

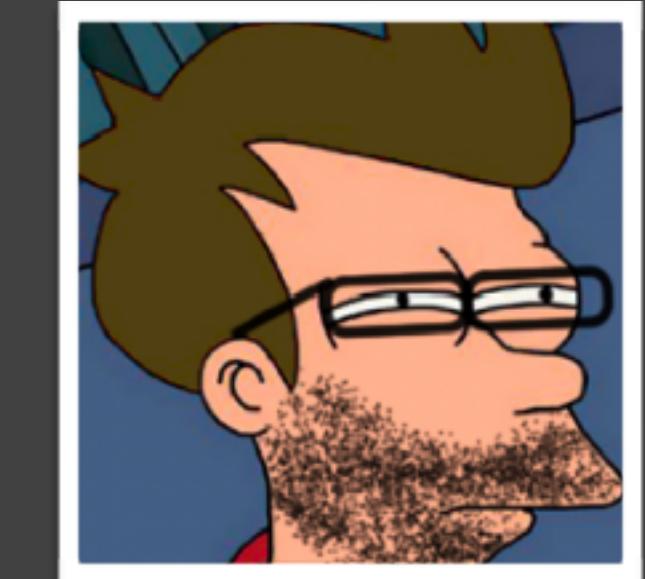
Feb 28th, 2018

**Matthias Bussonnier**

bussonniermatthias@gmail.com

GitHub: @carreau

Twitter: @mbussonn



# Jupyter

## Jupyter in HPC

# 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

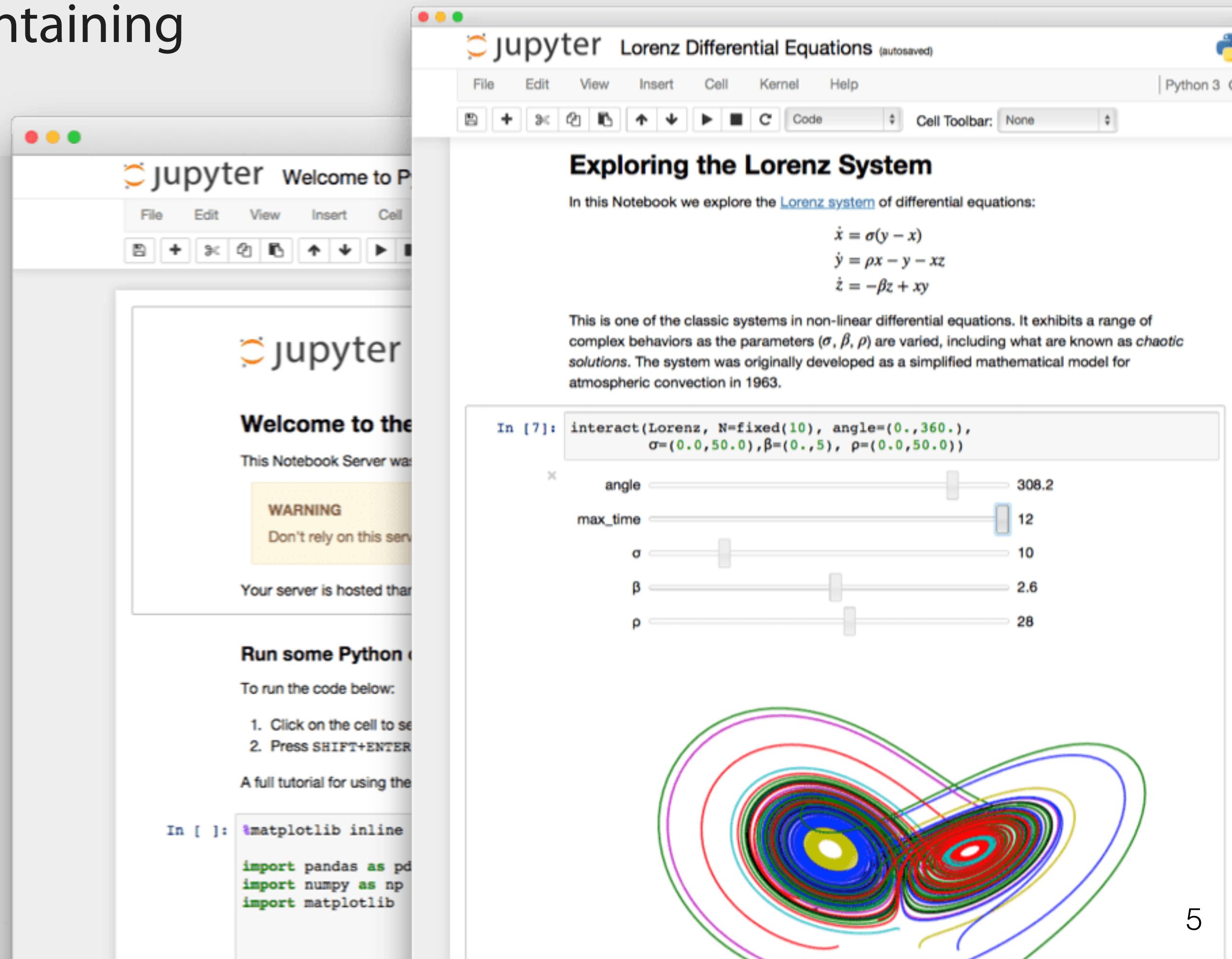


- 2001: Fernando Perez Wrote “**IPython**”
  - Create IPython for Interactive Python with prompt number, gnu plot integration
  - Replace a bunch on perl/make/C/C++ files with only Python.
- 2011: QtConsole
- 2012: Birth of current **Notebook** (6th prototype)
  - Make IPython “network enabled”
  - Made 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



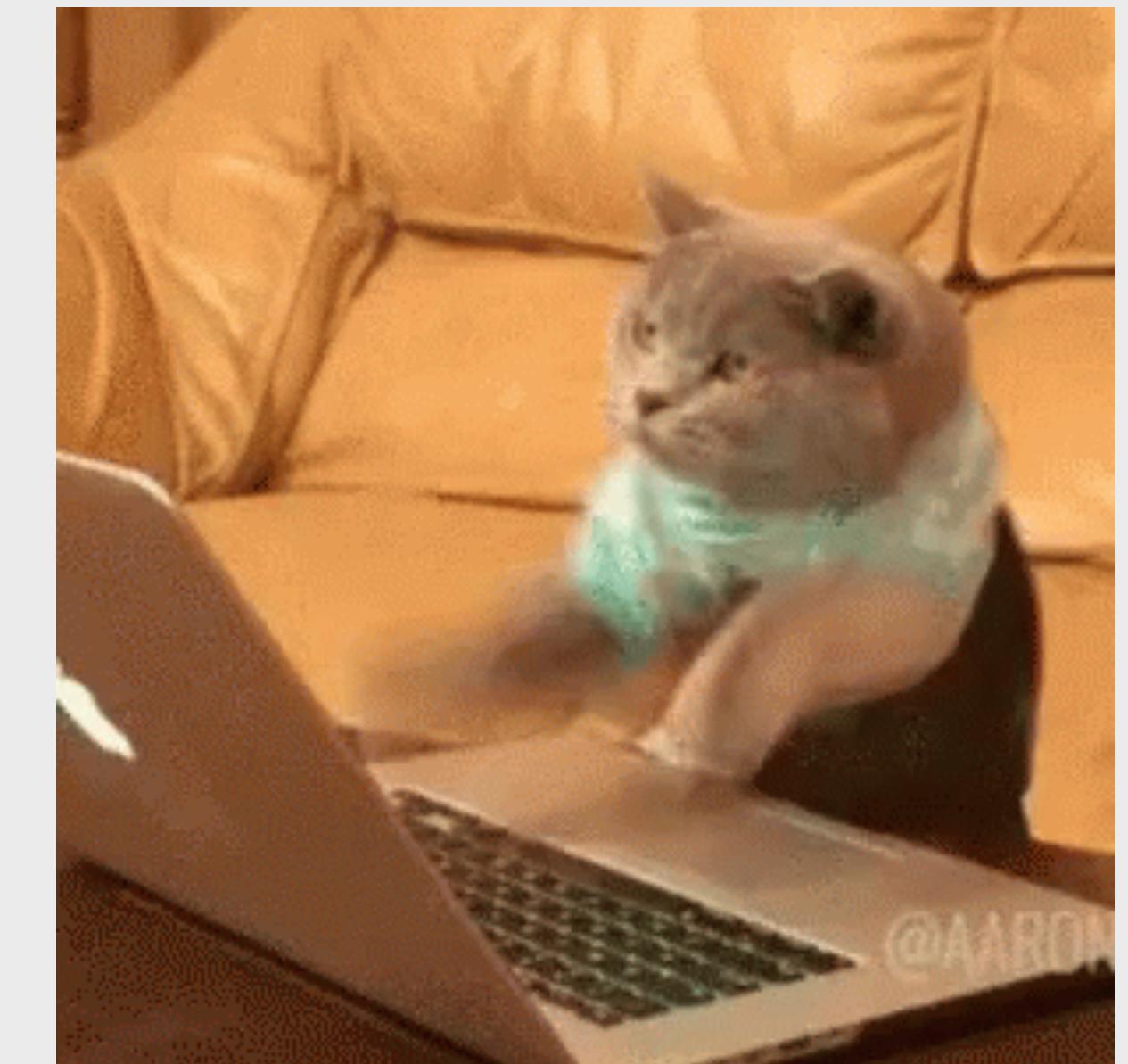
# What is Jupyter

- 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 scroll/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

– 谢国华 师姐

# 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...



demo.py — ~/Desktop/Pythagorean

demo.py

```
11 # <codecell> One line outputs
12 print('Hello World!') Hello World!
13
14 # %% Render LaTeX
15 x, y, z = sp.symbols('x y z')
16 f = sp.sin(x * y) + sp.cos(y * z)
17 sp.integrate(f, x)

$$x \cos(yz) + \begin{cases} 0 & \text{for } y = 0 \\ -\frac{1}{y} \cos(xy) & \text{otherwise} \end{cases}$$

18
19 # In[1]: Display arrays
20 t = np.linspace(0, 20, 500)
21 t
array([ 0.        ,  0.04900015,  0.09816032,  0.12924043, ...]
```

Python 3

# % Plot inline figures  
plt.plot(t, np.sin(t))  
plt.show()

+ Add watch Remove watch

9

# JupyterLab

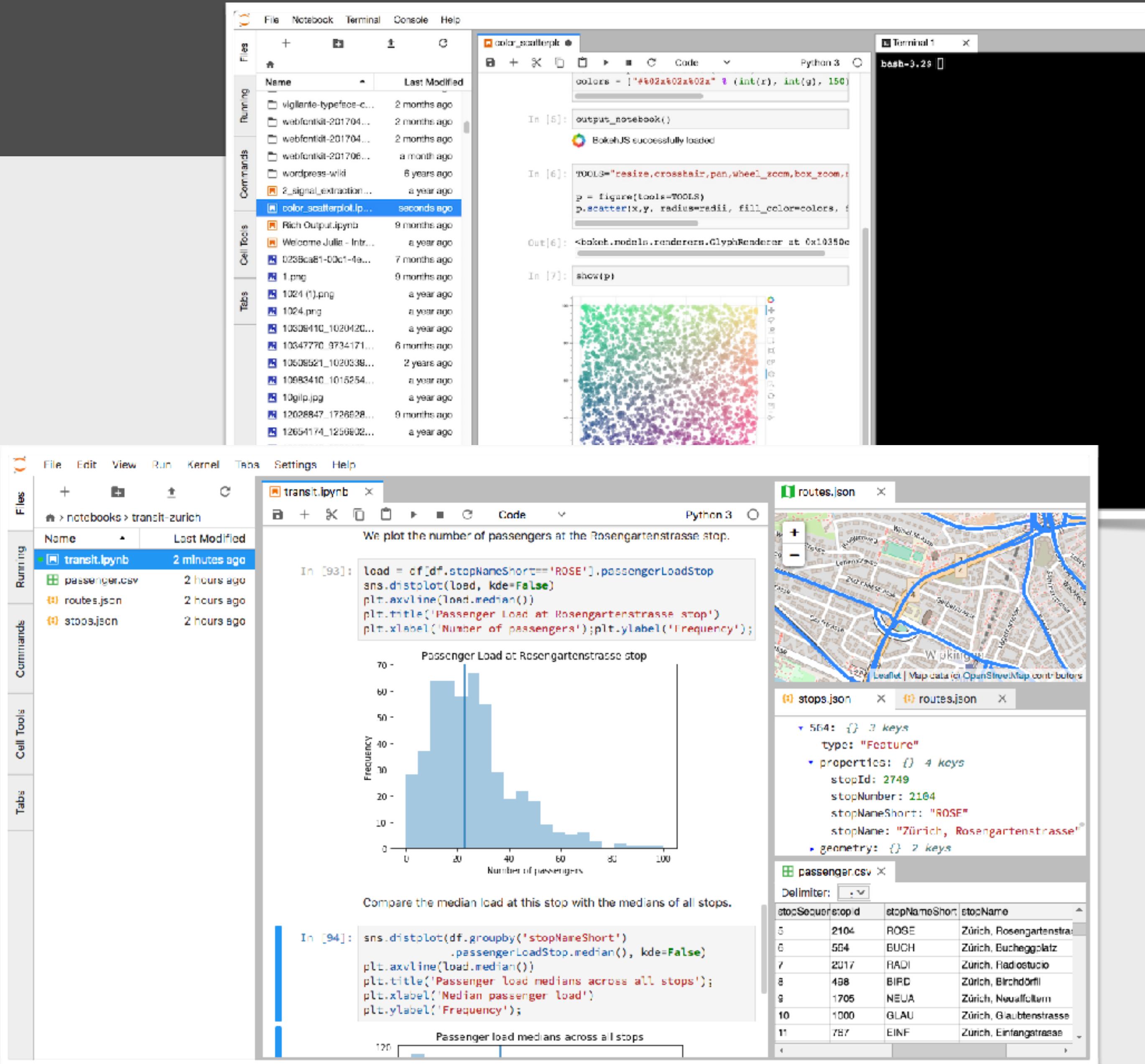
- Extends the notebook interface

with text editor, shell, ...etc

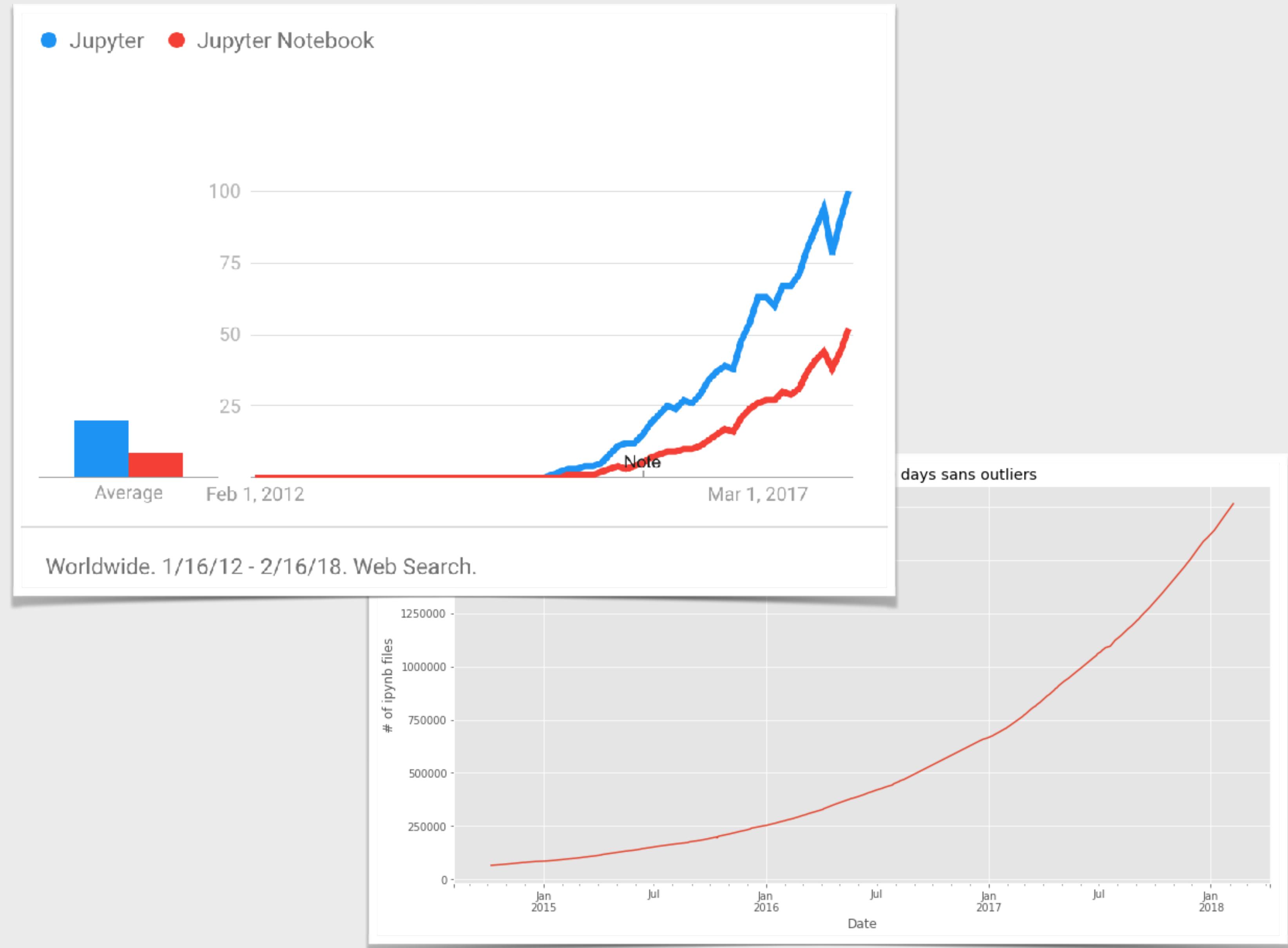
- is it and IDE ?

- If by I you mean Interactive,

then yes



# 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



# 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



Bojan Marković

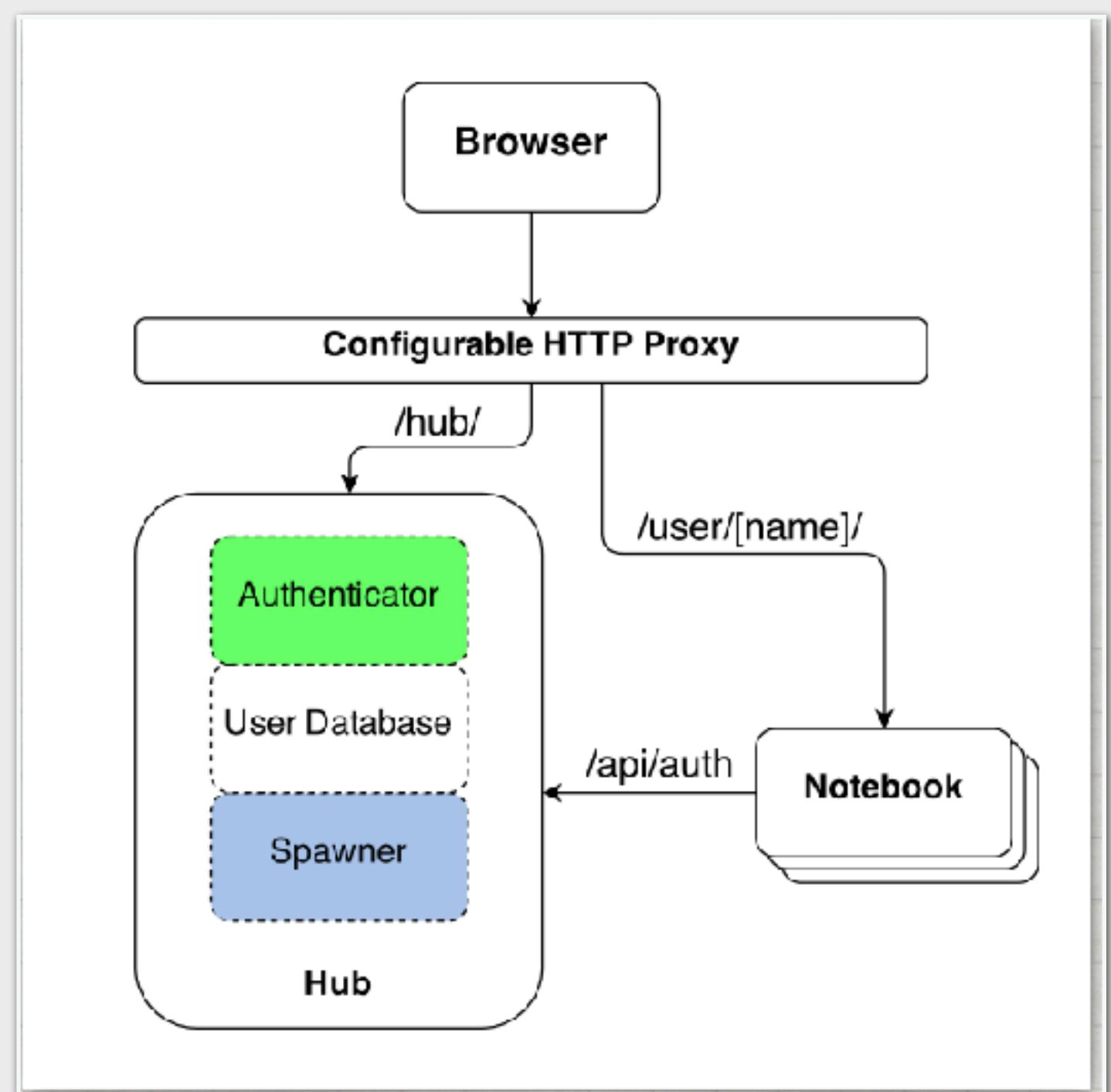
Feb 20

You'll only take Spyder from my cold, dead... Oooooh, 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.

# 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...)



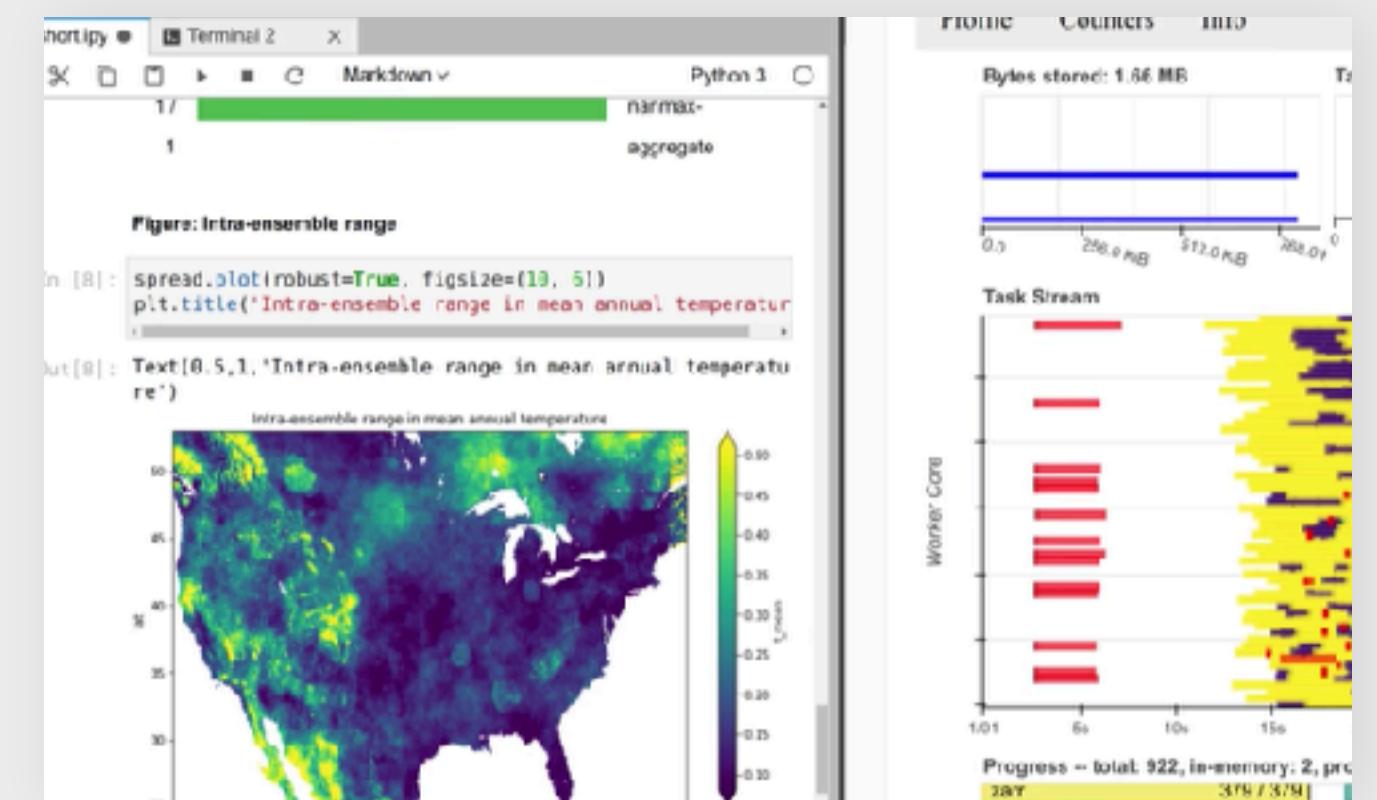
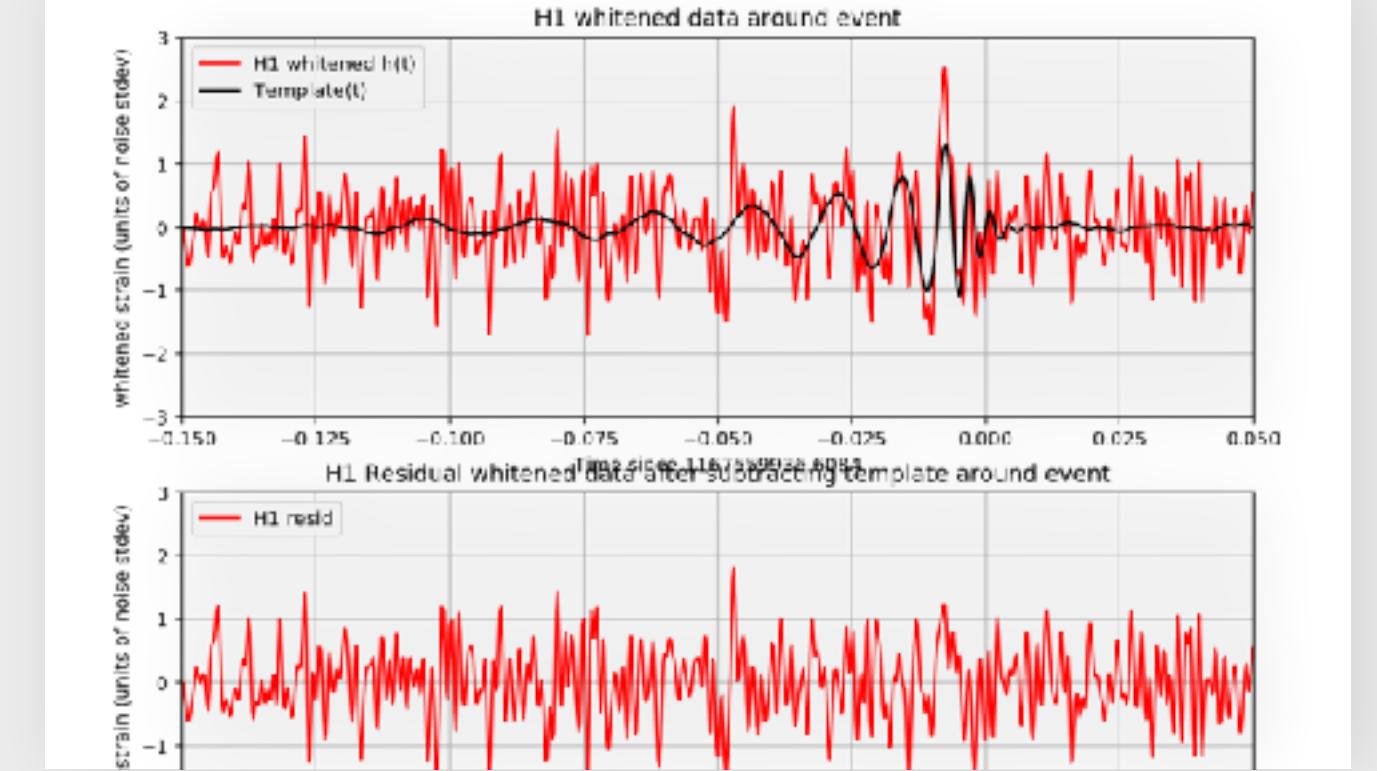
# Use Cases

## 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



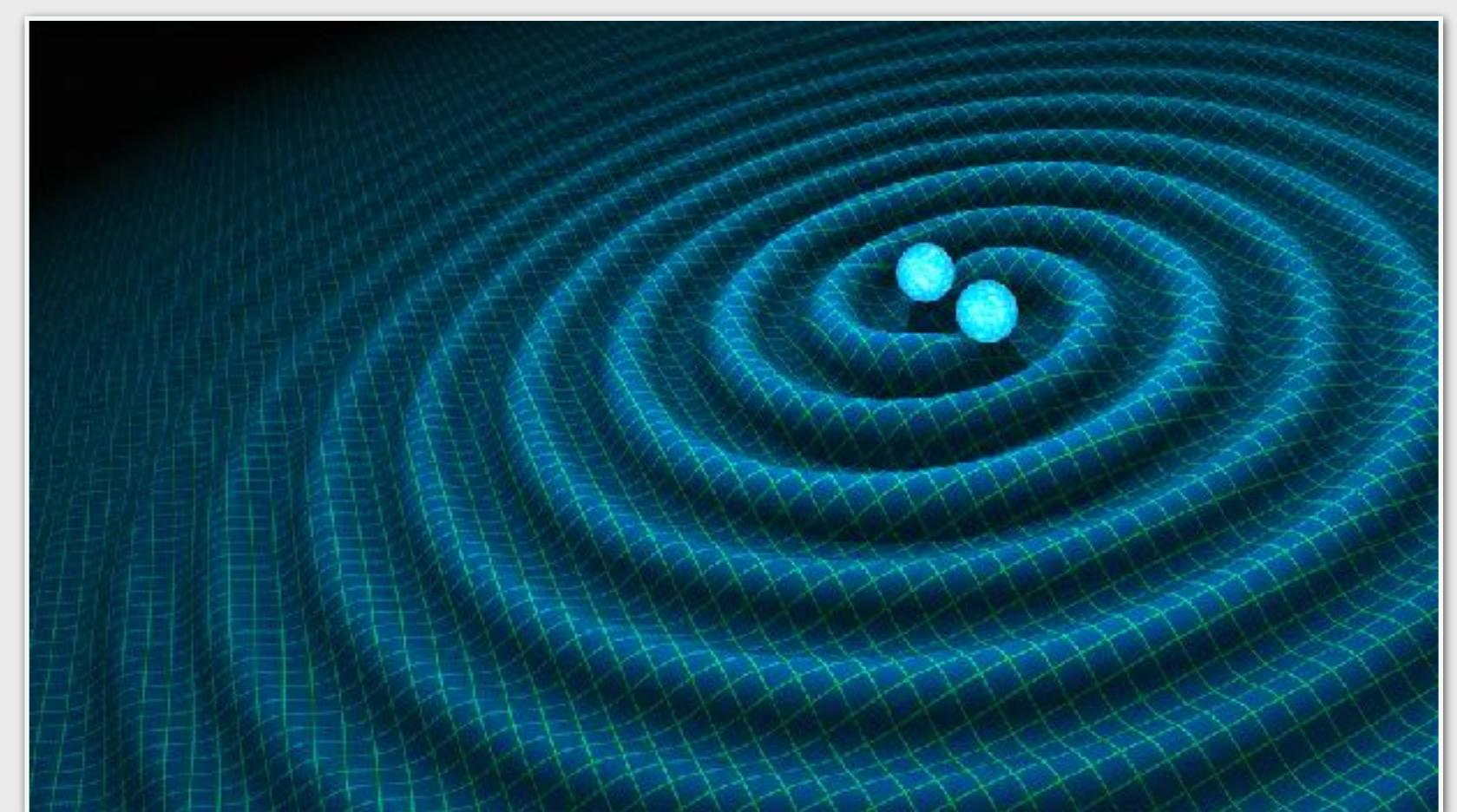
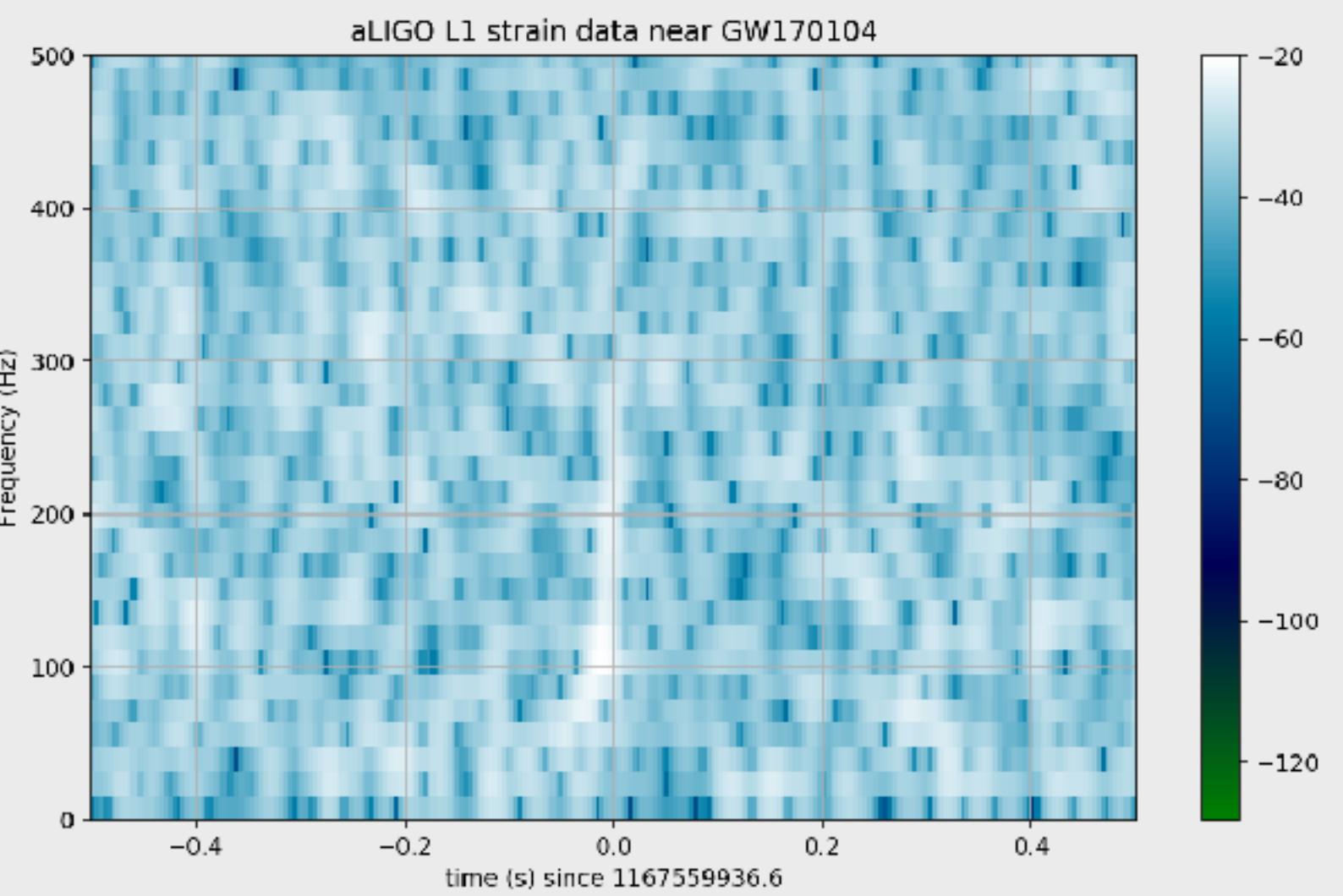
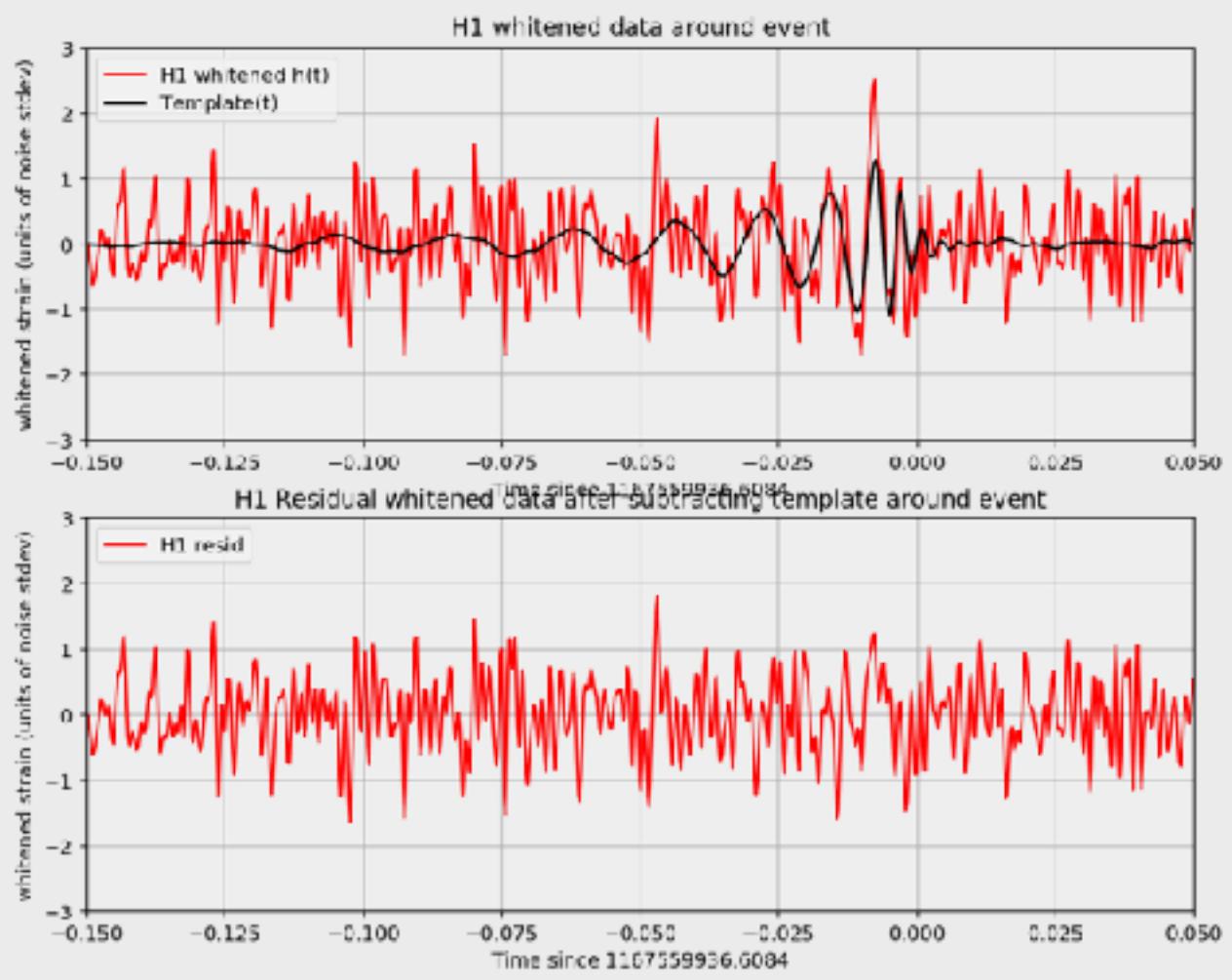
Cern's SWAN

Ligo

Pangeo

# 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

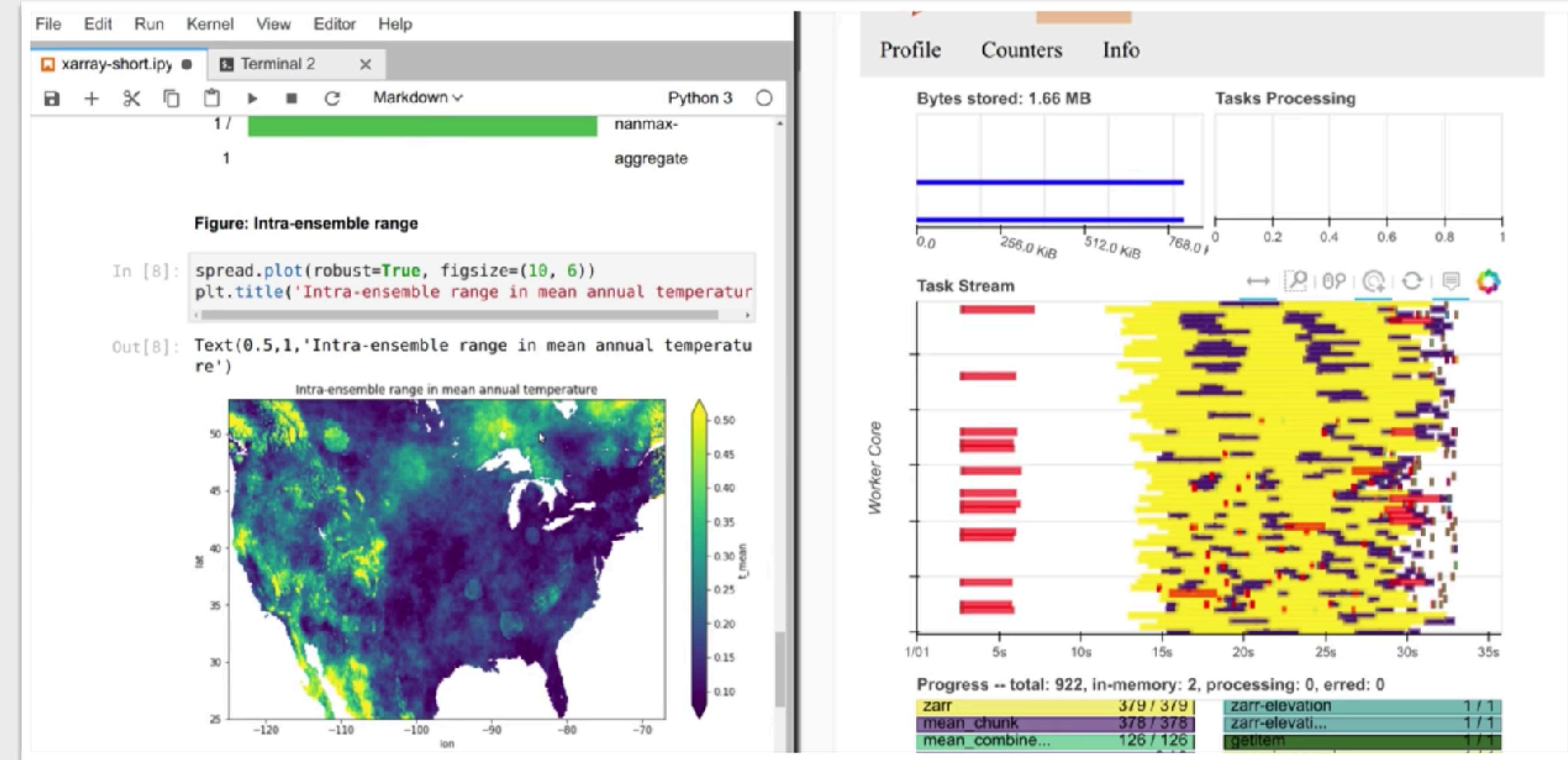
community

- 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

### 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! 0.01/0.01  
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_double = ROOT.B("double")
```



**CMS Opendata: di-muon analysis**

```
In [5]: invMass = ROOT.TH1F("invMass","CMS Opendata: #mu#mu mass;#mu#mu mass [GeV];Events",312, 2, 100)
invMass.Sumw2()
CUT = "sqrt((E1+E2)^2 - ((px1+px2)^2 + (py1+py2)^2 + (pz1+pz2)^2))>=1"
L = ROOT.TCanvas()
dimons.Draw(invMassFormula + " >= invMass",cut,"hist")
e.SetLabel()
e.SetLogy()
e.Draw()
```

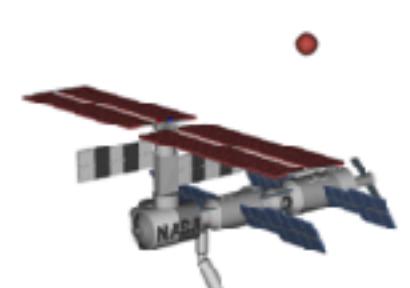
Open in SWAN

CMS Opendata:  $\mu\mu$  mass

invMass  
Entries: 63045  
Mean: 9.966  
Std Dev: 18.82

$\mu\mu \rightarrow \pi\pi \rightarrow$

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


### 3D Visualisation

```
In [1]: auto topVolume = geometry->getTopVolume();
topVolume->Draw();
```

Open in SWAN

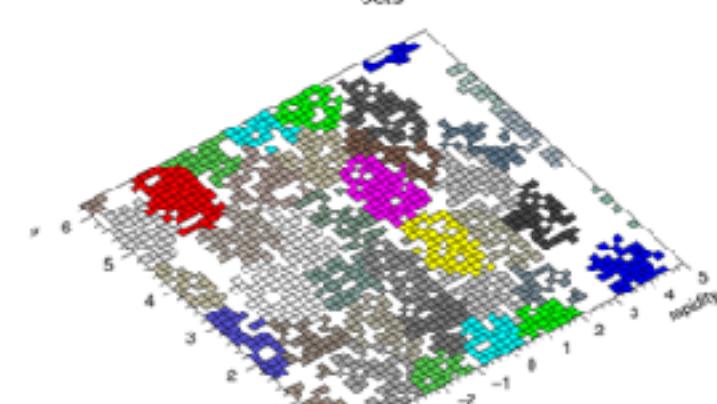
**Interactive usage of 3rd party libraries**

```
#components>forrtl=QUIET 0
1.Dim();
Fastjet release 3.1.1
M. Cacciari, C.P. Salam and G. Soyez
A software package for jet finding and analysis at colliders
http://fastjet.fr

# Please cite EPJ C73 (2012) 1896 [arXiv:1111.6997] if you use this package
# for scientific work and optionally PLB641 (2006) 57 [hep-ph/0512218].
# Fastjet is provided without warranty under the terms of the GNU GPL v2.
# It uses T.~Chen's closest pair algorithm, S.~Fortan's Voronoi code
# and 3rd party plugin jet algorithms. See COPYING file for details.
```

Open in SWAN

Jets







CFP- Ends March 6th



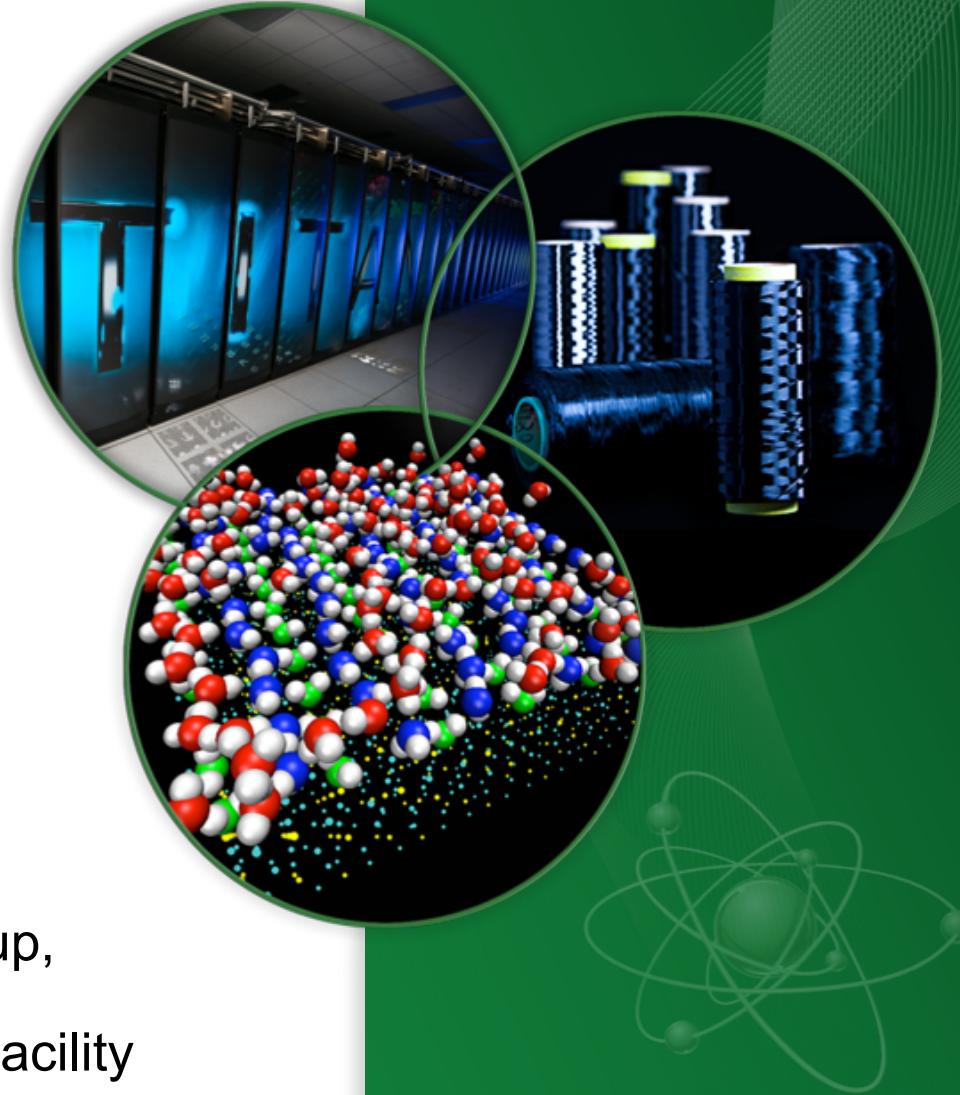
Question(s)  
while we change  
speakers ?



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

Suhas Somnath

Advanced Data and Workflows Group,  
Oak Ridge Leadership Computing Facility

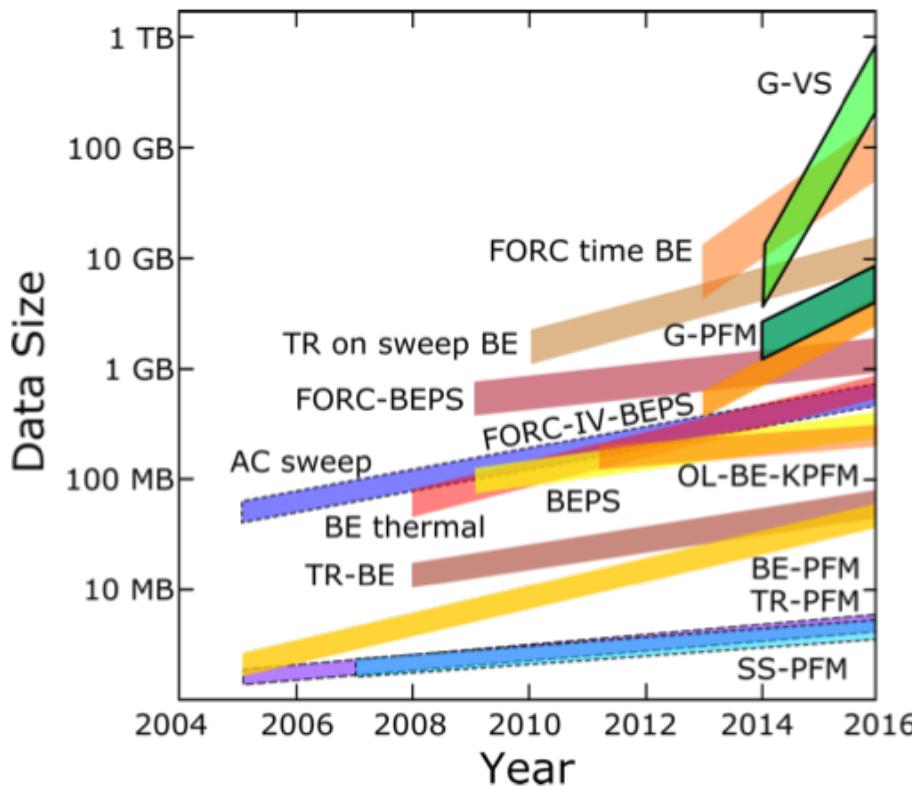


# Opportunities in Computing

- 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

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



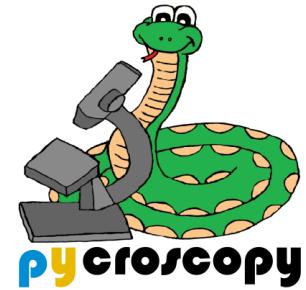
Centralized hub / repository for data pre-processing, analysis



Data ready for interactive visualization + analysis on HPC

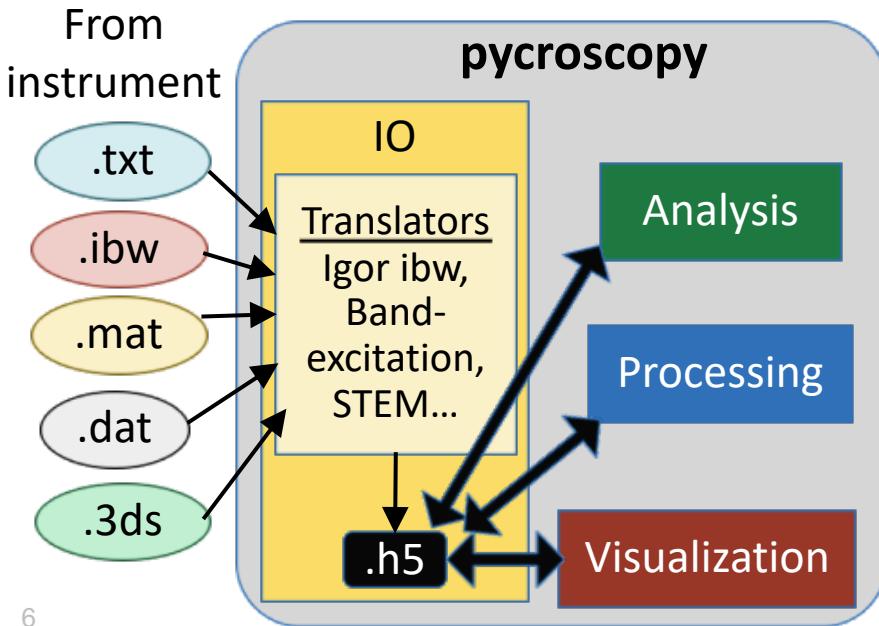
# Pycroscopy

Open-source python package for analyzing + formatting microscopy data



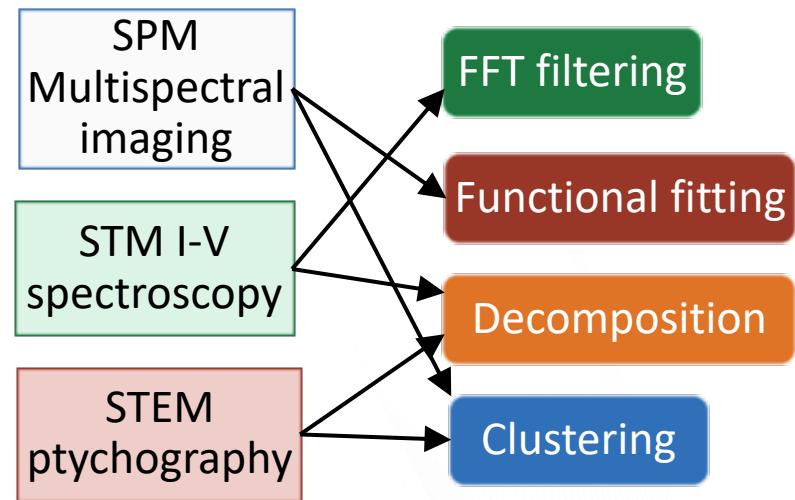
## Universal Data Format

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



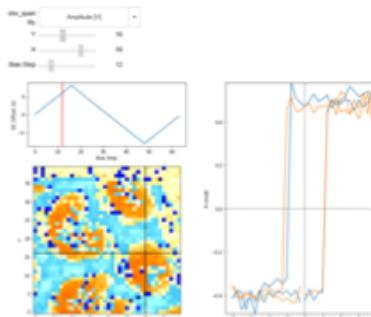
## Instrument agnostic code

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



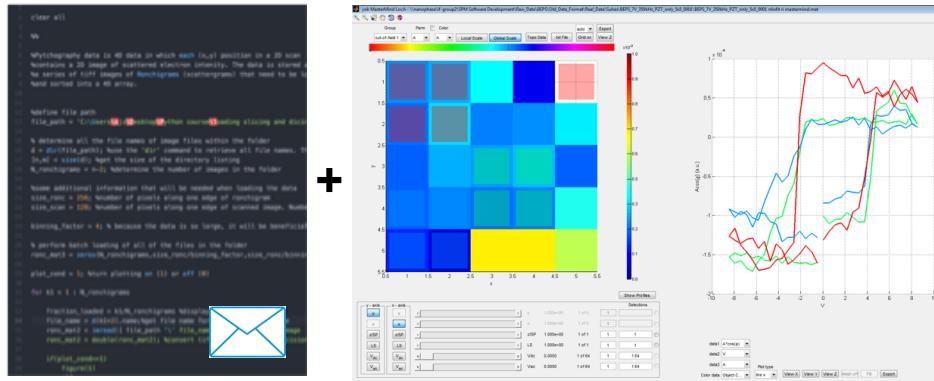
## Conveying information

- Interactive jupyter notebooks

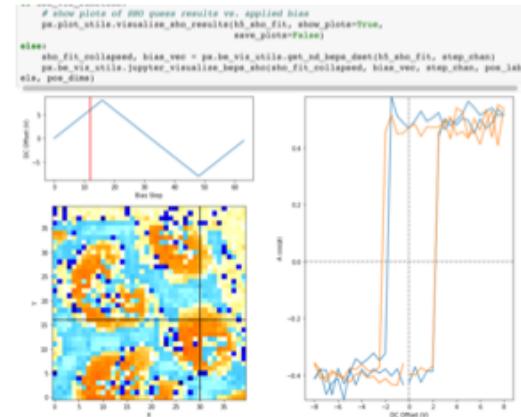


# Supporting User Research

## 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

The screenshot shows a Nature Communications article. The title is "Ultrafast current imaging by Bayesian inversion". The authors listed are S. Somnath, K.J.H. Law, A.N. Morozovska, P. Maksymovych, Y. Kim, X. Lu, M. Alexe, R. Archibald, S.V. Kalinin, S. Jesse, and R.K. Vasudevan. The abstract discusses spectroscopic measurements of current-voltage curves in scanning probe microscopy, highlighting a new approach for dynamic spectroscopic current imaging via full information capture and Bayesian inference, which is three orders of magnitude faster than previous methods.

Jupyter notebook associated with paper

The Jupyter notebook cell contains Python code for visualizing filtering results. It includes a code block and a corresponding plot. The plot is titled "Raw Signal" and shows a log-linear plot of Magnitude (a.u.) versus Bias (V). The plot displays three data series: "Raw" (blue line), "Composite Filter" (orange line), and "Noise Threshold" (red line). The x-axis ranges from 0 to 10 V, and the y-axis ranges from  $10^{-1}$  to  $10^5$  a.u.

```
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_rainbow_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_AO, [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)

The screenshot shows the Oak Ridge National Laboratory Leadership Computing Facility (OLCF) page. The page features the DOI 10.13139/OLCF/1410993 prominently. It includes sections for Authors, a list of names with their email addresses, and the Oak Ridge National Laboratory logo.

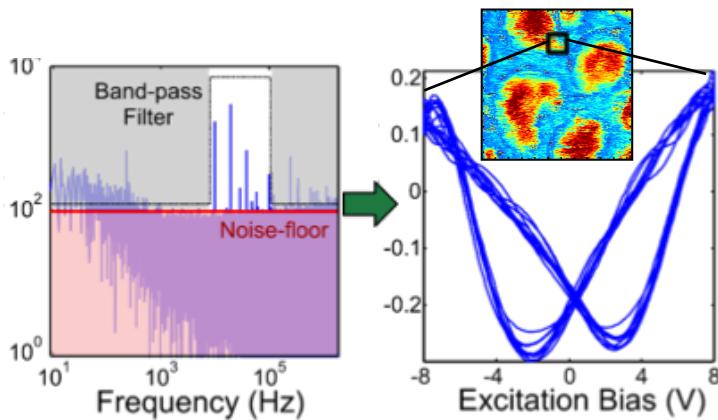
Authors

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, Xiaodi	xlu@xidian.edu.cn
Alexe, Marin	M.Alexe@warwick.ac.uk

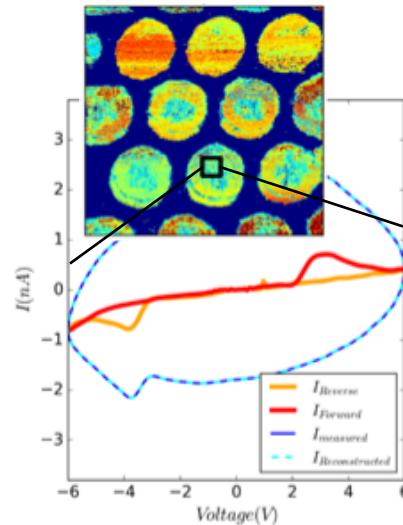
OAK RIDGE National Laboratory

# Scientific Advancements with Jupyter

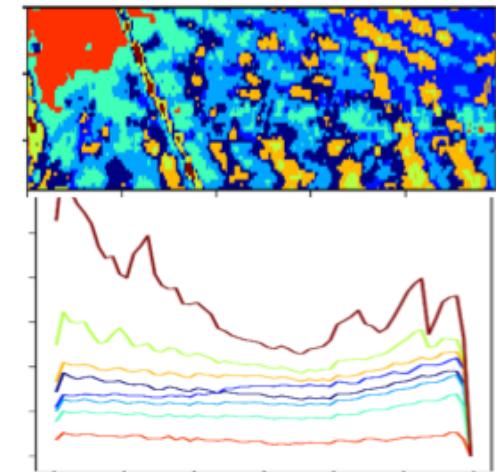
3,500x faster imaging via adaptive signal filtering, linear unmixing of signals



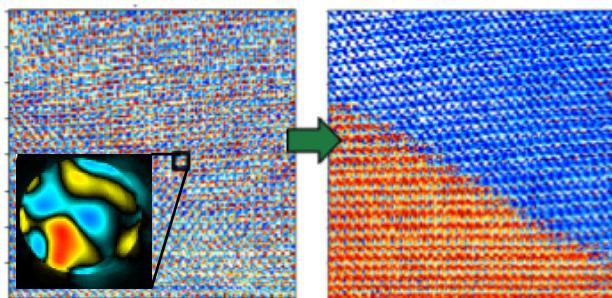
200x faster spectroscopy via Bayesian inference



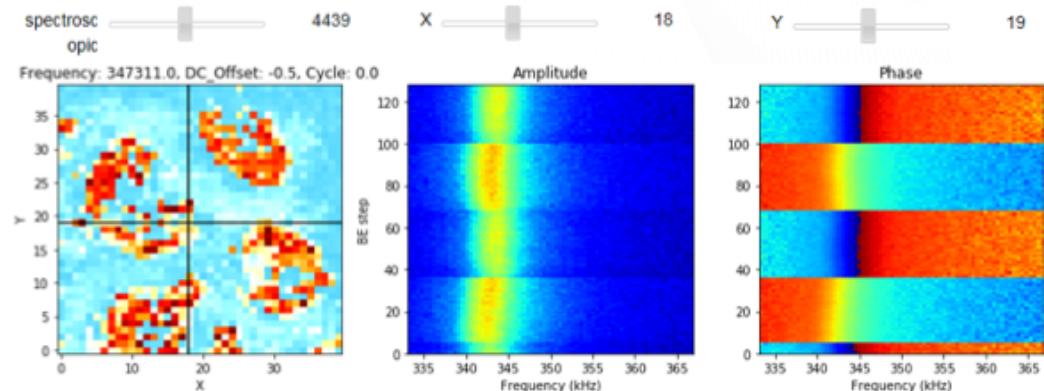
Denoising and clustering to identify superconductivity at the nanoscale



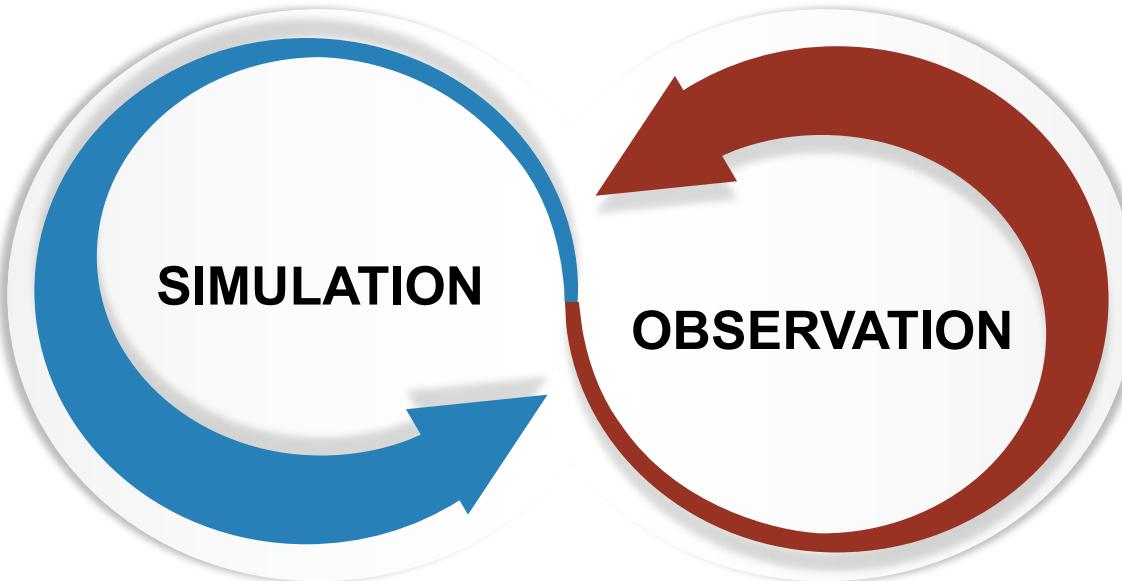
Identifying invisible patterns using multivariate analysis



Simplified navigation multidimensional data - users



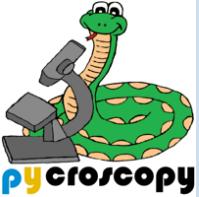
# Completing a Discovery Paradigm



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  
?



**CADES**  
(Cloud + Cluster)



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:

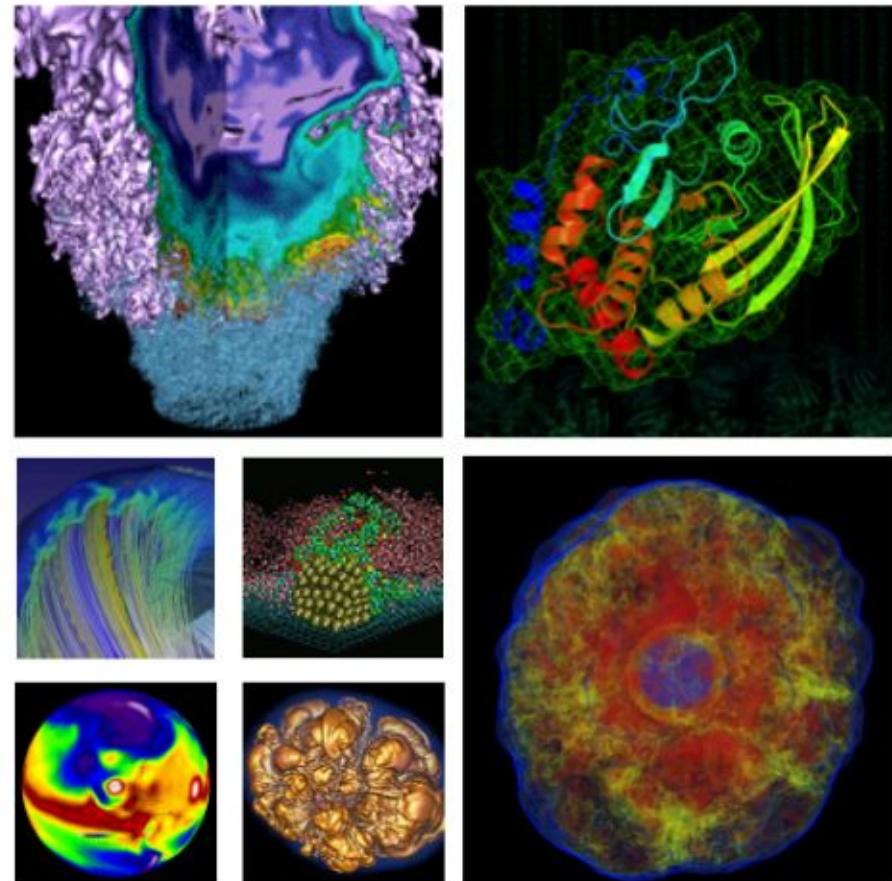
- Sergei V. Kalinin
- Stephen Jesse
- Rama K. Vasudevan

## 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”

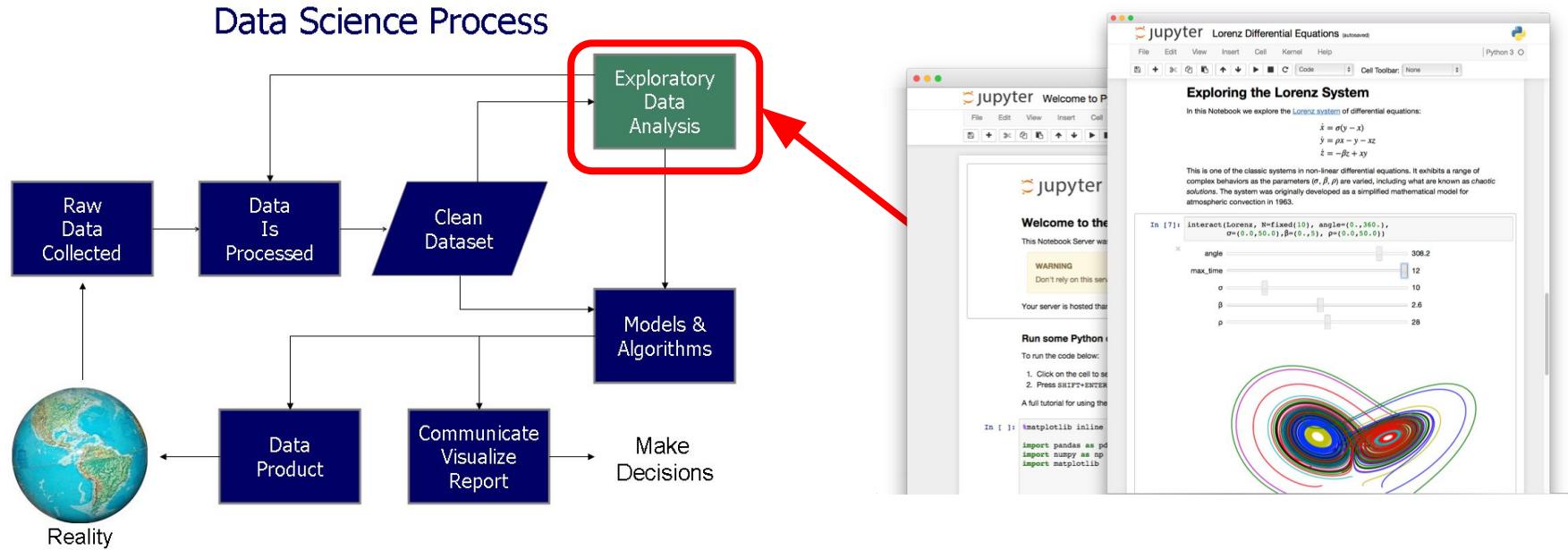


Gerty Cori: Biochemist and first American woman to win a Nobel Prize in science

- 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

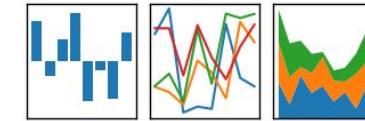
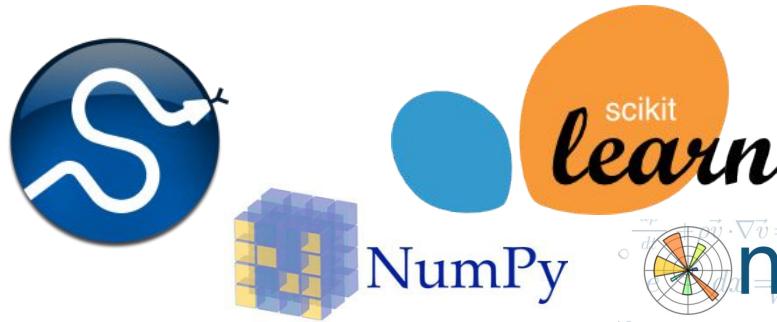


# Enter Jupyter



- **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



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**

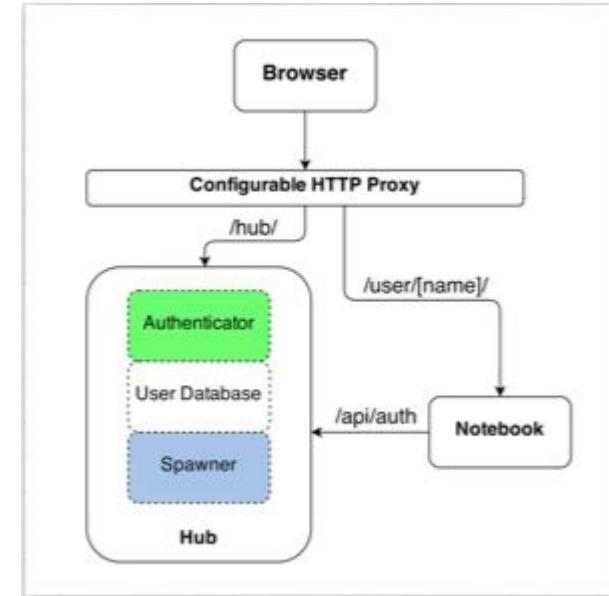
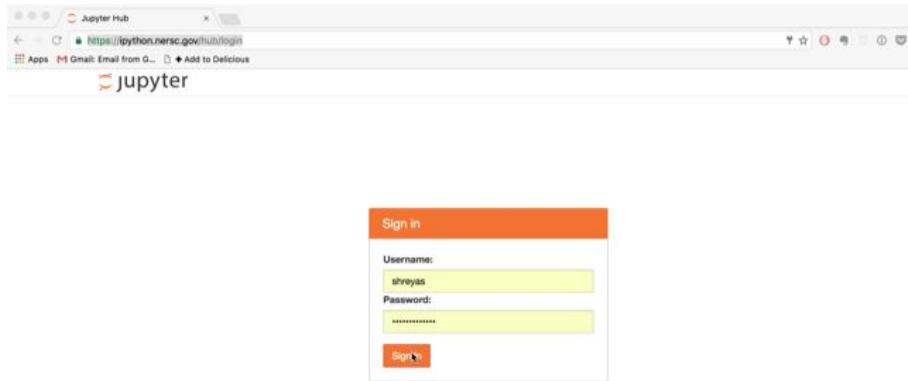
## Jupyterhub to rescue

- ✓ **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





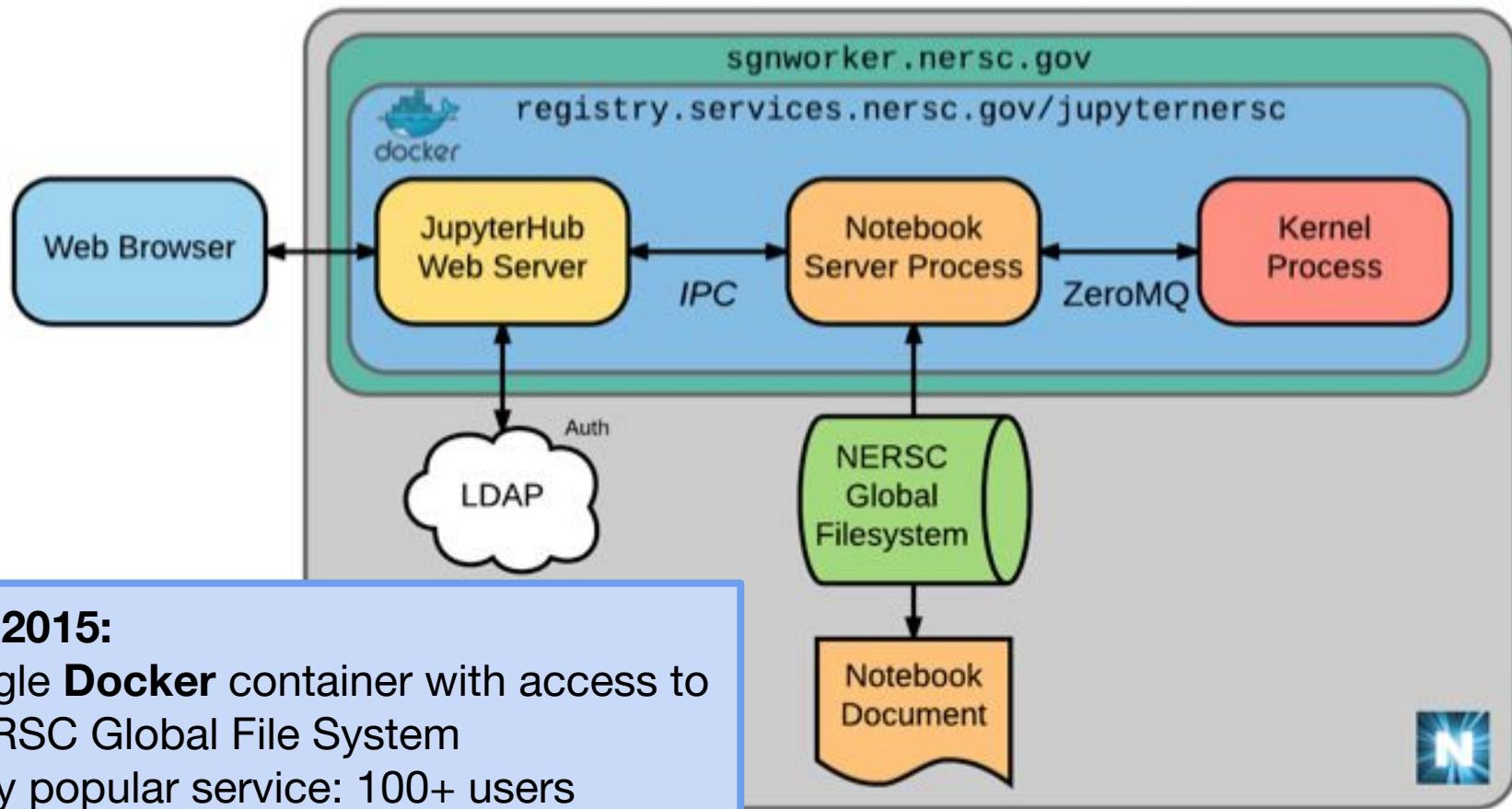
---

# 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  
...

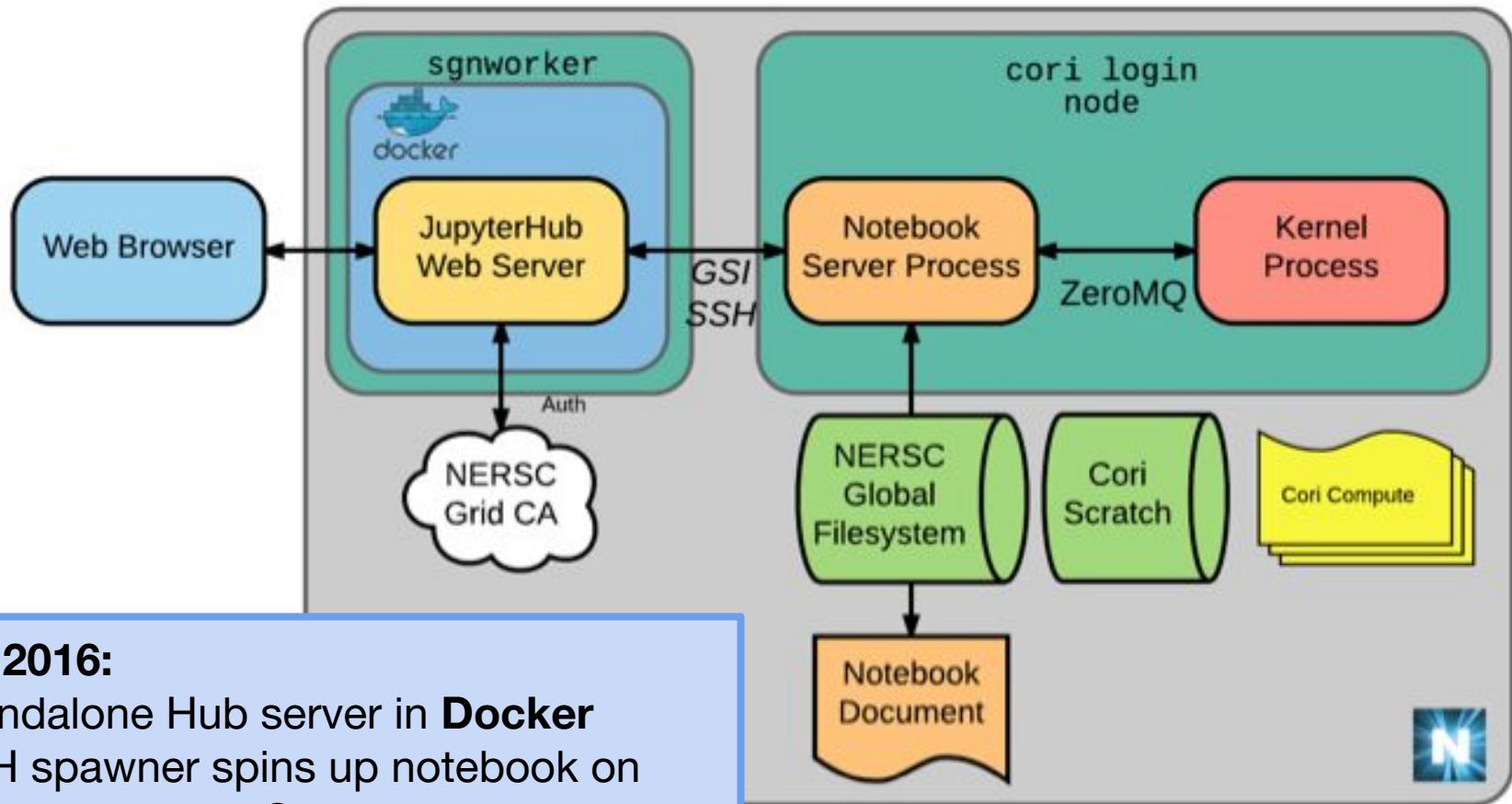
---

# 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`

- 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>

# 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/SSHSawner>

# 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

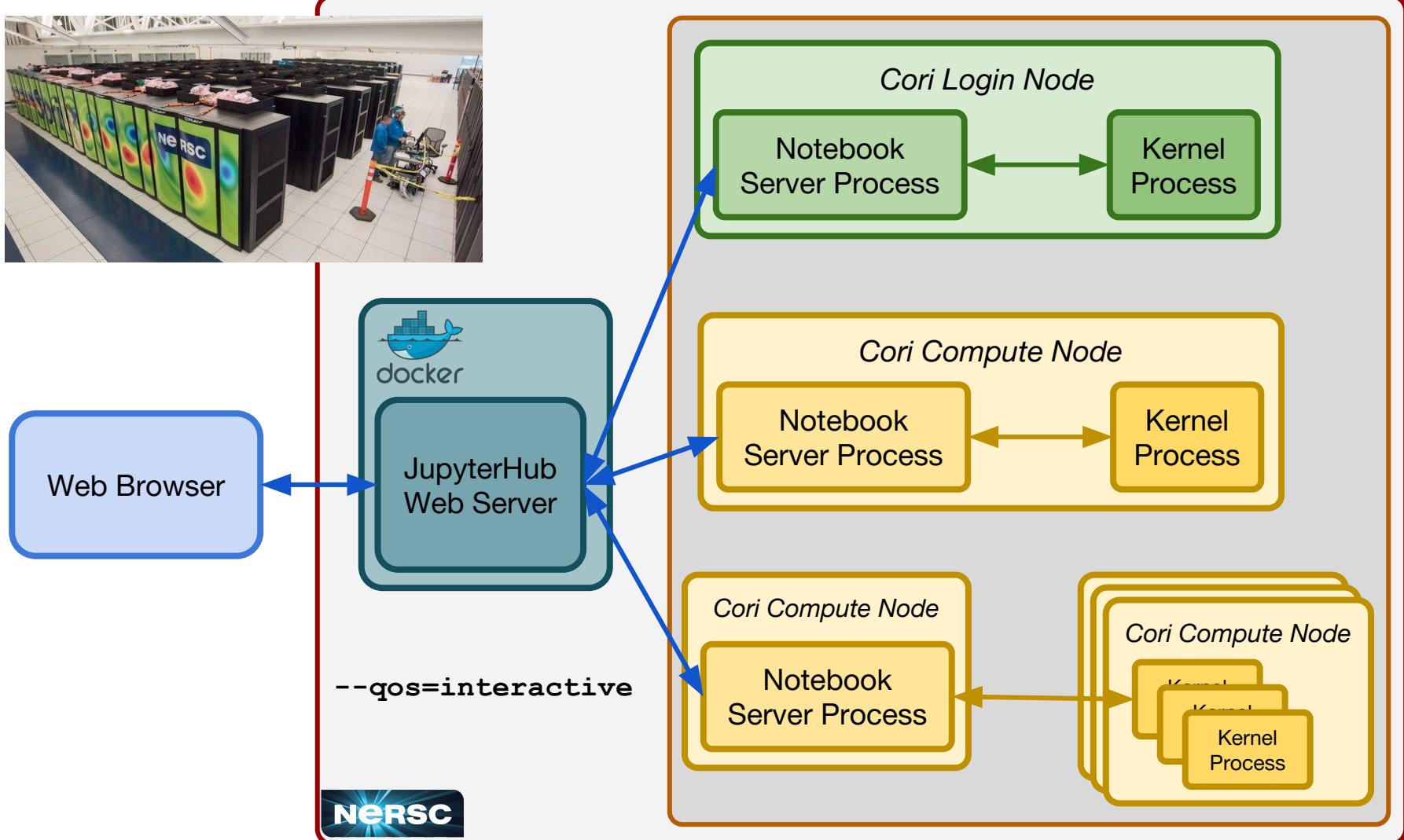
- **Users customize their notebooks with libraries and APIs of their own design or from third parties.**
- **NERSC wants to offer Jupyter to users so they don't set it up themselves in an insecure way.**

---

# Jupyter@NERSC Evolution of Architecture

## Step 3: The Future

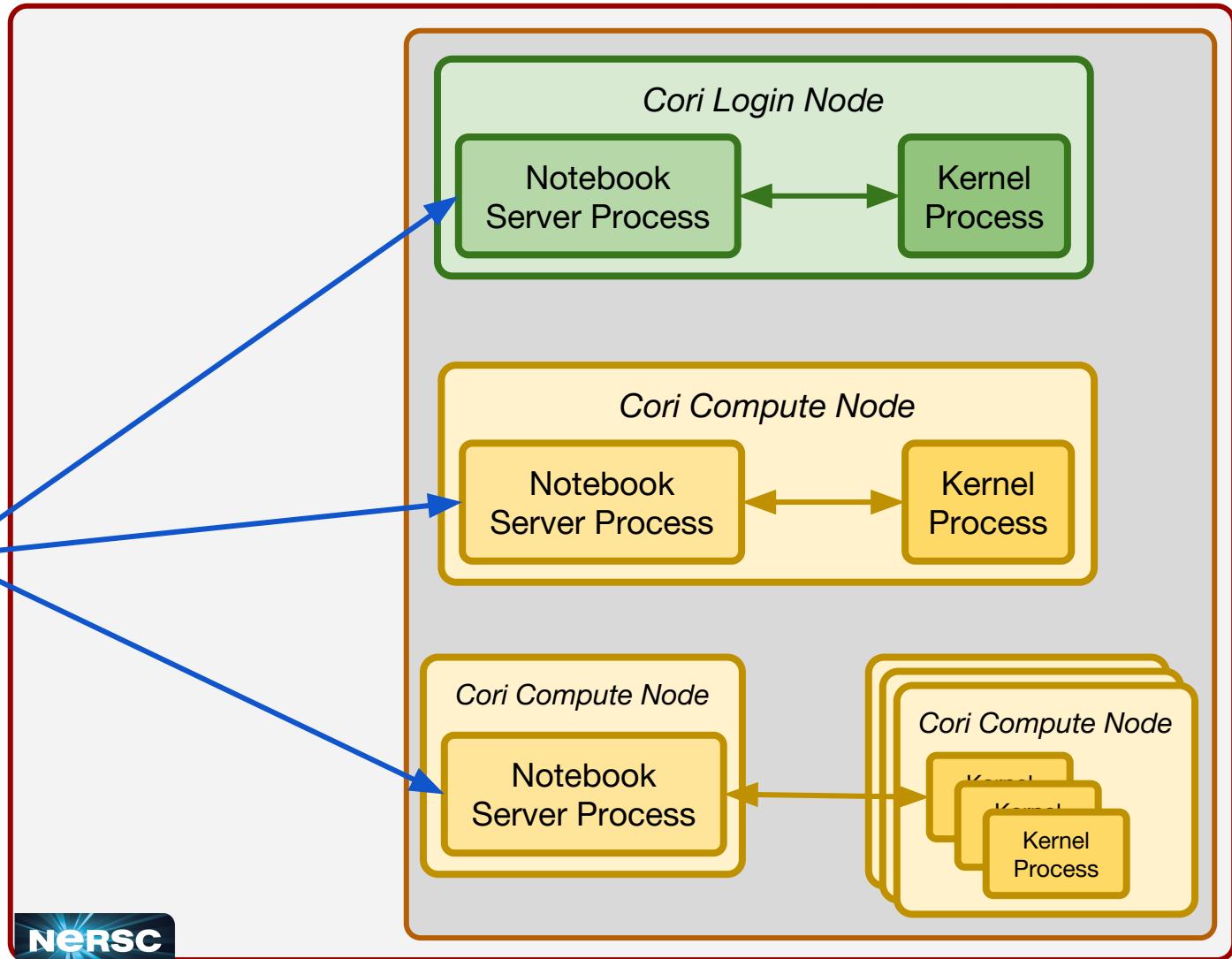
# Next: Cori Compute Nodes



# Role of Software Defined Networking



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



**NERSC**

# 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 figure illustrates the Kale workflow interface, which integrates Jupyter notebooks with HPC management. It shows four main components:

- WorkflowWidget:** A graphical representation of the workflow structure, showing a complex network of tasks and dependencies.
- Workflow Description:** A Jupyter notebook cell for "MD Simulations of Nanodroplet Wetting Dynamics". It contains descriptive text, a visualization of a nanodroplet on a surface, and a code cell with the text: "We find that  $\sigma_{\text{int}} = \sigma_0 \tau^2$  is an accurate model for droplets relative to values of  $\sigma_0$  and  $\tau$  which can be determined from the force radius."
- sim.plot\_steps():** A Jupyter notebook cell for a peptide simulation. It displays a 3D scatter plot titled "Peptide Simulation" showing the positions of peptides over time steps (0 to 120). The plot axes are labeled x, y, and z.
- QueueWidget(a):** A Jupyter notebook cell for managing HPC jobs. It shows a table of current jobs:

User	status	qos	name	timeuse	hostname	jobid	queue	submittime	memory	nodes	timereq	procs
0	oevans	PD	debug	big_simula*	cori	6380576	debug	2017-08-16T15:20:12	0	32	10:00	32
1	oevans	PD	debug	medium_job	cori	6380580	debug	2017-08-16T15:20:31	0	5	10:00	5
2	oevans	PD	debug	medium_job	cori	6380610	debug	2017-08-16T15:20:40	0	5	10:00	5
3	oevans	PD	debug	testtn	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

# 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

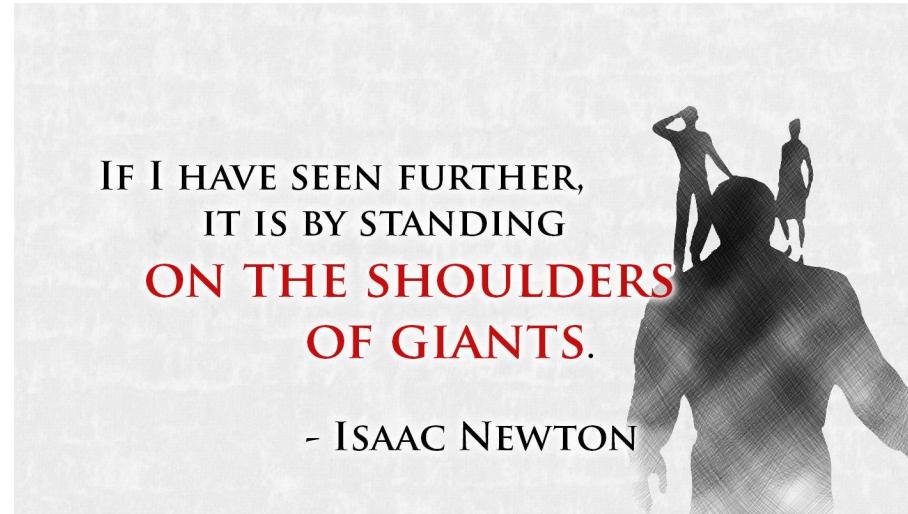
*Notebook/scheduler on Haswell, kernels on KNL?*

# 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”**