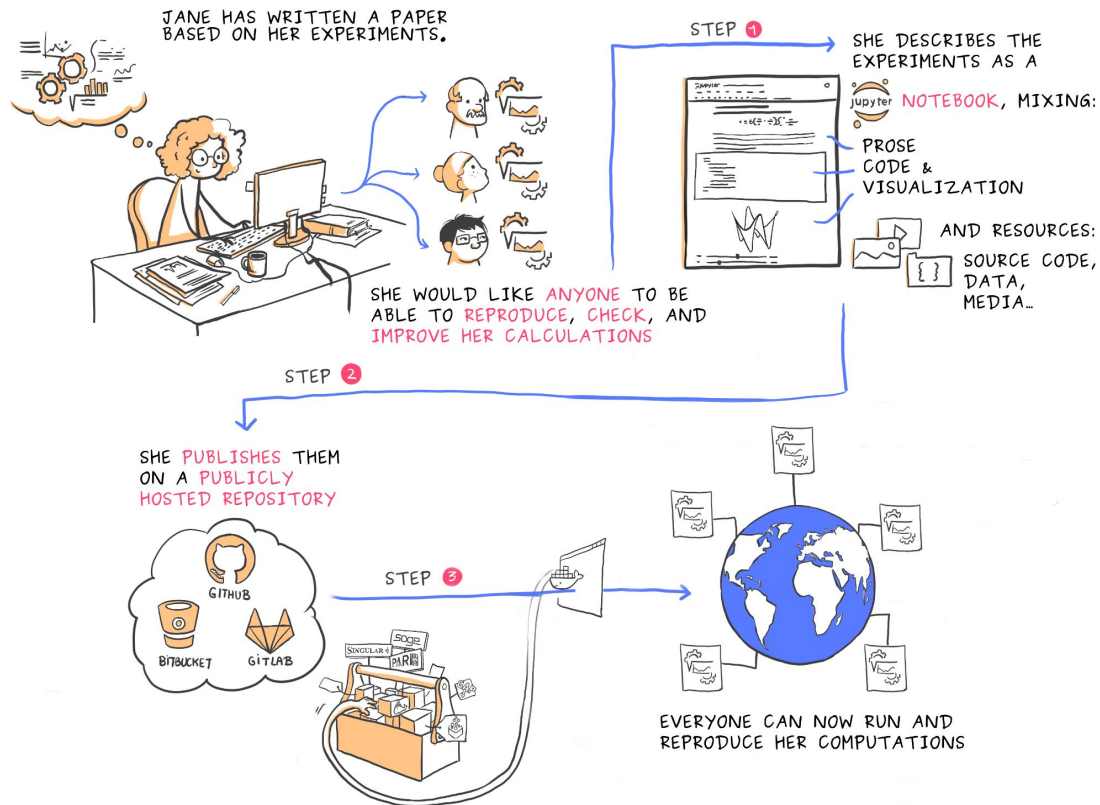
Voilà



Share insights by converting
notebooks into interactive
dashboards

Notebooks Revolution

- Annotations
- Code
- Plots
- Reports
- Presentations
- Reproducibility



Notebooks in a Nutshell

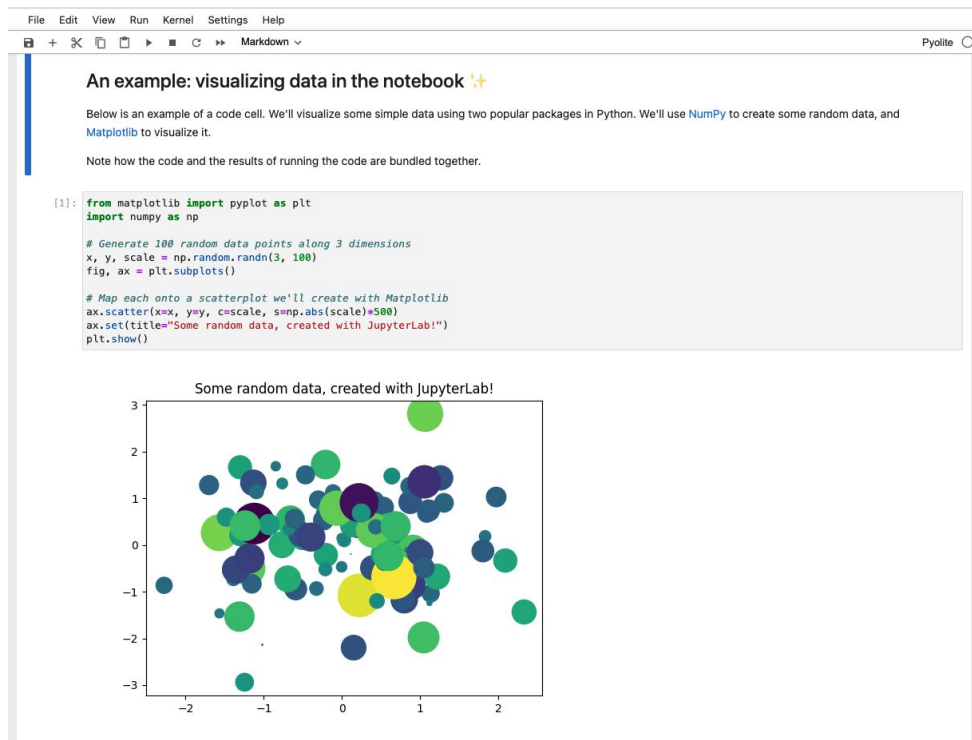
Notebooks are **documents** combining live runnable code with narrative text.

They consist in a list of cells of different type:

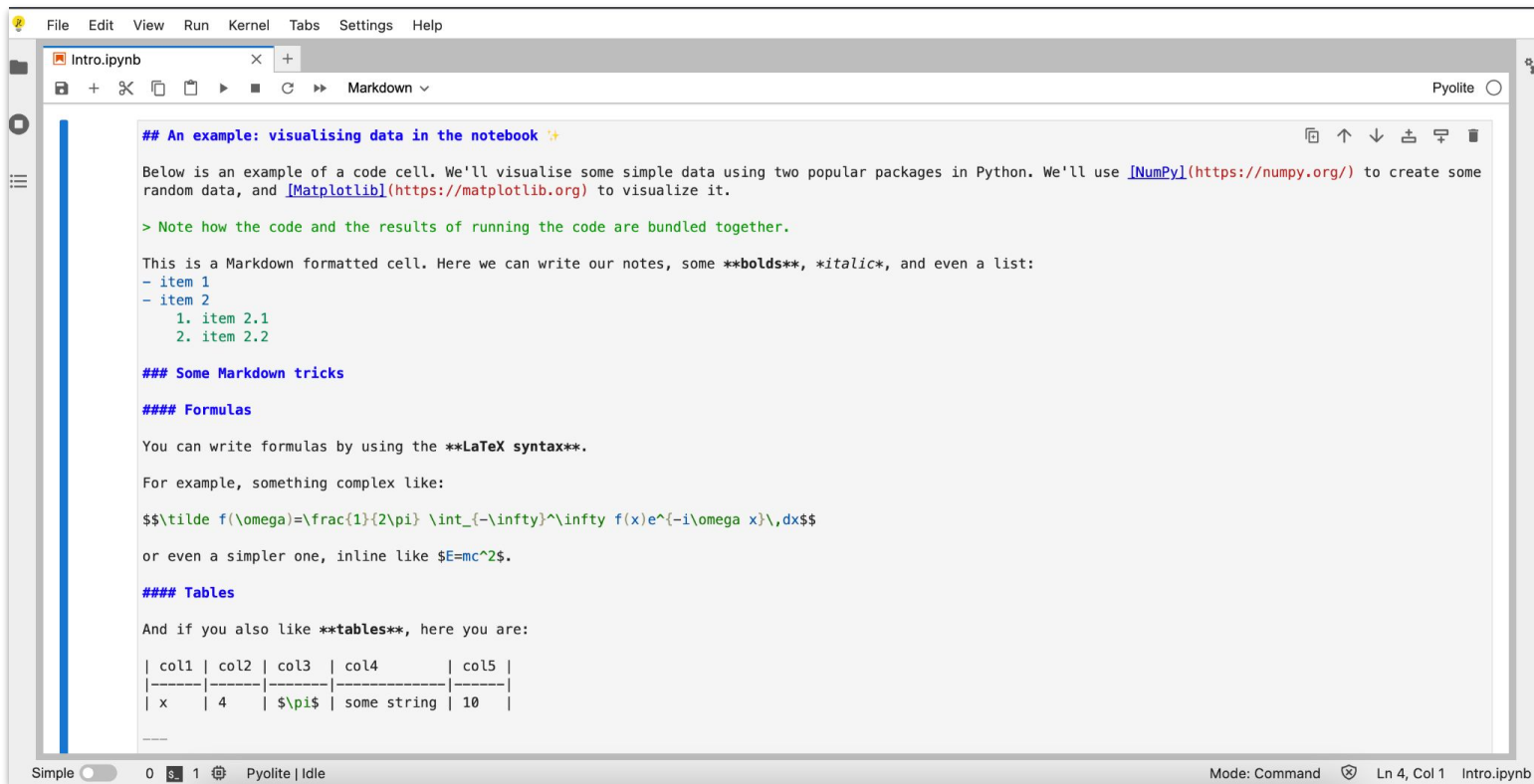
- **Code**: executable code
- **Markdown**: Markdown formatted text
- **Raw**: plain text

A cell has an input and an output area.

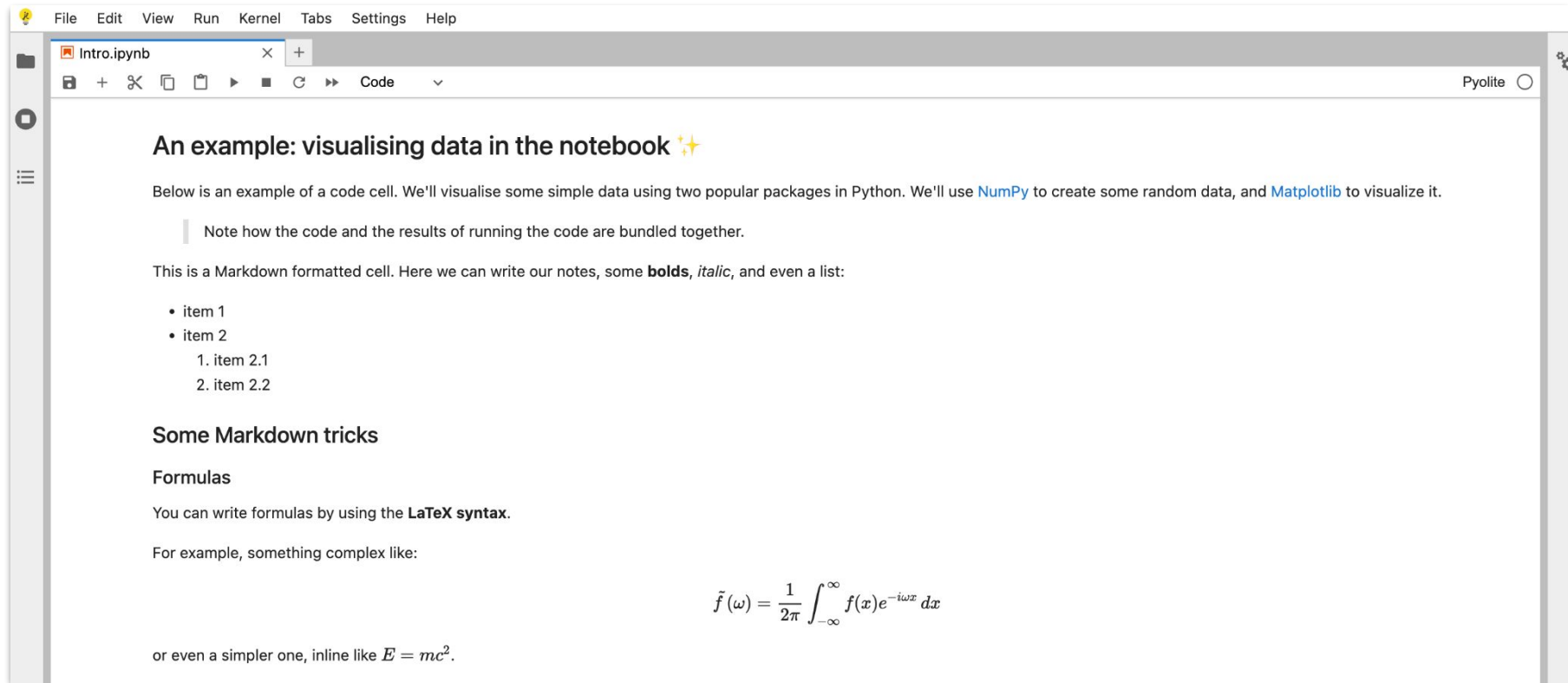
Hitting the Play button will run the cell content on an **execution kernel**.



Notebooks Sample (1)



Notebooks Sample (2)



The screenshot shows a Jupyter Notebook interface with a dark blue header. The notebook is titled "Intro.ipynb" and has a menu bar with options: File, Edit, View, Run, Kernel, Tabs, Settings, Help. Below the menu bar is a toolbar with icons for saving, adding, deleting, copying, pasting, running, and other actions. The notebook content is displayed in a light gray area. It starts with a title "An example: visualising data in the notebook" followed by a paragraph explaining the purpose of the notebook. A code cell follows, containing a note about bundling code and results. This is followed by a Markdown cell with a list of items. The notebook then shows a section titled "Some Markdown tricks" with a subsection "Formulas". It explains how to write formulas using LaTeX syntax and provides an example of a complex formula and a simpler one.

File Edit View Run Kernel Tabs Settings Help

Intro.ipynb

Code

Pyolite

An example: visualising data in the notebook ✨

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

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

This is a Markdown formatted cell. Here we can write our notes, some **bolds**, *italic*, and even a list:

- item 1
- item 2
 - 1. item 2.1
 - 2. item 2.2

Some Markdown tricks

Formulas

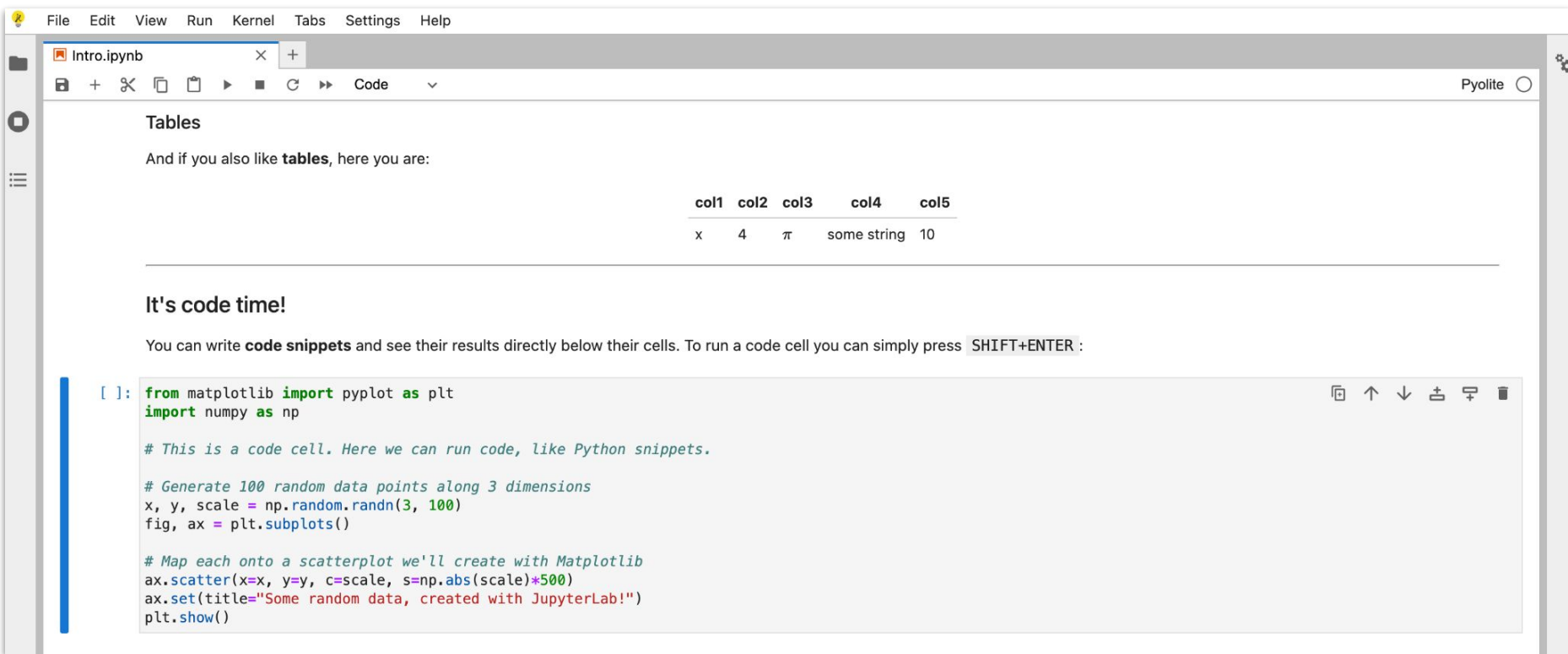
You can write formulas by using the **LaTeX syntax**.

For example, something complex like:

$$\tilde{f}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx$$

or even a simpler one, inline like $E = mc^2$.

Notebooks Sample (3)



The screenshot shows a Jupyter Notebook window titled "Intro.ipynb". The interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help) and a toolbar with icons for saving, adding, deleting, and running code cells. The notebook content is divided into sections: "Tables" and "It's code time!".

Tables

And if you also like **tables**, here you are:

col1	col2	col3	col4	col5
x	4	π	some string	10

It's code time!

You can write **code snippets** and see their results directly below their cells. To run a code cell you can simply press **SHIFT+ENTER** :

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

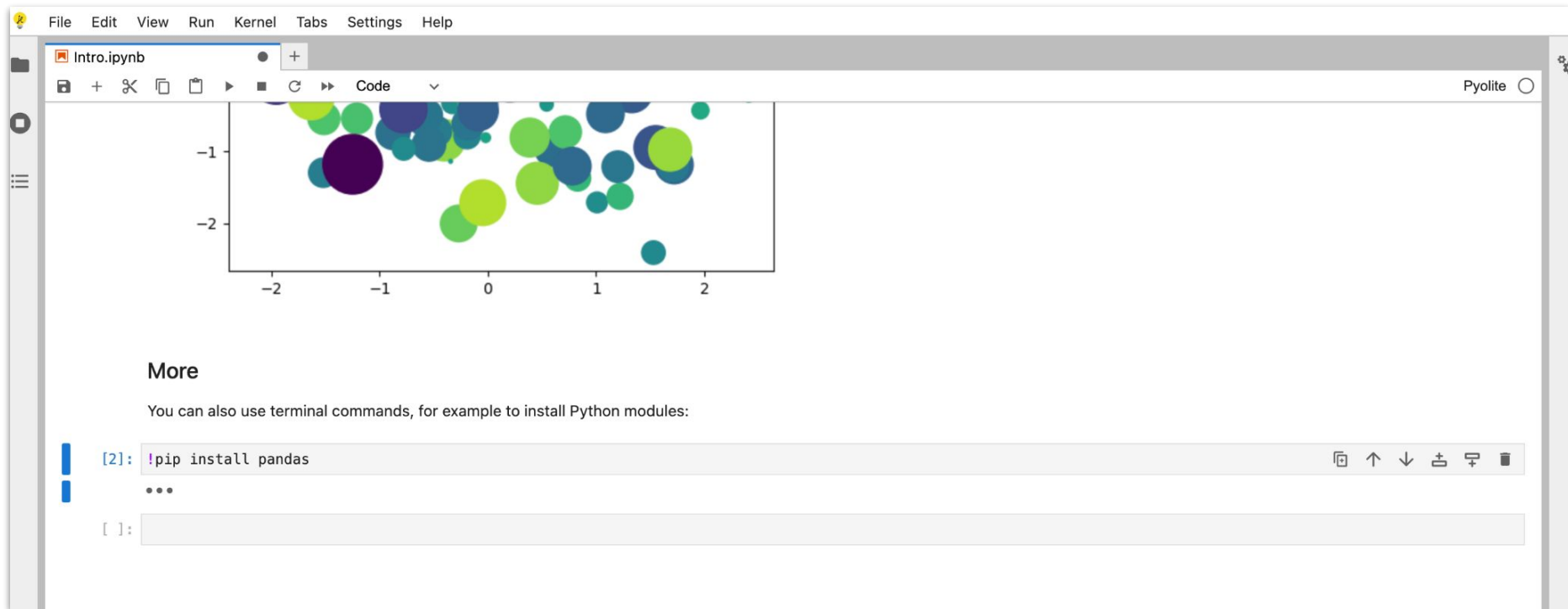
# This is a code cell. Here we can run code, like Python snippets.

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

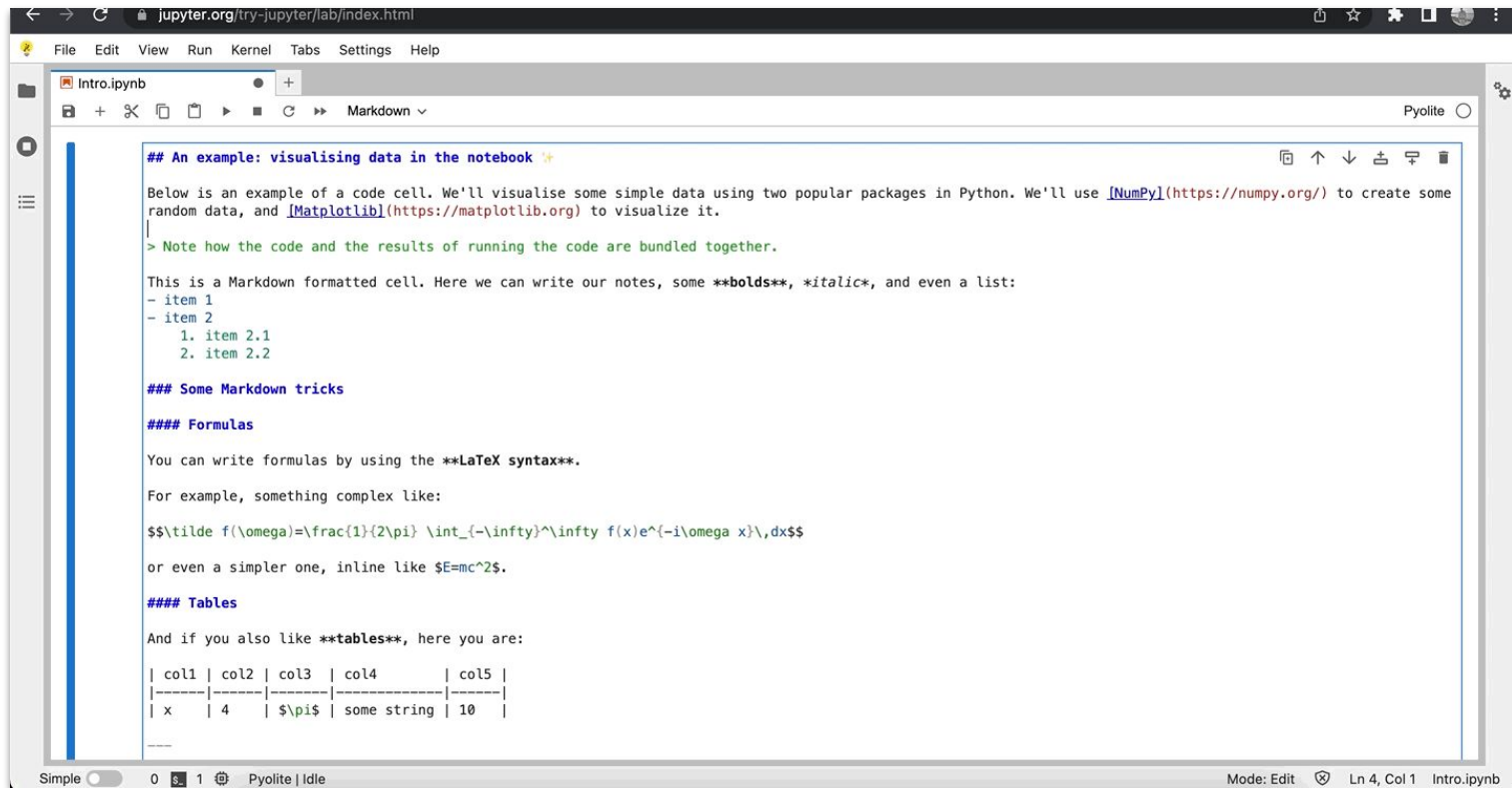
# Map each onto a scatterplot we'll create with Matplotlib
ax.scatter(x=x, y=y, c=scale, s=np.abs(scale)*500)
ax.set(title="Some random data, created with JupyterLab!")
plt.show()
```

The code cell output shows a scatter plot of 100 random data points along 3 dimensions. The plot is titled "Some random data, created with JupyterLab!". The x and y axes are labeled, and the scale of the data points is represented by the size of the markers. The plot is displayed in a separate window titled "Pyolite".

Notebooks Sample (4)



Notebooks Sample (5)



Notebooks Local Setup (1)

JupyterLab

Install JupyterLab with **pip**:

```
pip install jupyterlab
```

Note: If you install JupyterLab with conda or mamba, we recommend using **the conda-forge channel**.

Once installed, launch JupyterLab with:

```
jupyter-lab
```

Jupyter Notebook

Install the classic Jupyter Notebook with:

```
pip install notebook
```

To run the notebook:

```
jupyter notebook
```

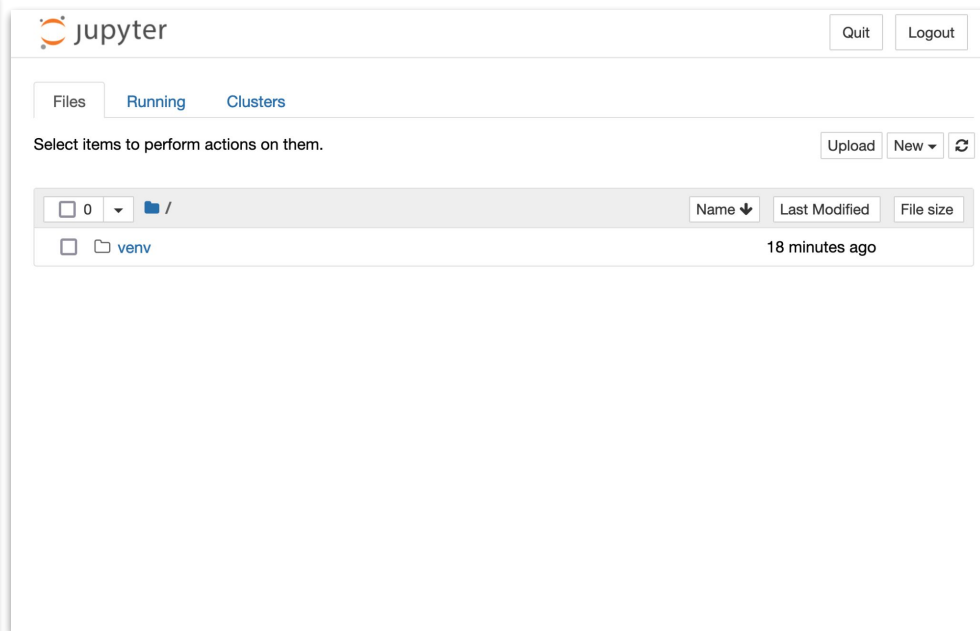
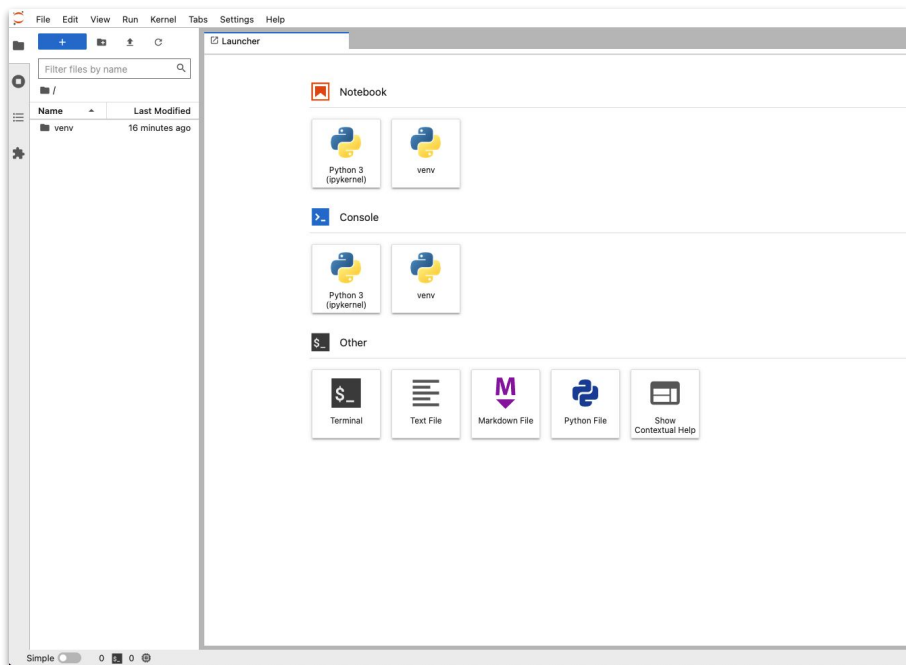
Notebooks Local Setup (2)

```
~/Desktop/VS-Gateway-Presentation  VS-Gateway-Presentation at 04:04:31 PM
> jupyter-lab
[I 2022-04-20 16:04:34.557 ServerApp] jupyterlab | extension was successfully linked.
[I 2022-04-20 16:04:34.567 ServerApp] nbclassic | extension was successfully linked.
[I 2022-04-20 16:04:34.932 ServerApp] notebook_shim | extension was successfully linked.
[I 2022-04-20 16:04:34.985 ServerApp] notebook_shim | extension was successfully loaded.
[I 2022-04-20 16:04:34.986 LabApp] JupyterLab extension loaded from /Users/mario/Desktop/VS-Gateway-Presentation/venv/lib/python3.9/site-packages/jupyterlab
[I 2022-04-20 16:04:34.987 LabApp] JupyterLab application directory is /Users/mario/Desktop/VS-Gateway-Presentation/venv/share/jupyter/lab
[I 2022-04-20 16:04:34.990 ServerApp] jupyterlab | extension was successfully loaded.
[I 2022-04-20 16:04:34.996 ServerApp] nbclassic | extension was successfully loaded.
[I 2022-04-20 16:04:34.997 ServerApp] Serving notebooks from local directory: /Users/mario/Desktop/VS-Gateway-Presentation
[I 2022-04-20 16:04:34.997 ServerApp] Jupyter Server 1.16.0 is running at:
[I 2022-04-20 16:04:34.997 ServerApp] http://localhost:8888/lab?token=fbca12339da8c6f77941f16ca2e020dc7be5af3dd77d6558
[I 2022-04-20 16:04:34.997 ServerApp] or http://127.0.0.1:8888/lab?token=fbca12339da8c6f77941f16ca2e020dc7be5af3dd77d6558
[I 2022-04-20 16:04:34.997 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).

[C 2022-04-20 16:04:35.005 ServerApp]

To access the server, open this file in a browser:
    file:///Users/mario/Library/Jupyter/runtime/jpserver-2576-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/lab?token=fbca12339da8c6f77941f16ca2e020dc7be5af3dd77d6558
    or http://127.0.0.1:8888/lab?token=fbca12339da8c6f77941f16ca2e020dc7be5af3dd77d6558
[I 2022-04-20 16:04:39.883 LabApp] Build is up to date
```

Notebooks Local Setup (3)



To Cloud and Beyond!

A multi-user Hub that spawns, manages, and proxies multiple instances of the single-user Jupyter notebook server.



Jupyter Hub in a Nutshell



A multi-user version of the notebook designed for companies, classrooms and research labs



Pluggable authentication

Manage users and authentication with PAM, OAuth or integrate with your own directory service system.



Centralized deployment

Deploy the Jupyter Notebook to thousands of users in your organization on centralized infrastructure on- or off-site.



Container friendly

Use Docker and Kubernetes to scale your deployment, isolate user processes, and simplify software installation.



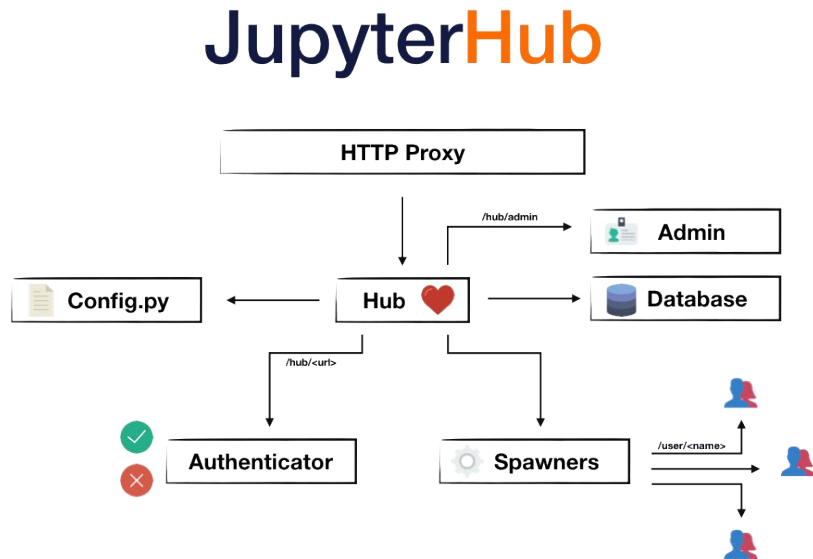
Code meets data

Deploy the Notebook next to your data to provide unified software management and data access within your organization.

Jupyter Hub Overview (1)

Four subsystems make up JupyterHub:

- a **Hub** (tornado process) that is the heart of JupyterHub
- a configurable **HTTP proxy** that receives the requests from the client's browser
- **multiple single-user Jupyter notebook servers** (Python/IPython/tornado) that are monitored by Spawners
- an **authentication class** that manages how users can access the system

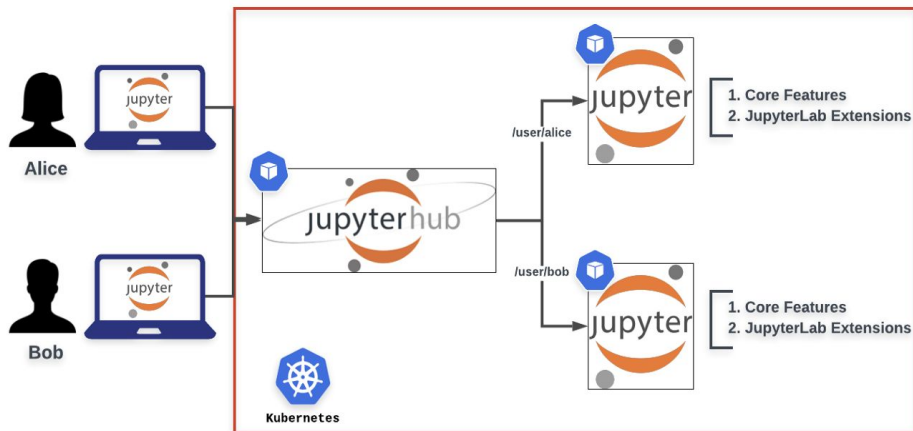


Jupyter Hub Overview (2)

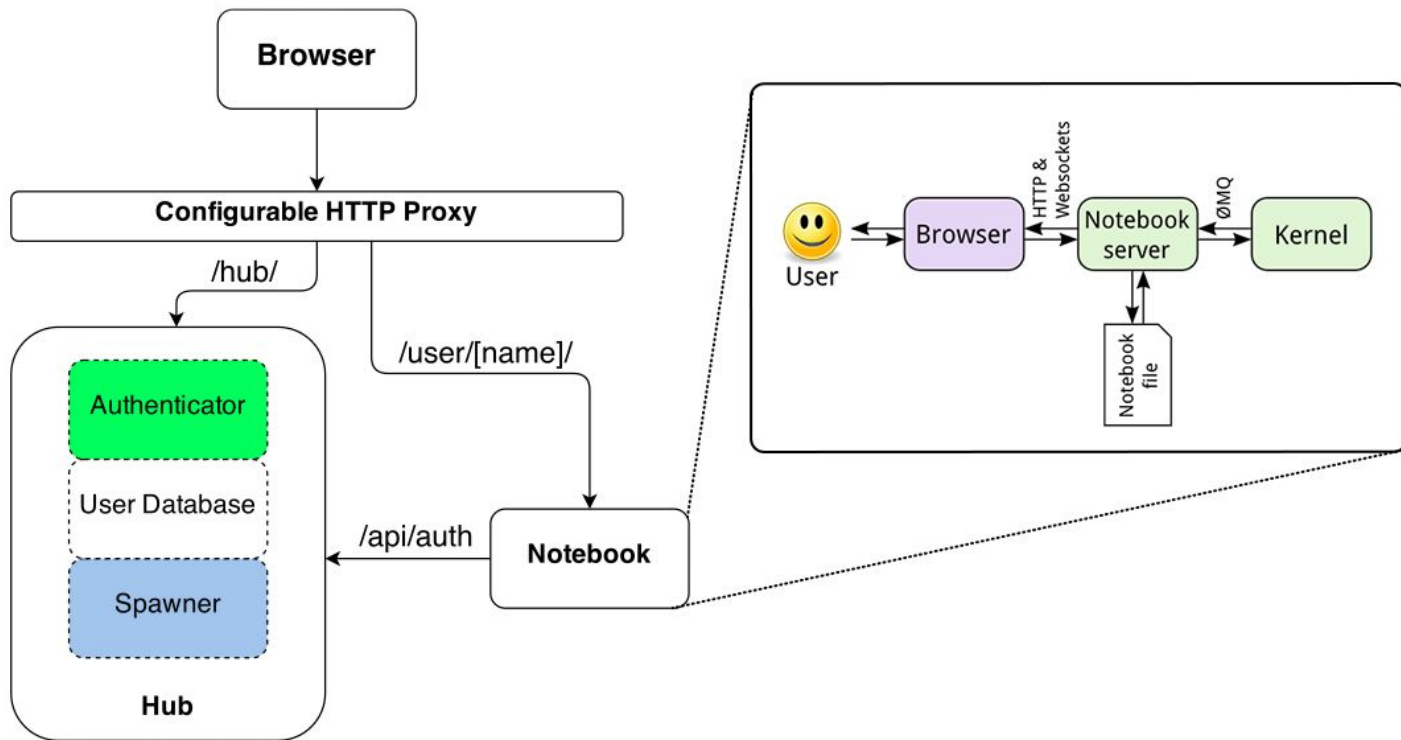
JupyterHub performs the following functions:

- the Hub launches a **proxy**
- the proxy forwards all requests to the Hub by default
- the Hub handles **user login** and spawns **single-user servers on demand**
- the Hub configures the proxy to forward URL prefixes to the single-user notebook servers

JupyterHub also provides a **REST API** for administration.



Jupyter Hub Overview (3)



Jupyter Hub Deployment

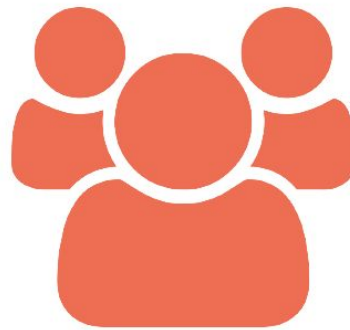
Zero to JupyterHub for Kubernetes (Z2JH)

1. Spread users on a scalable cluster (more users)
2. Use container technology

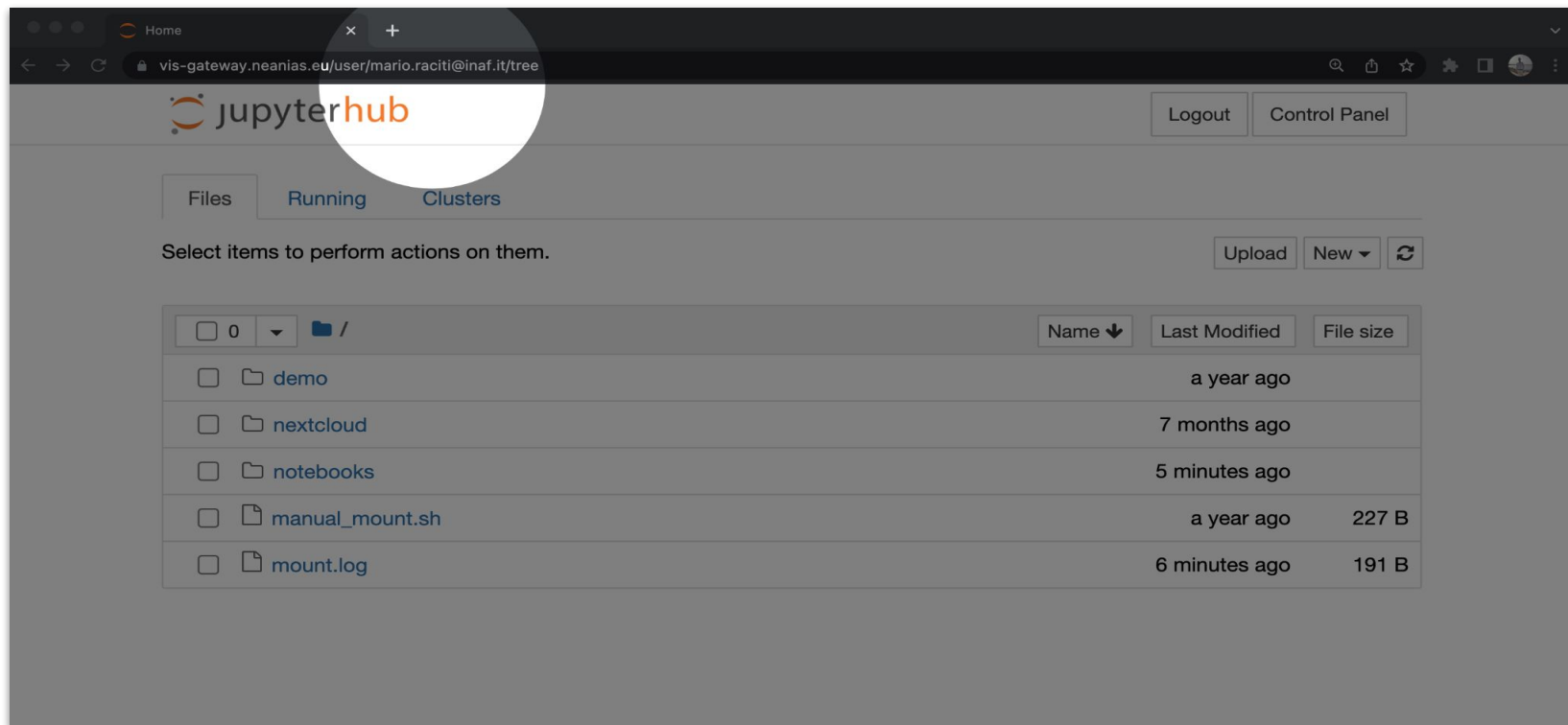


The Littlest JupyterHub (TLJH)

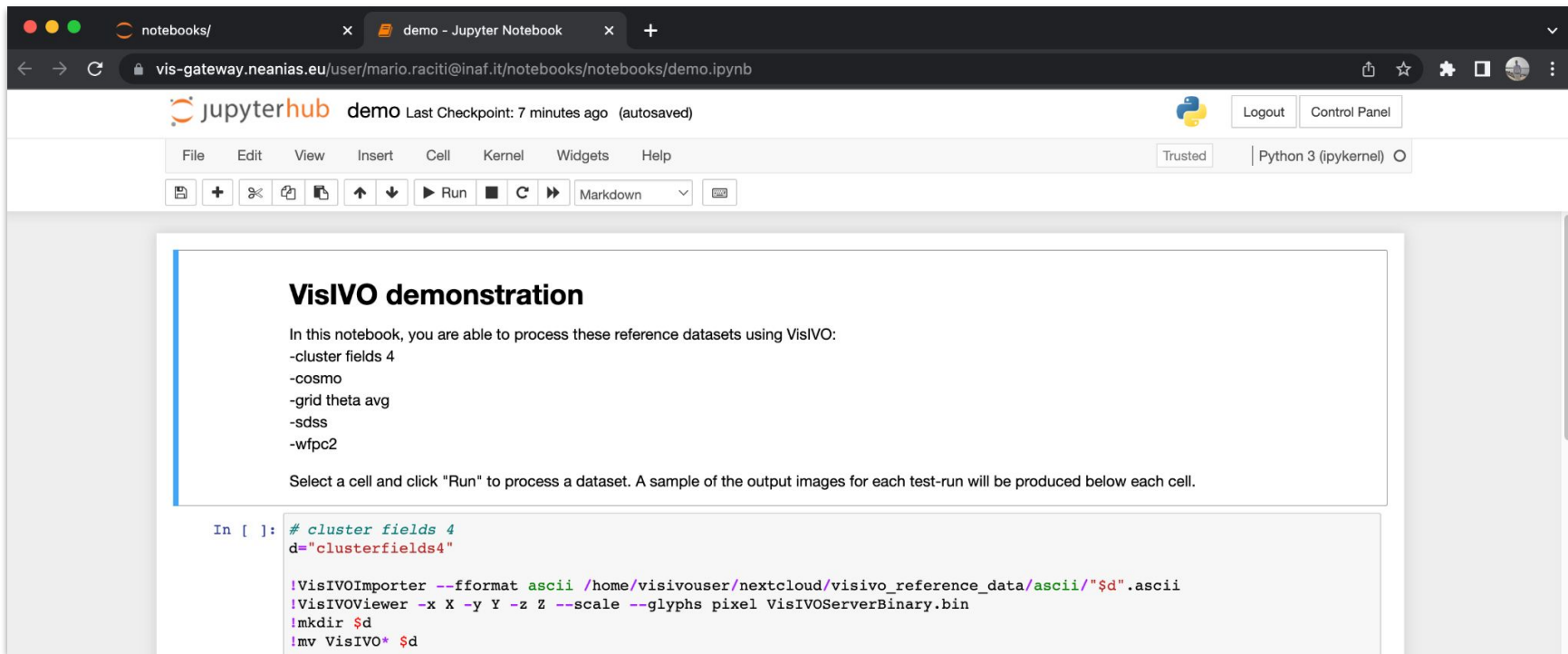
1. Single machine (less users)
2. No containers



Jupyter Hub Interface (1)



Jupyter Hub Interface (2)



The screenshot displays the Jupyter Hub web interface in a browser. The address bar shows the URL `vis-gateway.neanias.eu/user/mario.raciti@inaf.it/notebooks/notebooks/demo.ipynb`. The page header includes the Jupyter logo, the word "demo", and a status message "Last Checkpoint: 7 minutes ago (autosaved)". On the right, there are "Logout" and "Control Panel" buttons. Below the header is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A secondary bar contains icons for file operations, a "Run" button, and a "Markdown" dropdown. The main content area features a heading "VisIVO demonstration" followed by a paragraph: "In this notebook, you are able to process these reference datasets using VisIVO:". Below this is a list of datasets: `-cluster fields 4`, `-cosmo`, `-grid theta avg`, `-sdss`, and `-wfp2`. A note states: "Select a cell and click 'Run' to process a dataset. A sample of the output images for each test-run will be produced below each cell." The bottom section shows a code cell with the following content:

```
In [ ]: # cluster fields 4
        d="clusterfields4"

!VisIVOImporter --fformat ascii /home/visivouser/nextcloud/visivo_reference_data/ascii/"$d".ascii
!VisIVOViewer -x X -y Y -z Z --scale --glyphs pixel VisIVOServerBinary.bin
!mkdir $d
!mv VisIVO* $d
```

Use Case: NEANIAS VG

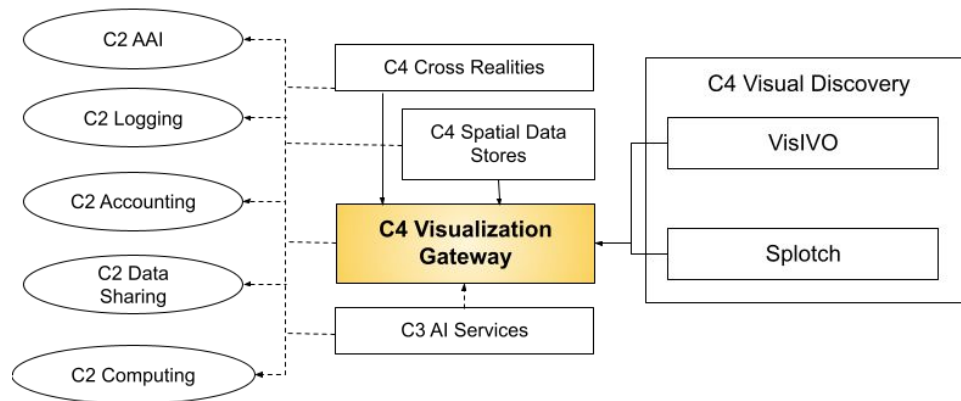
A development environment for designing, rapid prototyping, implementing and fully testing complex visualisation solutions for realising common data exploration workflows.



NEANIAS VG in a Nutshell

The Visualization Gateway:

- is based on **Z2JH** to serve as a universal core service for multiple users
- runs on the **GARR K8s Cluster**
- currently includes two visualisation frameworks (**Splotch** and **VisIVO**)
- includes a **Data Sharing Service** (WebDAV) to auto-mount to remote reference-data filestore (Nextcloud)



More @ [Sciacca E. et al. Journal of Grid Computing](#)

vis-gateway.neanias.eu/hub/spawn

jupyterhub Home Token mario.raciti@inaf.it Logout

Server Options

Remote storage mode

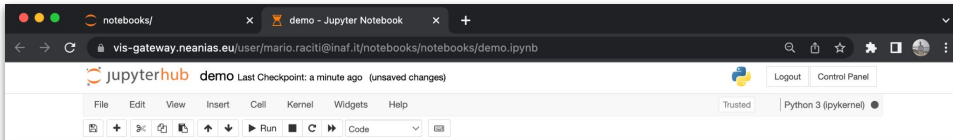
- ☒ **Auto-mount**
Auto-mount to remote reference-data filestore (Nextcloud)
- ☐ **Other**
Mount to other filestore via WebDAV

☒ **VD-Splotch**
VD-Splotch command line tools

☐ **VD-VisIVO**
VD-VisIVO command line tools

☐ **DS-AdamSpace**
Jupyter notebook+adamapi and examples

Start



Spotlch demonstration

In this notebook, you are able to run Spotlch on the reference datasets, D1-D5, using the 'run-spotlch' command. This command also 'stitches' the frames produced by Spotlch to form an animation. This is particularly interesting when the Spotlch 'camera' orbits the data points of interest.

The animation scripts have been designed to last around 1-2 minutes. By default, the dataset D1, "snap92", is used. This is a significantly smaller dataset than the others, and so, typically, more frames are produced in the given time. This means that the D1 animation can have either a longer orbit time or a higher frame rate than other animations, given that the Spotlch execution time is constant. There is a frame delay parameter which can be used to control the frame rate. This is the time for which each frame is displayed before changing to the next frame.

```
In [*]: # Run this cell to produce an animation for the chosen dataset.
# Use the parameters to select:
#   a dataset
#   a frame delay (time for which each frame is displayed)

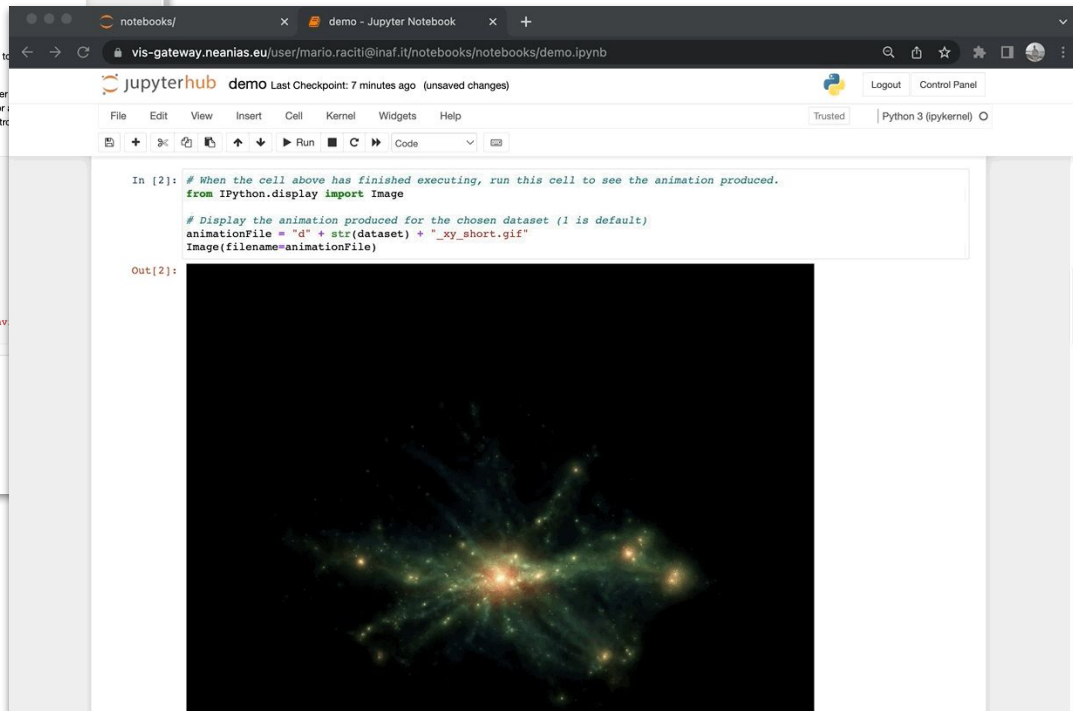
# D1
dataset = 1

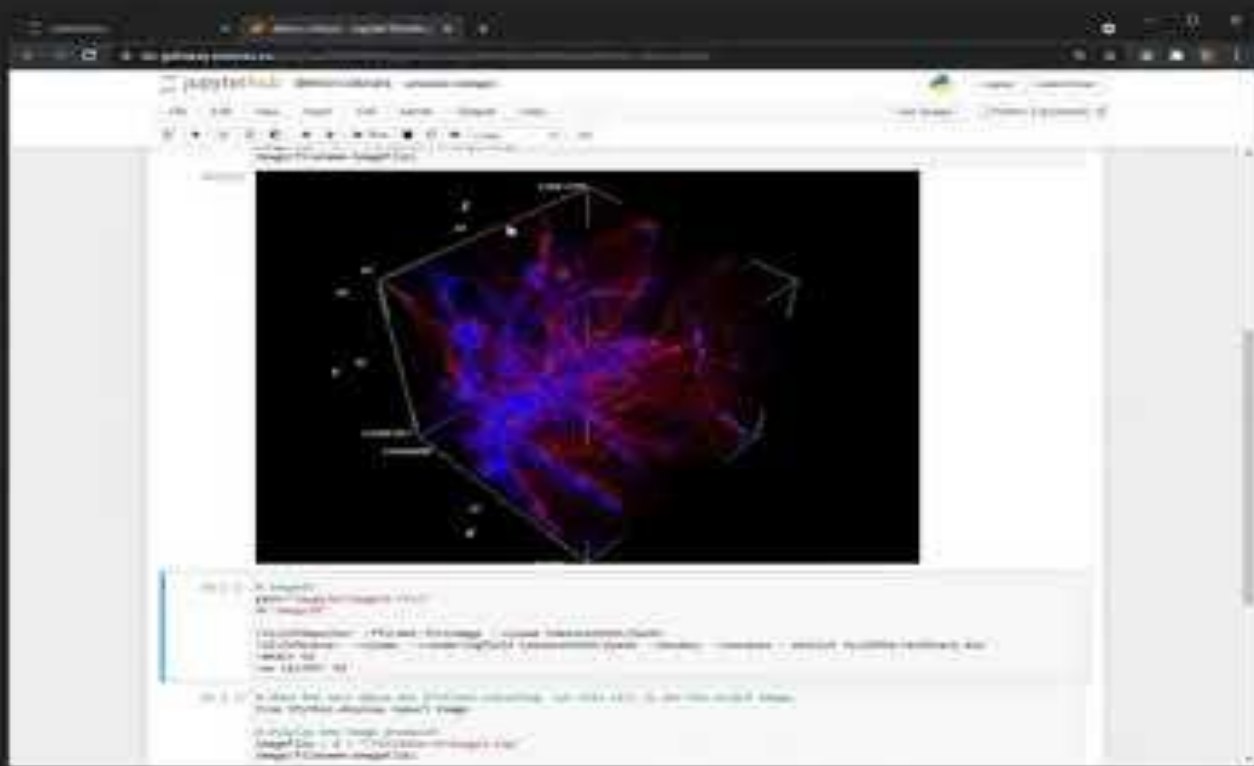
# 100ms delay
delay = 10

# run spotlch and produce short animation where by the camera rotates about the z-axis
print("Dataset D" + str(dataset) + "\n")
!run-spotlch D{dataset} xy_short $delay /home/spotlchuser/nextcloud/spotlch_reference_data | grep -E -o "(save|print)"
```

Dataset D1

```
saving file d1_0000 ...
saving file d1_0020 ...
saving file d1_0040 ...
saving file d1_0060 ...
saving file d1_0080 ...
saving file d1_0100 ...
saving file d1_0120 ...
saving file d1_0140 ...
saving file d1_0160 ...
```





What's the deal in short?



Mix of different components – **text, graphs, code** – presented in a nice and easily understandable way.

Multi-language support.

Extensions for data visualisation, HPC simulations, dashboards and automatic test scoring.



Users are not bothered with installations, dependency handling or administration procedures.

Research results can be **easily distributed**, as notebooks are easy to share with others.

One step closer to collaborative ecosystems promoting **Open Science** and **FAIR** principles.

References

- [Project Jupyter](#)
- [Jupyter Notebook](#)
- [Jupyter Lab](#)
- [Markdown Cheat Sheet](#)
- [Jupyter Hub](#)
- [Z2JH](#)
- [TLJH](#)
- [NEANIAS Visualization Gateway](#)
- [VG Journal of Grid Computing](#)
- [VisiVO](#)
- [VisiVO YT video](#)



Source credits: vecteezy.com