



Blue Water Webinar

April 24th,  
2019

Jupyter Team.

Presented by:  
**Matthias Bussonnier**

bussonniermatthias@gmai  
l.com

GitHub: @carreau

Twitter: @mbussonn



Slides <https://github.com/carreau/talks>

# IPython – 2001



```
IPython
$ ipython
Python 3.6.0
Type 'copyright', 'credits' or 'license' for more information
IPython 6.0.0.dev -- An enhanced Interactive Python. Type '?' for help.

In [1]: from string import hexdigits
...: from random import choice
...:
...: def randhex(length=10):
...:     return '0x'+''.join([choice(hexdigits) for x in range(10)]).l
ljust
lower
rstrip
```

(BTW, IPython is uppercase I)

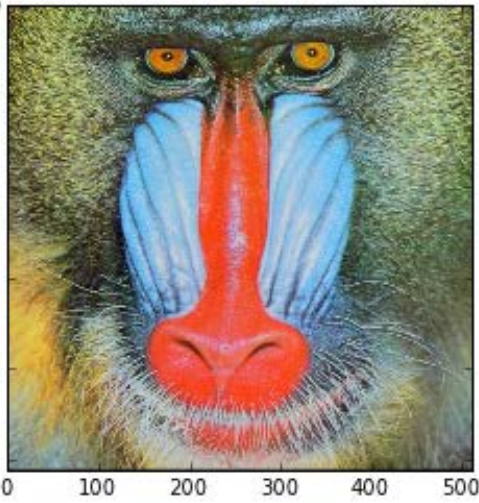


# QtConsole 2010-2011



```
IPython
IPython 0.11 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.
%gui       -> A brief reference about the graphical user interface.

In [1]: imshow(imread("baboon.png"))
Out[1]: <matplotlib.image.AxesImage at 0x9fe274c>
```



```
In [2]:
```

The screenshot shows a terminal window titled "IPython" with a white background and a dark grey border. The window contains text output from the IPython shell. The first part shows help text for various commands. The second part shows the execution of a command to load and display an image of a baboon's face. The image is displayed in a plot with x and y axes ranging from 0 to 500. The baboon's face is the central focus, showing its characteristic blue and red facial features. The plot has a white background and black axes. The text "In [2]:" is visible at the bottom of the window.

# The IPython Notebook – 2012



IPython Notebook Spectrogram Save Idle

Notebook

Actions: New, Open, Download, ipynb, Print

Cell

Actions: Delete

Format: Code, Markdown

Output: Toggle, ClearAll

Insert: Above, Below

Move: Up, Down

Run: Selected, All

Autoindent:

Kernel

Actions: Interrupt, Restart

Kill kernel upon exit:

Help

Links: Python, IPython, NumPy, SciPy, MPL, SymPy

Shift-Enter: run selected cell  
Ctrl-Enter: run in terminal mode  
Ctrl-m h: show keyboard shortcuts

## Simple spectral analysis

An illustration of the [Discrete Fourier Transform](#)

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \quad k = 0, \dots, N-1$$

using windowing, to reveal the frequency content of a sound signal.  
We begin by loading a datafile using SciPy's audio file support:

```
In [1]: from scipy.io import wavfile
rate, x = wavfile.read('/home/fperez/teach/py4science/book/examples/test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin specgram routine:

```
In [3]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');
```

The figure displays two plots side-by-side. The left plot, titled 'Raw audio signal', shows a blue waveform of a sound signal over time, with the x-axis ranging from 0 to 50,000 and the y-axis from -10,000 to 8,000. The right plot, titled 'Spectrogram', shows the frequency content of the signal over time, with the x-axis ranging from 0 to 25,000 and the y-axis from 0.0 to 1.0. The spectrogram uses a color scale from blue (low intensity) to red (high intensity) to represent the magnitude of the signal at different frequencies and times.





# The IPython Notebook – 2012



IPython Notebook Spectrogram Save Idle

Notebook

Actions: New, Open, Download, ipynb, Print

Cell

Actions: Delete

Format: Code, Markdown

Output: Toggle, ClearAll

Insert: Above, Below

Move: Up, Down

Run: Selected, All

Autoindent:

Kernel

Actions: Interrupt, Restart

Kill kernel upon exit:

Help

Links: Python, IPython, NumPy, SciPy, MPL, SymPy

Shift-Enter: run selected cell  
Ctrl-Enter: run in terminal mode  
Ctrl-m h: show keyboard shortcuts

## Simple spectral analysis

An illustration of the [Discrete Fourier Transform](#)

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \quad k = 0, \dots, N-1$$

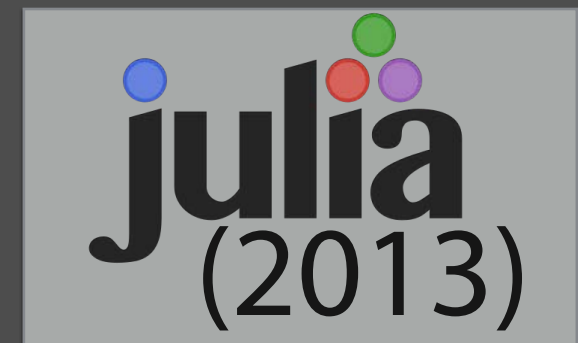
using windowing, to reveal the frequency content of a sound signal.  
We begin by loading a datafile using SciPy's audio file support:

```
In [1]: from scipy.io import wavfile
rate, x = wavfile.read('/home/fperez/teach/py4science/book/examples/test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin specgram routine:

```
In [3]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');
```

The figure contains two subplots. The left subplot, titled 'Raw audio signal', shows a blue waveform of an audio signal over time, with the x-axis ranging from 0 to 50,000 and the y-axis from -10,000 to 8,000. The right subplot, titled 'Spectrogram', shows a heatmap of the signal's frequency content over time, with the x-axis from 0 to 25,000 and the y-axis from 0.0 to 1.0. The spectrogram shows a series of horizontal lines of varying intensity, indicating the presence of different frequencies over time.



# Jupyter – 2014

Renames the Python Agnostic Part to “Jupyter” – an homage to Galileo first Notebooks.



The image displays two overlapping screenshots of the Jupyter web interface. The background screenshot shows the "Welcome to the Jupyter Notebook Server" page, which includes a "WARNING" message: "Don't rely on this server" and instructions on how to run Python code. The foreground screenshot shows a notebook titled "Exploring the Lorenz System" (autosaved). The notebook content includes the following text:

In this Notebook we explore the [Lorenz system](#) of differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters  $(\sigma, \beta, \rho)$  are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

The notebook also shows 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))
```

Below the code cell, there are interactive sliders for the parameters: angle (308.2), max\_time (12),  $\sigma$  (10),  $\beta$  (2.6), and  $\rho$  (28). At the bottom of the notebook, a plot of the Lorenz attractor is shown, featuring multiple overlapping trajectories in various colors.





# Jupyter: 2019



1000+ Contributors

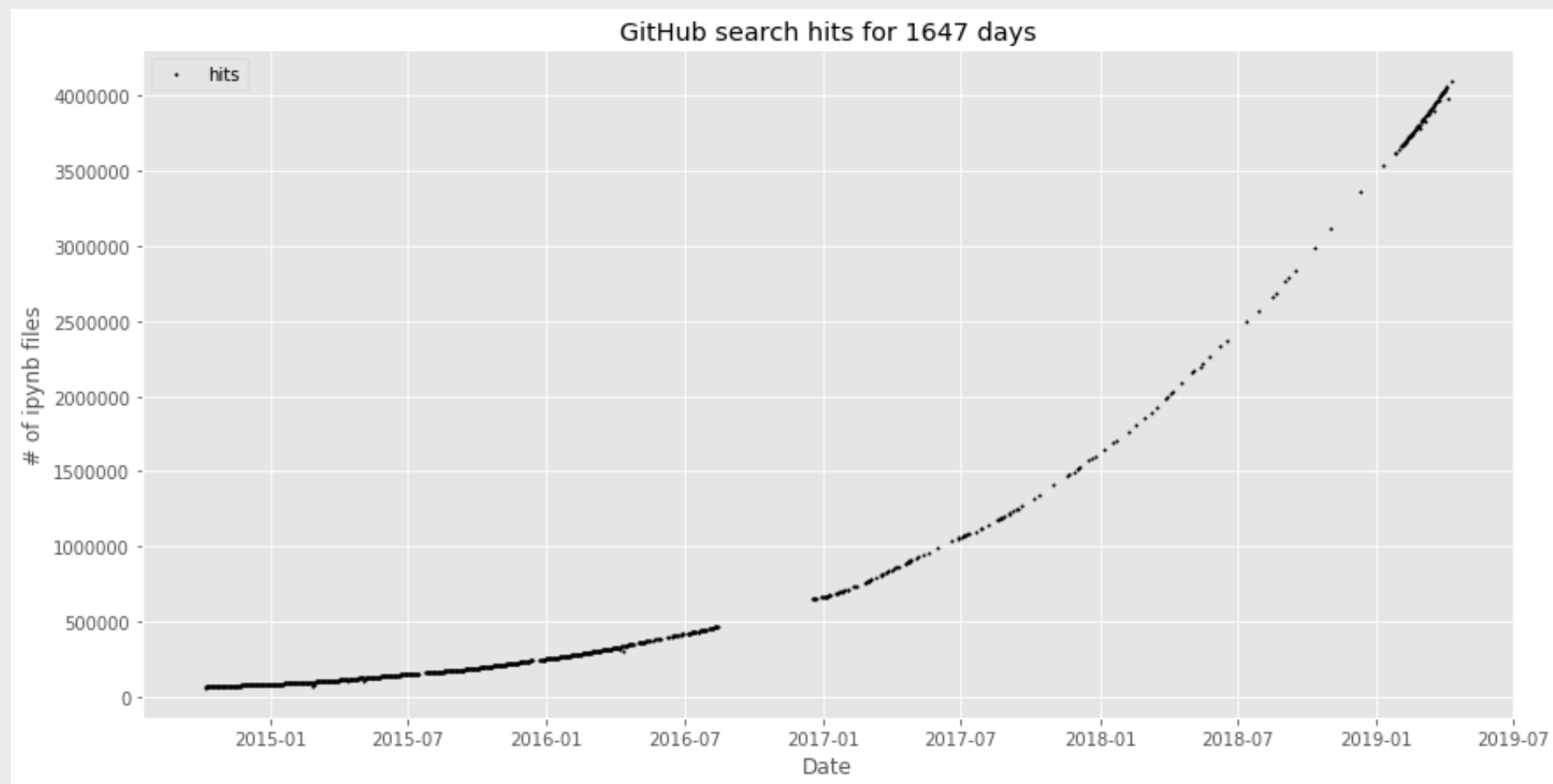
Dozens of Projects



2017 ACM System Software Award

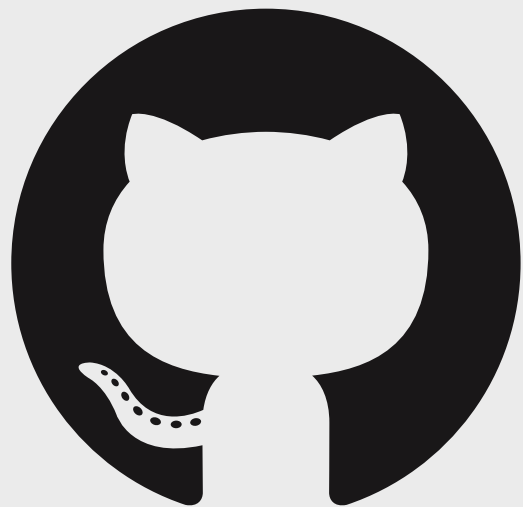
# A few Numbers

~4M on GitHub



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





150+ repositories across multiple organizations  
(IPython, Jupyter, JupyterHub, JupyterLab, ..  
at 2 release/year that's ~ 1 release per day

1000+ Contributors

8+ Millions Users,  
(with conservative estimates)

Worldwide ~21M developers – North America ~4.4M  
VS Code ~2.6 M Active Users  
GitHub 24M Users

# Core Contributors



- 1000+ Open source contributors, Majority Volunteer
- Organization with Open Governance (currently restructuring)





# Sponsors



# How Jupyter came to be



# Life cycle of a Scientific Idea

- **Individual** exploratory work (Repl, Scripts)
- **Collaborative** development (Dropbox/ Google Doc / emails / git )
- **Parallel** production runs (MPI, rewrite C++, batch jobs)
- **Publication** & communication (Word, Latex, ppt...)
- **Education**
- Goto 1





# Tools Overhead

Each Tool brings (cognitive) overhead, time to install, deploy, and master.

Can we create a (set of) tools, with minimal overhead and enough flexibility ?

Parallel with popular DataScience languages

Fortran/C/C++ are fast, but take significant development time and skills

Python/R/Julia are (usually) slower, but are useful immediately.



# Rise of Jupyter

- An increasing number of disciplines have a fast growing amount of data
- Technology is **a tool** that should
  - Empower **the User**
  - Amplify **Domain Knowledge and Expertise**
  - Facilitate **Sharing and Collaboration**

**Jupyter provide a framework that can be use in all the step in the cycle of a scientific idea**

- BSD Licensed (Free to use and redistribute even Commercially)
- Open Source, Community Maintained
- Important for sustainability, diversity, and equal access



# Life cycle of a Scientific Idea in 2020

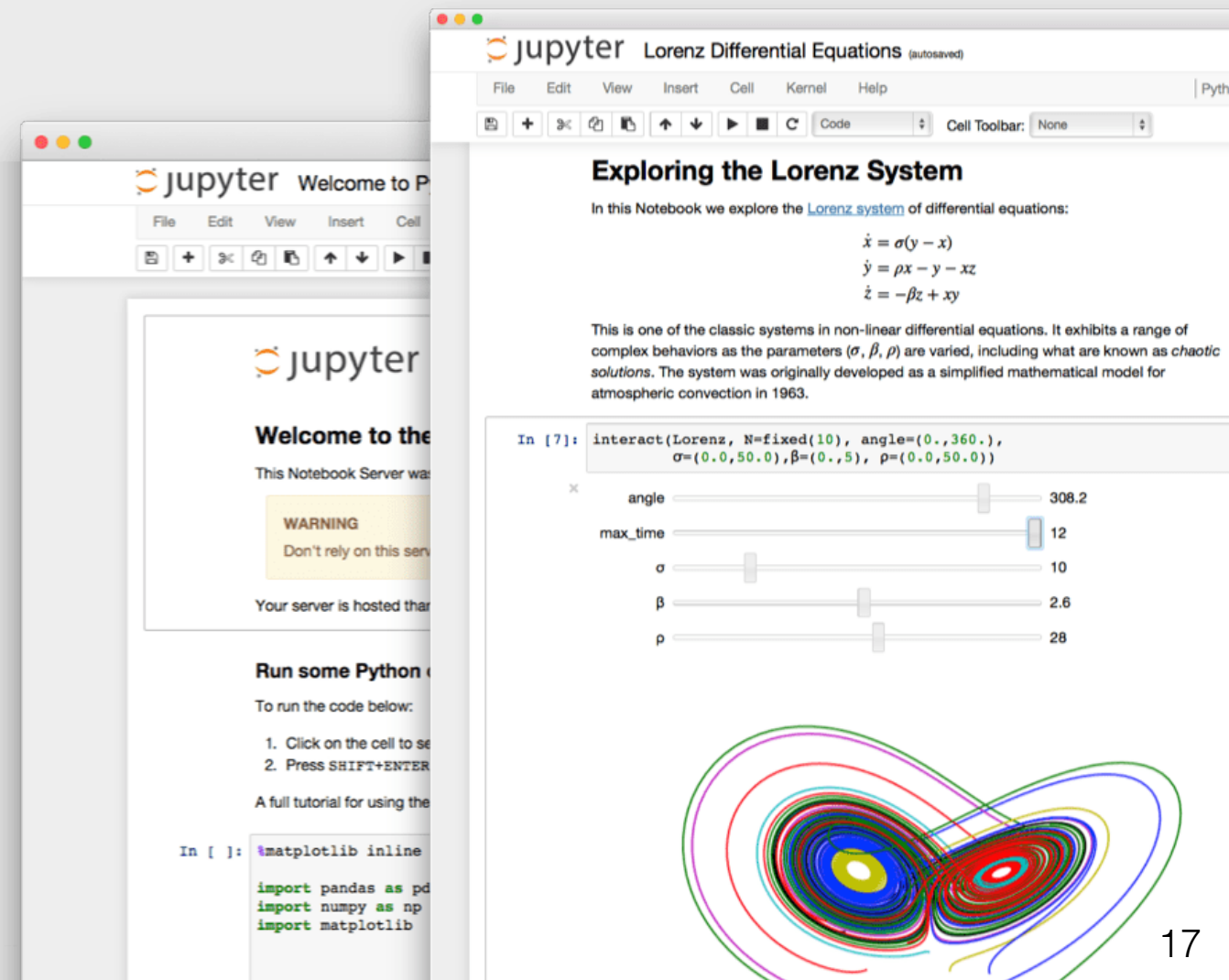
- **Exploratory work:** not "small" anymore.
- **Collaboration:** a rich, dynamic network.
- **Scholarly output:** new and diverse types.
- **Consumers of output:** from academia and education to decision making and the public.





# What is Jupyter

- Mainly Known for **The Notebook**
  - Web server, a web app, containing code, narrative, math and results.
  - Attached to a **Kernel** doing computation.
- Results can be:
  - Static
  - Interactive (client side)
  - Dynamic (trigger compute)



The image shows a Jupyter Notebook interface with a Python kernel. The notebook is titled "Lorenz Differential Equations (autosaved)". The main content area displays the Lorenz attractor, a 3D plot of the Lorenz system of differential equations. The equations are:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

The notebook also includes 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 a table of sliders for the parameters:

Parameter	Value
angle	308.2
max_time	12
$\sigma$	10
$\beta$	2.6
$\rho$	28

The Lorenz attractor is a 3D plot showing the trajectory of the Lorenz system, characterized by its butterfly-like shape. The plot is rendered in a 3D perspective view with a white background and a gray grid. The trajectory is shown in a multi-colored line, with the colors transitioning from blue to red to yellow to green. The plot is centered in the lower right quadrant of the notebook's main content area.



Narrative



Code  
Result



jupyter Lorenz Differential Equations Last Checkpoint: 21 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

### Exploring the Lorenz System of Differential Equations

In this Notebook we explore the Lorenz system of differential equations:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy \end{aligned}$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of different behaviors as the parameters ( $\sigma, \beta, \rho$ ) are varied.

```
In [11]: w = interactive(solve_lorenz, angle=(0.,360.), N=(0,50),sigma=(0.0,50.0), rho=(0.0,50.0))
display(w)
```

N 10  
 angle 0.00  
 max\_time 4.00  
 $\sigma$  10.00  
 $\beta$  2.67  
 $\rho$  28.00

Math

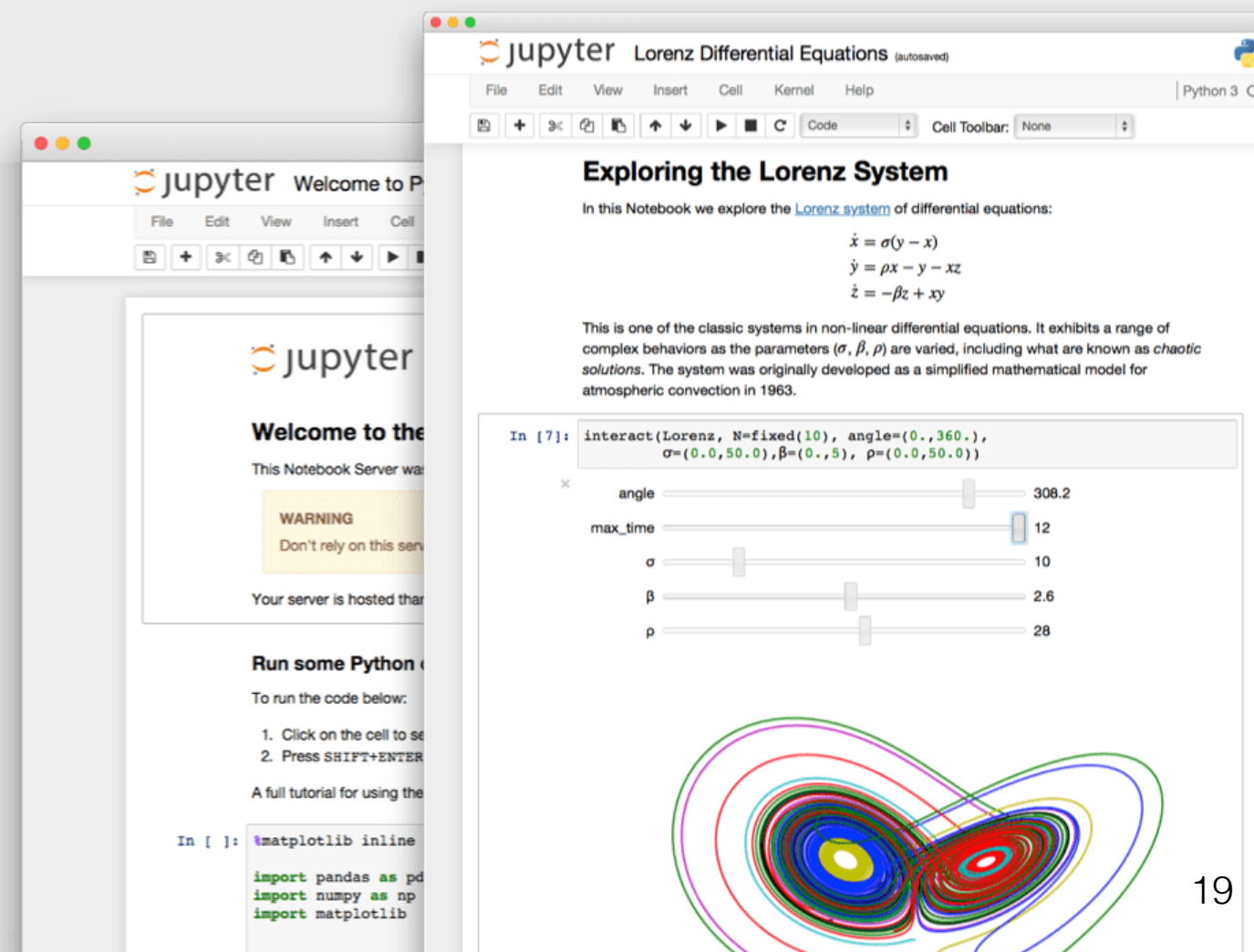


Dynamic  
Controls

aka "widgets"

# Web Based Notebook Application

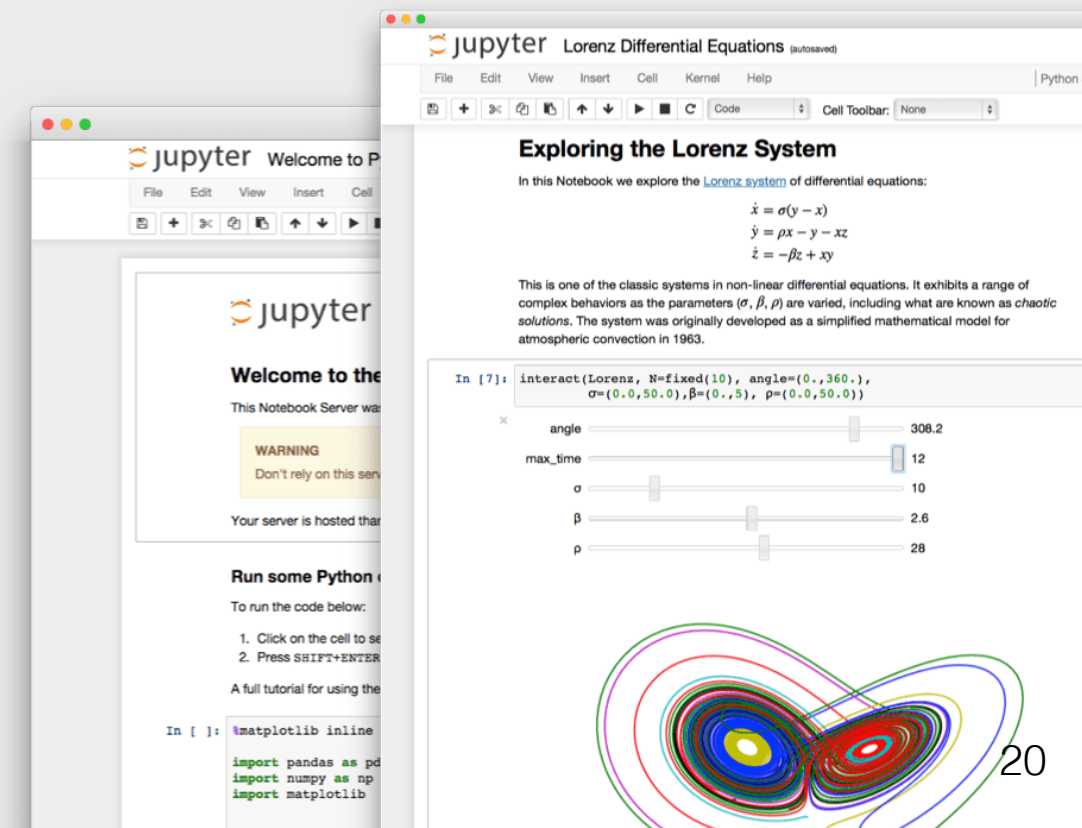
- Web technologies are accessible.
  - Only need a web browser to control an HPC Cluster
  - Familiar to users
- Rapid increase in performance and functionality
  - V8, 3D, Wasm, ...
- Identical for local and remote use.
- Allow multiple domain collaboration





# Open Notebook Document Format

- Notebooks get saved as JSON documents, which contain **narrative, code, and results**
  - Ubiquitous, JSON is readable in ~all languages.
  - Result embedding ensure trust (no Copy Past errors)
  - Make it easy to share and modify (Nbviewer, Binder)



# JupyterLab



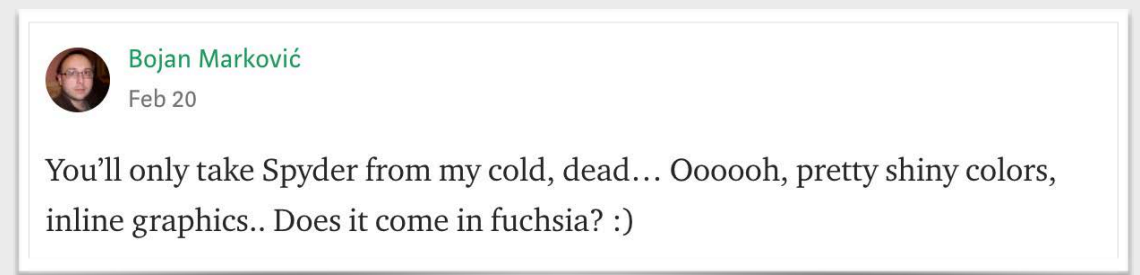
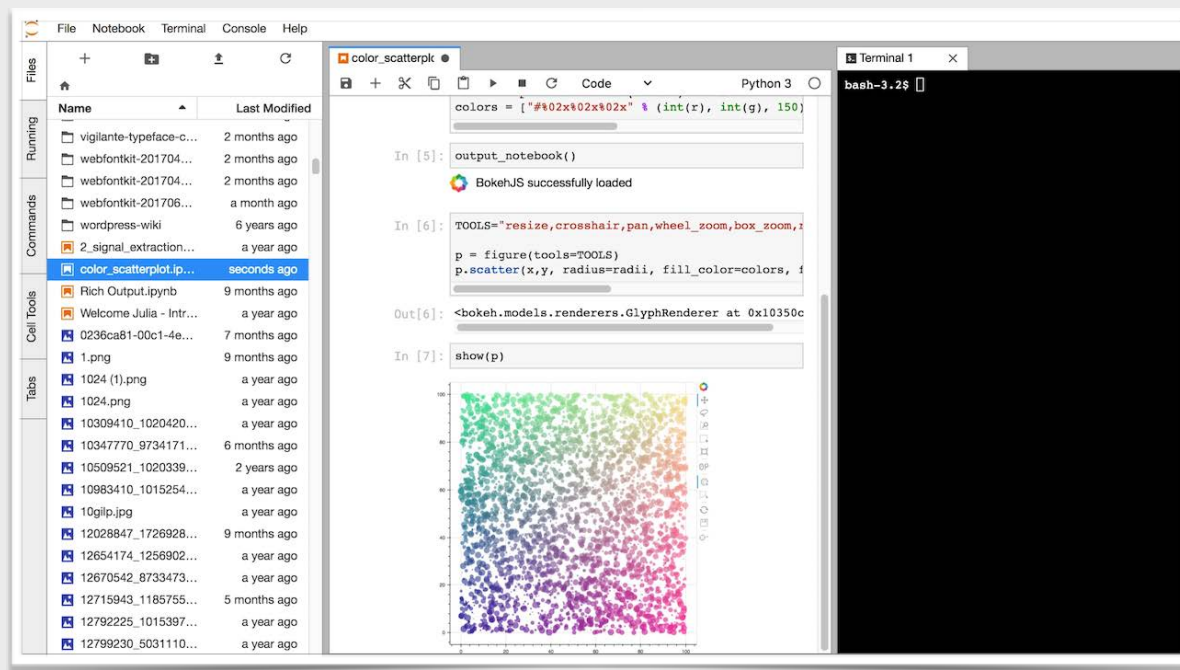
The screenshot displays the JupyterLab environment with three main notebooks open:

- color\_scatterplot.ipynb:** Shows a scatter plot with points colored by a gradient from blue to red. The code defines a color palette and uses Bokeh for rendering.
- transit.ipynb:** Analyzes passenger load data for the Rosengartenstrasse stop. It includes a histogram titled "Passenger Load at Rosengartenstrasse stop" and a comparison plot titled "Passenger load medians across all stops".
- routes.json:** A JSON viewer showing the structure of a stop feature, including keys like stopId, stopNumber, stopNameShort, and stopName.

The interface also shows a file browser on the left and a terminal window at the top right.

# JupyterLab

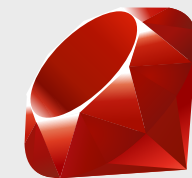
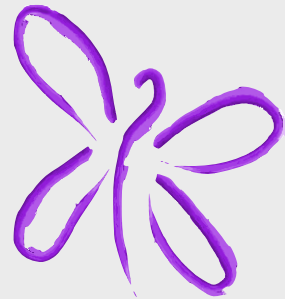
- Install Side by Side with Classic Notebook
- No Change in File Format, or protocol
- Better Architecture (all extensions are first class)



- Classic Notebook will be deprecated at some point



# Many languages



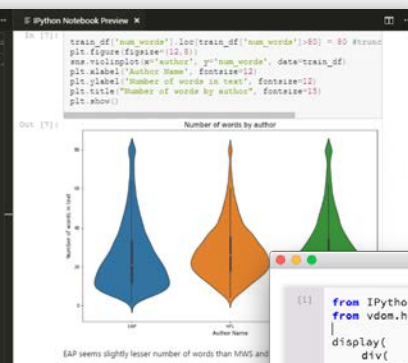
...



# Tools Integrations

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


```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import nltk
6 from nltk.corpus import stopwords
7 import string
8 from sklearn.feature_extraction.text import TfidfVectorizer, CountVec
9 from sklearn.decomposition import TruncatedSVD
10 from sklearn.ensemble import RandomForestClassifier
11 from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
12 color = sns.color_palette()
13
14 # Read the train and test datasets and check the top few lines
15 train_df = pd.read_csv("../input/train.csv")
16 test_df = pd.read_csv("../input/test.csv")
17 print("Number of rows in train dataset: ", train_df.shape[0])
18 print("Number of rows in test dataset: ", test_df.shape[0])
19
20 # Vectorize the text
21 vectorizer = TfidfVectorizer(stop_words=stopwords.words('english'))
22 train_vectors = vectorizer.fit_transform(train_df['text']).toarray()
23 test_vectors = vectorizer.transform(test_df['text']).toarray()
24
25 # Train the model
26 model = RandomForestClassifier(n_estimators=100, random_state=42)
27 model.fit(train_vectors, train_df['target'])
28
29 # Predict on the test set
30 predictions = model.predict(test_vectors)
31
32 # Evaluate the model
33 accuracy = accuracy_score(test_df['target'], predictions)
34 print("Accuracy: ", accuracy)
```



```
1 from IPython.display import display
2 from vdom.helpers import h1, p, img, div, b
3
4 display(
5     div(
6         h1('Our Incredibly Declarative Example'),
7         p('Can you believe we wrote this in Python?'),
8         img(src="https://media.giphy.com/media/xUPGcGwZHRc2HyBRS/giphy.gif"),
9         p('What will you create next?')
10     )
11 )
```

**Our Incredibly Declarative Example**

Can you believe we wrote this in Python?



What will you create next?



```
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)
18 x cos(yz) + { 0 for y = 0, -1/y cos(xy) otherwise
19
20 # In[1]: Display arrays
21 t = np.linspace(0, 20, 500)
22
23 array([[ 0.,  0.4088816,  0.80810932,  0.12024068,
24         0.16832884,  0.20940988,  0.24948996,  0.28866112,
25         0.32684128,  0.36392144,  0.4000016,  0.43608176,
26         0.47216192,  0.50824208,  0.54432224,  0.5804024,
27         0.61648256,  0.65256272,  0.68864288,  0.72472304,
28         0.7608032,  0.79688336,  0.83296352,  0.86904368,
29         0.90512384,  0.941204,  0.97728416,  1.01336432,
30         1.04944448,  1.08552464,  1.1216048,  1.15768496,
31         1.19376512,  1.22984528,  1.26592544,  1.3020056,
32         1.33808576,  1.37416592,  1.41024608,  1.44632624,
33         1.4824064,  1.51848656,  1.55456672,  1.59064688,
34         1.62672704,  1.6628072,  1.69888736,  1.73496752,
35         1.77104768,  1.80712784,  1.843208,  1.87928816,
36         1.91536832,  1.95144848,  1.98752864,  2.0236088,
37         2.05968896,  2.09576912,  2.13184928,  2.16792944,
38         2.2040096,  2.24008976,  2.27616992,  2.31225008,
39         2.34833024,  2.3844104,  2.42049056,  2.45657072,
40         2.49265088,  2.52873104,  2.5648112,  2.60089136,
41         2.63697152,  2.67305168,  2.70913184,  2.745212,
42         2.78129216,  2.81737232,  2.85345248,  2.88953264,
43         2.9256128,  2.96169296,  2.99777312,  3.03385328,
44         3.06993344,  3.1060136,  3.14209376,  3.17817392,
45         3.21425408,  3.25033424,  3.2864144,  3.32249456,
46         3.35857472,  3.39465488,  3.43073504,  3.4668152,
47         3.50289536,  3.53897552,  3.57505568,  3.61113584,
48         3.647216,  3.68329616,  3.71937632,  3.75545648,
49         3.79153664,  3.8276168,  3.86369696,  3.90000008,
50         3.93608024,  3.9721604,  4.00824056,  4.04432072,
51         4.08040088,  4.11648104,  4.1525612,  4.18864136,
52         4.22472152,  4.26080168,  4.29688184,  4.332962,
53         4.36904216,  4.40512232,  4.44120248,  4.47728264,
54         4.5133628,  4.54944296,  4.58552312,  4.62160328,
55         4.65768344,  4.6937636,  4.72984376,  4.76592392,
56         4.80200408,  4.83808424,  4.8741644,  4.91024456,
57         4.94632472,  4.98240488,  5.01848504,  5.0545652,
58         5.09064536,  5.12672552,  5.16280568,  5.19888584,
59         5.234966,  5.27104616,  5.30712632,  5.34320648,
60         5.37928664,  5.4153668,  5.45144696,  5.48752712,
61         5.52360728,  5.55968744,  5.5957676,  5.63184776,
62         5.66792792,  5.70400808,  5.74008824,  5.7761684,
63         5.81224856,  5.84832872,  5.88440888,  5.92048904,
64         5.9565692,  5.99264936,  6.02872952,  6.06480968,
65         6.10088984,  6.13697,  6.17305016,  6.20913032,
66         6.24521048,  6.28129064,  6.3173708,  6.35345096,
67         6.38953112,  6.42561128,  6.46169144,  6.4977716,
68         6.53385176,  6.56993192,  6.60601208,  6.64209224,
69         6.6781724,  6.71425256,  6.75033272,  6.78641288,
70         6.82249304,  6.8585732,  6.89465336,  6.93073352,
71         6.96681368,  7.00289384,  7.038974,  7.07505416,
72         7.11113432,  7.14721448,  7.18329464,  7.2193748,
73         7.25545496,  7.29153512,  7.32761528,  7.36369544,
74         7.40000008,  7.43608024,  7.4721604,  7.50824056,
75         7.54432072,  7.58040088,  7.61648104,  7.6525612,
76         7.68864136,  7.72472152,  7.76080168,  7.79688184,
77         7.832962,  7.86904216,  7.90512232,  7.94120248,
78         7.97728264,  8.0133628,  8.04944296,  8.08552312,
79         8.12160328,  8.15768344,  8.1937636,  8.22984376,
80         8.26592392,  8.30200408,  8.33808424,  8.3741644,
81         8.41024456,  8.44632472,  8.48240488,  8.51848504,
82         8.5545652,  8.59064536,  8.62672552,  8.66280568,
83         8.69888584,  8.734966,  8.77104616,  8.80712632,
84         8.84320648,  8.87928664,  8.9153668,  8.95144696,
85         8.98752712,  9.02360728,  9.05968744,  9.0957676,
86         9.13184776,  9.16792792,  9.20400808,  9.24008824,
87         9.2761684,  9.31224856,  9.34832872,  9.38440888,
88         9.42048904,  9.4565692,  9.49264936,  9.52872952,
89         9.56480968,  9.60088984,  9.63697,  9.67305016,
90         9.70913032,  9.74521048,  9.78129064,  9.8173708,
91         9.85345096,  9.88953112,  9.92561128,  9.96169144,
92         9.9977716,  10.03385176,  10.06993192,  10.10601208,
93         10.14209224,  10.1781724,  10.21425256,  10.25033272,
94         10.28641288,  10.32249304,  10.3585732,  10.39465336,
95         10.43073352,  10.46681368,  10.50289384,  10.538974,
96         10.57505416,  10.61113432,  10.64721448,  10.68329464,
97         10.7193748,  10.75545496,  10.79153512,  10.82761528,
98         10.86369544,  10.8997756,  10.93585576,  10.97193592,
99         11.00801608,  11.04409624,  11.0801764,  11.11625656,
100        11.15233672,  11.18841688,  11.22449704,  11.2605772,
101        11.29665736,  11.33273752,  11.36881768,  11.40489784,
102        11.440978,  11.47705816,  11.51313832,  11.54921848,
103        11.58529864,  11.6213788,  11.65745896,  11.69353912,
104        11.72961928,  11.76569944,  11.8017796,  11.83785976,
105        11.87393992,  11.91002008,  11.94610024,  11.9821804,
106        12.01826056,  12.05434072,  12.09042088,  12.12650104,
107        12.1625812,  12.19866136,  12.23474152,  12.27082168,
108        12.30690184,  12.342982,  12.37906216,  12.41514232,
109        12.45122248,  12.48730264,  12.5233828,  12.55946296,
110        12.59554312,  12.63162328,  12.66770344,  12.7037836,
111        12.73986376,  12.77594392,  12.81202408,  12.84810424,
112        12.8841844,  12.92026456,  12.95634472,  12.99242488,
113        13.02850504,  13.0645852,  13.10066536,  13.13674552,
114        13.17282568,  13.20890584,  13.244986,  13.28106616,
115        13.31714632,  13.35322648,  13.38930664,  13.4253868,
116        13.46146696,  13.49754712,  13.53362728,  13.56970744,
117        13.6057876,  13.64186776,  13.67794792,  13.71402808,
118        13.75010824,  13.7861884,  13.82226856,  13.85834872,
119        13.89442888,  13.93050904,  13.9665892,  14.00266936,
120        14.03874952,  14.07482968,  14.11090984,  14.14699,
121        14.1830702,  14.21915036,  14.25523052,  14.29131068,
122        14.32739084,  14.363471,  14.39955116,  14.43563132,
123        14.47171148,  14.50779164,  14.5438718,  14.57995196,
124        14.61603212,  14.65211228,  14.68819244,  14.7242726,
125        14.76035276,  14.79643292,  14.83251308,  14.86859324,
126        14.9046734,  14.94075356,  14.97683372,  15.01291388,
127        15.04899404,  15.0850742,  15.12115436,  15.15723452,
128        15.19331468,  15.22939484,  15.265475,  15.30155516,
129        15.33763532,  15.37371548,  15.40979564,  15.4458758,
130        15.48195596,  15.51803612,  15.55411628,  15.59019644,
131        15.6262766,  15.66235676,  15.69843692,  15.73451708,
132        15.77059724,  15.8066774,  15.84275756,  15.87883772,
133        15.91491788,  15.95099804,  15.9870782,  16.02315836,
134        16.05923852,  16.09531868,  16.13139884,  16.167479,
135        16.20355916,  16.23963932,  16.27571948,  16.31179964,
136        16.3478798,  16.38395996,  16.42004012,  16.45612028,
137        16.49220044,  16.5282806,  16.56436076,  16.60044092,
138        16.63652108,  16.67260124,  16.7086814,  16.74476156,
139        16.78084172,  16.81692188,  16.85300204,  16.8890822,
140        16.92516236,  16.96124252,  16.99732268,  17.03340284,
141        17.069483,  17.10556316,  17.14164332,  17.17772348,
142        17.21380364,  17.2498838,  17.28596396,  17.32204412,
143        17.35812428,  17.39420444,  17.4302846,  17.46636476,
144        17.50244492,  17.53852508,  17.57460524,  17.6106854,
145        17.64676556,  17.68284572,  17.71892588,  17.75500604,
146        17.7910862,  17.82716636,  17.86324652,  17.89932668,
147        17.93540684,  17.971487,  18.00756716,  18.04364732,
148        18.07972748,  18.11580764,  18.1518878,  18.18796796,
149        18.22404812,  18.26012828,  18.29620844,  18.3322886,
150        18.36836876,  18.40444892,  18.44052908,  18.47660924,
151        18.5126894,  18.54876956,  18.58484972,  18.62092988,
152        18.65701004,  18.6930902,  18.72917036,  18.76525052,
153        18.80133068,  18.83741084,  18.873491,  18.90957116,
154        18.94565132,  18.98173148,  19.01781164,  19.0538918,
155        19.09000008,  19.12608024,  19.1621604,  19.19824056,
156        19.23432072,  19.27040088,  19.30648104,  19.3425612,
157        19.37864136,  19.41472152,  19.45080168,  19.48688184,
158        19.522962,  19.55904216,  19.59512232,  19.63120248,
159        19.66728264,  19.7033628,  19.73944296,  19.77552312,
160        19.81160328,  19.84768344,  19.8837636,  19.91984376,
161        19.95592392,  19.99200408,  20.02808424,  20.0641644,
162        20.10024456,  20.13632472,  20.17240488,  20.20848504,
163        20.2445652,  20.28064536,  20.31672552,  20.35280568,
164        20.38888584,  20.424966,  20.46104616,  20.49712632,
165        20.53320648,  20.56928664,  20.6053668,  20.64144696,
166        20.67752712,  20.71360728,  20.74968744,  20.7857676,
167        20.82184776,  20.85792792,  20.89400808,  20.93008824,
168        20.9661684,  21.00224856,  21.03832872,  21.07440888,
169        21.11048904,  21.1465692,  21.18264936,  21.21872952,
170        21.25480968,  21.29088984,  21.32697,  21.36305016,
171        21.39913032,  21.43521048,  21.47129064,  21.5073708,
172        21.54345096,  21.57953112,  21.61561128,  21.65169144,
173        21.6877716,  21.72385176,  21.75993192,  21.79601208,
174        21.83209224,  21.8681724,  21.90425256,  21.94033272,
175        21.97641288,  22.01249304,  22.0485732,  22.08465336,
176        22.12073352,  22.15681368,  22.19289384,  22.228974,
177        22.26505416,  22.30113432,  22.33721448,  22.37329464,
178        22.4093748,  22.44545496,  22.48153512,  22.51761528,
179        22.55369544,  22.5897756,  22.62585576,  22.66193592,
180        22.69801608,  22.73409624,  22.7701764,  22.80625656,
181        22.84233672,  22.87841688,  22.91449704,  22.9505772,
182        22.98665736,  23.02273752,  23.05881768,  23.09489784,
183        23.130978,  23.16705816,  23.20313832,  23.23921848,
184        23.27529864,  23.3113788,  23.34745896,  23.38353912,
185        23.41961928,  23.45569944,  23.4917796,  23.52785976,
186        23.56393992,  23.60002008,  23.63610024,  23.6721804,
187        23.70826056,  23.74434072,  23.78042088,  23.81650104,
188        23.8525812,  23.88866136,  23.92474152,  23.96082168,
189        23.99690184,  24.032982,  24.06906216,  24.10514232,
190        24.14122248,  24.17730264,  24.2133828,  24.24946296,
191        24.28554312,  24.32162328,  24.35770344,  24.3937836,
192        24.42986376,  24.46594392,  24.50202408,  24.53810424,
193        24.5741844,  24.61026456,  24.64634472,  24.68242488,
194        24.71850504,  24.7545852,  24.79066536,  24.82674552,
195        24.86282568,  24.89890584,  24.934986,  24.97106616,
196        25.00714632,  25.04322648,  25.07930664,  25.1153868,
197        25.15146696,  25.18754712,  25.22362728,  25.25970744,
198        25.2957876,  25.33186776,  25.36794792,  25.40402808,
199        25.44010824,  25.4761884,  25.51226856,  25.54834872,
200        25.58442888,  25.62050904,  25.6565892,  25.69266936,
201        25.72874952,  25.76482968,  25.80090984,  25.83699,
202        25.8730702,  25.90915036,  25.94523052,  25.98131068,
203        26.01739084,  26.053471,  26.08955116,  26.12563132,
204        26.16171148,  26.19779164,  26.2338718,  26.26995196,
205        26.30603212,  26.34211228,  26.37819244,  26.4142726,
206        26.45035276,  26.48643292,  26.52251308,  26.55859324,
207        26.5946734,  26.63075356,  26.66683372,  26.70291388,
208        26.73899404,  26.7750742,  26.81115436,  26.84723452,
209        26.88331468,  26.91939484,  26.955475,  26.99155516,
210        27.02763532,  27.06371548,  27.10000008,  27.13608024,
211        27.1721604,  27.20824056,  27.24432072,  27.28040088,
212        27.31648104,  27.3525612,  27.38864136,  27.42472152,
213        27.46080168,  27.49688184,  27.532962,  27.56904216,
214        27.60512232,  27.64120248,  27.67728264,  27.7133628,
215        27.74944296,  27.78552312,  27.82160328,  27.85768344,
216        27.8937636,  27.92984376,  27.96592392,  28.00200408,
217        28.03808424,  28.0741644,  28.11024456,  28.14632472,
218        28.18240488,  28.21848504,  28.2545652,  28.29064536,
219        28.32672552,  28.36280568,  28.39888584,  28.434966,
220        28.47104616,  28.50712632,  28.54320648,  28.57928664,
221        28.6153
```

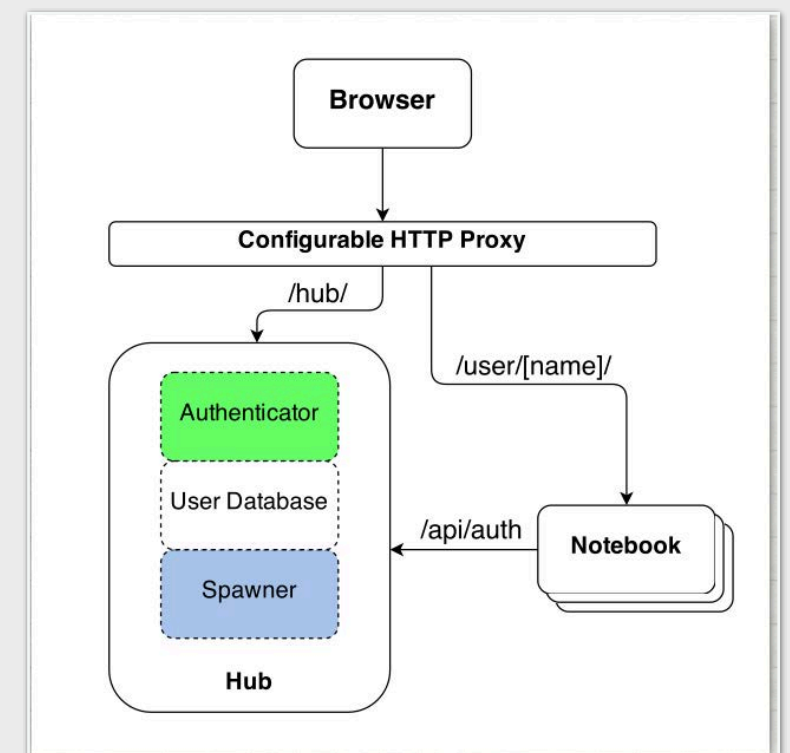


# Easy, Scalable Deployment



# JupyterHub

- A notebook application is a **Single User** application
- Quick and easy multi-user deployments are critical to lower overhead.
- JupyterHub Provides way a simple way to deploy Jupyter at scale.
  - <https://z2jh.jupyter.org/> for a guide.





**Amazon SageMaker**



**GRYD**



See <https://discourse.jupyter.org/t/in-depth-comparison-of-cloud-based-services-that-run-jupyter-notebook/460/14>

# Binder

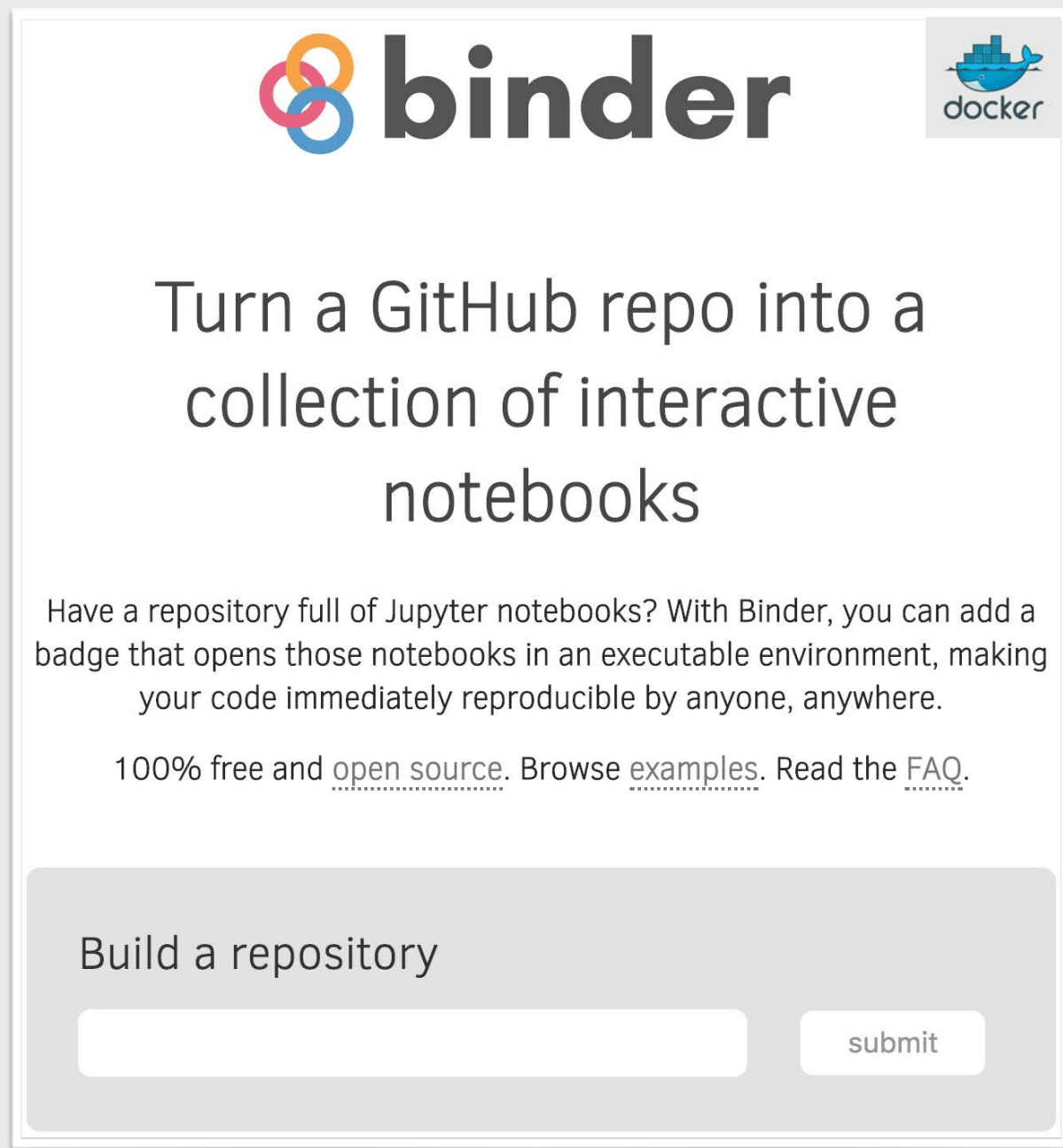


- Technology which takes any GitHub repository with Jupyter notebooks
- Turn it into a Docker image to ensure reproducibility and quick deployment.
- Starts an isolated, ephemeral server in a few seconds, for user to interact with.

\* Not limited to GitHub, Notebooks, Jupyter, Docker, or Ephemeral



# MyBinder.org





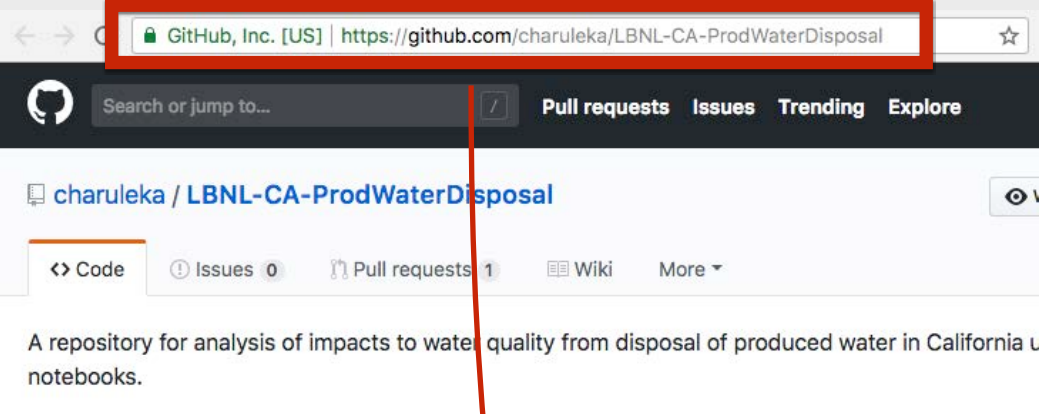
The screenshot shows the MyBinder.org homepage. At the top left is the Binder logo (three interlocking circles in orange, red, and blue) followed by the word "binder" in a bold, dark grey font. To the right of the logo is a small Docker logo (a blue whale) with the word "docker" underneath. Below the logo and text, the main heading reads "Turn a GitHub repo into a collection of interactive notebooks". Underneath this is a paragraph: "Have a repository full of Jupyter notebooks? With Binder, you can add a badge that opens those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere." Below the paragraph is another line of text: "100% free and open source. Browse examples. Read the FAQ." At the bottom of the page is a grey rectangular box containing the text "Build a repository" on the left, a white input field in the center, and a white button labeled "submit" on the right.

- One Public instance of Binder
  - [mybinder.org](https://mybinder.org)
  - (stats at [grafana.mybinder.org](https://grafana.mybinder.org))
  - Limited CPU/Memory/network
- Anonymous Login
- Ephemeral (2h) and restricted to 50 parallel launch
- Build on demand
- <sup>29</sup> • Caches images for fast launch





# MyBinder.org

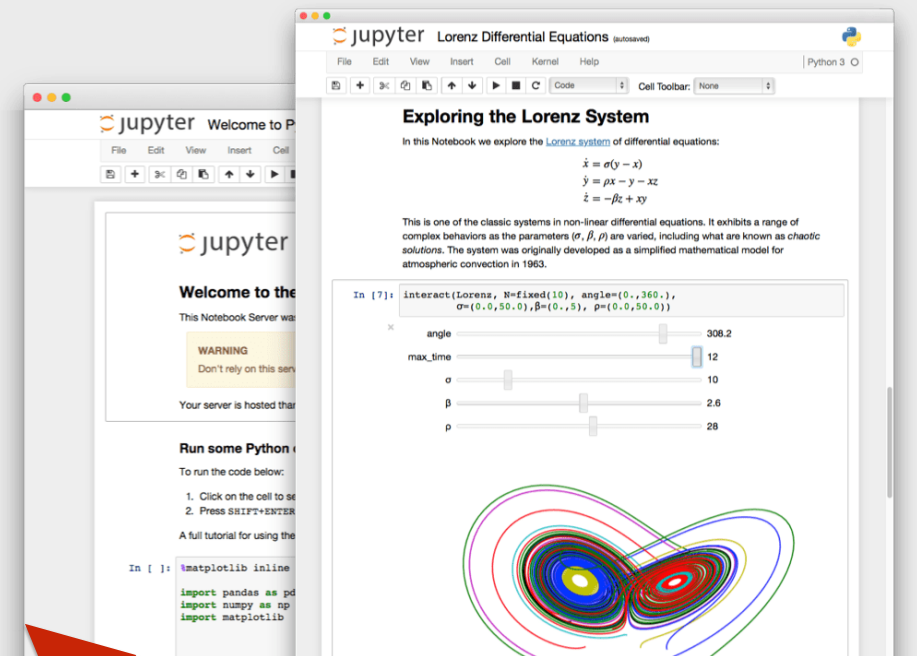


Turn a GitHub repo into a collection of interactive notebooks

Have a repository full of Jupyter notebooks? With Binder, you can add a badge that opens those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.

100% free and [open source](#). Browse [examples](#). Read the [FAQ](#).

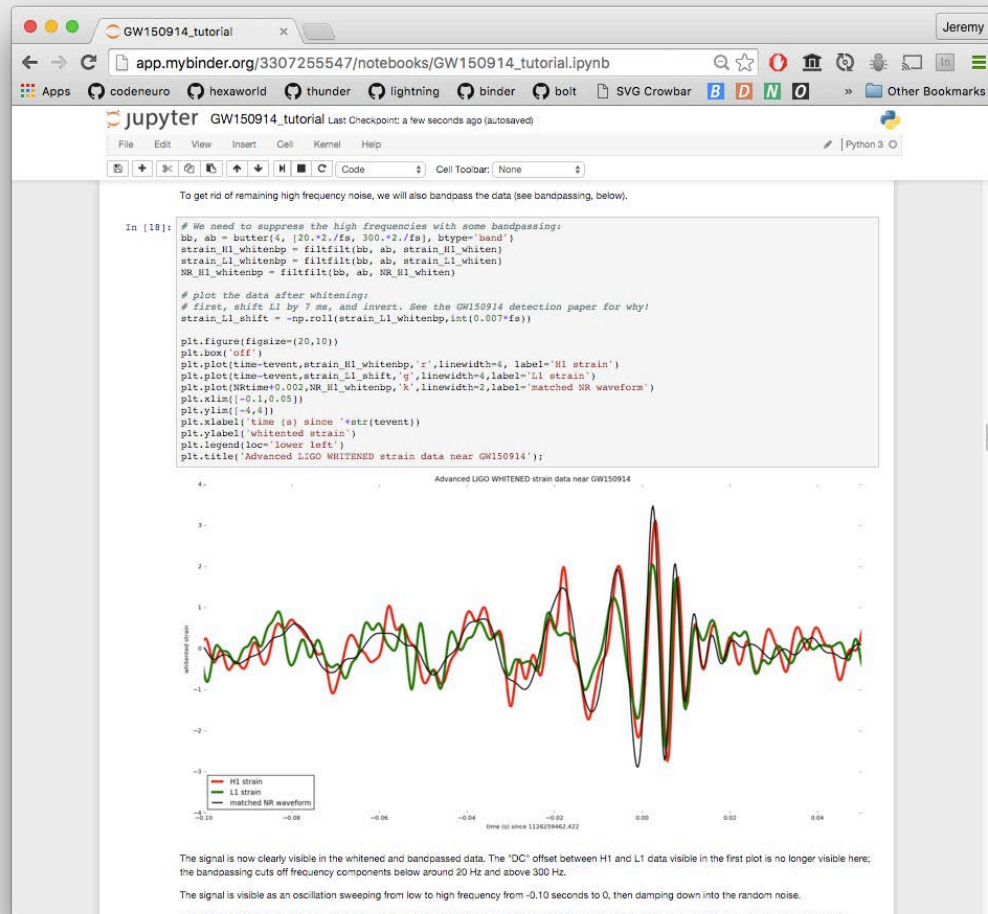
Build a repository



launch binder



# Ligo binder



# NbInteract

## nbinteract.hist

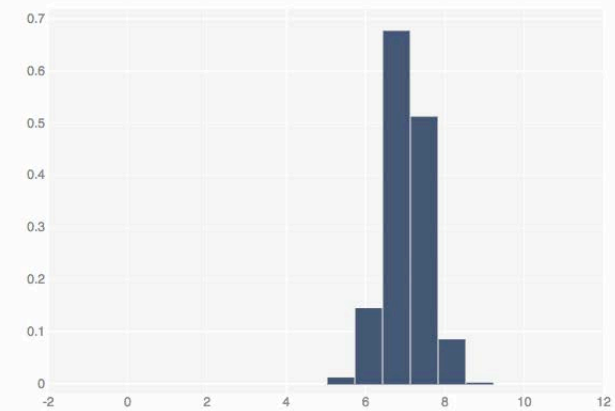
`hist` generates a histogram that allows interaction with the parameters for the response function.

`hist` takes in a single response function. The response function returns the array of numerical values that will be shown in the histogram. The `hist` function allows interaction with the response function's parameters by specifying them as keyword arguments in the same format as `ipywidgets.interact`. Any argument that can be used for `ipywidgets.interact` can be used for `hist`.

```
def hist_response_function(mean, sd, size=1000):
    """
    Returns 1000 values picked at random from the normal
    distribution with the mean and SD given.
    """
    return np.random.normal(loc=mean, scale=sd, size=1000)
```

```
nbinteract.hist(hist_response_function, mean=(0, 10), sd=(0, 2.0, 0.2))
```

mean  7  
sd  0.50



# In the Classroom



DataHub

[datahub.berkeley.edu](http://datahub.berkeley.edu)



<http://www.ds100.org/>



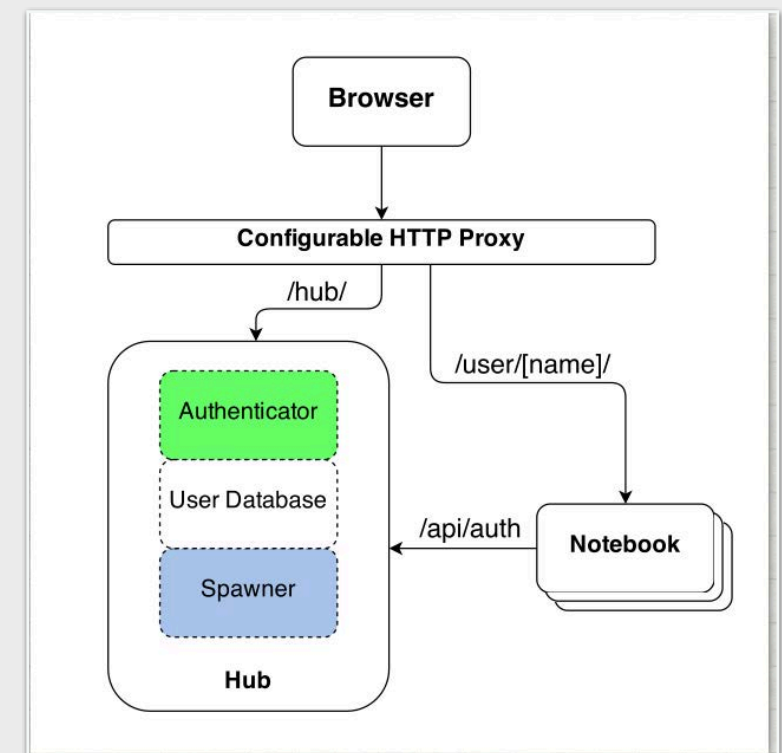


# Zero setup\*

- Campus Wide deployment
- Login with Cal ID
- Can focus on Domain

Knowledge

Students can still optionally install Jupyter on their machine later on.



\* At least for students



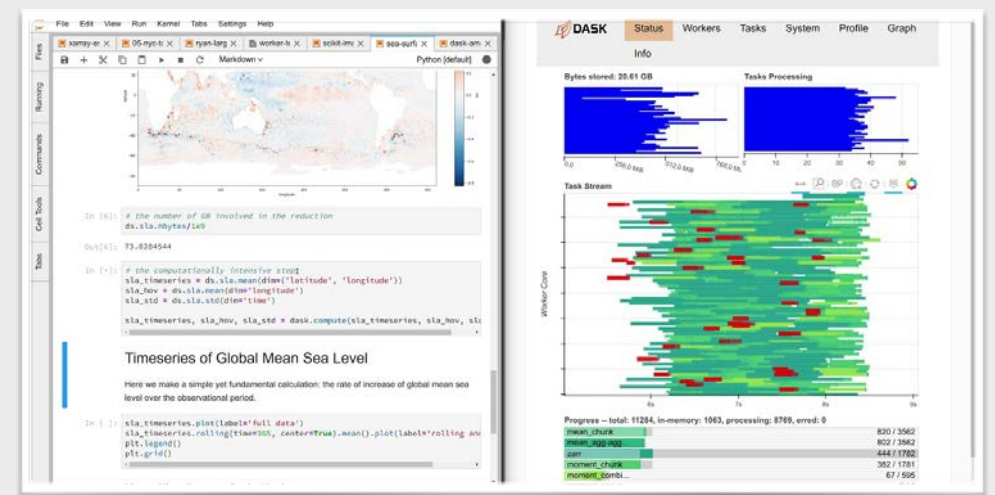
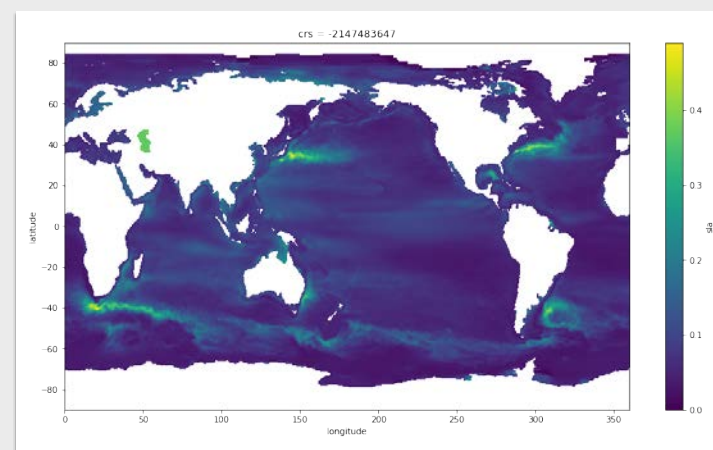
# In the Cloud

1. Foster collaboration around the open source scientific python ecosystem for ocean / atmosphere / land / climate science.
2. Support the development with domain-specific geoscience packages.
3. Improve scalability of these tools to to handle petabyte-scale datasets on HPC and cloud platforms.



**PANGEO**

<http://pangeo-data.org/>





# In the Cloud

- Completely managed JupyterHub on Kubernetes

- <http://pangeo.pydata.org/>

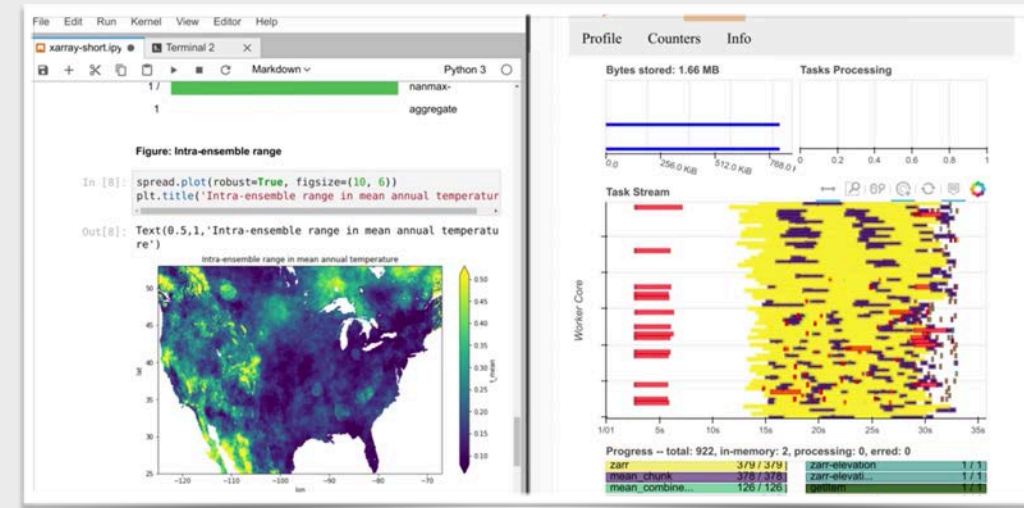
- Login via GitHub

- Customized for GeoScience

- Persisting servers on Google Cloud,

- Large amount of Ram/CPU/Nodes

- Dynamically scalable



**PANGEO**

<http://pangeo-data.org/>