Jupyter in HPC

Feb 28th, 2018

Matthias Bussonnier

bussonniermatthias@gmail.com
GitHub: @carreau
Twitter: @mbussonn

About Me

Matthias Bussonnier

- A Physicist/Bio-Physicist
- Core developer of IPython/Jupyter since 2012
    - Co-founder, and Steering Council member
- Post doctoral Scholar on Jupyter at BIDS

Webinar & Outline

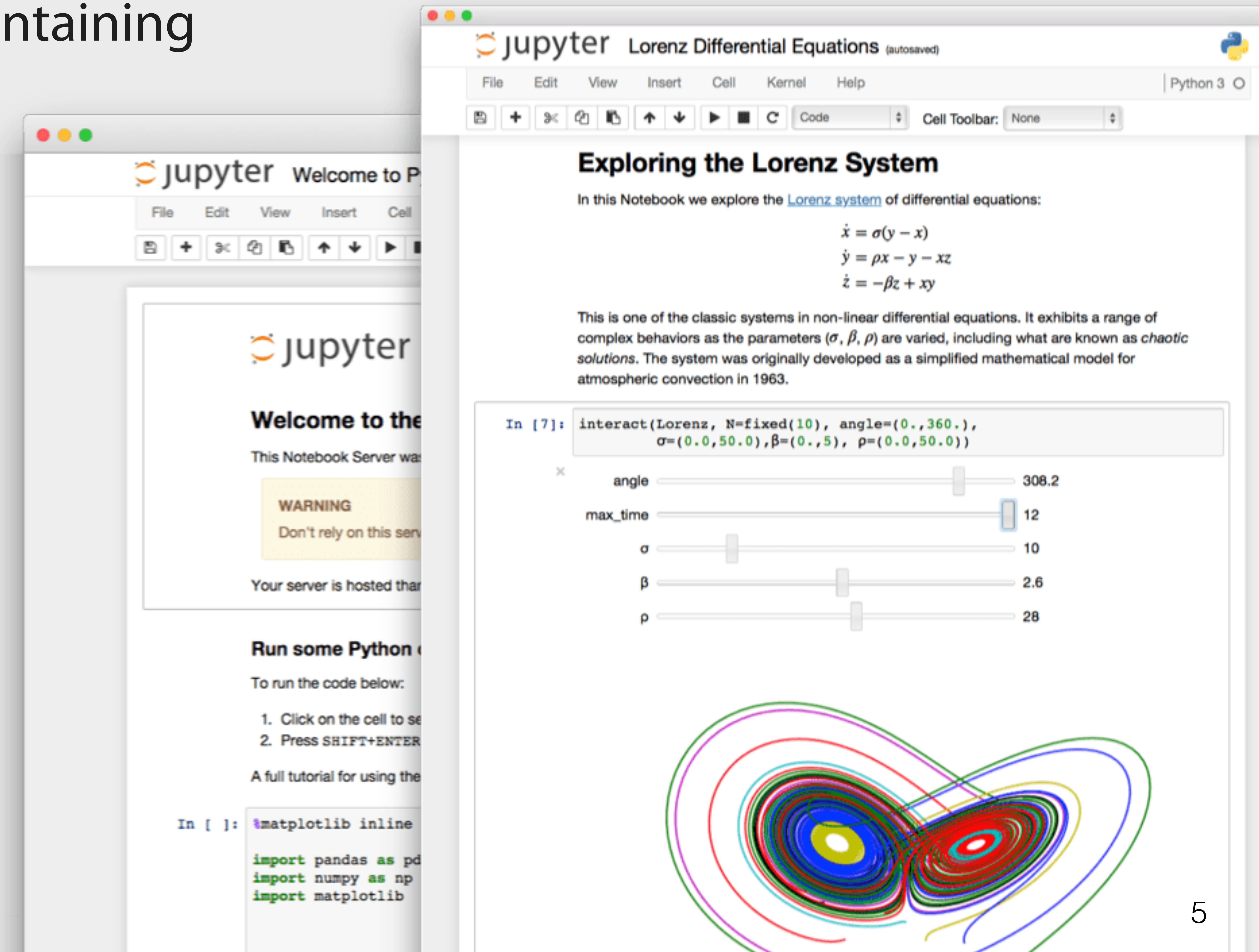
- This webinar will be in 3 parts
  - Overview of what is Jupyter + HPC
  - Use case : Suha Somnath
  - Use case : Shreyas Cholia
- Outline Part 1
  - From IPython to Jupyter
  - What is Jupyter
  - Jupyter Popularity
  - Some Jupyter Usage

From IPython to Jupyter

- 2001: Fernando Perez Wrote "**IPython**"
  - Create IPython for Interactive Python with prompt number, gnu plot integration
  - Replace a bunch on perl/make/C/C++ files with only Python.
- 2011: QtConsole
- 2012: Birth of current **Notebook** (6th prototype)
  - Make IPython "network enabled"
  - Made possible by mature web tech.
- 2013: First non-Python (**Julia)** kernel
- 2014: we **renamed** the Python-Agnostic part to **Jupyter**.
- 2018: several millions users & **JupyterLab** released

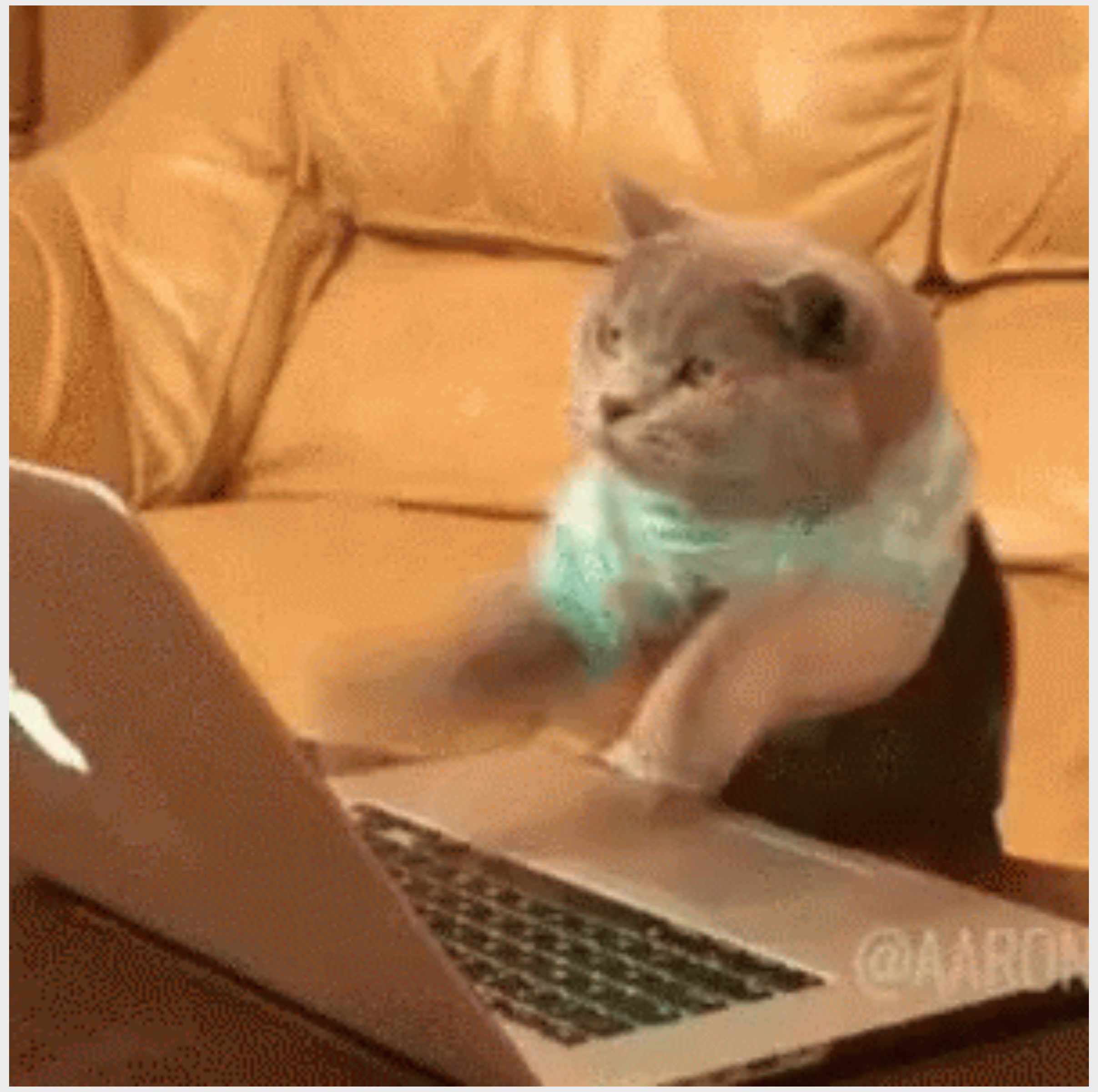
What is Jupyter

- Mainly Known for **The Notebook**
  - Web server, a web app, load .ipynb (json), containing code, narrative, math and results.
  - Attached to a **Kernel** doing computation.
- Results can be:
  - Static (Image)
  - Interactive (client-side scoll/pan/brush)
  - Dynamic (Call back into Kernel)



Focused on Exploratory Programming

- IPython was designed for exploratory programming, as a REPL (Read Eval Print Loop) and grew popular, especially among scientist who loved it to explore.



> "IPython have weaponized the tab [completion] key"
> – Fernando Pérez

Open Organisation

- Organisation with Open Governance (https://GitHub.com/jupyter/governance)
- Funded by Grants and Donations, and Collaborations

Protocols and Formats

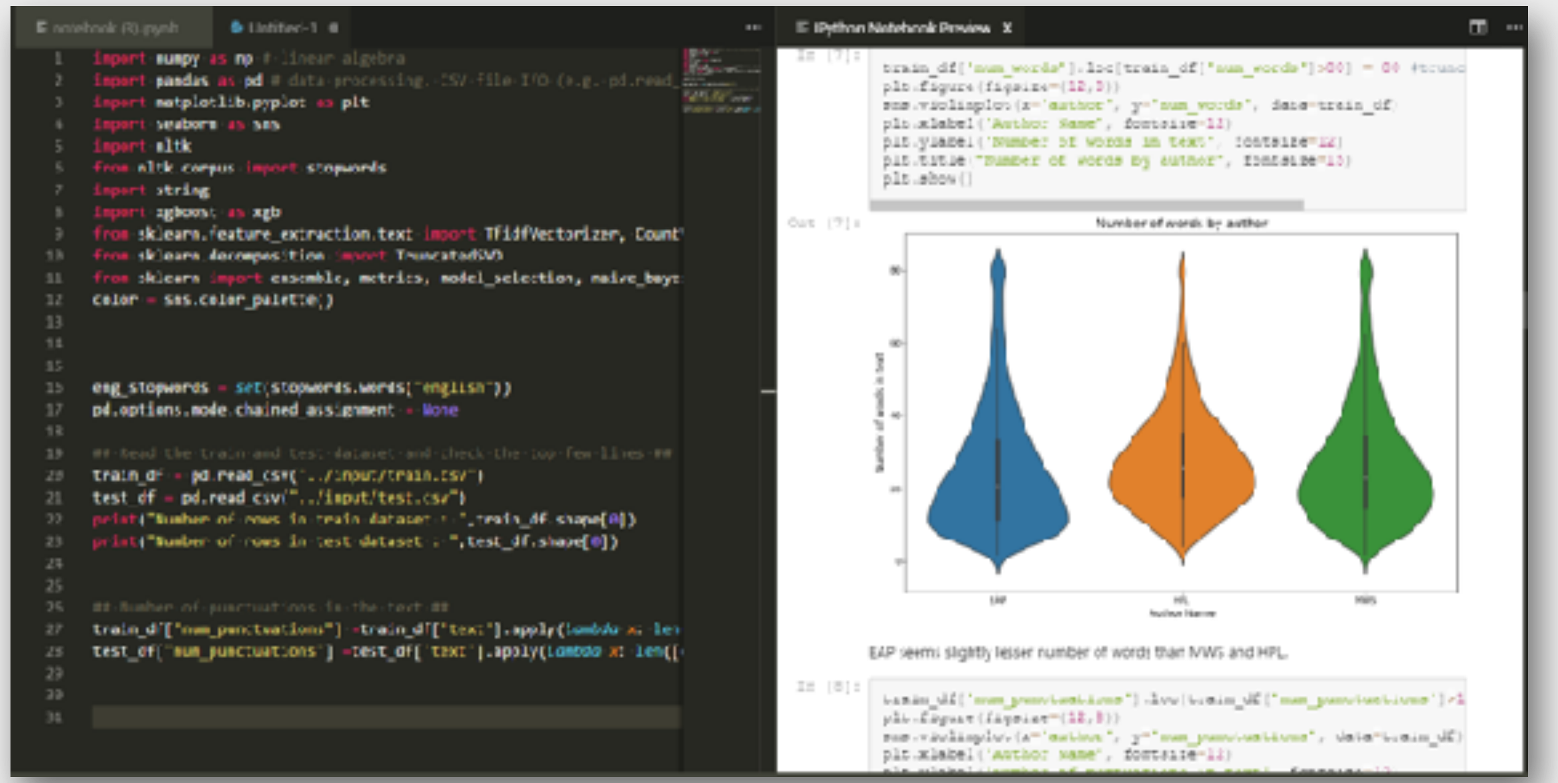
- Jupyter is also a set of **Protocols and Formats** that reduce the **N-frontends × M-backends** problem to a **M-Frontends + N-backends**,
  - Open, Free and Simple.
    - JSON (almost) everywhere
    - Notebook document format,
    - Wire protocol
  - Thought for Science and **Interactive** use case.
    - Results embedded in documents no "Copy past" mistake.
    - Scale from Education to HPC jobs.

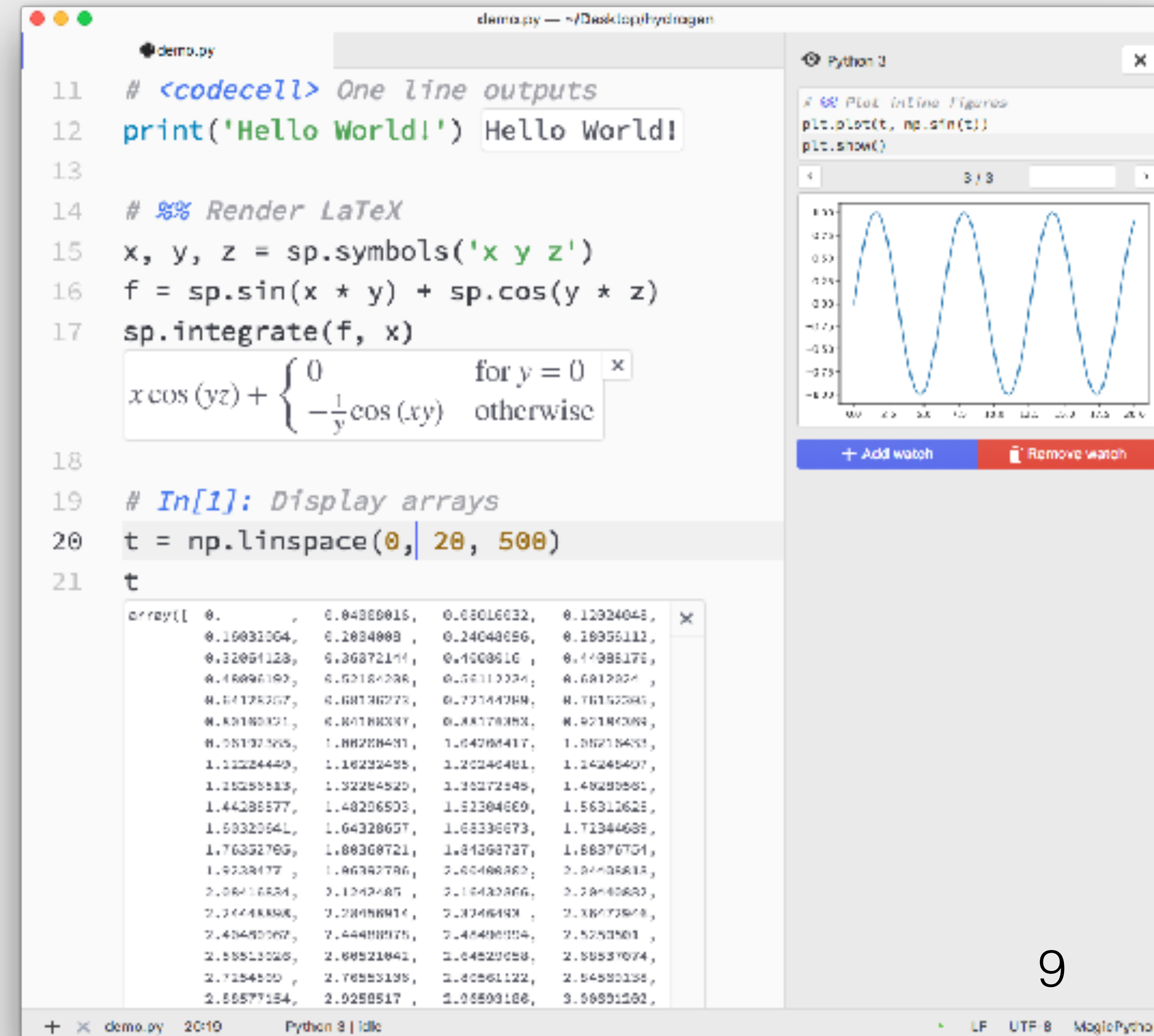
Ecosystem

Frontends: Notebook, JupyterLab, CLI, *Vim, Emacs, Visual Studio Code, Atom, Nteract, Juno...*

Kernels: Python, *Julia, R, Haskell, Perl, Fortran, Ruby, Javascript, C/C++, Go, Scala, Elixir... 60+*

Building Blocks: Nbformat, JupyterHub, Kernel Gateway...





JupyterLab

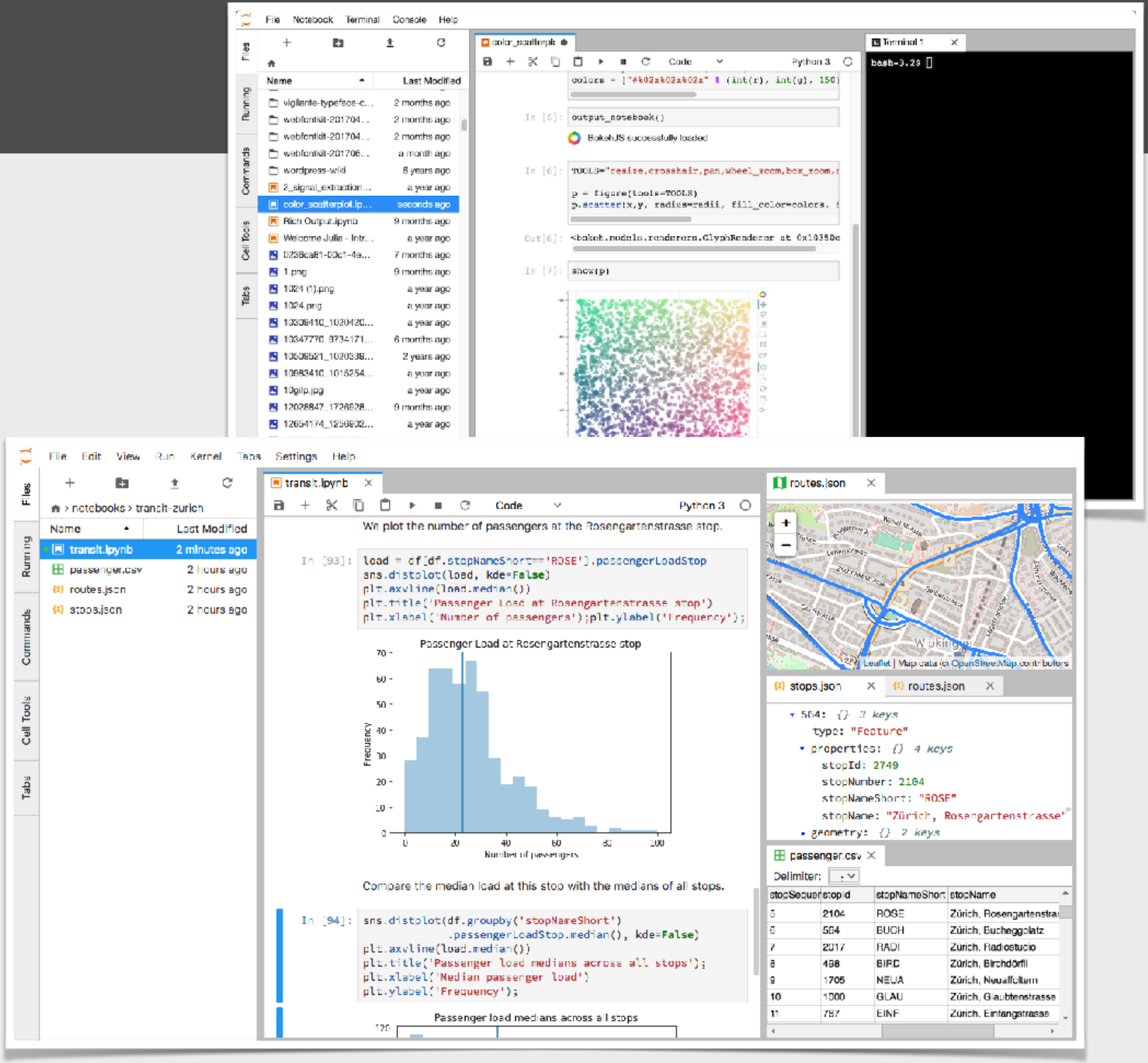
- Extends the notebook interface

  with text editor, shell, ...etc

- is it and IDE ?

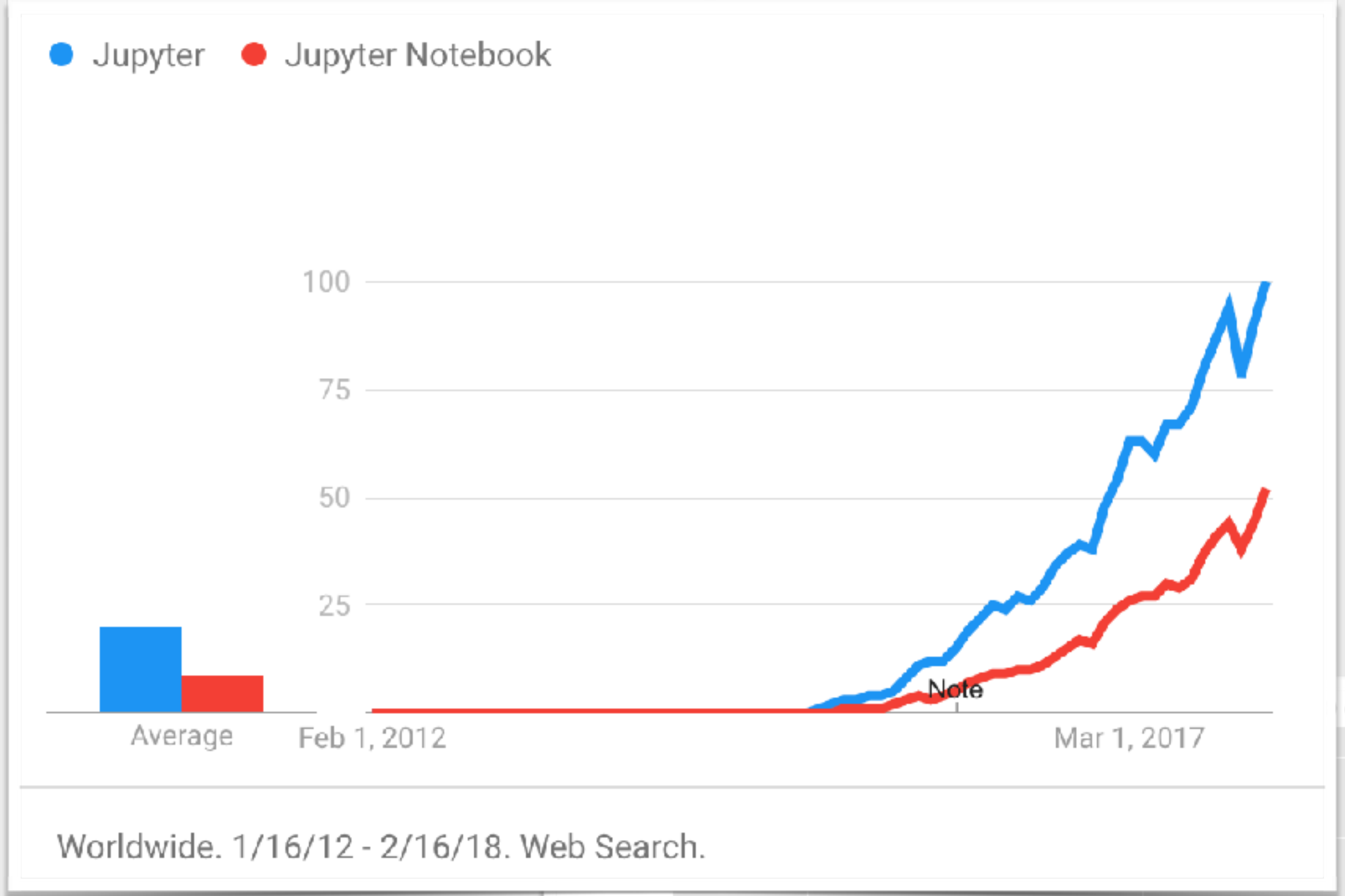
- If by I you mean Interactive,

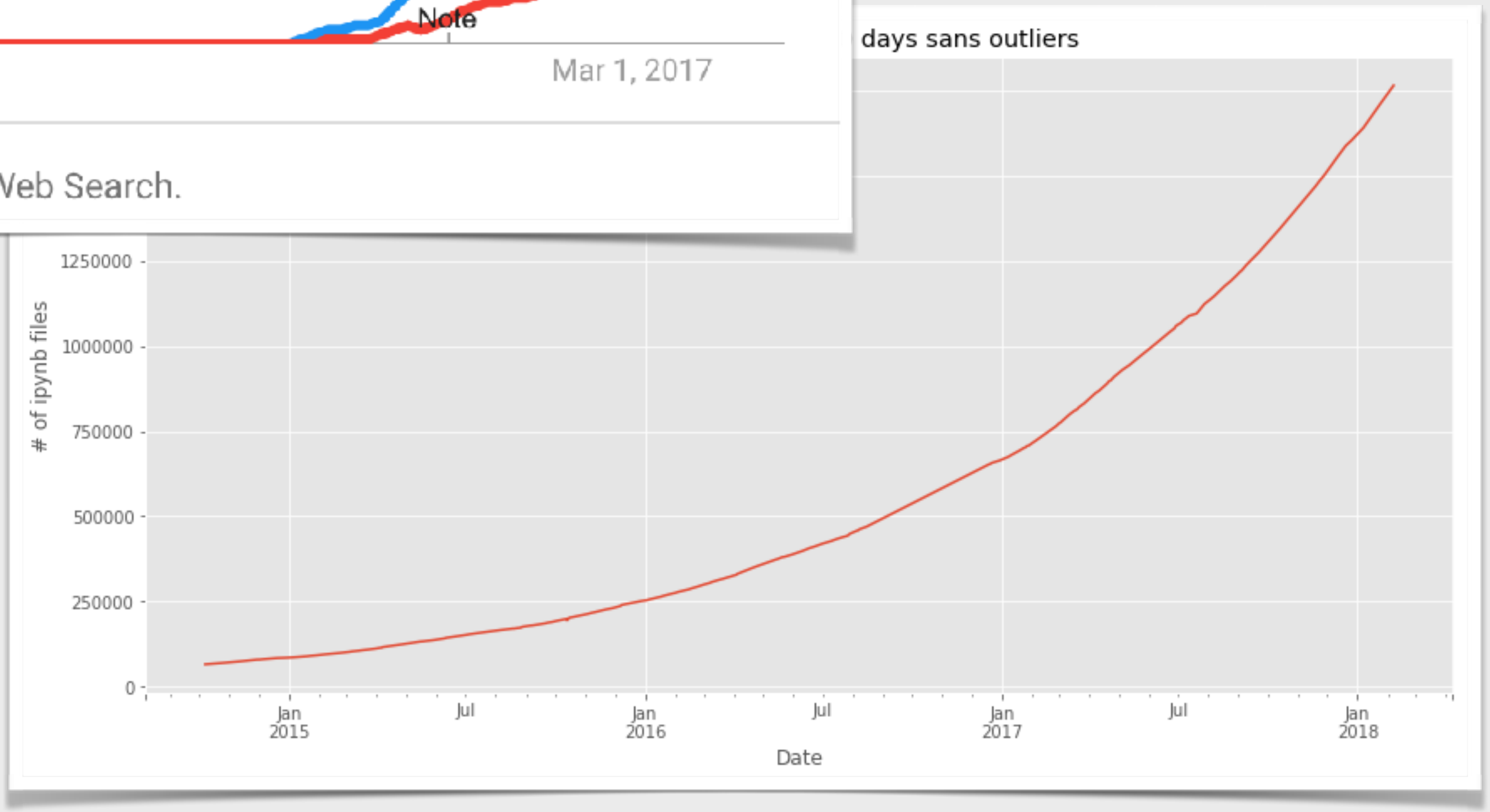
  then yes



https://blog.jupyter.org/jupyterlab-is-ready-for-users-5a6f039b8906

Popularity





https://github.com/parente/nbestimate

Popularity

Interactivity

- Coding is not the end goal of most of our users. A simple, single tool, with friendly interface helps.
- Persisting kernel state allows to iterate only on part of an analysis.
- Notebook interface give the interactivity of the REPL with the edit-ability and linearity of a script with intermediate result. Aka "Literate Computing”

Popularity

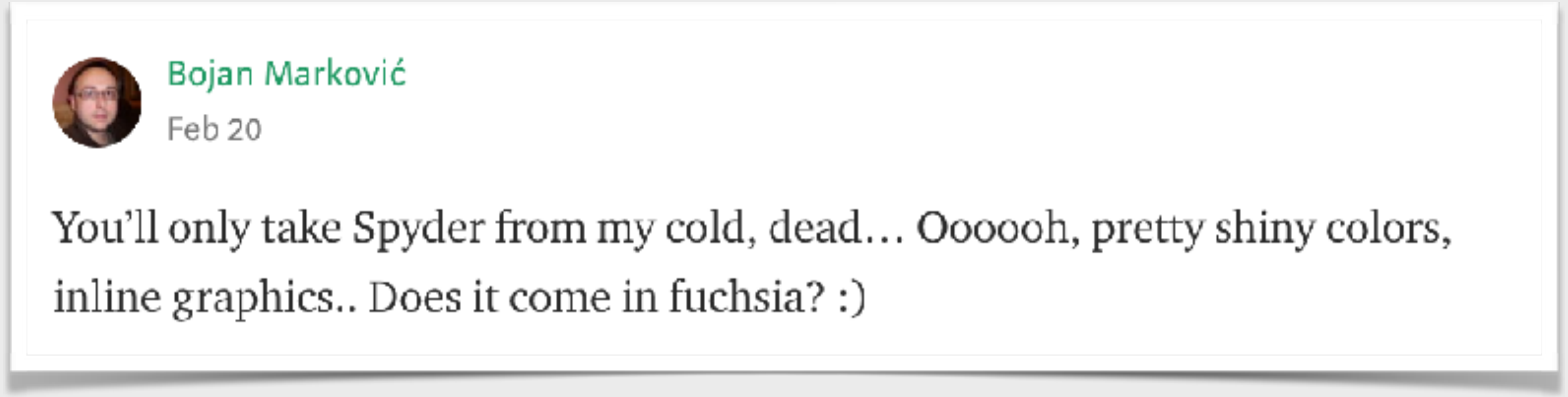
Separation of states

- Computation, narrative/visualisation in different processes.
  - Robust to crashes
  - Can "Share" and analysis / notebook without having to “rerun"
  - Trustworthy (No copy-past issues).
- Cons:
  - Understanding that document/kernel can have different states can be challenging.
  - Notebook format is not as widespread as others.

Popularity

Network enabled / web based

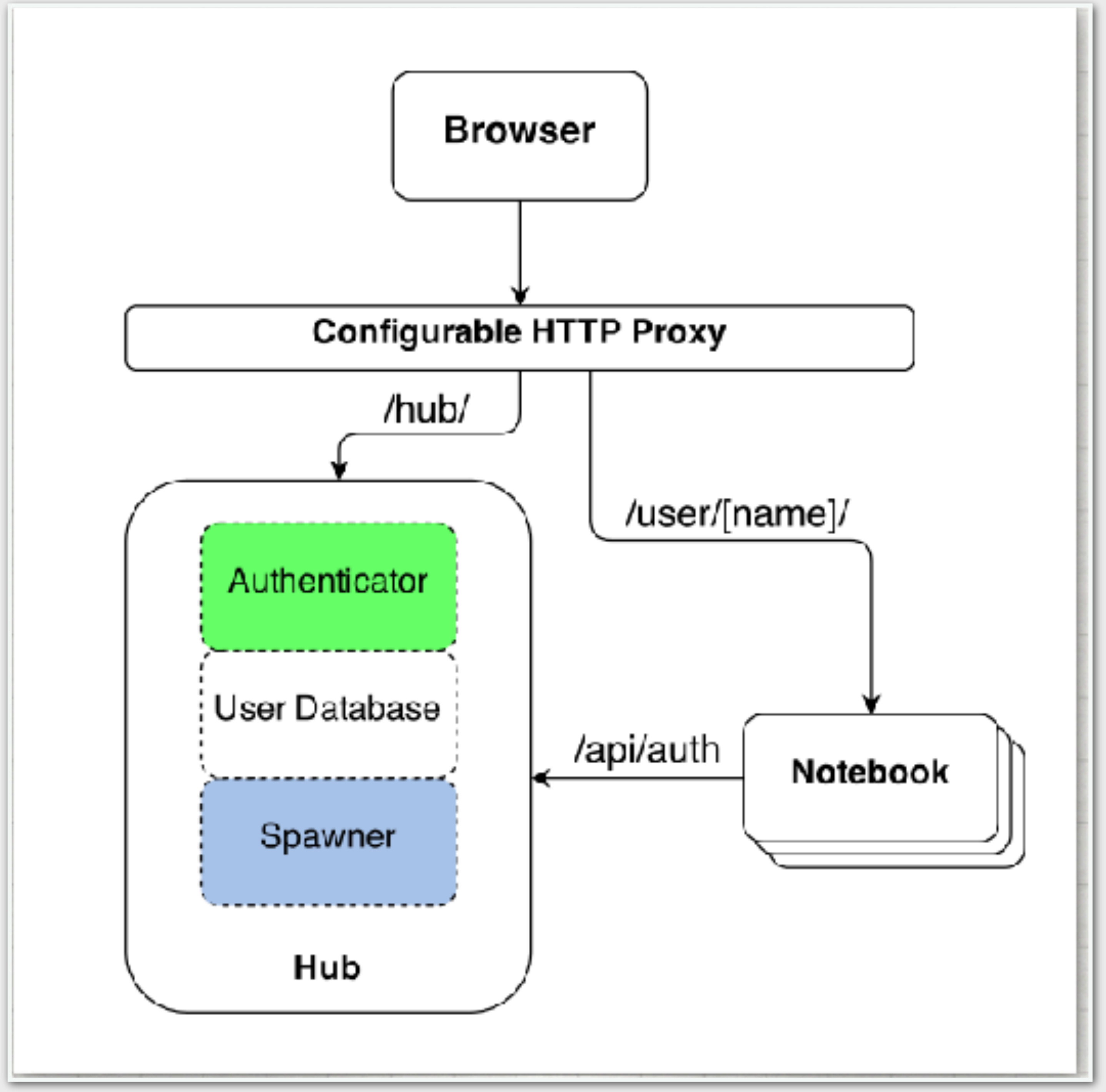
- User love fancy colors and things moving. Using D3 and other



  - dynamic libraries are highly popular
  - Usable by novices and power-users
  - Users w/ different expertise (Numerical Methods, Visualization,...)
- Seamless transition to HPC: Kernel Menu > Restart on Cluster
- Document persist if code crash.
- Can be Zero-Installation (See JupyterHub).
  - A web browser is all you need.

JupyterHub

- Multi-users Jupyter deployment
  - Not (Yet) Realtime collaboration
- Each user can get their own process/version(s)/configuration(s)
  - Hooks into any Auth
  - Only requires a browser
- Not limited to running Jupyter (e.g. work with RStudio, OpenRefine...)

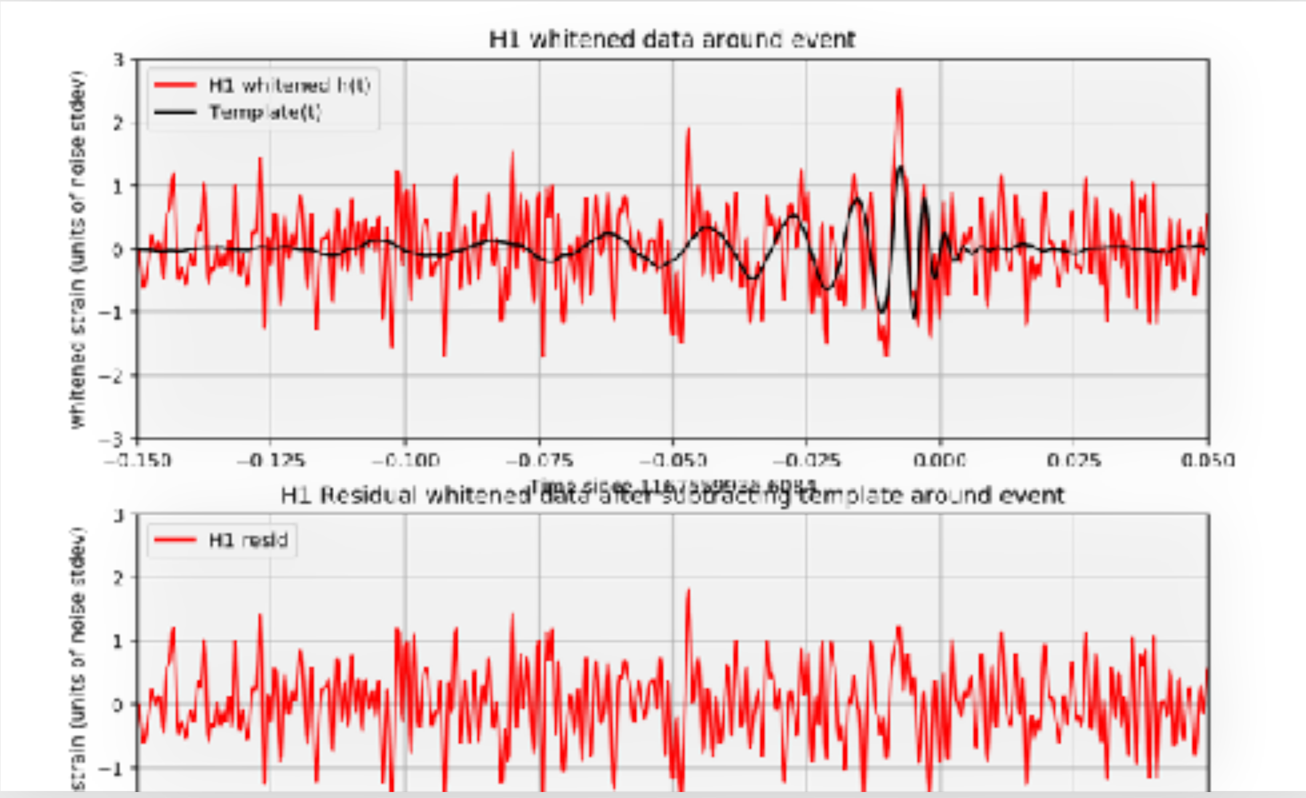


Use Cases

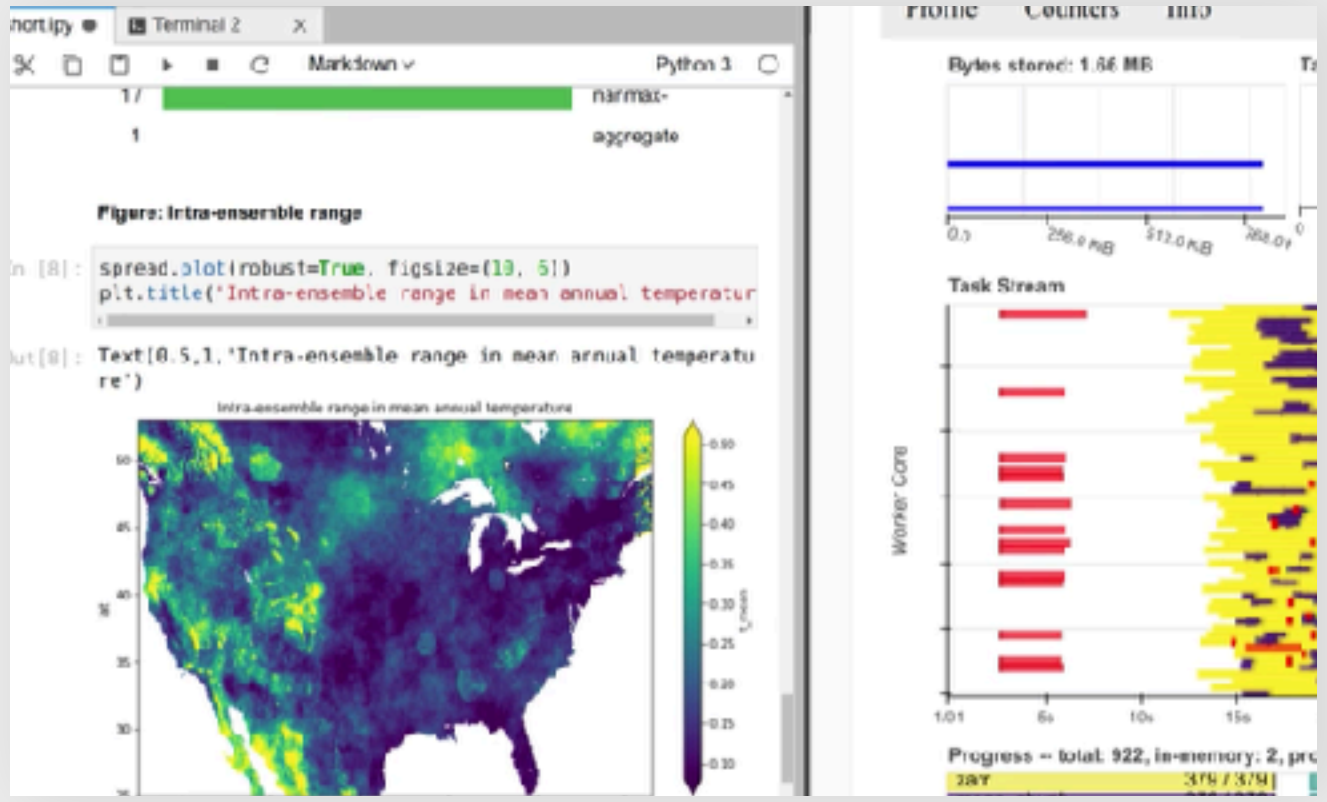
HPC

- Batch Jobs
  - You can run notebook "headless"
  - Parametrized notebook as "reports" you can interact with later
- Interactive Cluster.
  - Run a Hub (hook into LDAP/PAM...)
  - Run notebook servers on a Head node
  - Run Kernels on head Node/fast queue
  - Extra Workers (e.g. dask) on Batch queue/cluster.

Some Jupyter Usage



Ligo

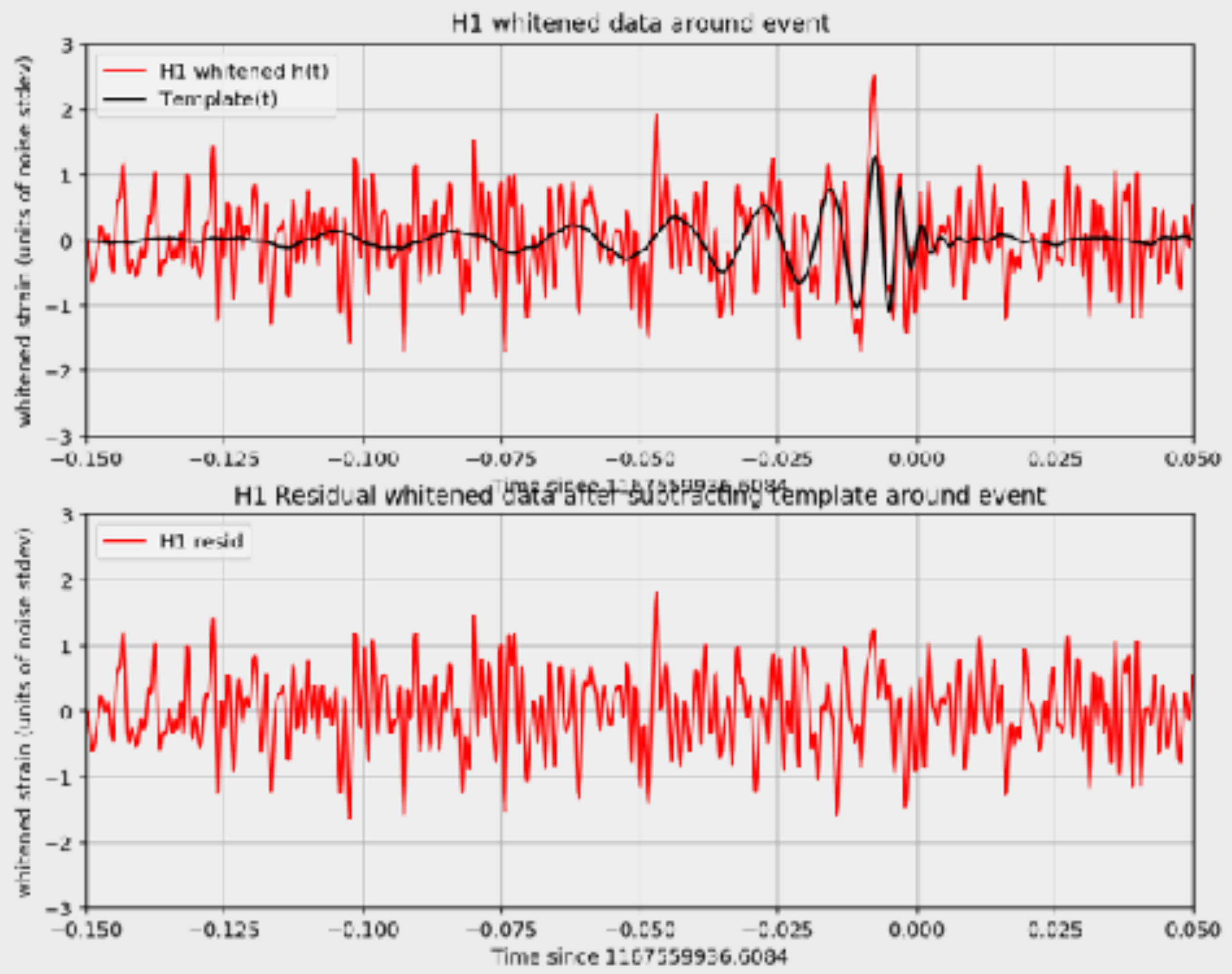


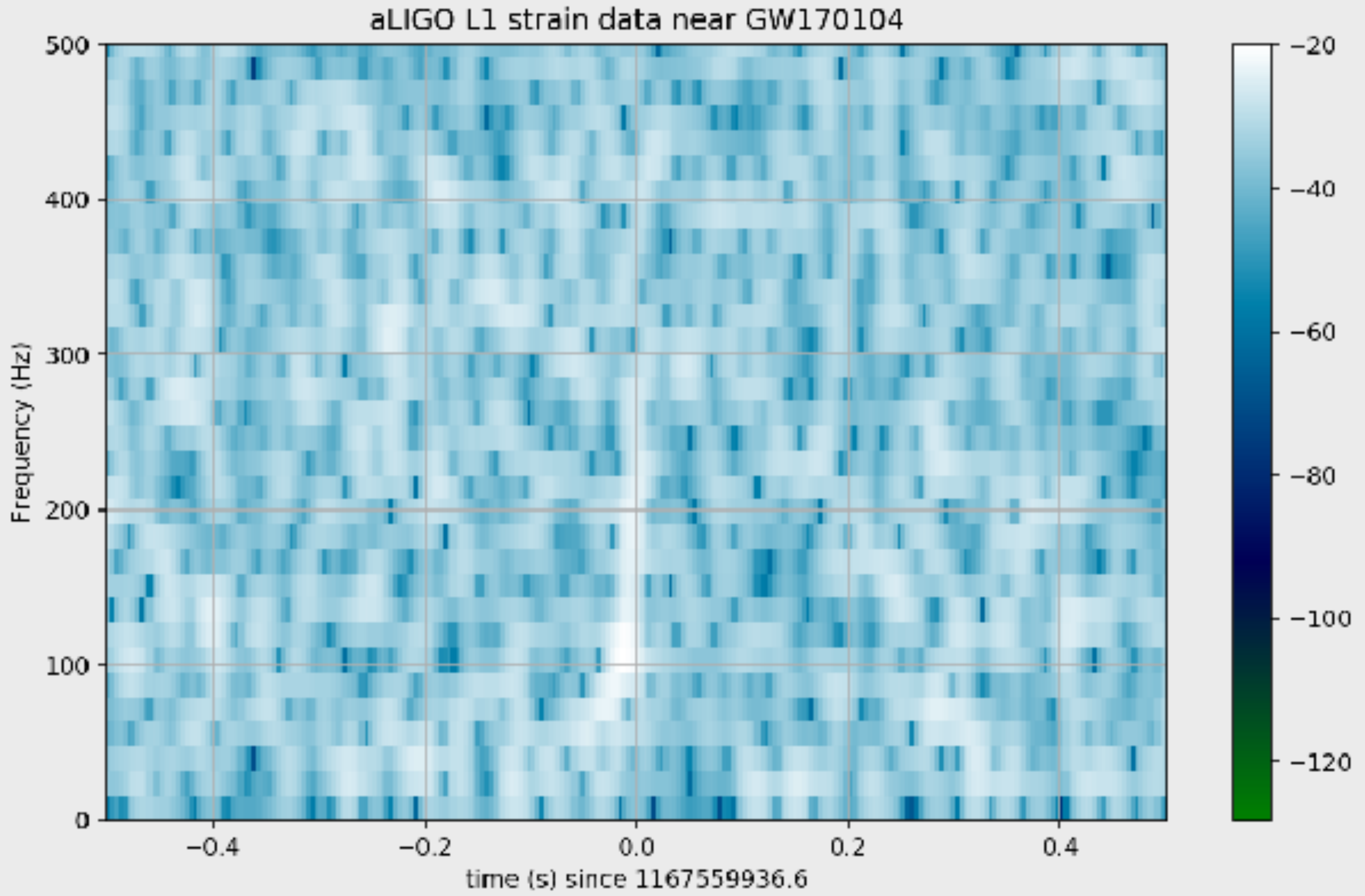
Pangeo

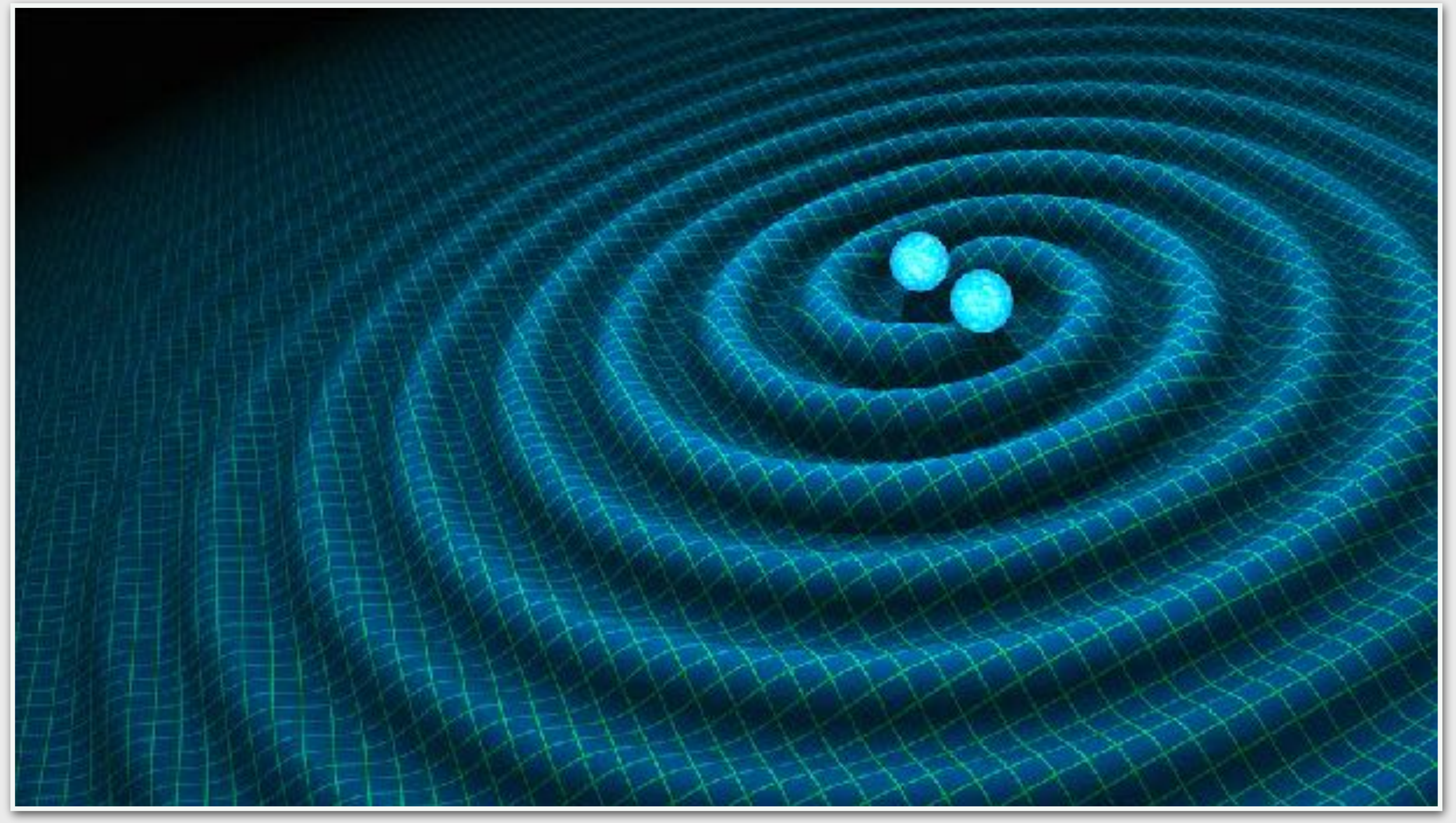
Cern's SWAN

Ligo

- Some events analysis with Jupyter
- Subset of data + env put online
- Run the analysis yourself on Binder[1] and listen to the waves



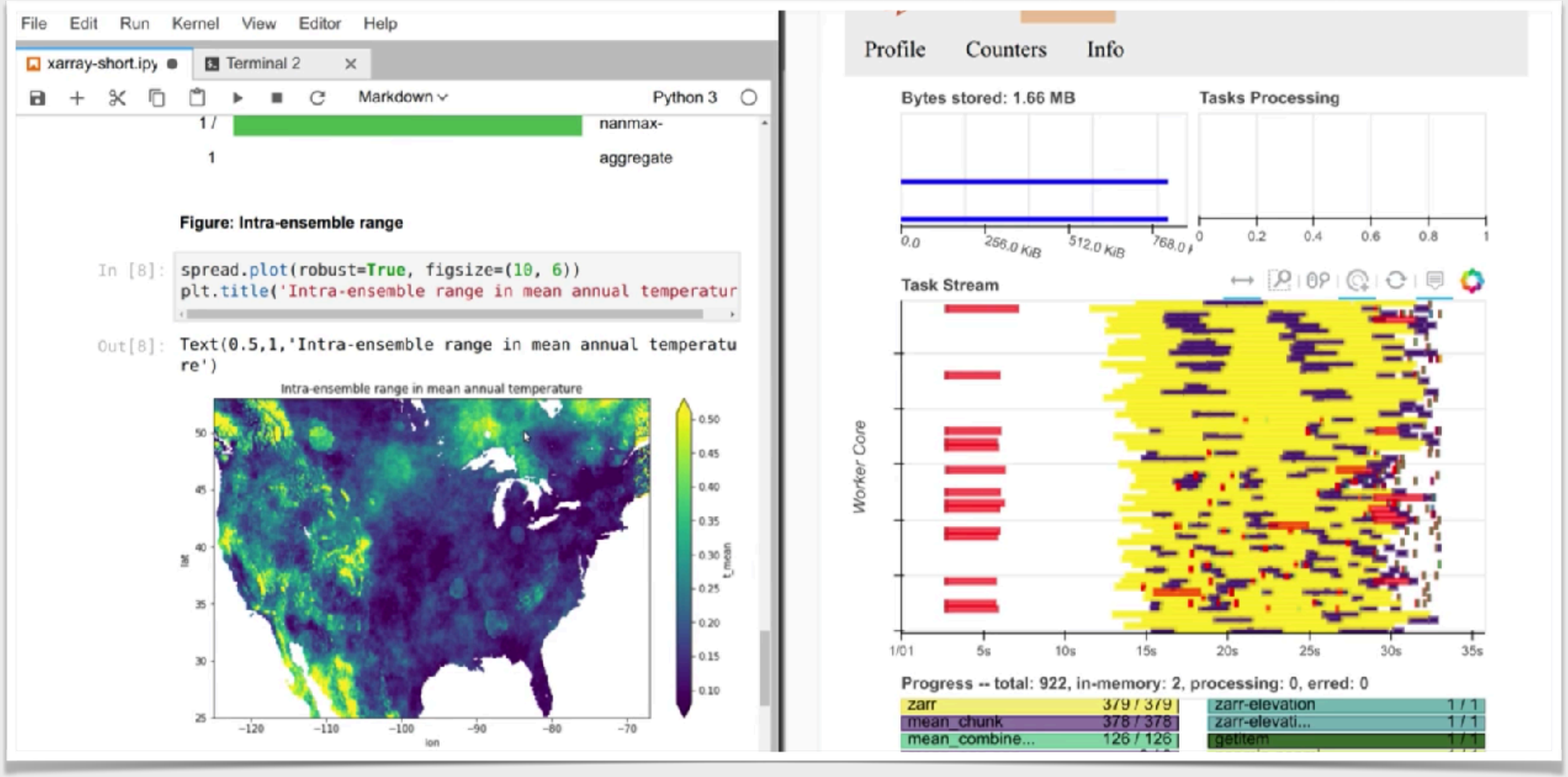




[1] https://github.com/minrk/ligo-binder

Pangeo (pangeo-data.github.io)

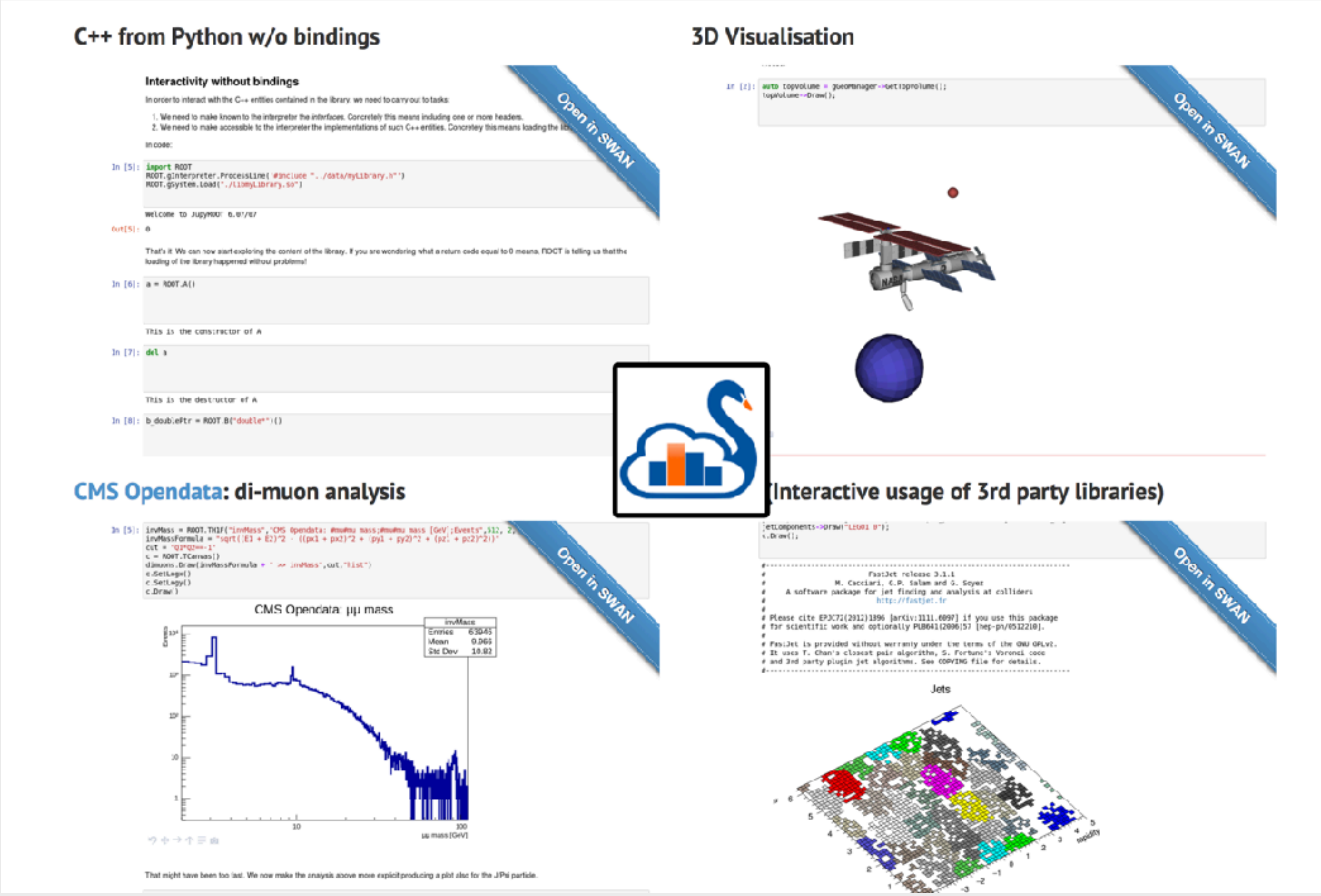
- Effort from Atmosphere / Ocean / Land / Climate (AOC) science community
- unified effort
- Cloud based
- Recent Technologies
  - Dask, Jupyter



Matt Rocklin Blog post on pangeo-data.github.io

Cern Swan (swan.web.cern.ch)

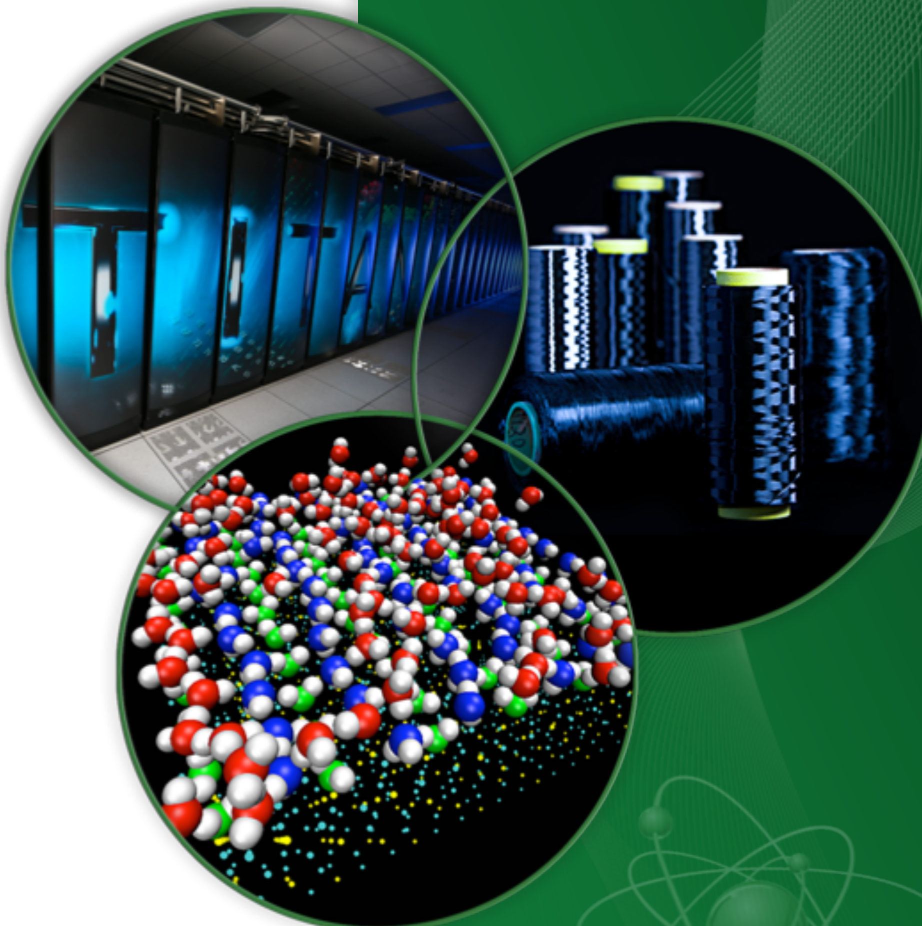
- Share platformed for Data Analysis
- Sync W/ $HOME directory
- 0-install
- Share Data
- Provide example gallery with 1-click-fork



CFP- Ends March 6th

Question(s)
while we change
speakers ?

Jupyter for Supporting a Materials Imaging User Facility (and beyond)



Suhas Somnath

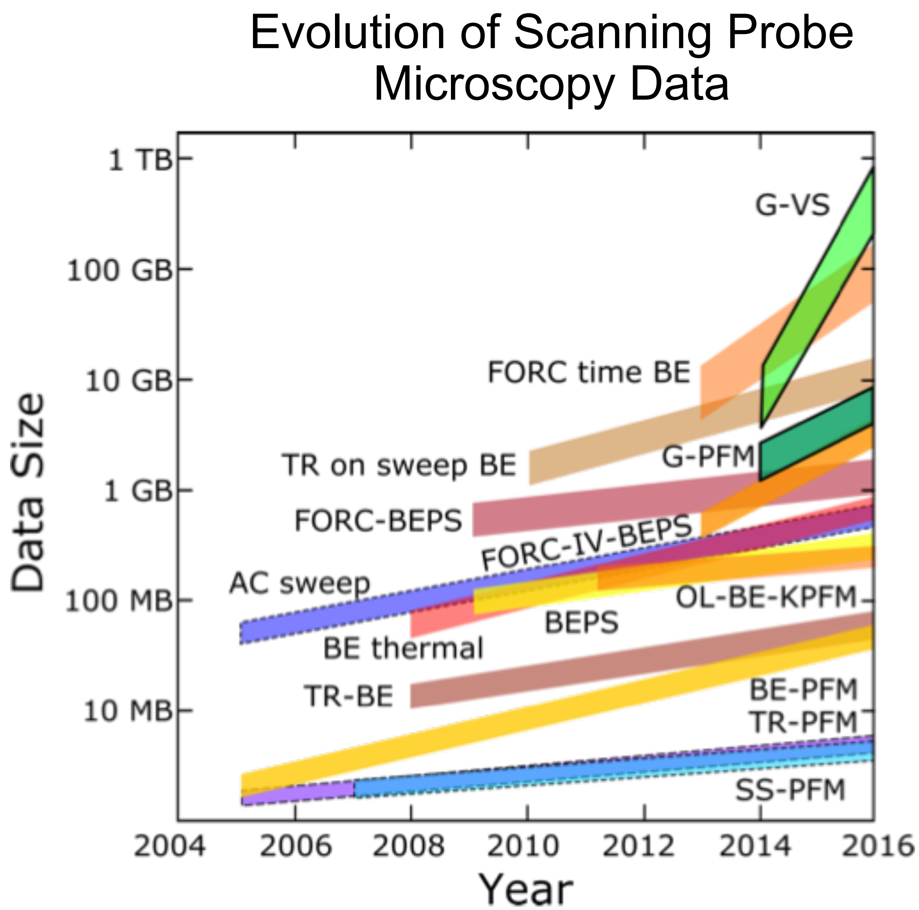
Advanced Data and Workflows Group,

Oak Ridge Leadership Computing Facility

Opportunities in Computing

- Numerical simulations already very popular
- Data analytics is growing
  - Plenty of simulation data
  - Numerous analytics software including ORNL's own:
    - Parallel Big Data with R (pbdR)
    - Spark on Demand ….
- Experimental / Observational data:
  - Few large / mature facilities already invested in analytics
  - Plenty of opportunities in other facilities too
    - Case Study – Imaging / Microscopy / Materials characterization
- Enough information-rich, structured, observational data to complete simulation-experiment feedback loop

Opportunities in Microscopy



- **Growing data sizes & dimensionality**
  - Cannot use desktop computers for analysis

Kalinin et al., *ACS Nano*, 9068-9086, 2015

- **Multiple file formats**
  - Multiple data structures
  - Incompatible for correlation
- **Disjoint and unorganized communities**
  - Similar analysis but reinventing the wheel
  - Norm: emailing each other scripts, data
- **No proper analysis software**
  - Instrumentation software is woefully inadequate
  - No central repository, version control
- **Closed Science**
  - Analysis software, data not shared
  - No guarantees on reproducibility

From 0 to Data Exploration on HPC

Instrument Tier

Data ready for interactive visualization + analysis on HPC

From 0 to Data Exploration on HPC

Instrument Tier

Automated + standardized + modularized data acquisition

Instrument-independent + self-describing data formatting

Centralized hub / repository for data pre-processing, analysis

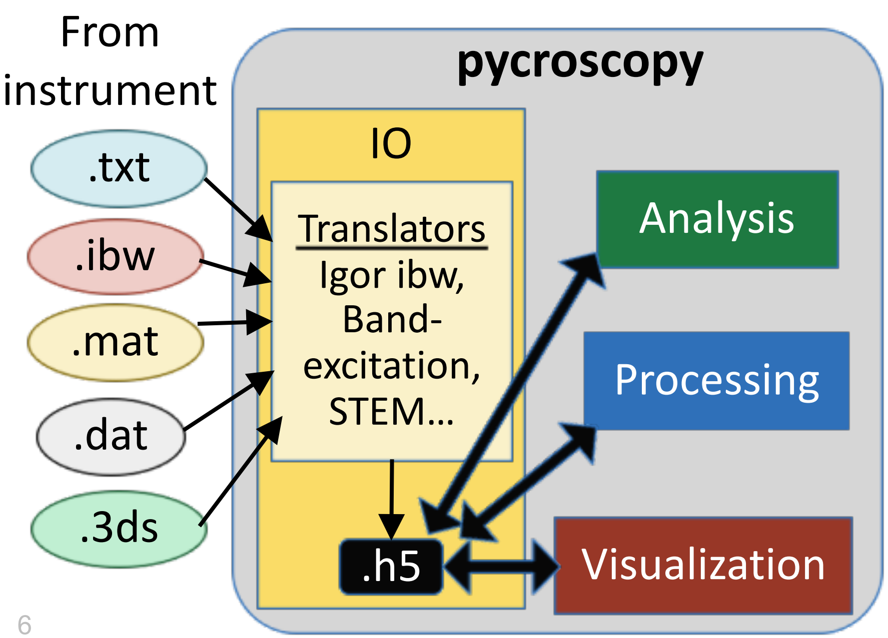
Data ready for interactive visualization + analysis on HPC

Pycroscopy

Open-source python package for analyzing + formatting microscopy data

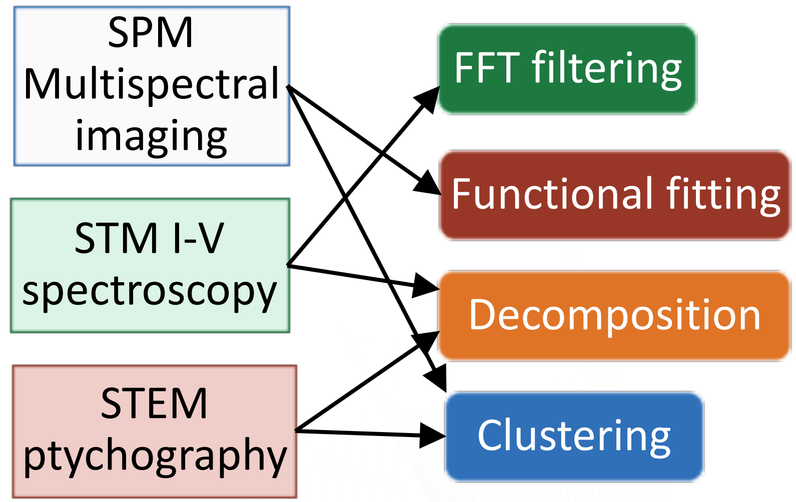
Universal Data Format

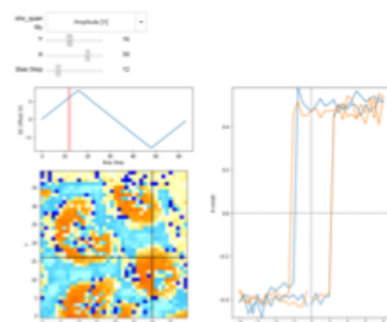
- Instrument-independent format
- HDF5 files for scalable storage
- HDF5 hierarchical structure leveraged for traceability



Instrument agnostic code

- Single version of (reusable) analysis routine
- Brings multiple microscopy fields together





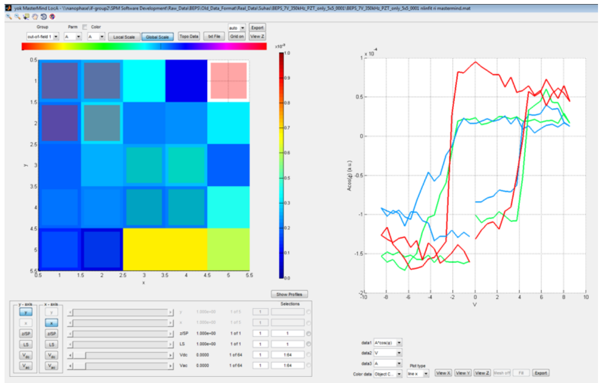
Conveying information

- Interactive jupyter notebooks

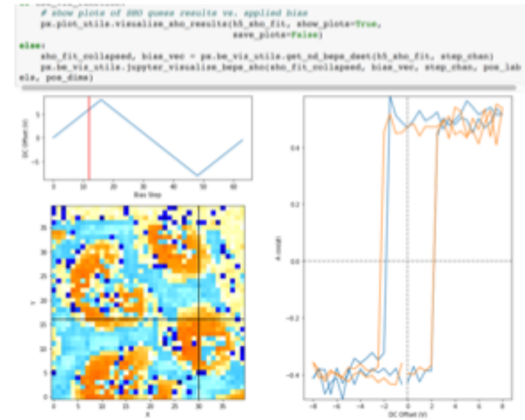
Supporting User Research

Before 2016

+



Since 2016



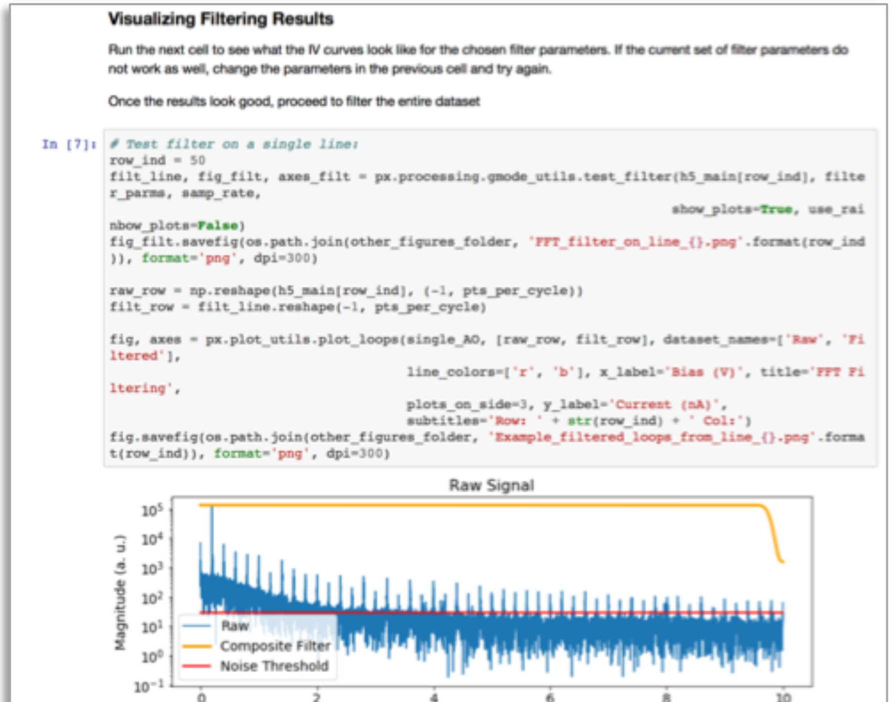
Scripts + complicated, monolithic, Matlab GUI	Set of simple Jupyter notebooks
Witten by dedicated software engineer	Written by material scientists
Not customizable on-the-fly	Completely customizable.
2-3 hours of training before use	Instructions embedded within notebook. NO training required!
Deployed only on two offline workstations due to licensing restrictions = queue	Each user gets VMs with jupyter notebook server
Will remain on off-line desktops	In the process of switching to computations on clusters, and then HPC

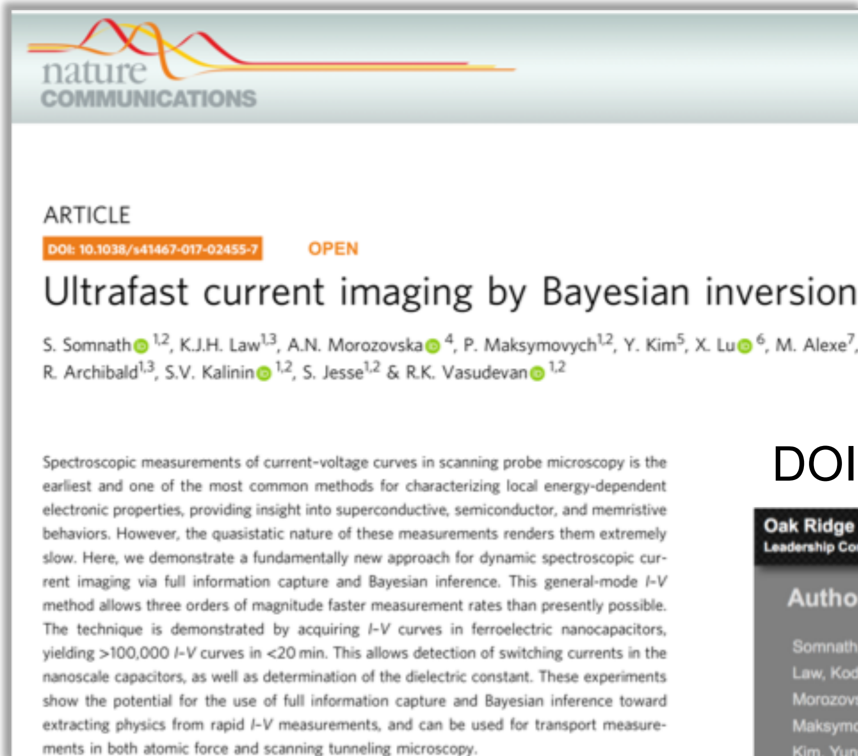
Truly Achieving Open Science, Reproducibility

Aim – ALL scientific journal papers accompanied with:

- Jupyter notebook that shows all analysis (raw data → figures).
- Data with DOI number

Jupyter notebook associated with paper





nature COMMUNICATIONS

ARTICLE

DOI: 10.1038/s41467-017-02455-7 OPEN

Ultrafast current imaging by Bayesian inversion

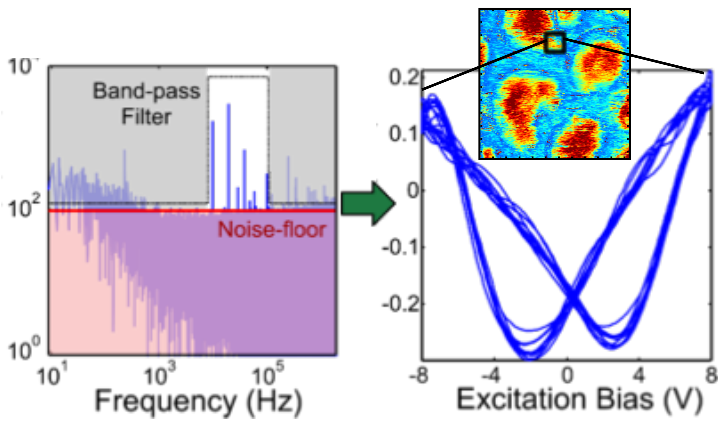
S. Somnath[1,2], K.J.H. Law[1,3], A.N. Morozovska[4], P. Maksymovych[1,2], Y. Kim[5], X. Lu[6], M. Alexe[7], R. Archibald[1,3], S.V. Kalinin[1,2], S. Jesse[1,2] & R.K. Vasudevan[1,2]

Spectroscopic measurements of current-voltage curves in scanning probe microscopy is the earliest and one of the most common methods for characterizing local energy-dependent electronic properties, providing insight into superconductive, semiconductor, and memristive behaviors. However, the quasistatic nature of these measurements renders them extremely slow. Here, we demonstrate a fundamentally new approach for dynamic spectroscopic current imaging via full information capture and Bayesian inference. This general-mode I-V method allows three orders of magnitude faster measurement rates than presently possible. The technique is demonstrated by acquiring I-V curves in ferroelectric nanocapacitors, yielding >100,000 I-V curves in <20 min. This allows detection of switching currents in the nanoscale capacitors, as well as determination of the dielectric constant. These experiments show the potential for the use of full information capture and Bayesian inference toward extracting physics from rapid I-V measurements, and can be used for transport measurements in both atomic force and scanning tunneling microscopy.

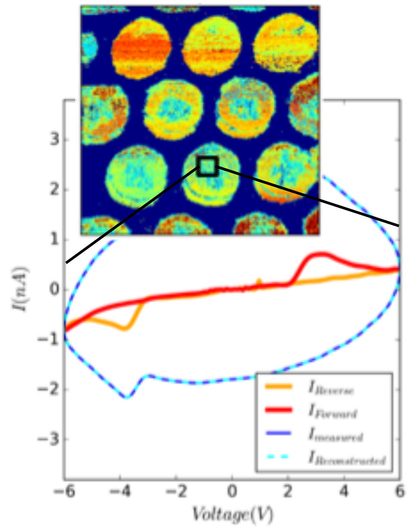
DOI associated with data (raw → paper figures)

Scientific Advancements with Jupyter

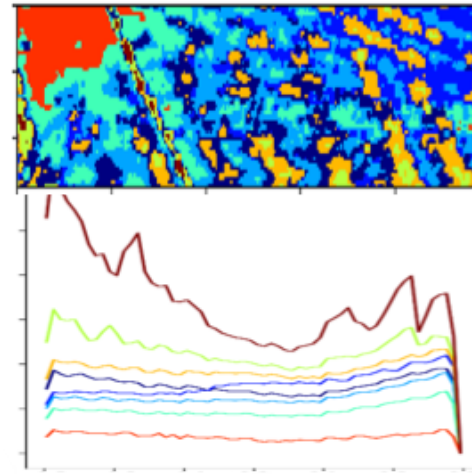
3,500x faster imaging via adaptive signal filtering, linear unmixing of signals



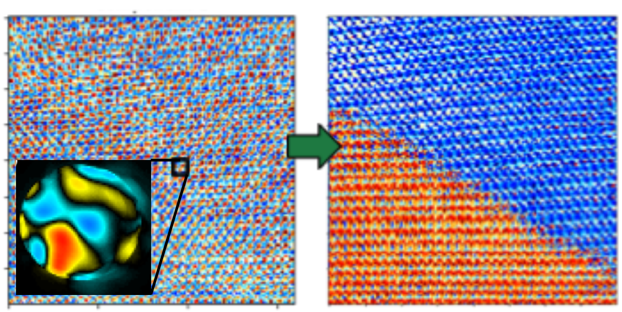
200x faster spectroscopy via Bayesian inference



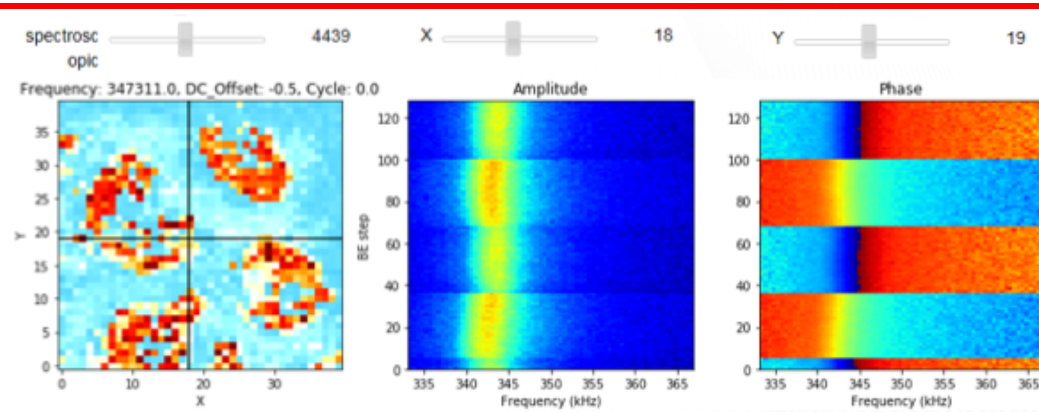
Denoising and clustering to identify superconductivity at the nanoscale



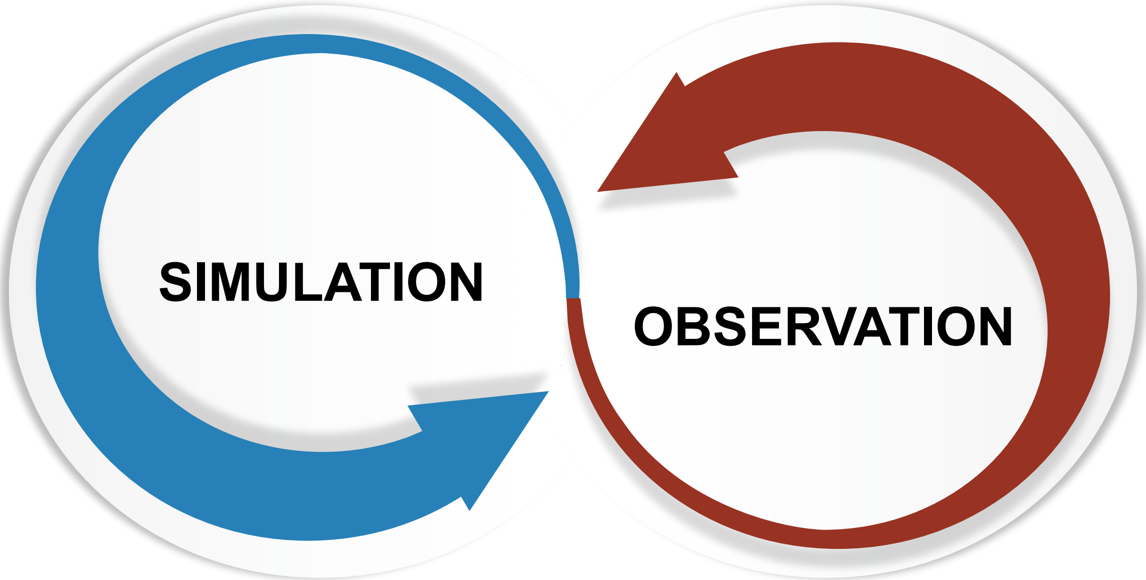
Identifying invisible patterns using multivariate analysis



Simplified navigation multidimensional data - users



Completing a Discovery Paradigm



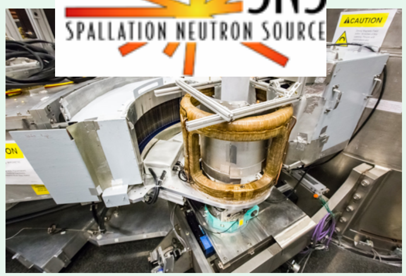
Enough information-rich, well-structured, observational data to complete simulation-experiment feedback loop

Scaling this approach to the lab

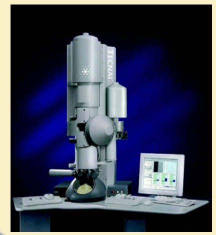
Institute for Functional Imaging of Materials

pycroscopy

SNS SPALLATION NEUTRON SOURCE



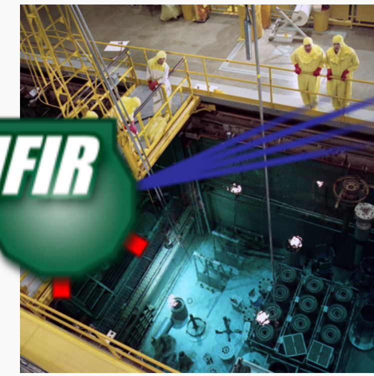
Electron Microscopy



pyEM ?

….

HFIR



jupyterhub

CADES

(Cloud + Cluster)

summit

OAK RIDGE National Laboratory | OAK RIDGE LEADERSHIP COMPUTING FACILITY

Acknowledgements

Pycroscopy Team:
- Stephen Jesse
- Chris R. Smith

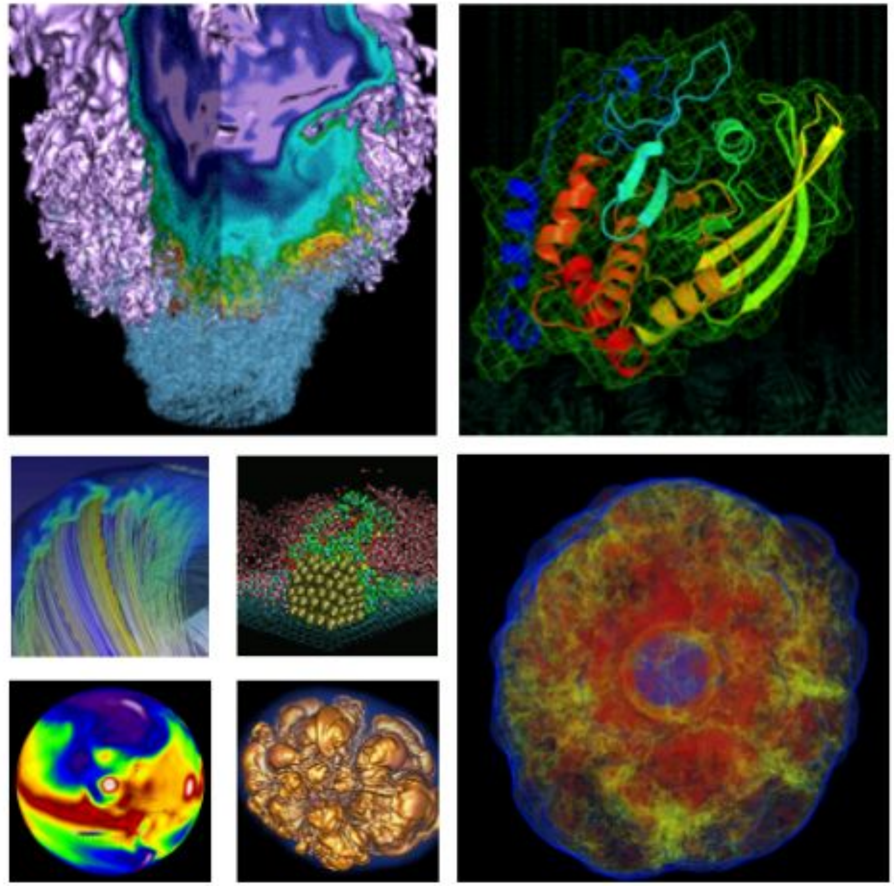
Analytics Team:
- Junqi Yin
- Arjun Shankar

IFIM members:
- Sergei V. Kalinin
- Stephen Jesse
- Rama K. Vasudevan

CADES Group:
- OpenStack team
- SHPC Condo team
- Arjun Shankar

Jupyter @ NERSC

Tales From a Supercomputing Center



Shreyas Cholia, Rollin Thomas, and Shane Canon

IDEAS Webinar
February 28 2018

Cori: Friendly for “Data Users”

Gerty Cori: Biochemist and first American woman to win a Nobel Prize in science

- Two architectures in one system:
  - **Data** 2388 nodes 32-core Intel Xeon “Haswell” 128 GB DDR4
  - **HPC** 9688 nodes 68-core Intel Xeon Phi “KNL” 96 GB DDR4 + 16 GB MCDRAM
- Haswell login and **special-purpose large memory nodes** (512 & 768 GB)
- NVRAM Burst Buffer for IO acceleration
- Shared and real-time queues
- Shifter for containerized HPC

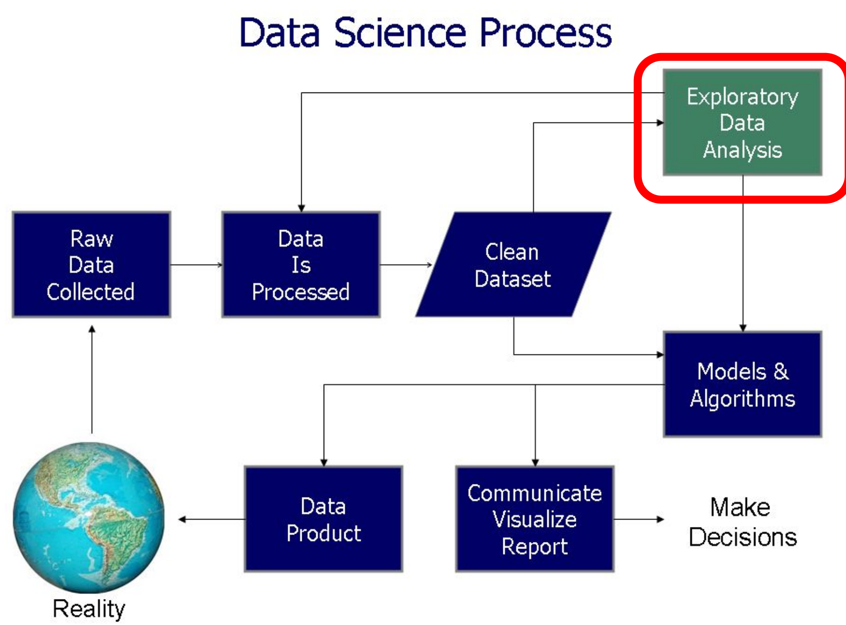
slurm workload manager

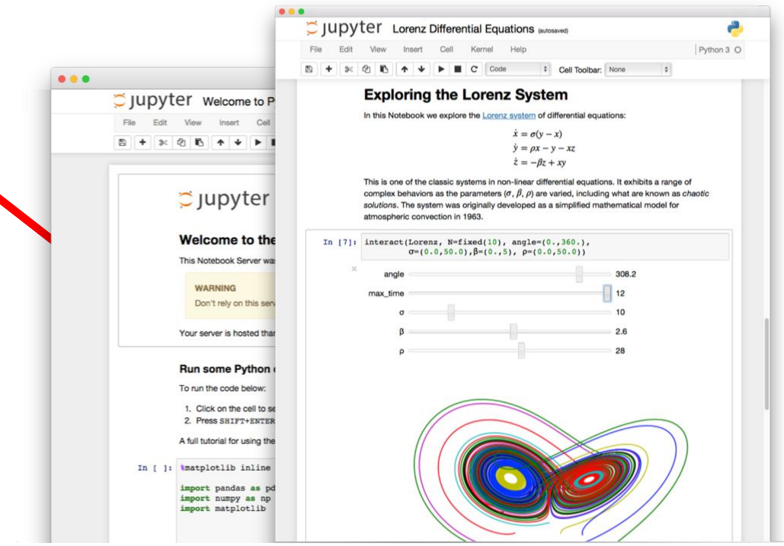
SHIFTER

U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

Enter Jupyter

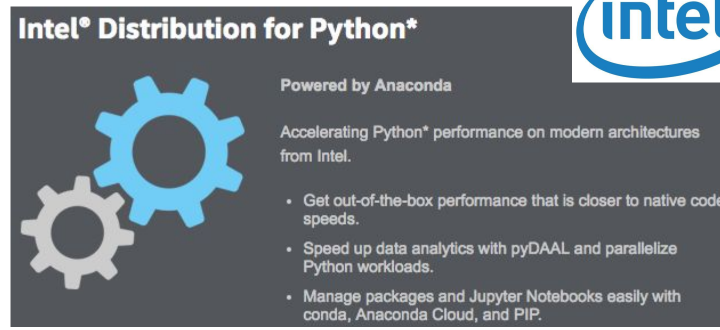




- **Jupyter Notebooks: *Literate Computing, "Narratives"***
  - **Code and comments: Reproducibility, show your work! Document your workflow**
  - **Rich text, plots, equations, widgets, etc.**
  - **Iterate and explore to arrive at meaningful insights**

Diagram courtesy of "Farcaster" at English Wikipedia

Central Role of Python at NERSC



Python is the most popular language at NERSC used to:

- **Script workflows for both data analysis and simulations**
- **Perform exploratory data analysis**

Motivation For Jupyterhub Service

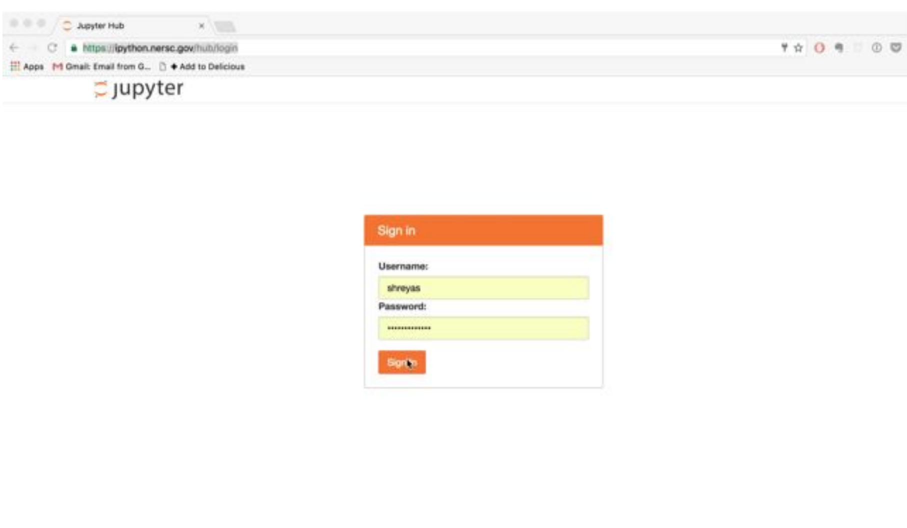
- × Users running their own notebook servers on a supercomputer makes security folks very nervous.
- × Difficult to support and manage different kernels and environments

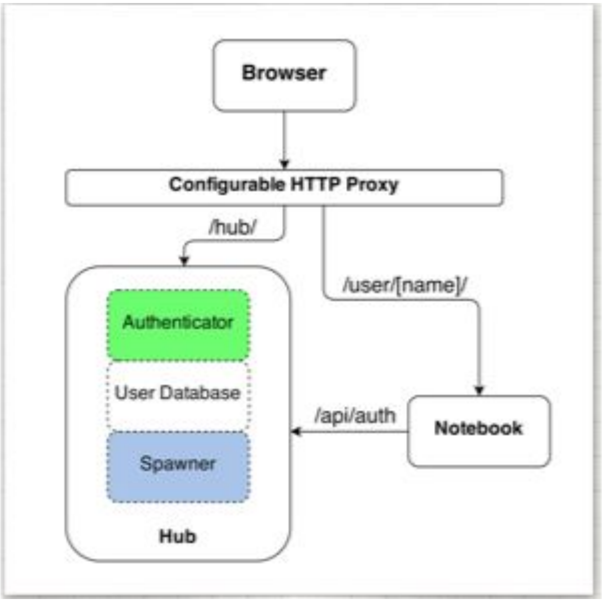
Jupyterhub to rescue

- ✓ Centralized service to deploy notebooks in a standard authenticated manner
- ✓ Package known kernels out of the box (Anaconda)
- ✓ Access to NERSC resources through this interfaces
    - Filesystems, Batch Queue, Network, DBs

Jupyterhub: Jupyter as a Service

- Service to deploy notebooks in a multi-user environment
- Manages user authentication, notebook deployment and web proxies

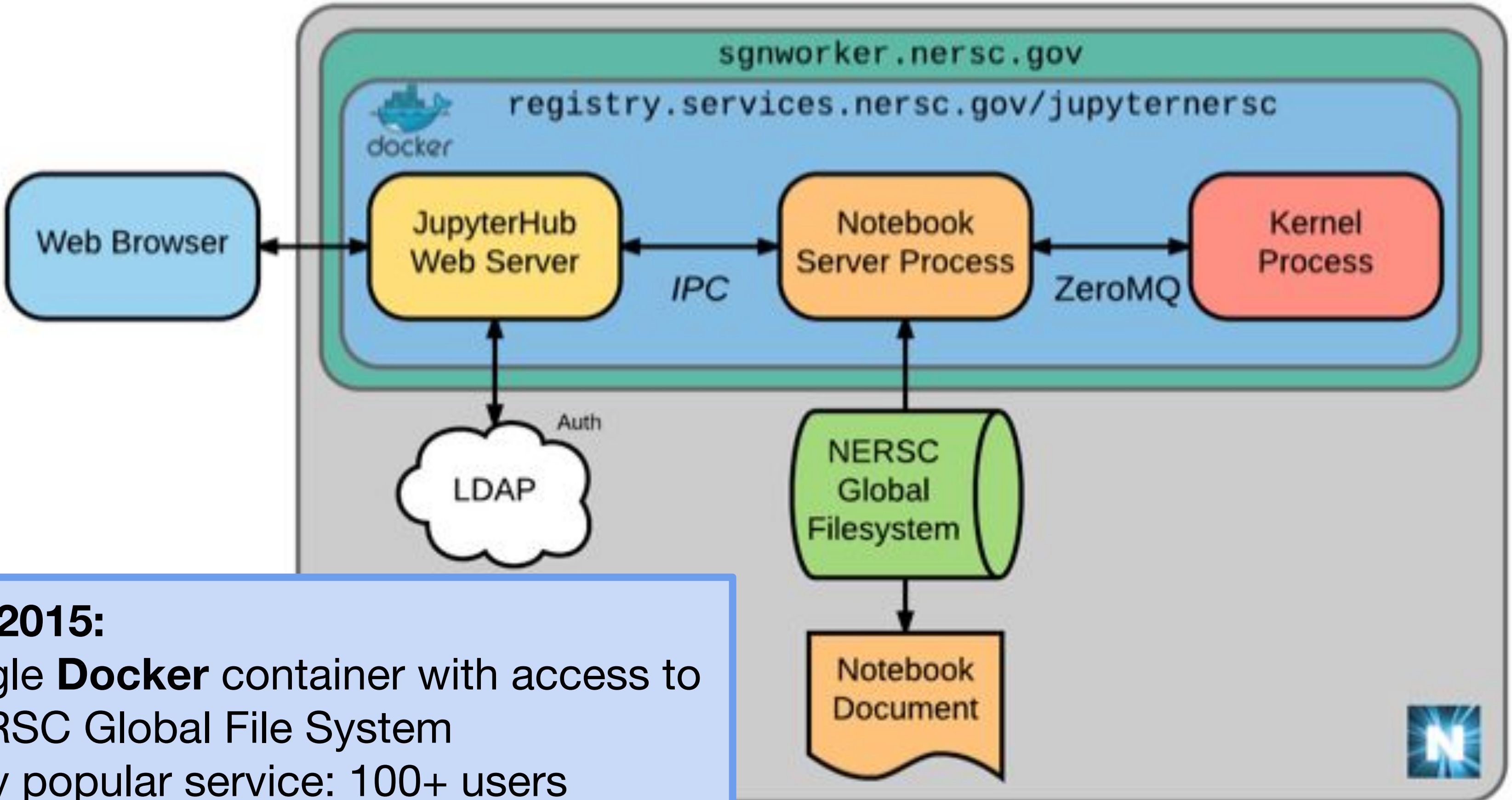




Jupyter@NERSC
Evolution of Architecture

Step 1: Give people access to their data

First Architecture: “Edge Service”



August 2015:

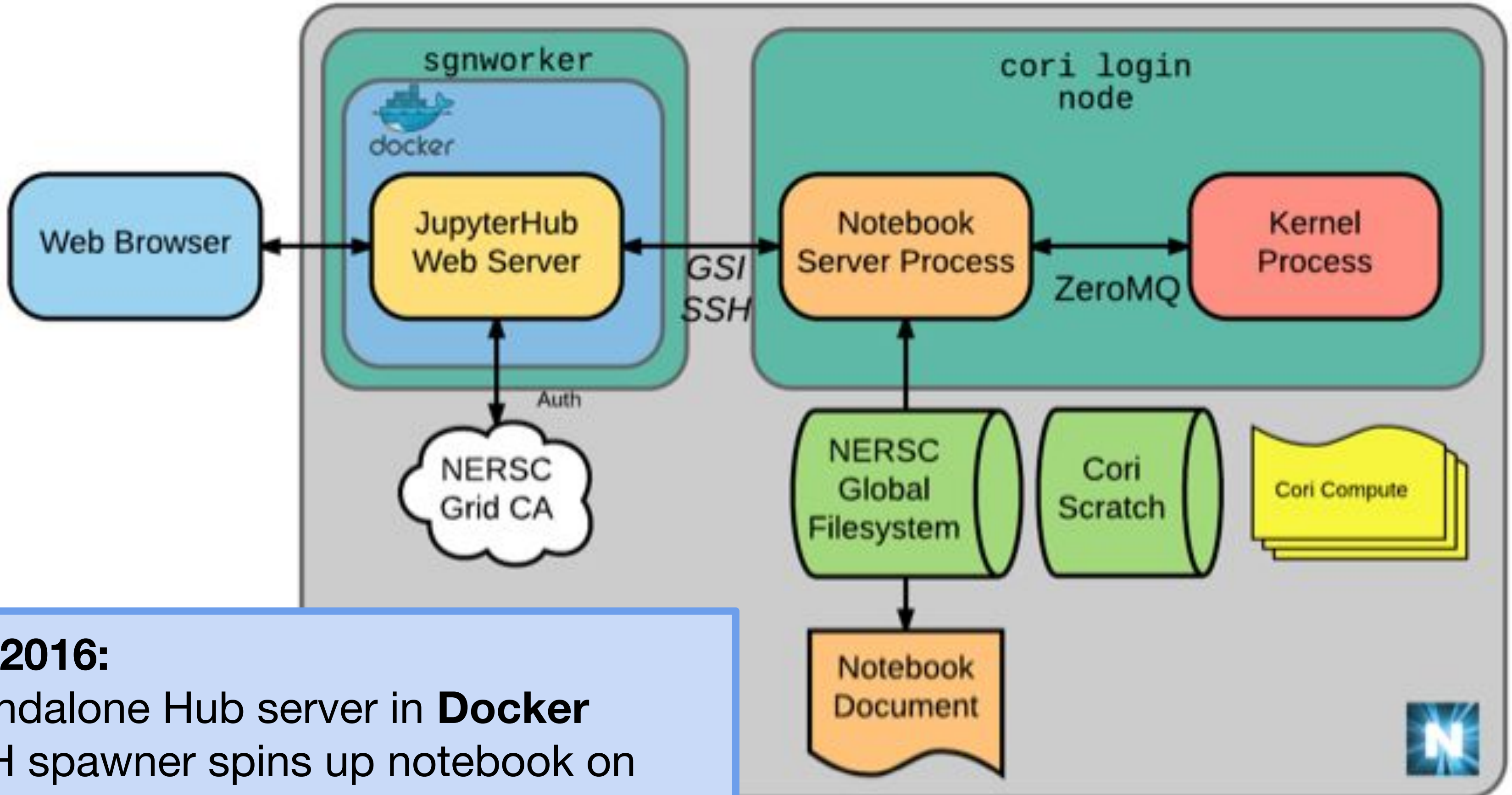
- Single **Docker** container with access to NERSC Global File System
- Very popular service: 100+ users
- Missing:
  - Access to Cori Lustre Scratch
  - Interactivity with Cori batch queues
  - Cori Python environment.

Projects:
OpenMSI
Metabolite Atlas
LUX
...

Jupyter@NERSC
Evolution of Architecture

Step 2: Integration with Cori compute and filesystems

Second Architecture: Cori Login Node



August 2016:

- Standalone Hub server in **Docker**
- SSH spawner spins up notebook on special-purpose Cori login node
- Access to Cori Lustre Scratch
- Same Python environment as Cori login
- Interactivity with batch queues

Projects:
LSST
DESI
MaterialsProject
...

Our Extensions to JupyterHub

`jupyterhub.auth.Authenticator`

↑

`GSIAuthenticator`

https://github.com/NERSC/GSIAuthenticator

- Use MyProxy to login to NERSC CA server with user/pass to get X509 certificate credentials.
- No need to run JupyterHub with additional privileges, or root access.

`jupyterhub.spawner.Spawner`

↑

`SSHSpawner`

https://github.com/NERSC/sshspawner

- SSH to Cori with user's credential. Uses GSISSH, but can use SSH.
- Notebook starts up, spawner goes away, Notebook communicates w/Hub, keep PID.

GSI Authenticator

- User logs in with username and password. Authenticator uses myproxy to login to NERSC CA server with username/password and retrieves credentials (X509 certificate)
- Jupyterhub runs as a standalone service and doesn't need root access. In fact, no root access needed across this architecture.
- https://github.com/NERSC/gsiauthenticator

SSH Spawner

- **We wrote an SSH Spawner that will will SSH into the Cori node with users credential**
  - Supports GSISSH (use with certificates from GSI authenticator)
  - Supports SSH key based auth
- **SSH Spawner starts up notebook server process and goes away; Notebook server communicates directly with hub**
  - No tunnels or persistent connections needed
- **Keep track of the PID for poll and shutdown functions (also via SSH)**
- **Inspired by Andrea Zonca's RemoteSpawner (SDSC)**
- **https://github.com/NERSC/SSHSpawner**

SLURM MAGIC

- Jupyter “%magic” commands:
  - Expose extra-language functionality
  - Outputs are first-class Notebook objects
- Developed wrappers around SLURM commands.
  https://github.com/NERSC/slurm-magic

- `%squeue`
  ```
  %squeue -u rthomas
  ```
- `%sbatch`
  ```
  %sbatch script.sh
  ```
- `%%sbatch`
  ```
  %%sbatch -N 1 -p debug -t 30 -C haswell
  #!/bin/bash
  srun ...
  ```

Enable Custom Kernels

```
{
    "display_name": "HEP",
    "language": "python",
    "argv": [
        "/global/common/cori/software/python/2.7-anaconda/bin/python",
        "-m",
        "IPython.kernel",
        "-f",
        "{connection_file}"
    ],
    "env": {
        "LD_LIBRARY_PATH": "/usr/common/software/root/6.06.06/lib/root",
        "PYTHONPATH" : "/usr/common/software/root/6.06.06/lib/root"
    }
}
```

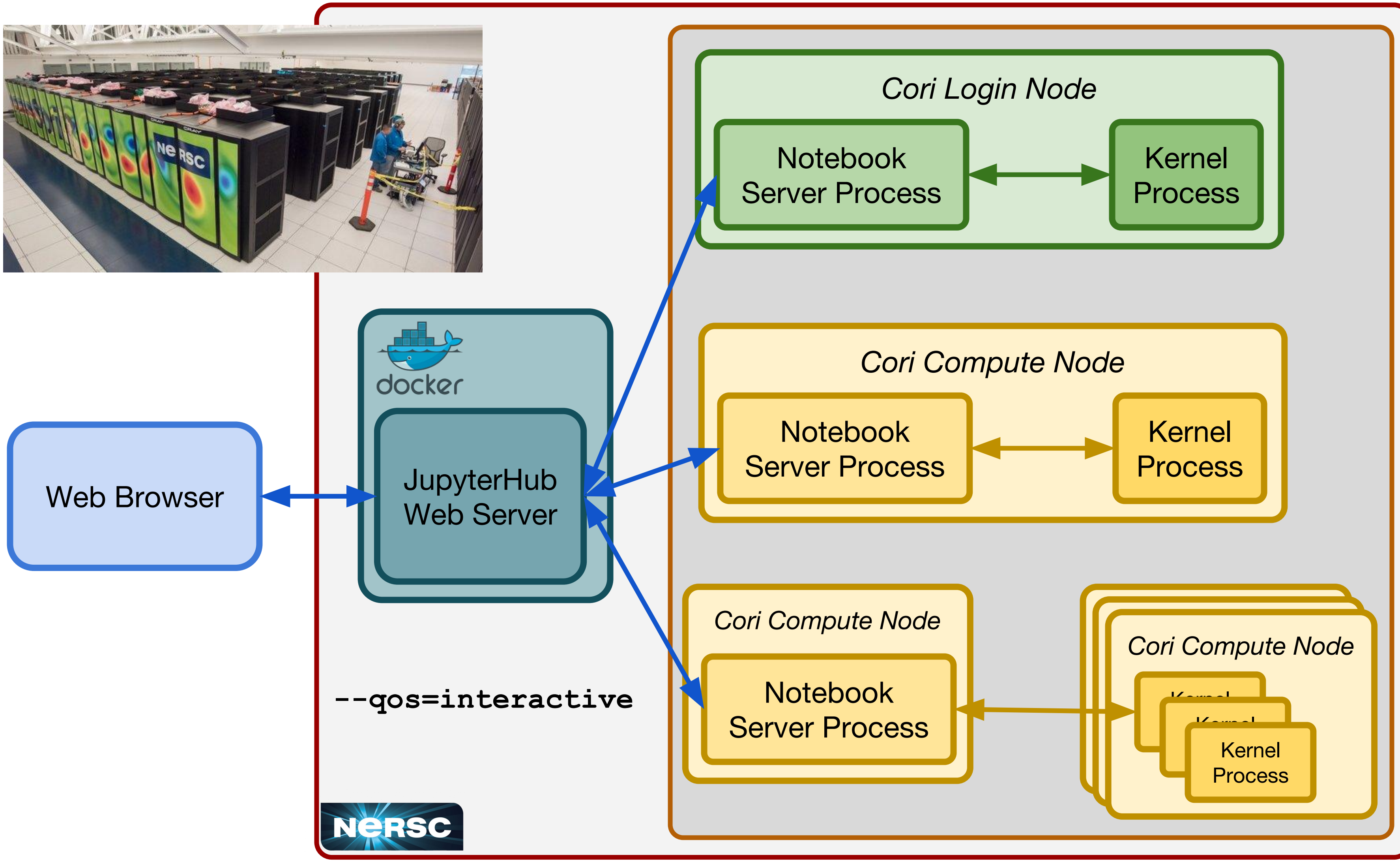
Example PyROOT Kernel Spec

- **Users customize their notebooks with libraries and APIs of their own design or from third parties.**
- **NERSC wants to offer Jupyter to users so they don't set it up themselves in an insecure way.**

Jupyter@NERSC
Evolution of Architecture

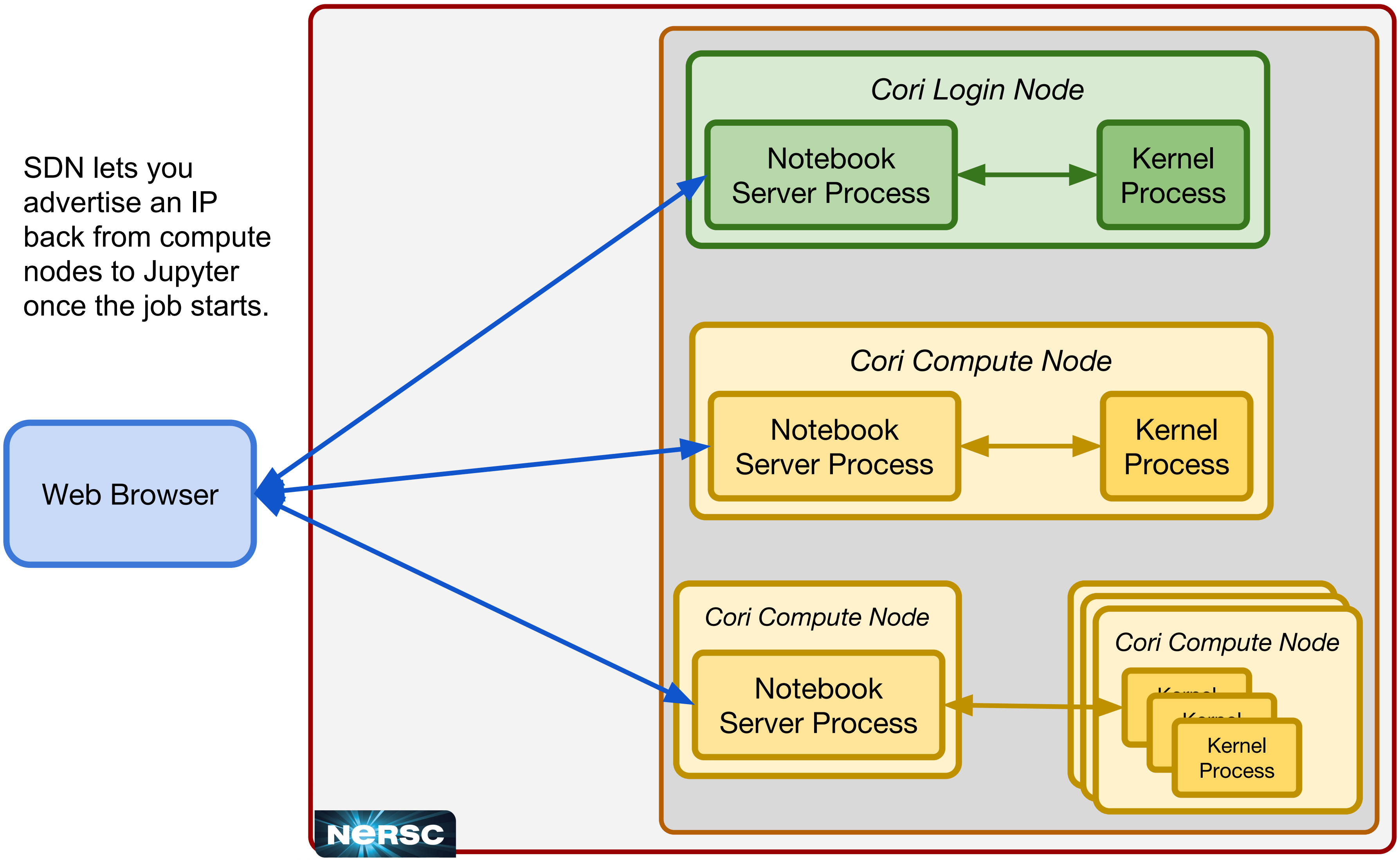
Step 3: The Future

Next: Cori Compute Nodes



Role of Software Defined Networking

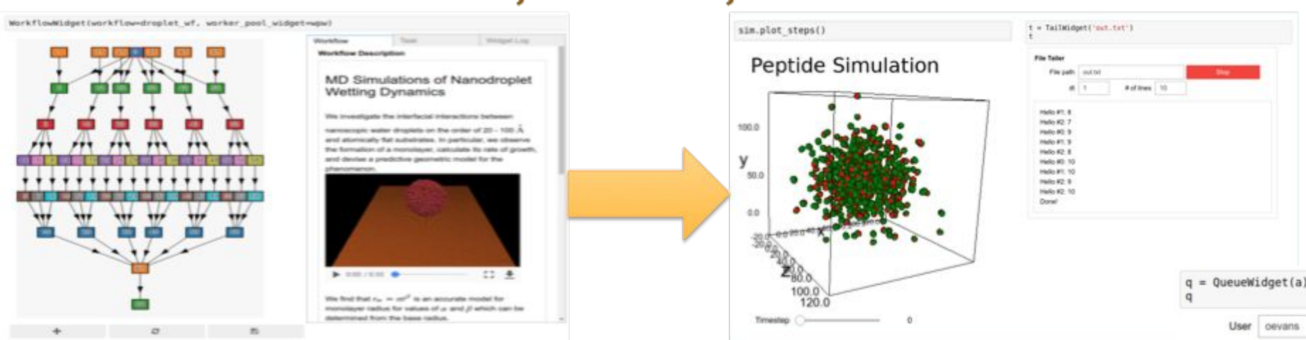
SDN lets you advertise an IP back from compute nodes to Jupyter once the job starts.

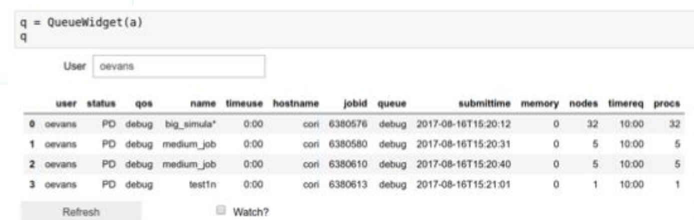


Kale: Human-in-the-loop HPC

Project Kale is a research effort focused on adapting the Jupyter machinery for HPC workflows

View, Control, Monitor





- Master notebook to control workflow
- Jupyter notebooks as **interactive workflow steps**
- Interaction with workflow tasks via kernels
- Realtime Monitoring of HPC jobs and output
- Widgets and dashboards for batch job management

The Ultimate Jupyter@NERSC

Software defined networking

- *Advertise IP of notebook server back to user.*
- *Notebook on login node, kernel on compute.*
- *Notebook+kernel on login, Spark job on computes.*

Leveraging interactive QOS

- *Immediate access to compute up to four hours.*

Docker/Shifter

- *Customize notebook/kernel's environment through containers.*
- *Make larger-scale analytics apps actually start up.*

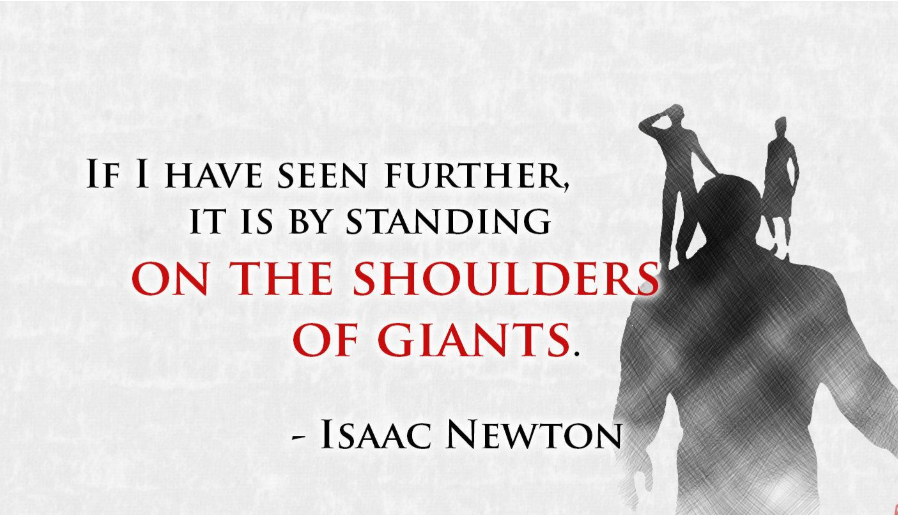
Other possibilities

- *Notebook/scheduler on Haswell, kernels on KNL?*

Acknowledgements

Big Thanks to the Community!

- MSI
- TACC
- SDSC
- Jupyter Dev Team



What Our Users Say

"I'll never have to leave a notebook again … that's like the ultimate dream"