



Jupyter

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)

The image displays two overlapping screenshots of the Jupyter Notebook web interface. The background screenshot shows the 'Welcome to the Jupyter Notebook' page, which includes a warning message: 'WARNING: Don't rely on this server for production use. Your server is hosted that...' and instructions on how to run code in a cell. The foreground screenshot shows a notebook titled 'Exploring the Lorenz System'. It contains the following text: 'In this Notebook we explore the [Lorenz system](#) of differential equations:'. Below this are the Lorenz equations: $\dot{x} = \sigma(y - x)$, $\dot{y} = \rho x - y - xz$, and $\dot{z} = -\beta z + xy$. A paragraph explains that this is a classic system in non-linear differential equations, exhibiting complex behaviors and chaotic solutions. Below the text is a code cell with the following Python code:

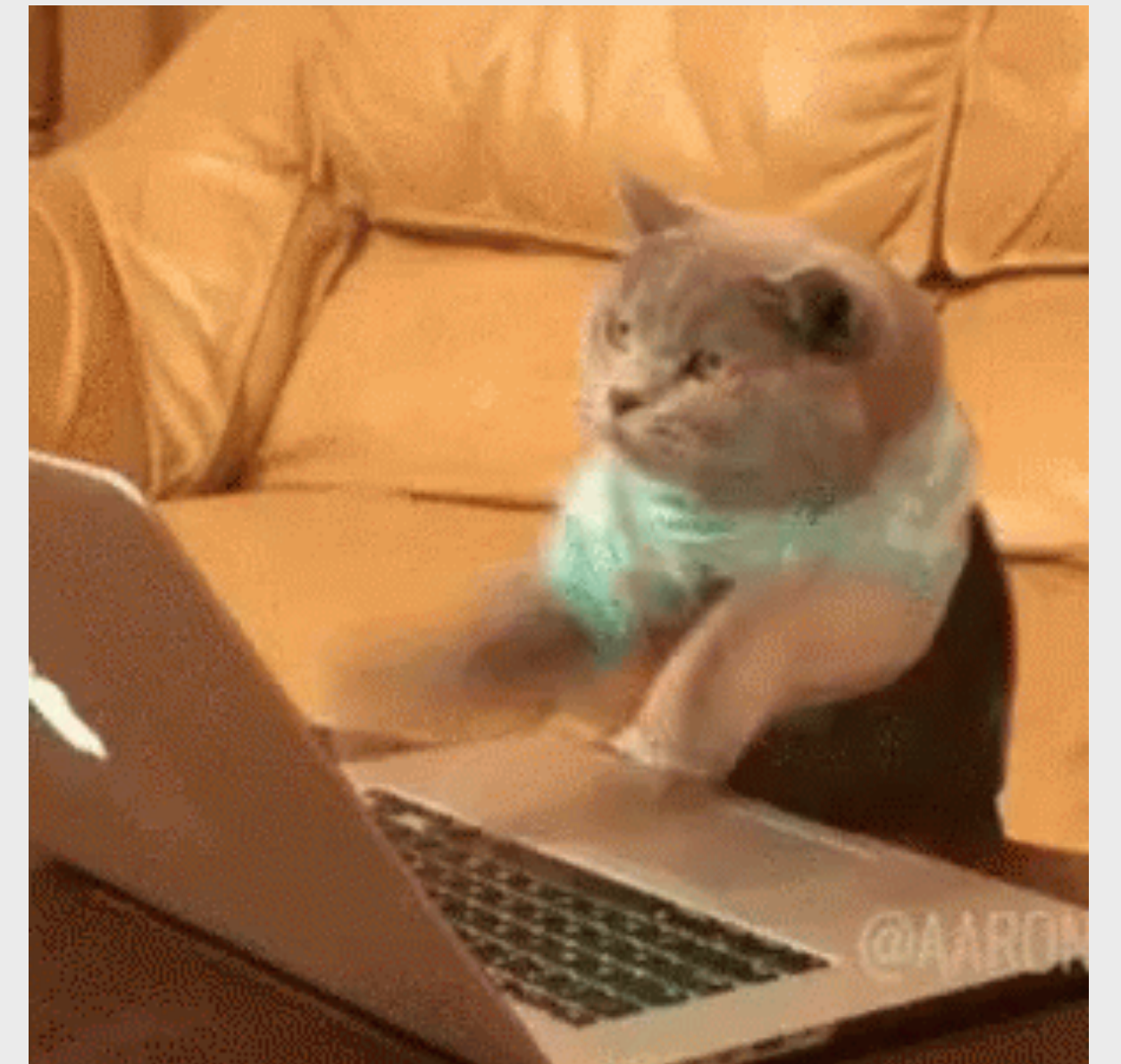
```
In [7]: interact(Lorenz, N=fixed(10), angle=(0., 360.),  
               sigma=(0.0, 50.0), beta=(0., 5), rho=(0.0, 50.0))
```

 The code cell is followed by an interactive plot with sliders for 'angle' (308.2), 'max_time' (12), 'sigma' (10), 'beta' (2.6), and 'rho' (28). Below the plot is a colorful Lorenz attractor plot.



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

NUMFOCUS
OPEN CODE = BETTER SCIENCE

GORDON AND BETTY
MOORE
FOUNDATION



THE LEONA M. AND HARRY B.
HELMSLEY
CHARITABLE TRUST



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.

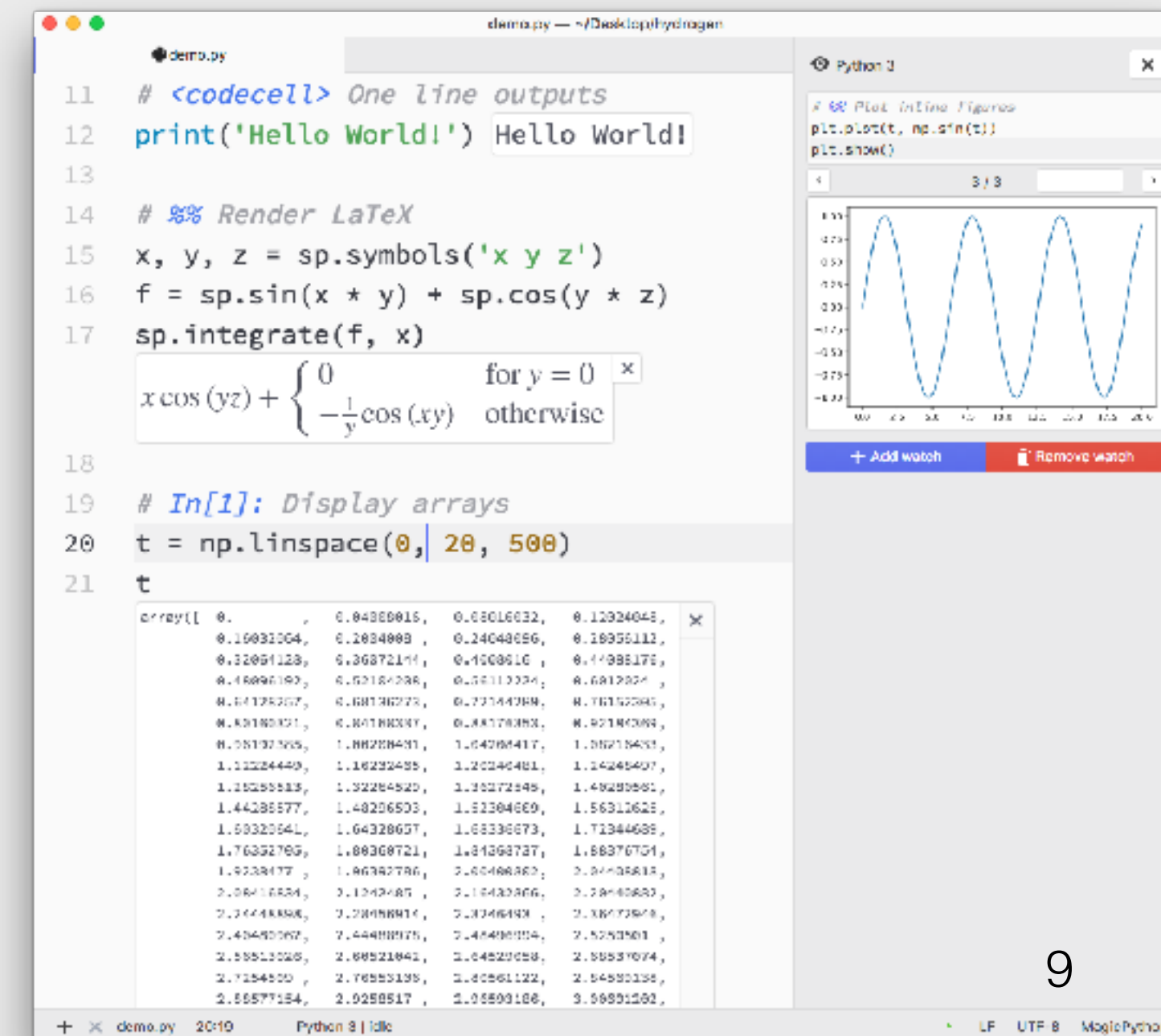
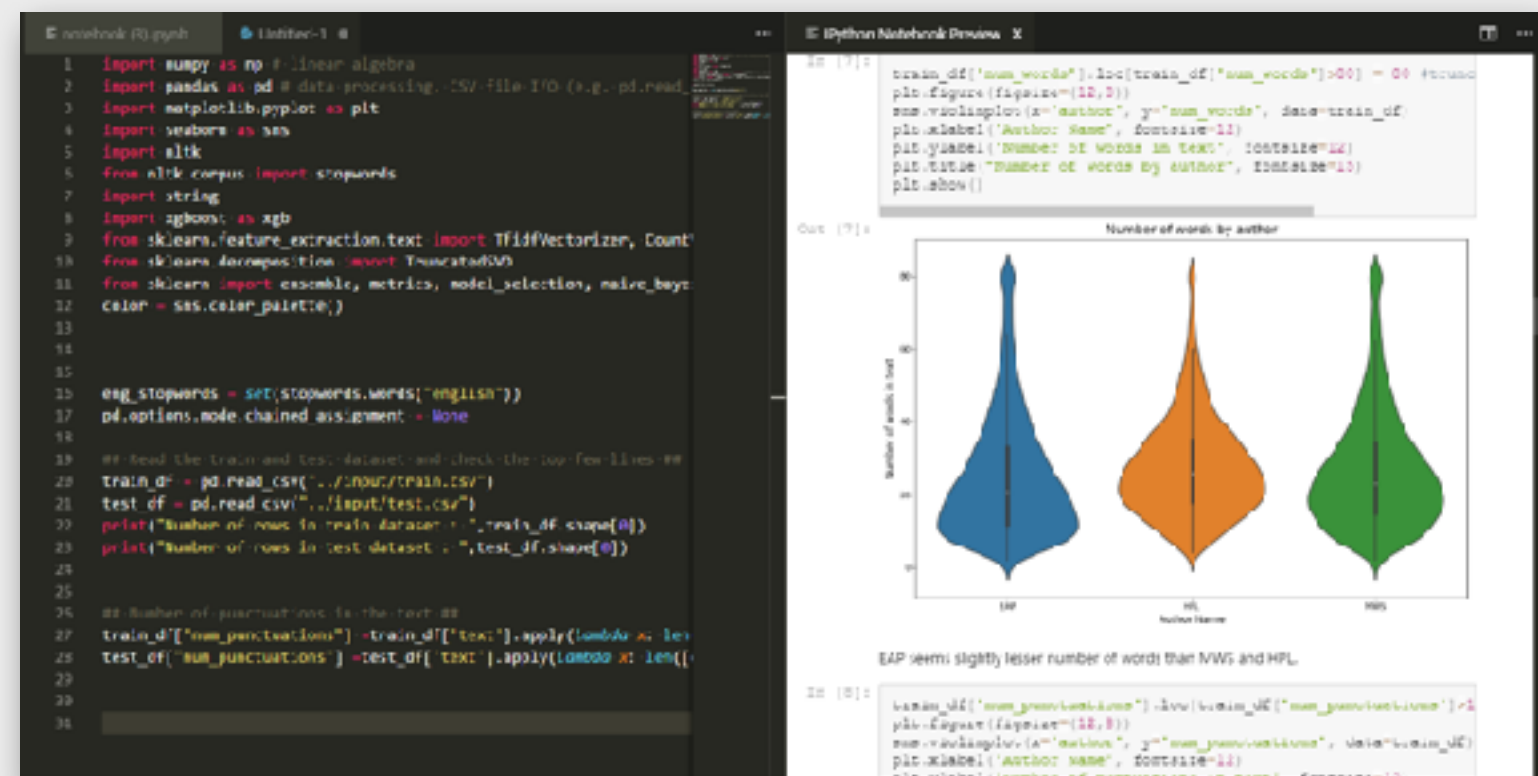


Ecosystem

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+*

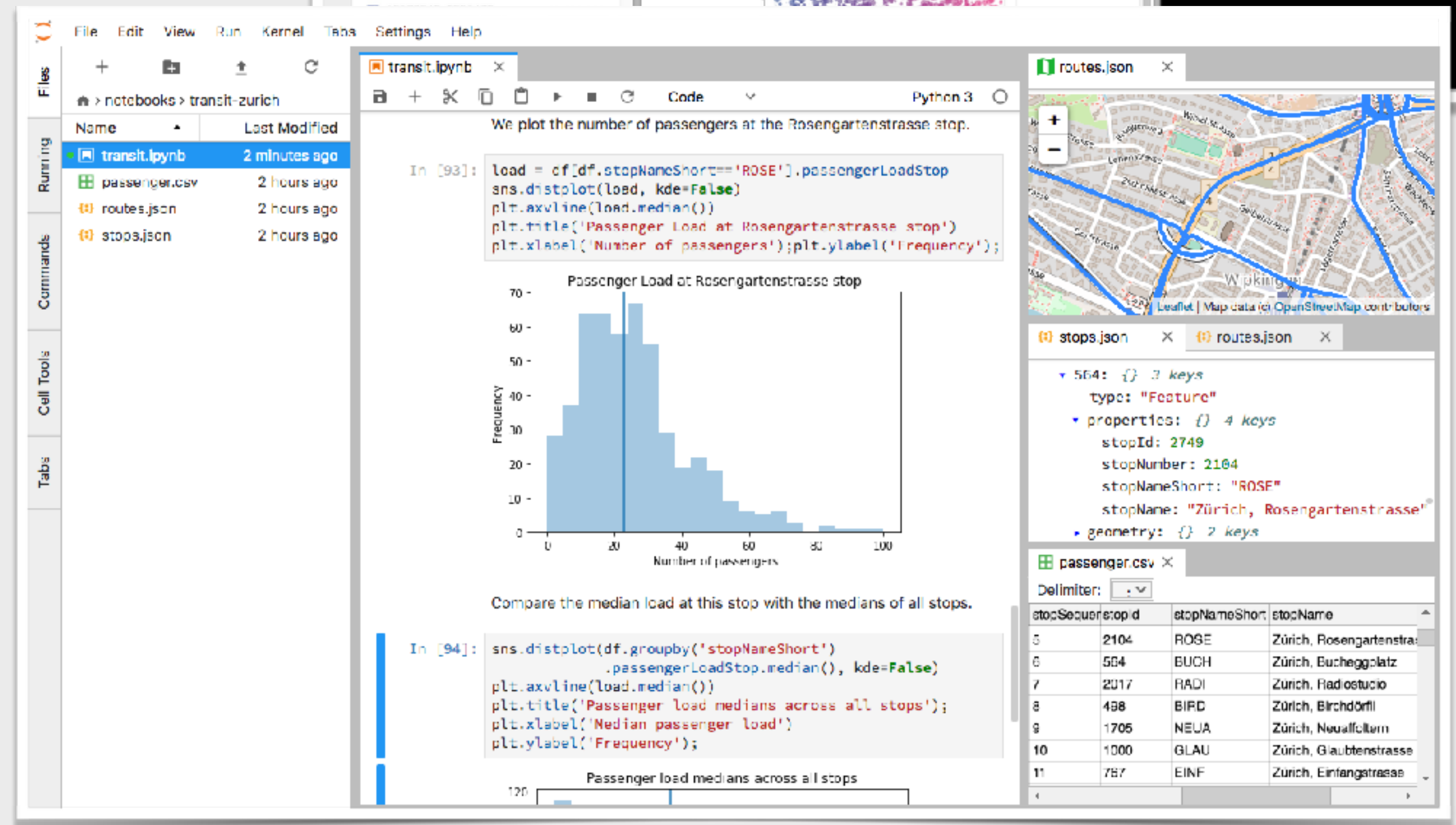
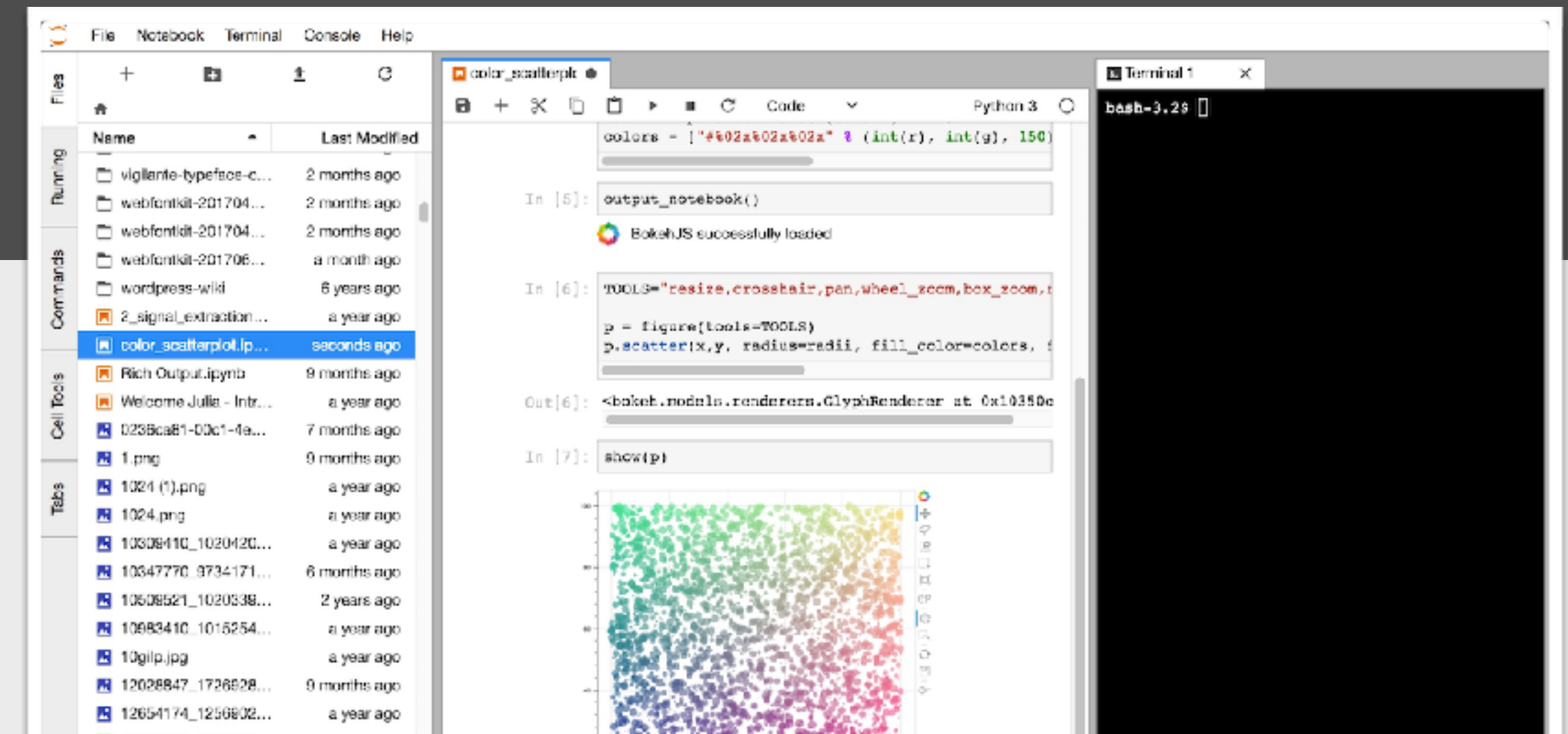
Building Blocks: Nbformat, JupyterHub, Kernel Gateway...



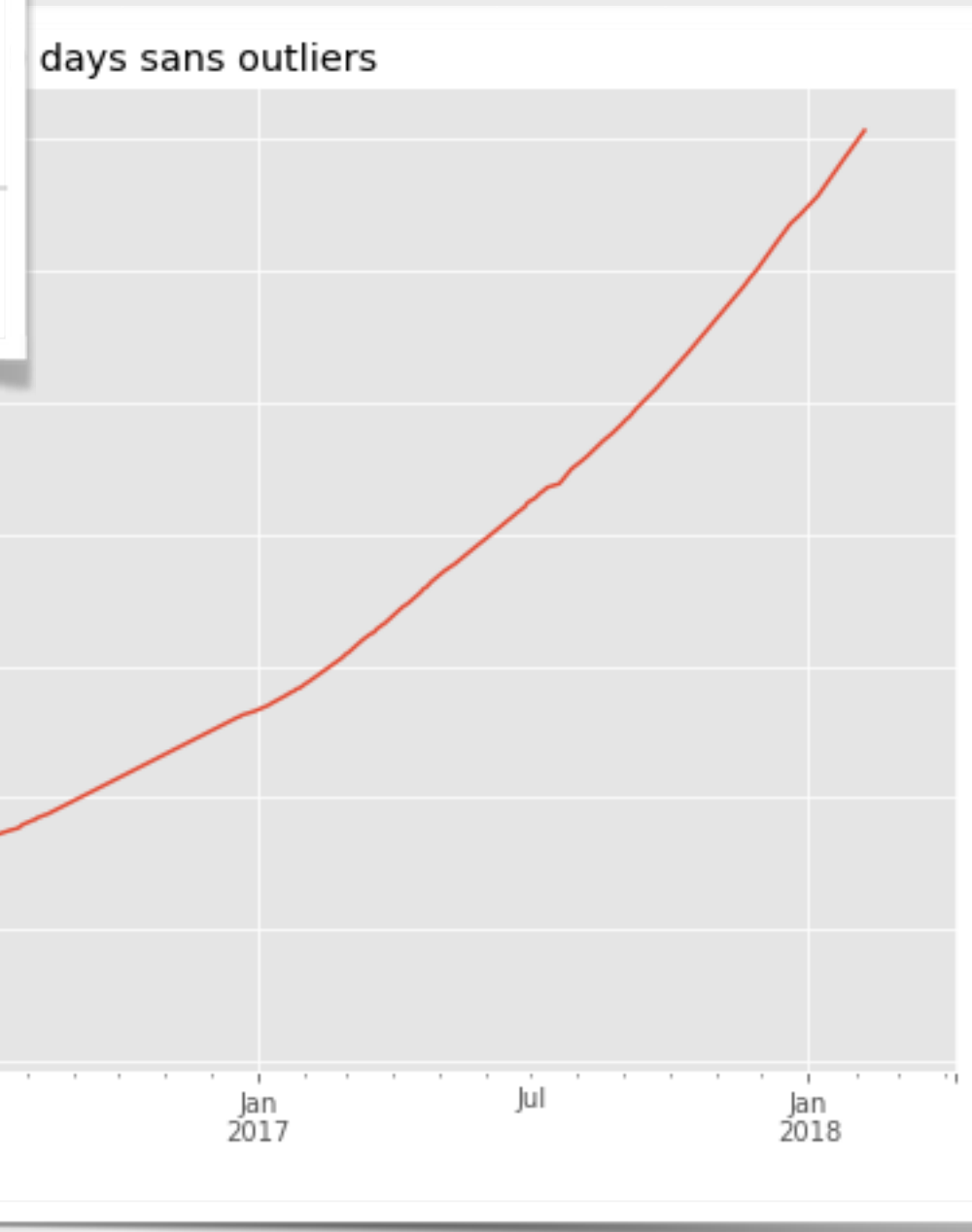
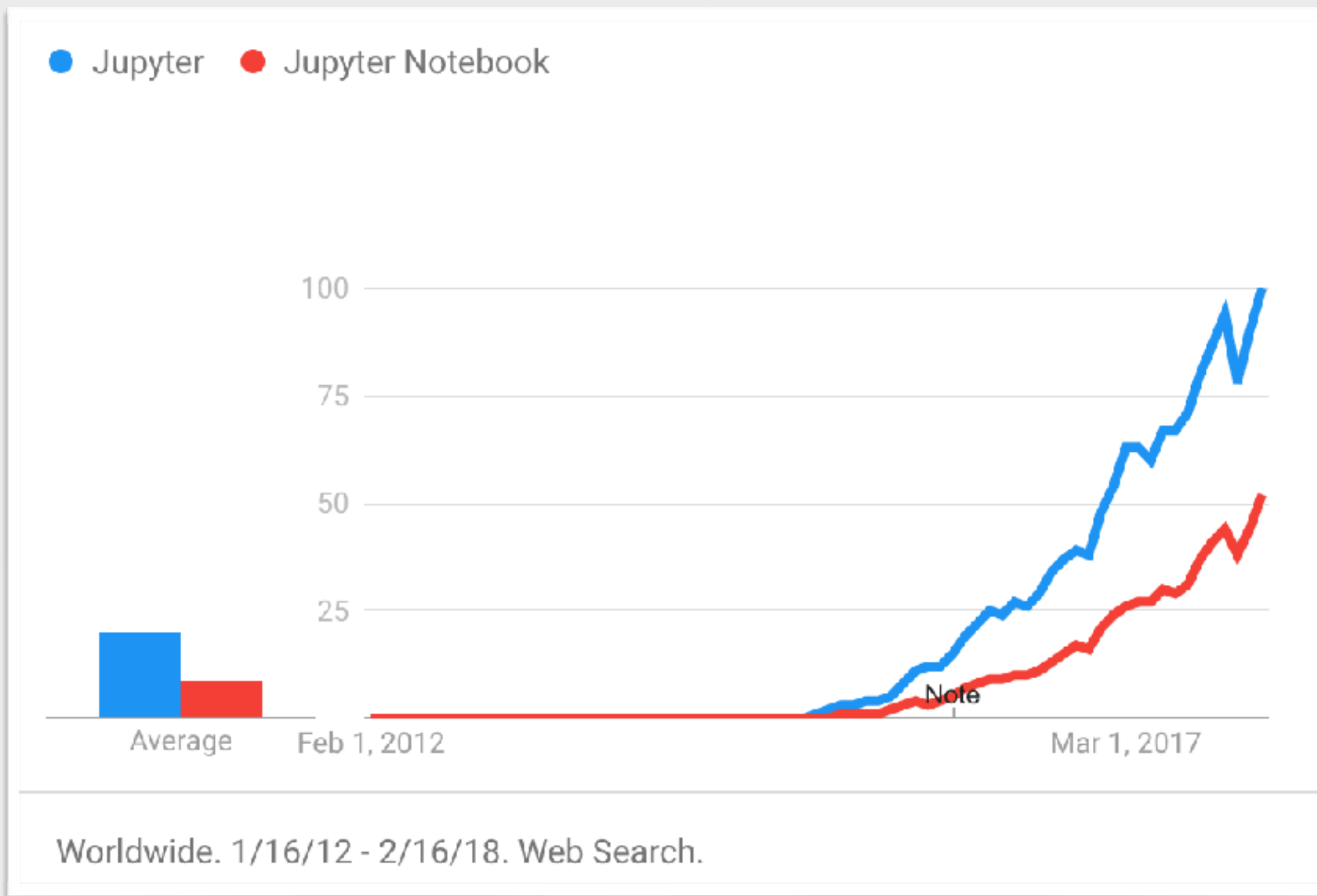
JupyterLab

- Extends the notebook interface with text editor, shell, ...etc
- is it an IDE ?

- If by I you mean Interactive, then yes



Popularity



<https://github.com/parente/nbestimate>



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



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.



Network enabled / web based

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



Bojan Marković

Feb 20

You'll only take Spyder from my cold, dead... Ooooooh, pretty shiny colors, inline graphics.. Does it come in fuchsia? :)

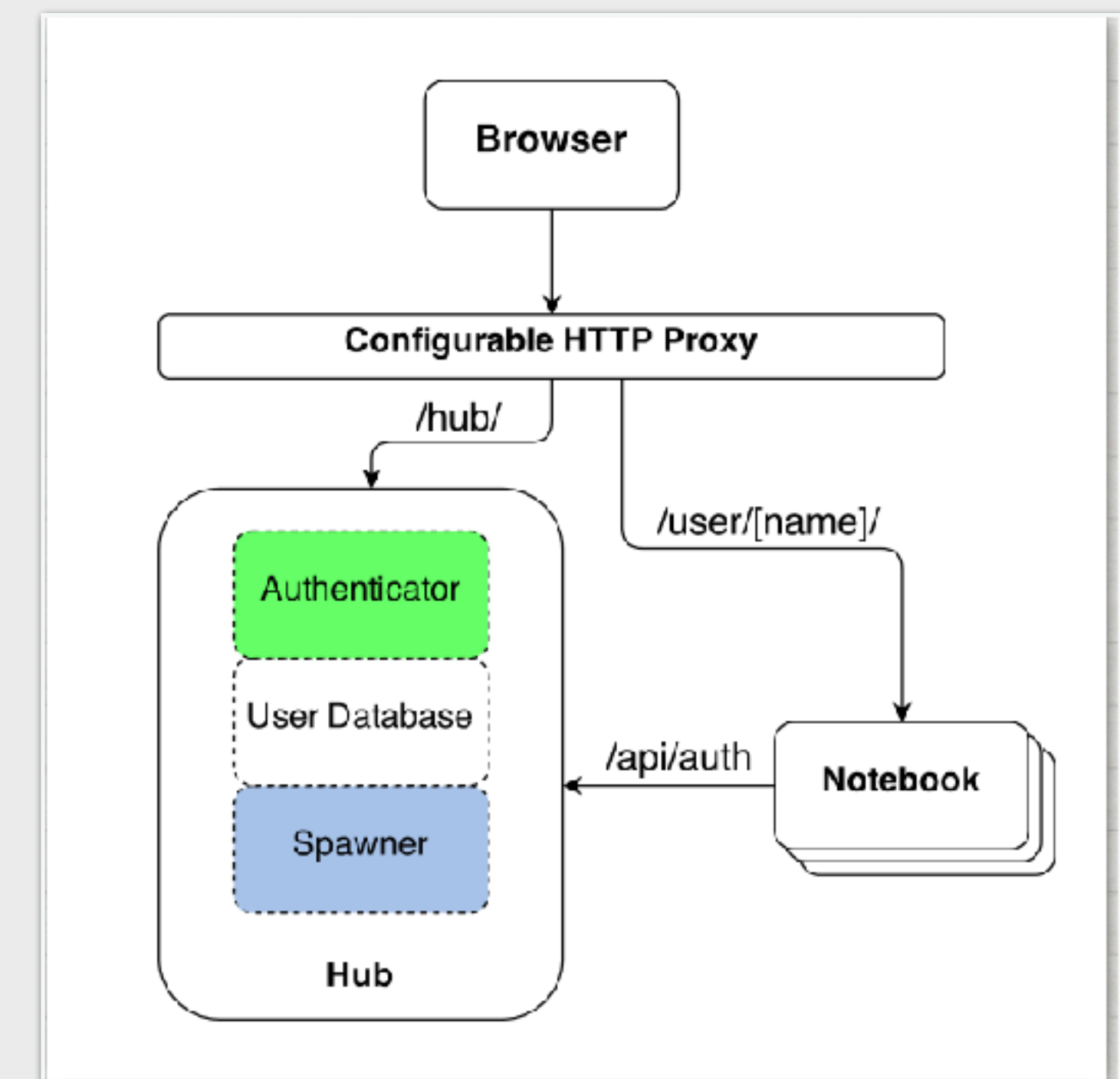
- dynamic libraries are highly popular
- Usable by novices and power-users
- Users w/ different expertise (Numerical Methods, Visualization,...)
- Seamless transition to HPC: Kernel Menu > Restart on Cluster
- Document persist if code crash.
- Can be Zero-Installation (See JupyterHub).
 - A web browser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

Popularity



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...)



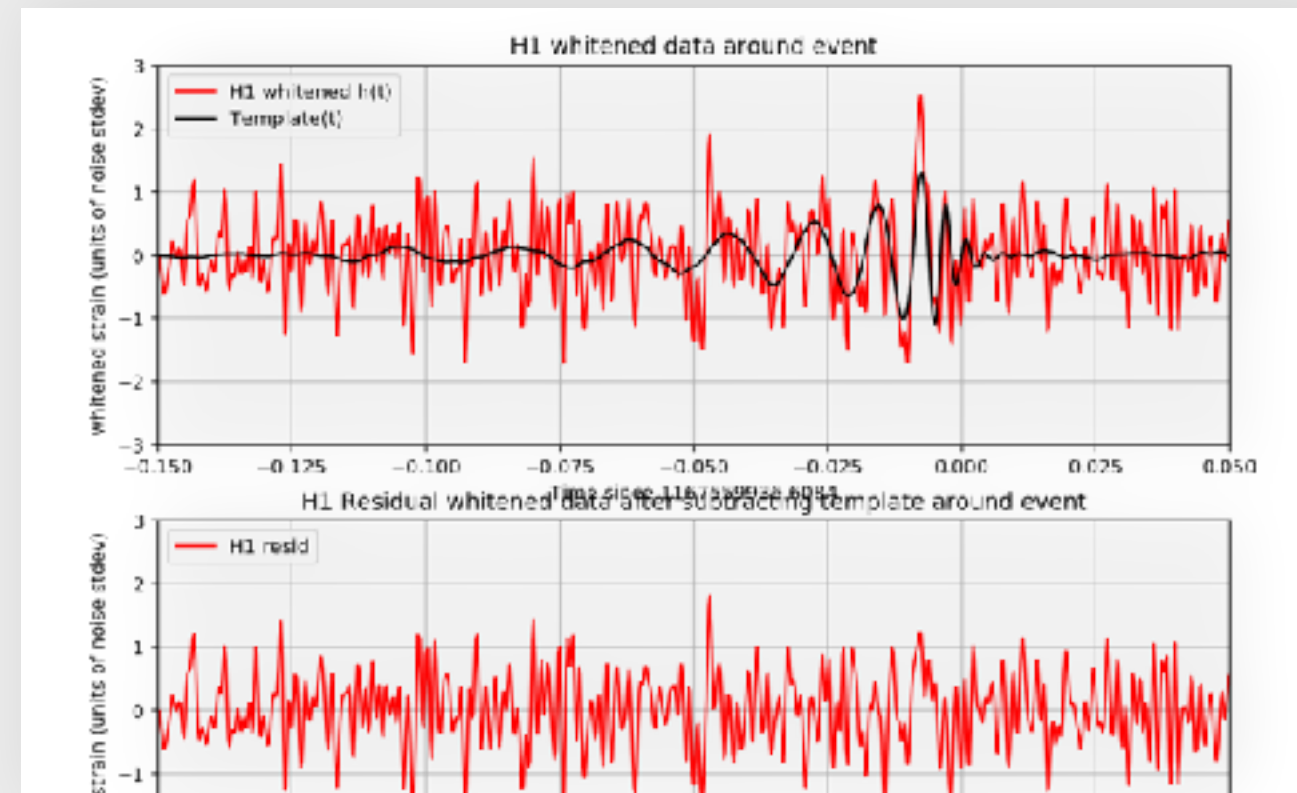
Use Cases

HPC

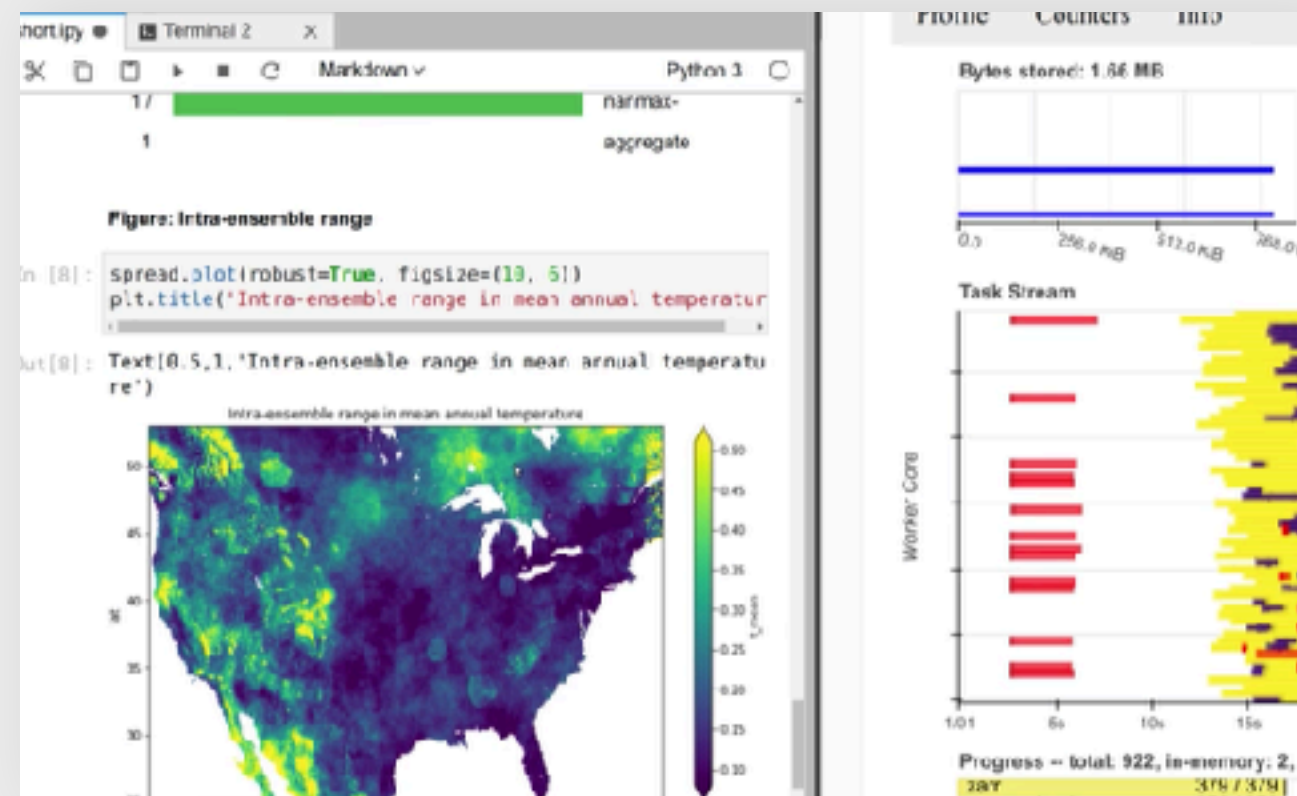
- Batch Jobs
 - You can run notebook “headless”
 - Parametrized notebook as “reports” you can interact with later
- Interactive Cluster.
 - Run a Hub (hook into LDAP/PAM...)
 - Run notebook servers on a Head node
 - Run Kernels on head Node/fast queue
 - Extra Workers (e.g. dask) on Batch queue/cluster.



Some Jupyter Usage



Ligo



Pangeo

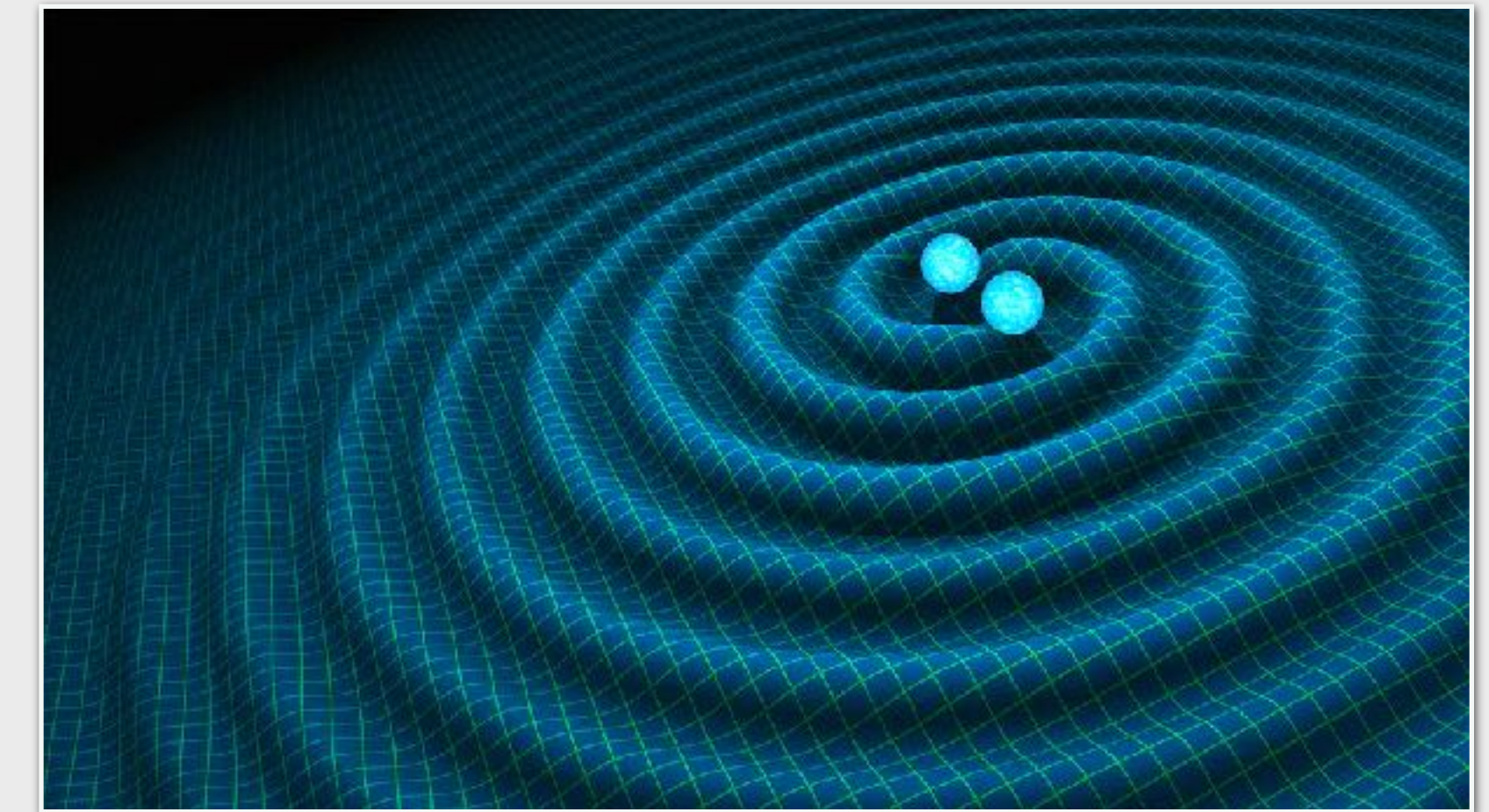
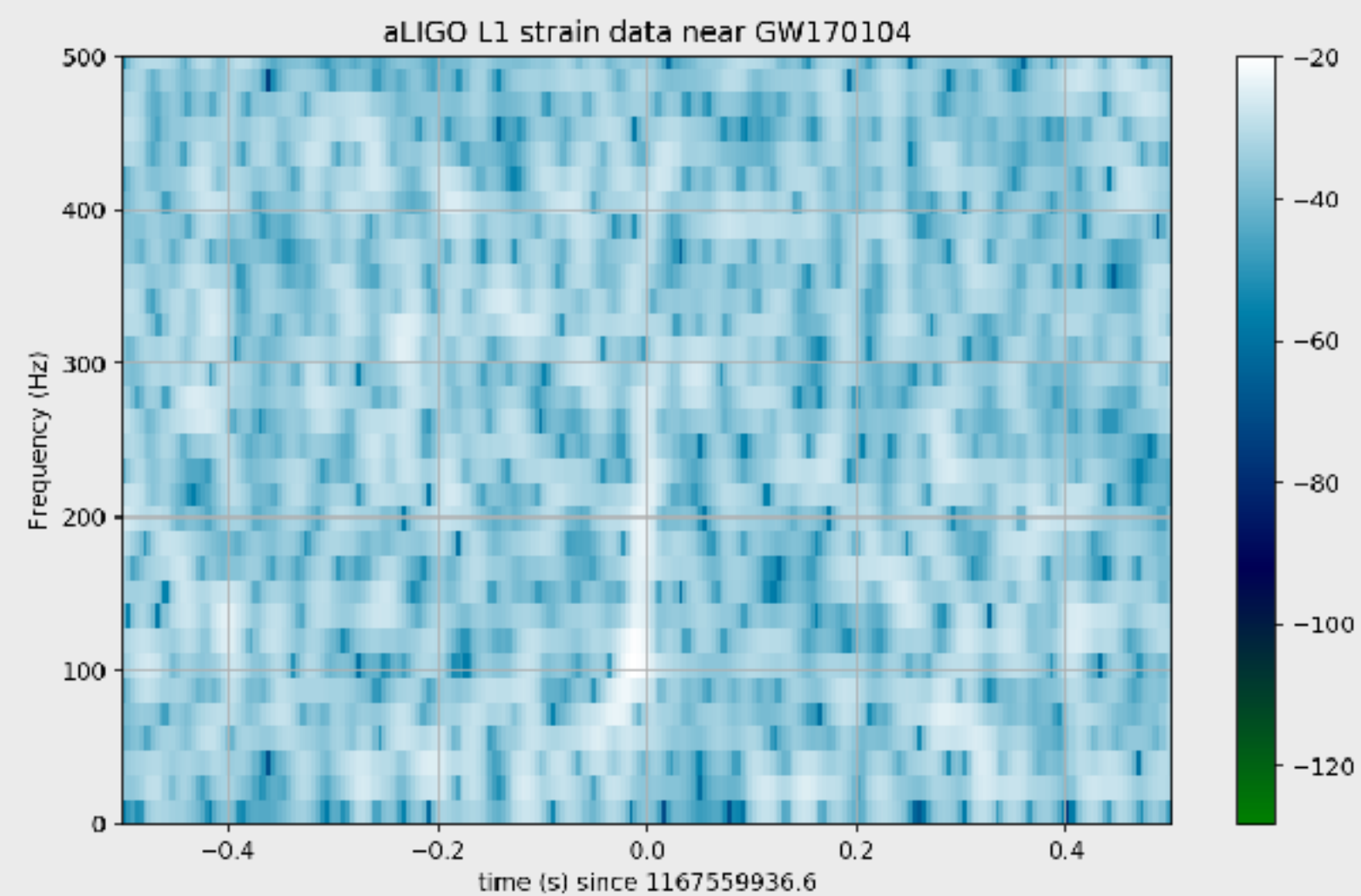
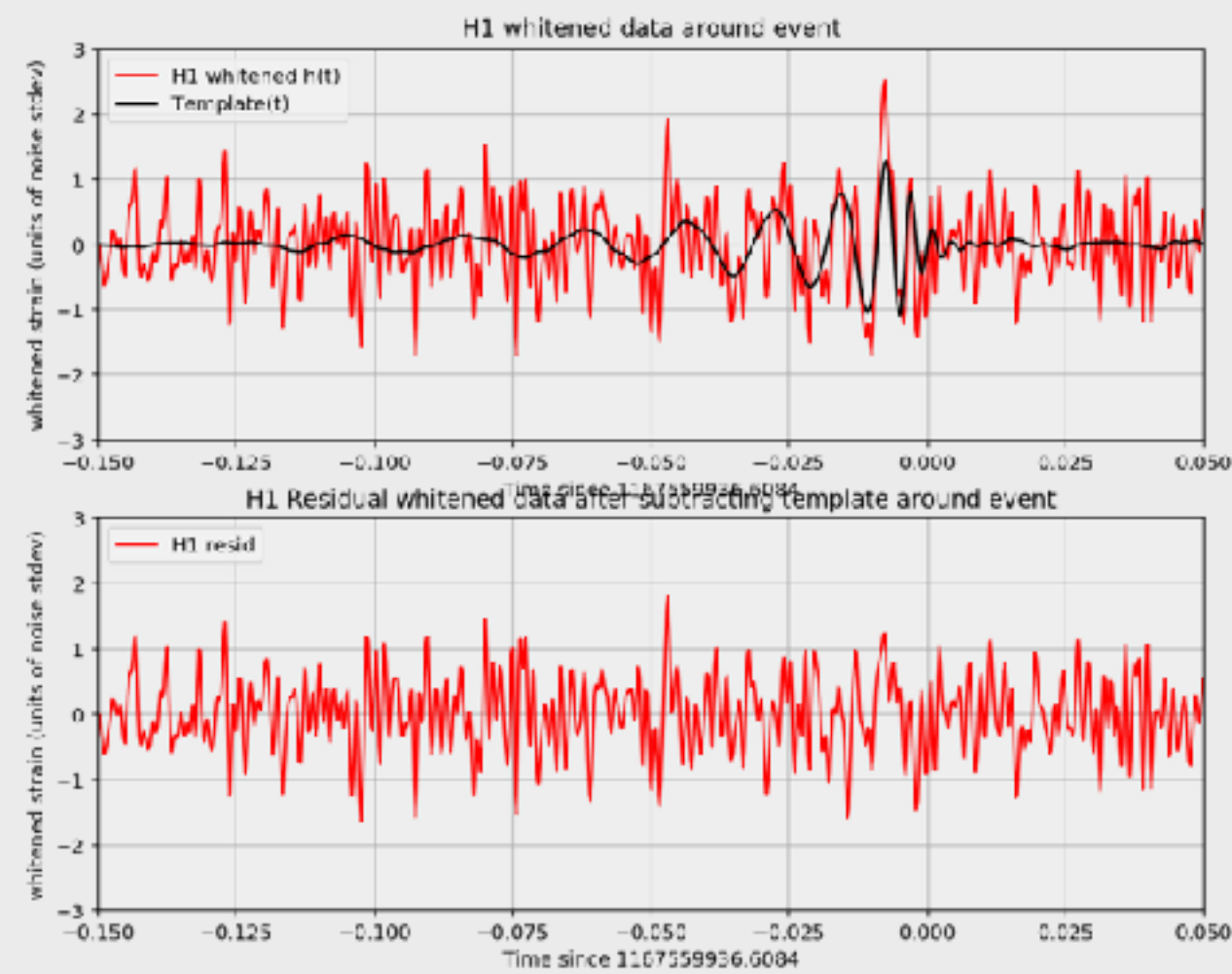


Cern's SWAN



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>₁₈

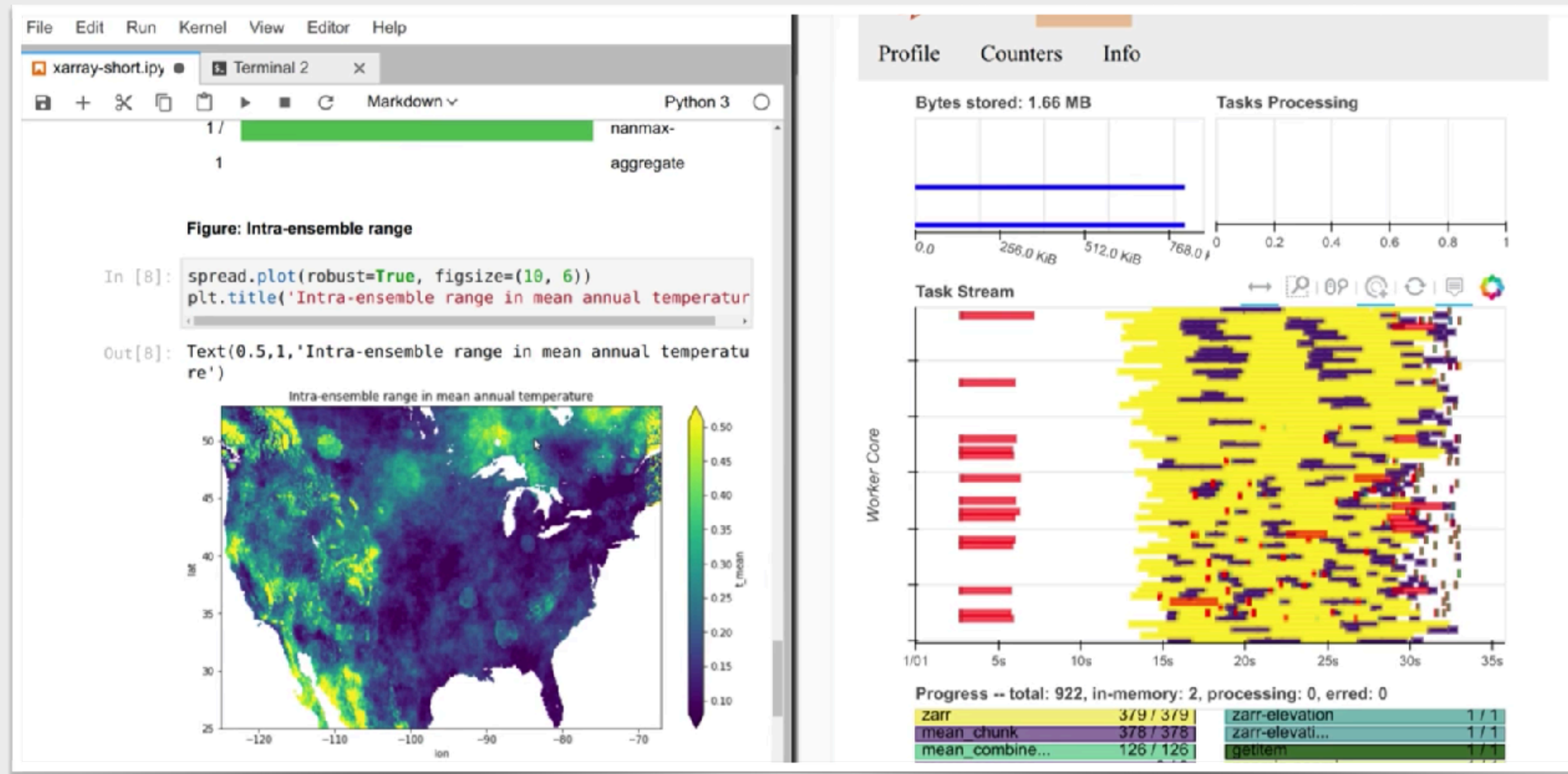


Pangeo (pangeo-data.github.io)

- 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

C++ from Python w/o bindings

Interactivity without bindings

In order to interact with the C++ entities contained in the library, we need to carry out two tasks:

1. We need to make known to the interpreter the interfaces. Concretely this means including one or more headers.
2. We need to make accessible to the interpreter the implementations of such C++ entities. Concretely this means loading the library.

In code:

```
In [5]: %import ROOT
ROOT.gInterpreter.ProcessLine( "#include <./data/myLibrary.h>" )
ROOT.gSystem.Load( "./libmyLibrary.so" )
```

Welcome to Jupyter 0.01/01

Out[5]: 0

That's it! We can now start exploring the content of the library. If you are wondering what a return code equal to 0 means, ROOT is telling us that the loading of the library happened without problems!

```
In [6]: a = ROOT.A()
```

This is the constructor of A

```
In [7]: del a
```

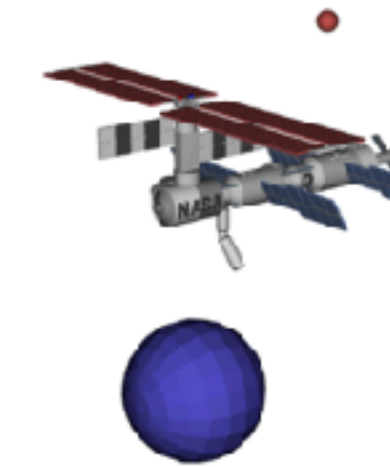
This is the destructor of A

```
In [8]: b.doublePtr = ROOT.B("double")()
```

Open in SWAN

3D Visualisation

```
if [1]: auto topVolume = gGeoManager->GetTopVolume();
topVolume->Draw();
```

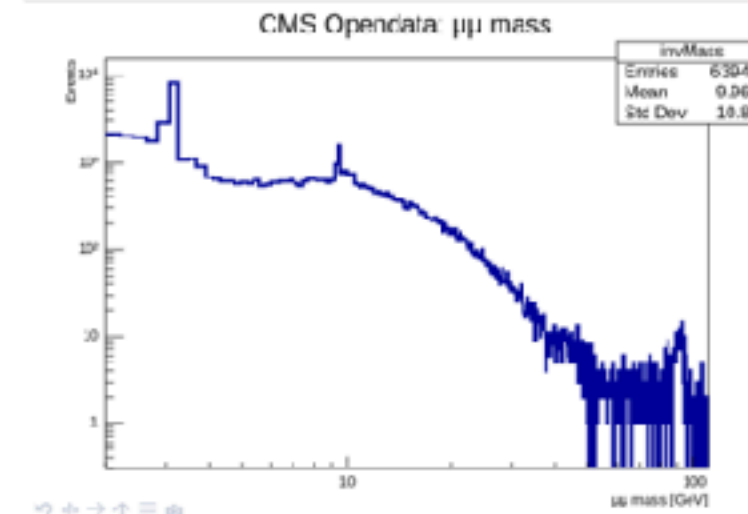


Open in SWAN



CMS Opendata: di-muon analysis

```
In [5]: invMass = ROOT.TH1F("invMass", "CMS Opendata: #muon mass;#muon mass [GeV];112, 22,
invMassFormula = "sqrt((E1 + E2)^2 - ((px1 + px2)^2 + (py1 + py2)^2 + (pz1 + pz2)^2))";
cut = "Q^2 >= 1";
c = ROOT.TCanvas();
c.Draw(invMassFormula + " >> invMass", cut, "112");
c.SetLogy();
c.Draw();
```



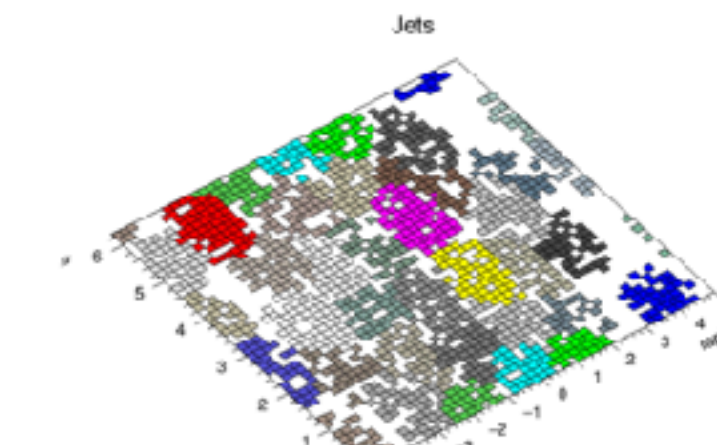
That might have been too fast. We now make the analysis above more explicit producing a plot also for the J/Psi particle.

Open in SWAN

Interactive usage of 3rd party libraries

```
[components->Draw("L1L1 B");
c.Draw();

# FastJet release 3.1.1
# M. Cacciari, G.P. Salam and G. Soyez
# A software package for jet finding and analysis at colliders
# http://fastjet.fr
# Please cite EPJ3(2012)1396 [arXiv:1111.0697] if you use this package
# for scientific work and optionally PUB041(2006)07 [hep-ph/0512218].
#
# FastJet is provided without warranty under the terms of the GNU GPLv2.
# It uses T. Chan's closest pair algorithm, S. Furuno's Voronoi code
# and 3rd party plugin jet algorithms. See COPYING file for details.
```



Open in SWAN





CFP- Ends March 6th



Question(s)
while we change
speakers ?

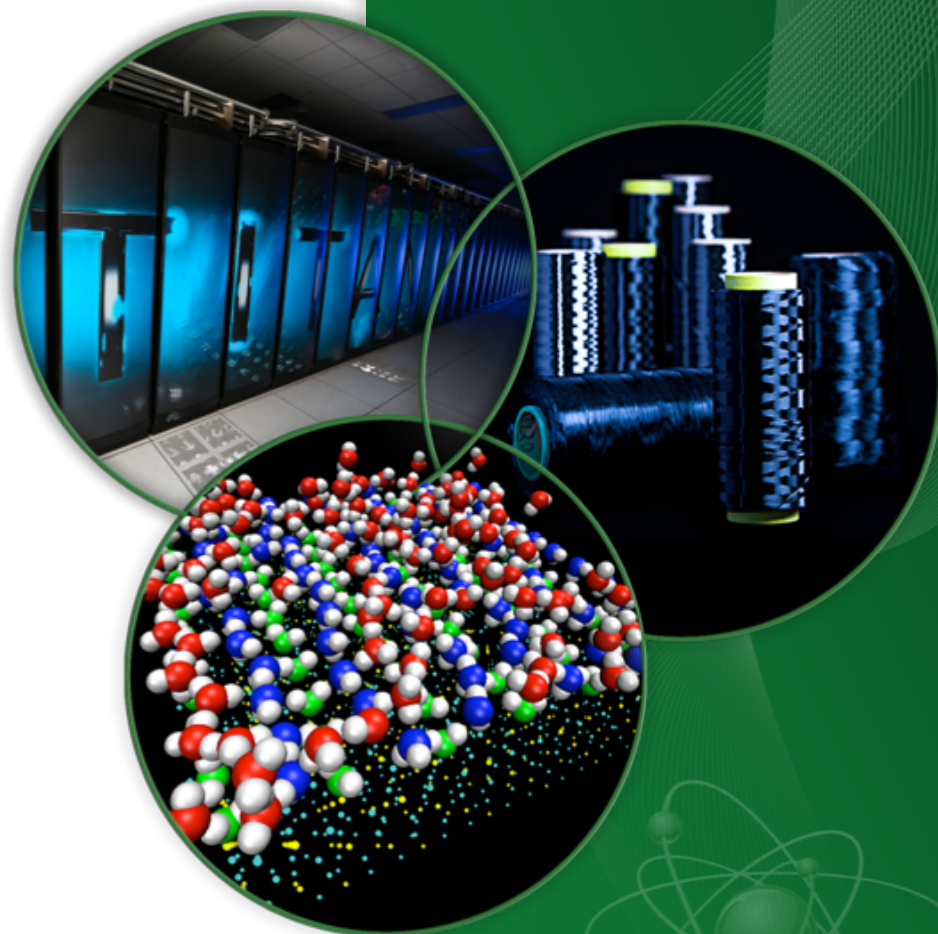


Jupyter for Supporting a Materials Imaging User Facility (and beyond)

Suhas Somnath

Advanced Data and Workflows Group,
Oak Ridge Leadership Computing Facility

ORNL is managed by UT-Battelle
for the US Department of Energy

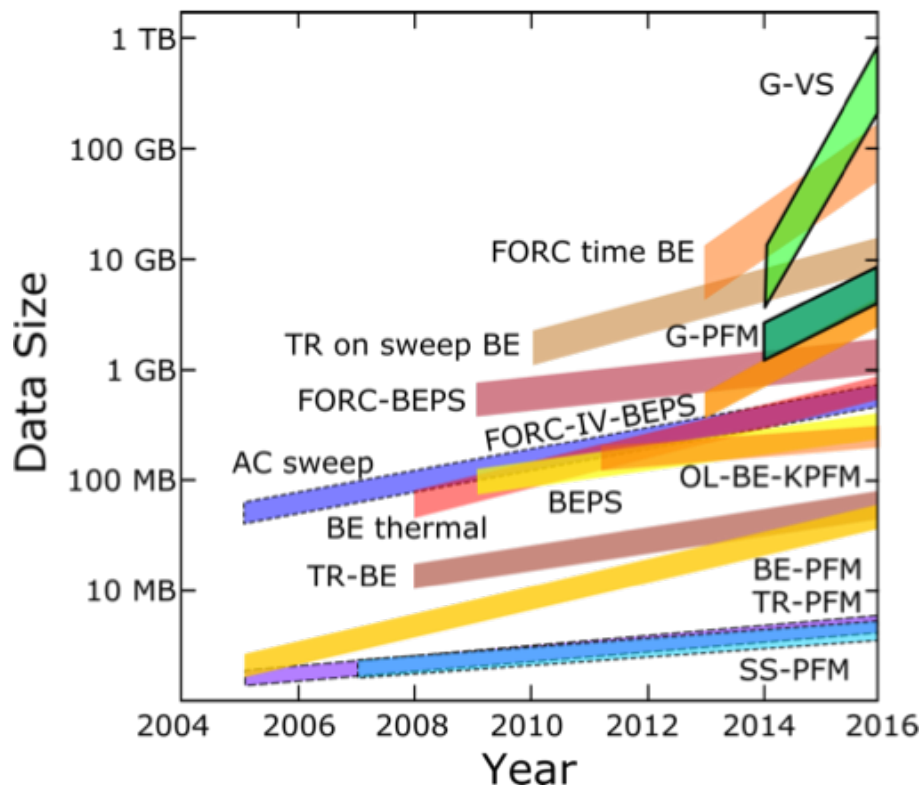


Opportunities in Computing

- 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

Evolution of Scanning Probe Microscopy Data



- **Growing data sizes & dimensionality**

- Cannot use desktop computers for analysis

- **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



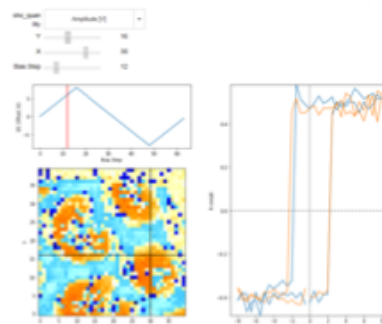
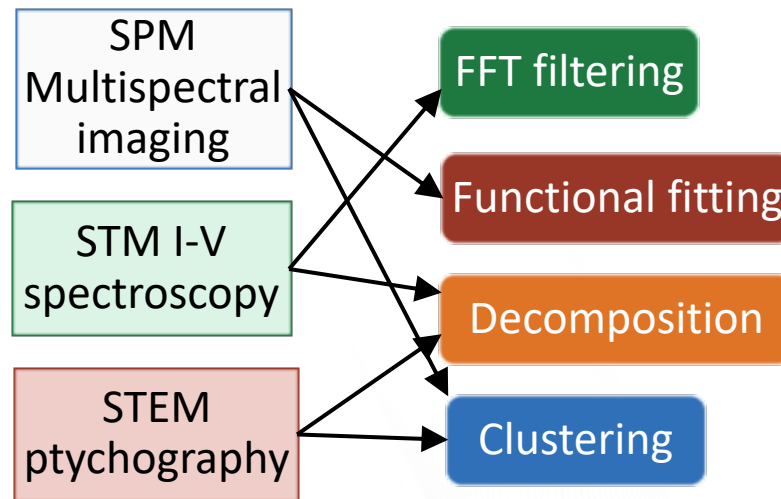
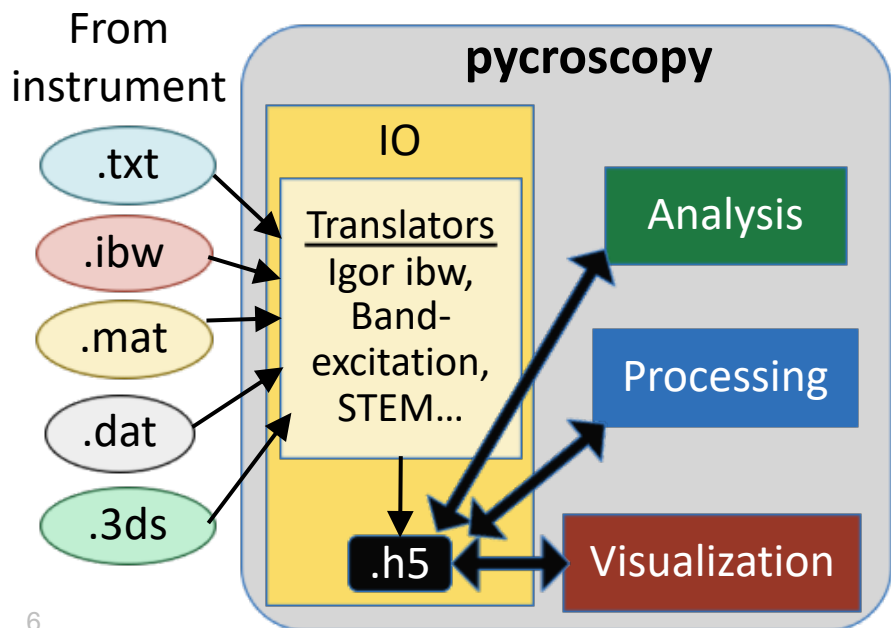
Open-source python package for analyzing + formatting microscopy data

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

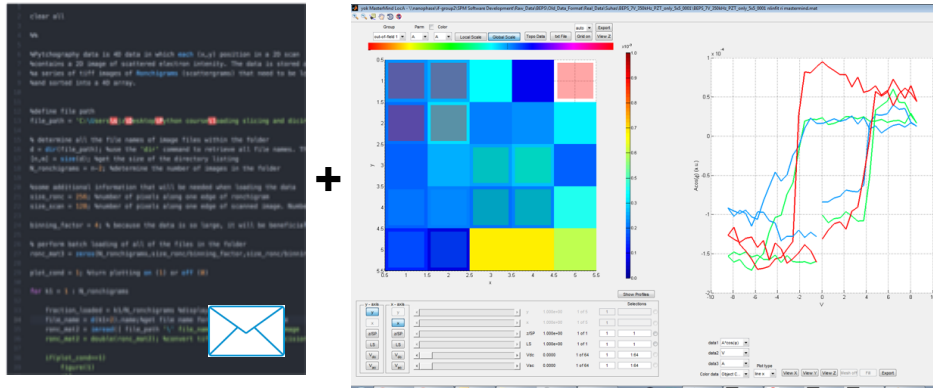


Conveying information

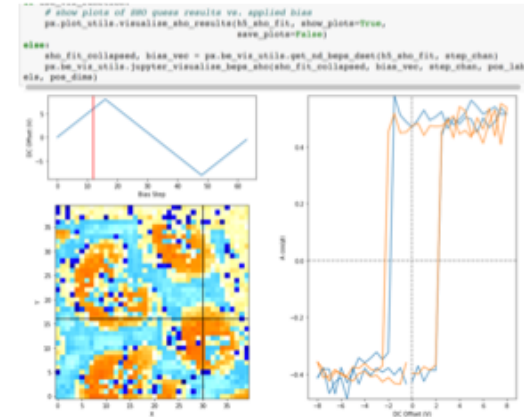
- Interactive jupyter notebooks

Supporting User Research

Before 2016



Since 2016



Scripts + complicated, monolithic, Matlab GUI

Written by dedicated software engineer

Not customizable on-the-fly

2-3 hours of training before use

Deployed only on two offline workstations due to licensing restrictions = queue

Will remain on off-line desktops

Set of simple Jupyter notebooks

Written by material scientists

Completely customizable.

Instructions embedded within notebook. NO training required!

Each user gets VMs with jupyter notebook server


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



ARTICLE

DOI: [10.1038/s41467-017-02455-7](https://doi.org/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. Morozovska⁴, P. Maksymovych^{1,2}, Y. Kim⁵, X. Lu⁶, M. Alexe⁷, 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.

Visualizing Filtering Results

Run the next cell to see what the IV curves look like for the chosen filter parameters. If the current set of filter parameters do not work as well, change the parameters in the previous cell and try again.

Once the results look good, proceed to filter the entire dataset

```
In [7]: # Test filter on a single line:
row_ind = 50
filt_line, fig_filt, axes_filt = px.processing.gmode_utils.test_filter(h5_main[row_ind], filter_params, samp_rate, show_plots=True, use_raw_plots=False)
fig_filt.savefig(os.path.join(other_figures_folder, 'FFT_filter_on_line_{}.png'.format(row_ind)), format='png', dpi=300)

raw_row = np.reshape(h5_main[row_ind], (-1, pts_per_cycle))
filt_row = filt_line.reshape(-1, pts_per_cycle)

fig, axes = px.plot_utils.plot_loops(single_NO, [raw_row, filt_row], dataset_names=['Raw', 'Filtered'], line_colors=['r', 'b'], x_label='Bias (V)', title='FFT Filtering', plots_on_side=3, y_label='Current (nA)', subtitles='Row: ' + str(row_ind) + ' Col:')
fig.savefig(os.path.join(other_figures_folder, 'Example_filtered_loops_from_line_{}.png'.format(row_ind)), format='png', dpi=300)
```



DOI associated with data (raw → paper figures)

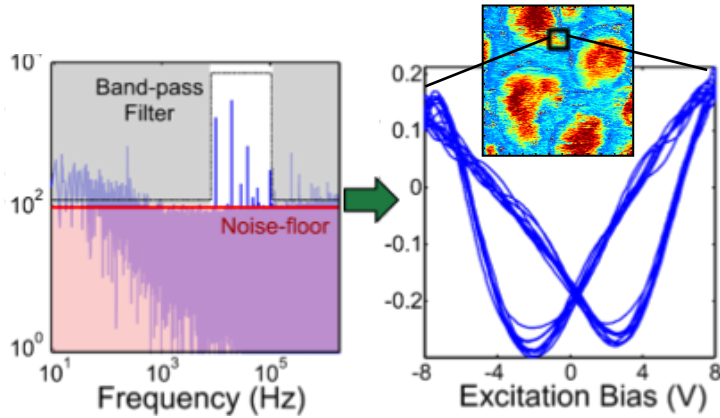
Oak Ridge National Laboratory Leadership Computing Facility **10.13139/OLCF/1410993** [Download](#)

Authors

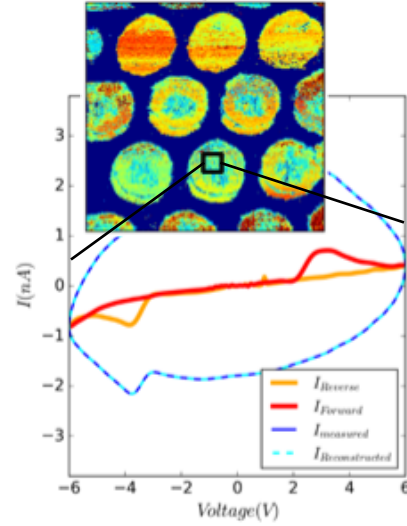
Somnath, Suhas	somnaths@ornl.gov
Law, Kody	lawkj@ornl.gov
Morozovska, Anna	anna.n.morozovska@gmail.com
Maksymovych, Petro	maksymovychp@ornl.gov
Kim, Yunseok	ykim943@gmail.com
Lu, Xiaoli	xilu@xidian.edu.cn
Alexe, Marin	M.Alexe@warwick.ac.uk

Scientific Advancements with Jupyter

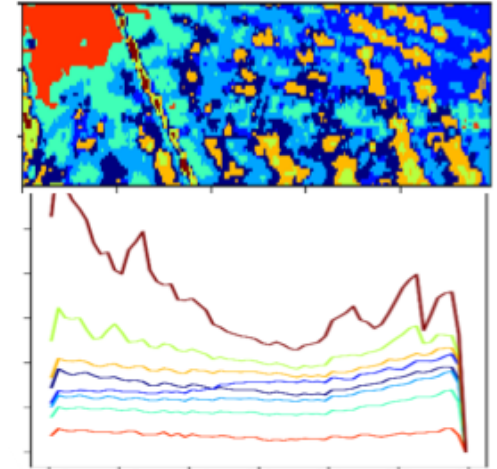
3,500x faster imaging via adaptive signal filtering, linear unmixing of signals



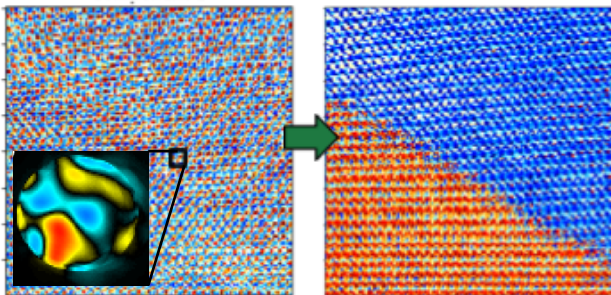
200x faster spectroscopy via Bayesian inference



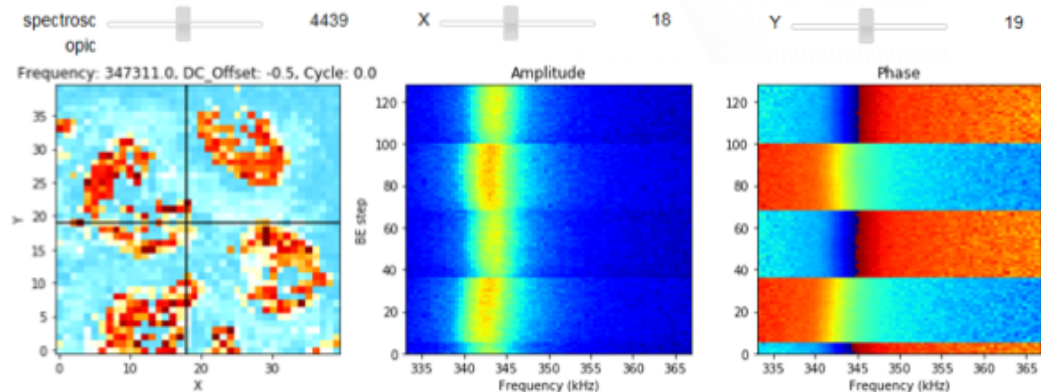
Denoising and clustering to identify superconductivity at the nanoscale



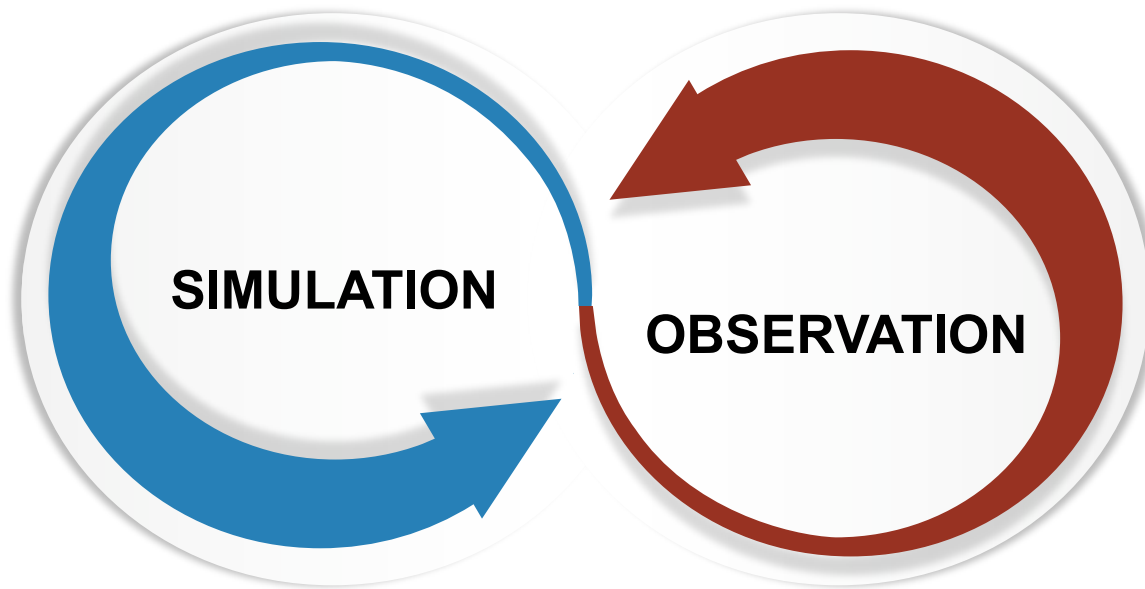
Identifying invisible patterns using multivariate analysis



Simplified navigation multidimensional data - users



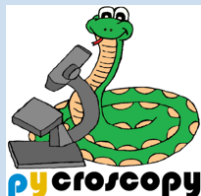
Completing a Discovery Paradigm



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

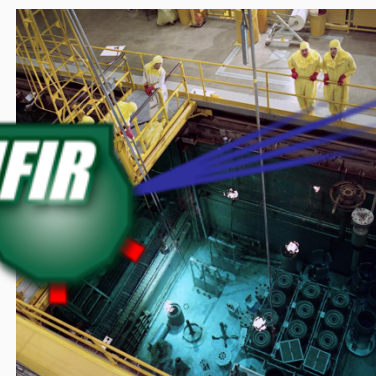


Electron Microscopy



pyEM ?

....



OAK RIDGE National Laboratory

Acknowledgements

Pycroscopy Team:

- Stephen Jesse
- Chris R. Smith

IFIM members:

- Sergei V. Kalinin
- Stephen Jesse
- Rama K. Vasudevan

Analytics Team:

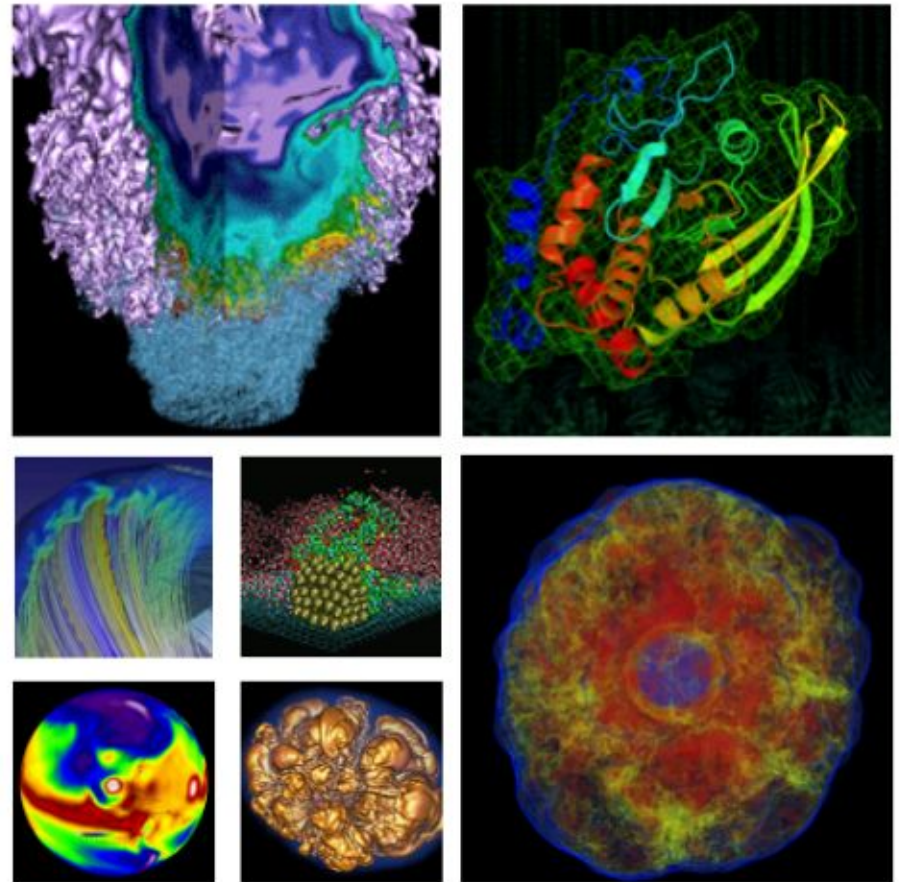
- Junqi Yin
- Arjun Shankar

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”

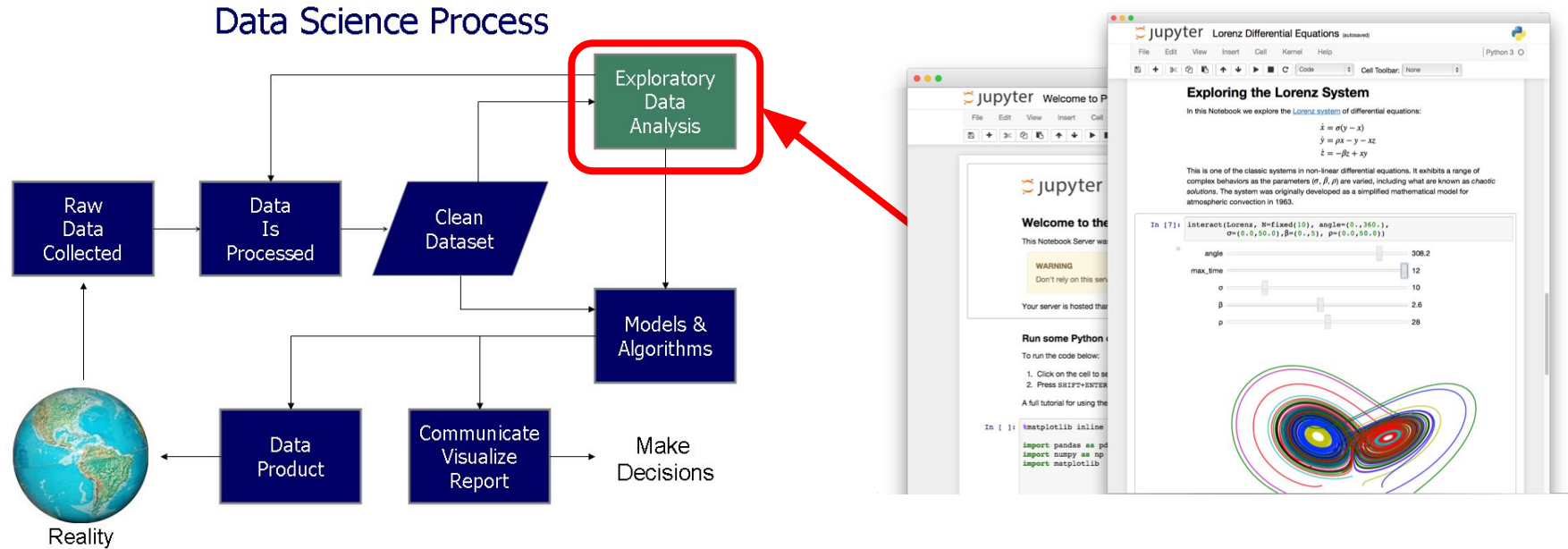


Gerty Cori: Biochemist and first American woman to win a Nobel Prize in science

- 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



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**

Central Role of Python at NERSC



Intel® Distribution for Python*



Powered by Anaconda

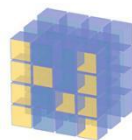
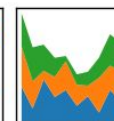
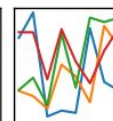
Accelerating Python* performance on modern architectures from Intel.

- Get out-of-the-box performance that is closer to native code speeds.
- Speed up data analytics with pyDAAL and parallelize Python workloads.
- Manage packages and Jupyter Notebooks easily with conda, Anaconda Cloud, and PIP.

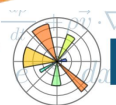


pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



NumPy



matplotlib



astropy

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**

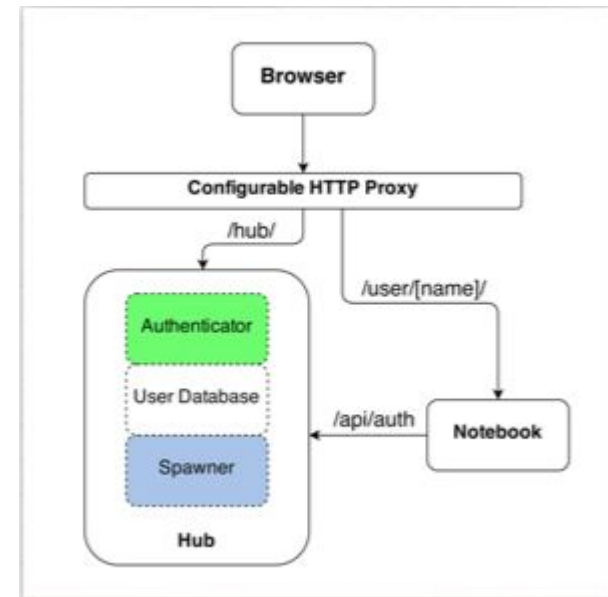
Jupyterhub to rescue

- ✓ **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



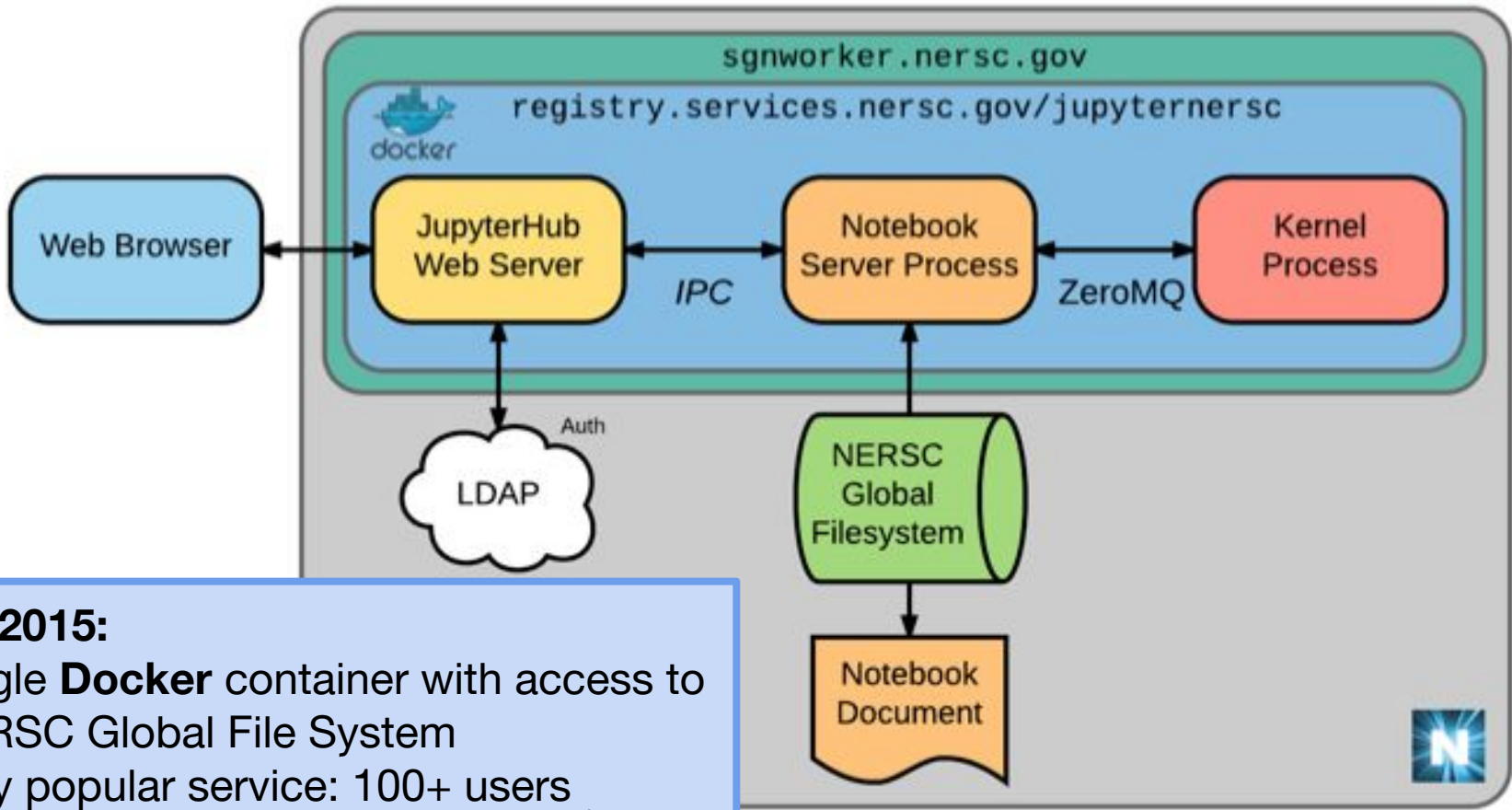
- Service to deploy notebooks in a multi-user environment
- Manages user authentication, notebook deployment and web proxies



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
 - ...

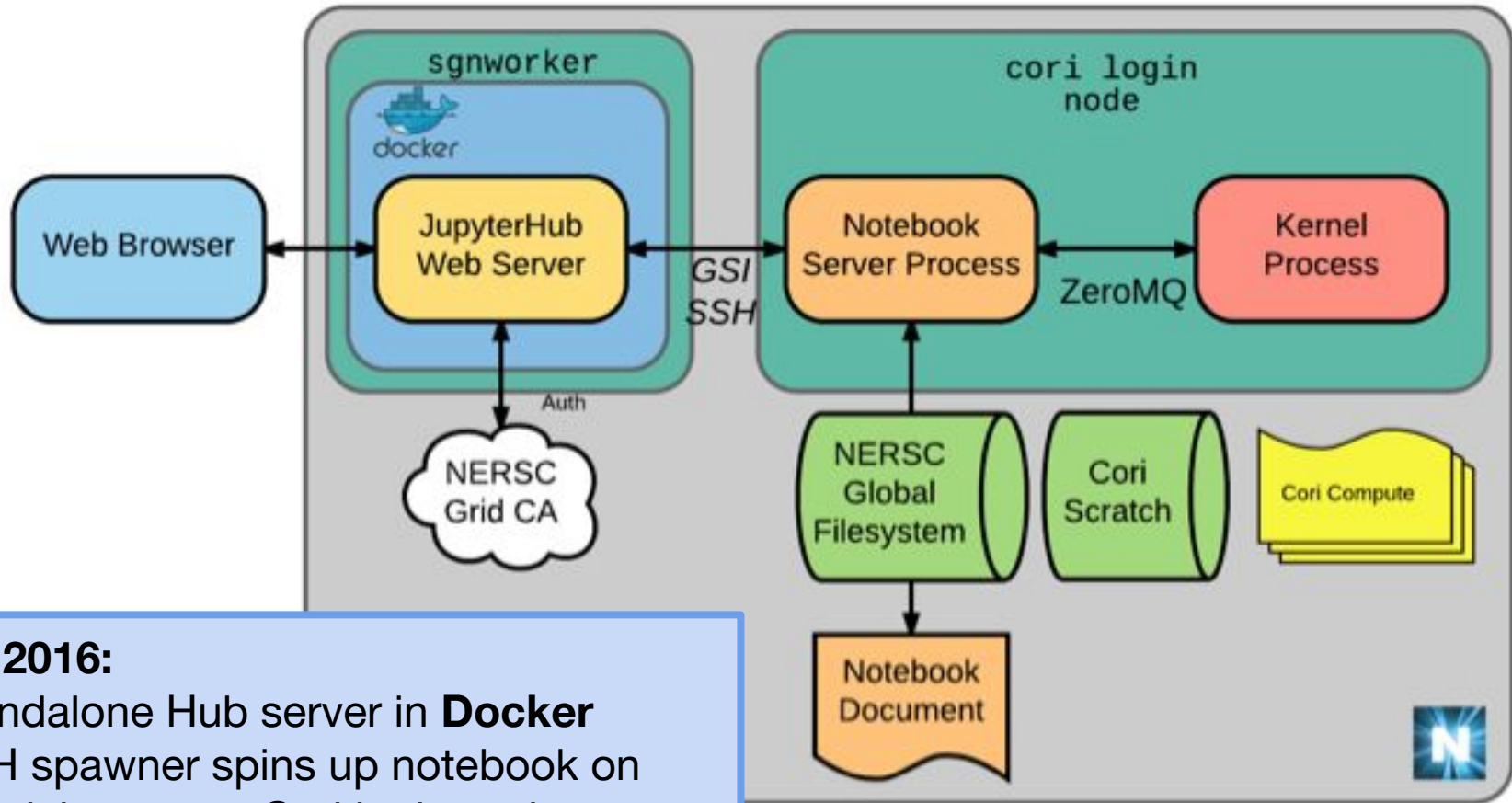
Jupyter@NERSC

Evolution of Architecture

Step 2: Integration with Cori compute and filesystems

Second Architecture: Cori Login Node

NERSC



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`

- 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.

`GSIAuthenticator`

<https://github.com/NERSC/GSIAuthenticator>

`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.

- 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>

- **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>**

- 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 ...
```



```
{
  "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

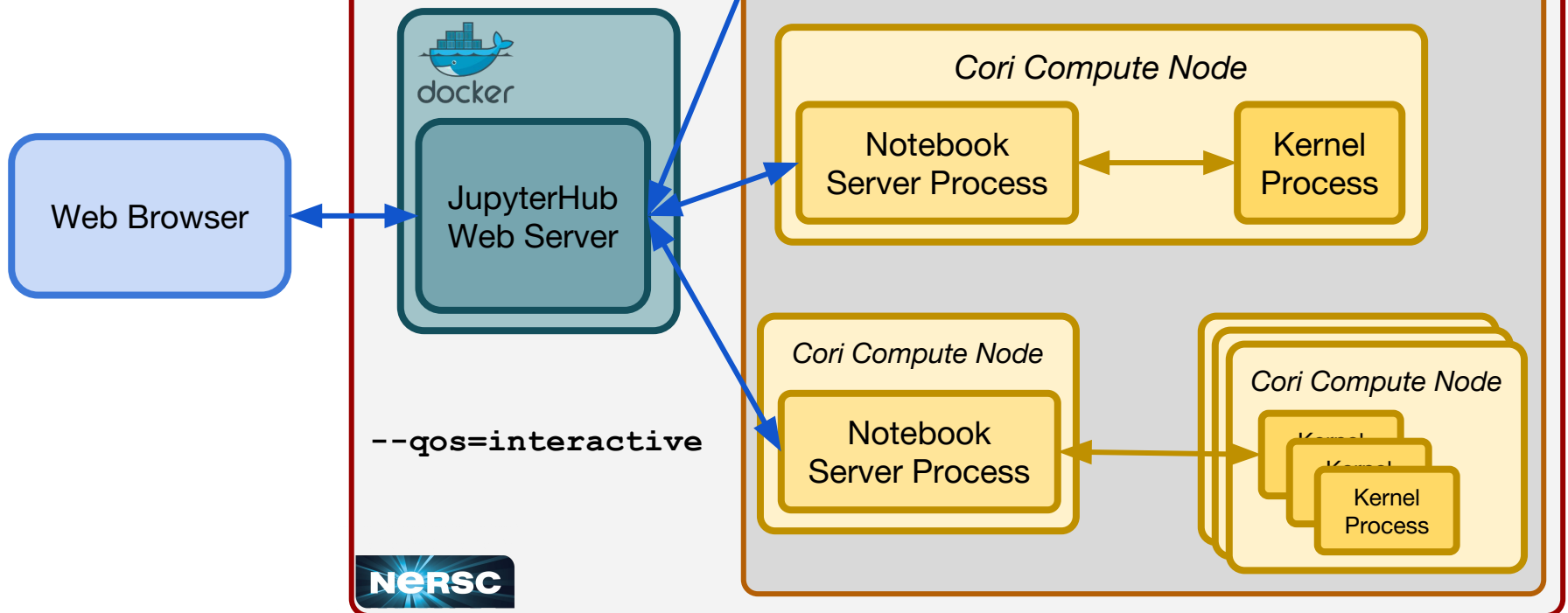
- **Users customize their notebooks with libraries and APIs of their own design or from third parties.**
- **NERSC wants to offer Jupyter to users so they don't set it up themselves in an insecure way.**

Jupyter@NERSC

Evolution of Architecture

Step 3: The Future

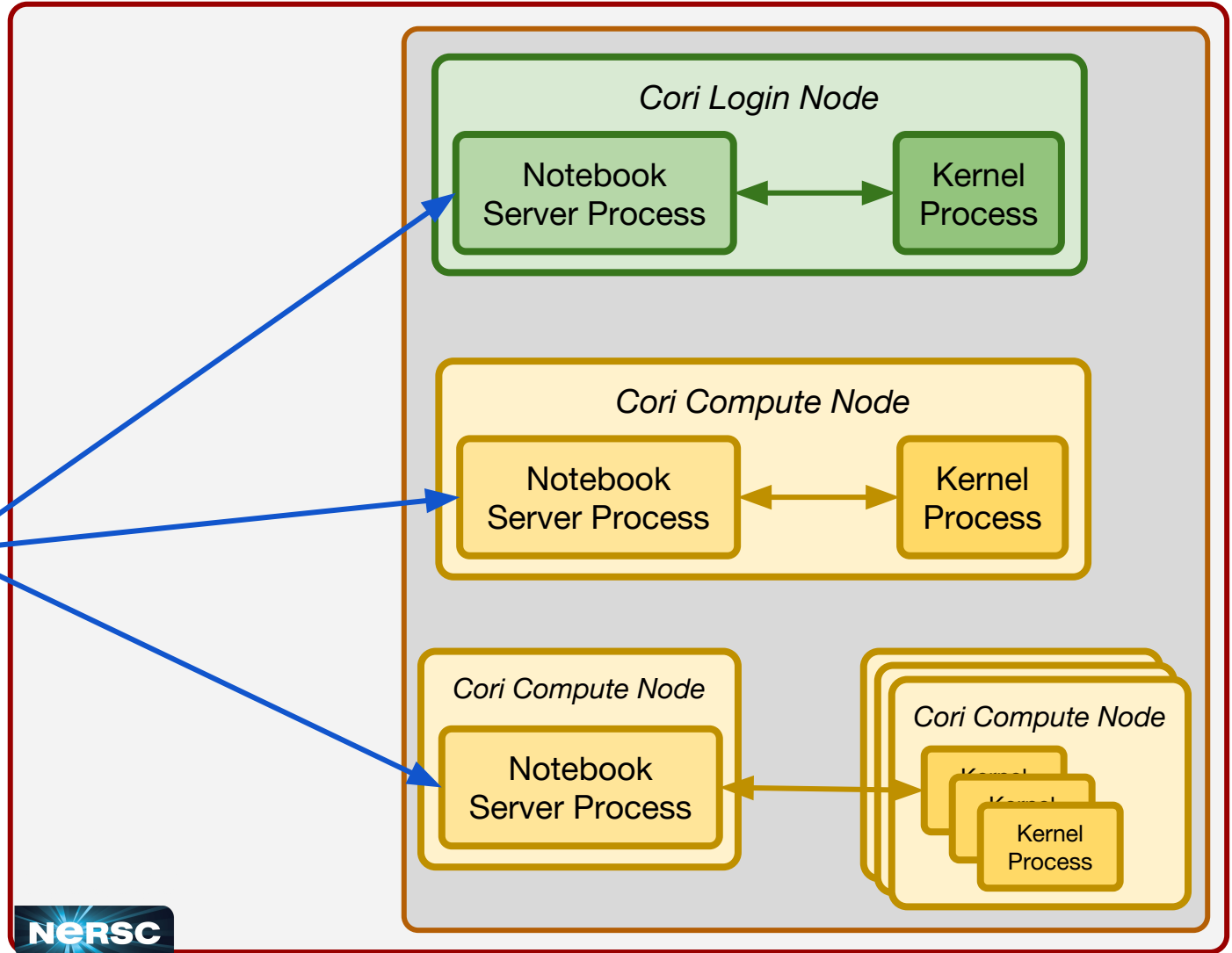
Next: Cori Compute Nodes



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

The screenshot displays a Jupyter notebook interface with several components:

- Workflow Description:** A diagram showing a hierarchical workflow of tasks (represented by colored boxes) connected by arrows. Below it is a text description of "MD Simulations of Nanodroplet Wetting Dynamics" and a 3D visualization of a droplet on a surface.
- Peptide Simulation:** A 3D plot titled "Peptide Simulation" showing a cluster of green and red spheres in a 3D coordinate system (x, y, z). The axes are labeled with values like 100.0, 50.0, 0.0, 100.0, and 120.0. A "Timestamp" slider is visible at the bottom.
- File Teller:** A small window showing a list of files with columns for "File path", "out.txt", and "Inp".
- QueueWidget:** A table showing the status of HPC jobs. The table has columns for user, status, qos, name, timeuse, hostname, jobid, queue, submittime, memory, nodes, timereq, and proc.

user	status	qos	name	timeuse	hostname	jobid	queue	submittime	memory	nodes	timereq	proc	
0	oelvans	PD	debug	big_simula*	0:00	cori	6380576	debug	2017-08-16T15:20:12	0	32	10:00	32
1	oelvans	PD	debug	medium_job	0:00	cori	6380580	debug	2017-08-16T15:20:31	0	5	10:00	5
2	oelvans	PD	debug	medium_job	0:00	cori	6380610	debug	2017-08-16T15:20:40	0	5	10:00	5
3	oelvans	PD	debug	testIn	0:00	cori	6380613	debug	2017-08-16T15:21:01	0	1	10:00	1

- 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

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

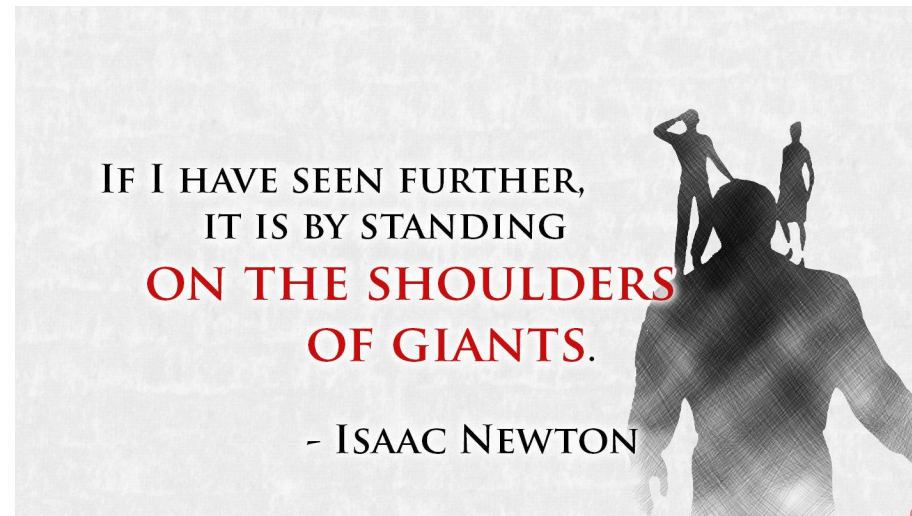
Notebook/scheduler on Haswell, kernels on KNL?

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”