



Interactive HPC

with Jupyter, IPython, and Dask

SOS 21

Min Ragan-Kelley  
Simula Research Lab

# Jupyter Notebooks

- Hugely popular in Data Science
- Part of reproducible research workflows
- Included in open science publications
- Used widely in education, including data science and computational sciences
- Interface to many existing Computational Environments (Google, Microsoft, IBM, Domino, more)



# Jupyter Notebooks



**Jupyter** Sampling\_Theorem (autosaved) IPython (Python 3) Logout

File Edit View Insert Cell Kernel Help

## Investigating the Sampling Theorem

In this section, we investigate the implications of the sampling theorem. Here is the usual statement of the theorem from wikipedia:

*"if a function  $x(t)$  contains no frequencies higher than  $B$  hertz, it is completely determined by giving its ordinates at a series of points spaced  $1/(2B)$  seconds apart."*

Since a function  $x(t)$  is a function from the real line to the real line, there are uncountably many points between any two ordinates, so sampling is a massive reduction of data since it only takes a tiny number of points to completely characterize the function. This is a powerful idea worth exploring. In fact, we have seen this idea of reducing a function to a discrete set of numbers before in Fourier series expansions where (for periodic  $x(t)$ )

$$a_n = \frac{1}{T} \int_0^T x(t) \exp(-j\omega_n t) dt$$

with corresponding reconstruction as:

$$x(t) = \sum_k a_n \exp(j\omega_n t)$$

But here we are generating discrete points  $a_n$  by integrating over the **entire** function  $x(t)$ , not just evaluating it at a single point. This means we are collecting information about the entire function to compute a single discrete point  $a_n$ , whereas with sampling we are just taking individual points in isolation.

# Jupyter Notebooks

JSON document format

Notebook = sequence of cells

Markdown cells with LaTeX math

Code cells with input and output

Convertible to HTML, TeX, PDF, etc.

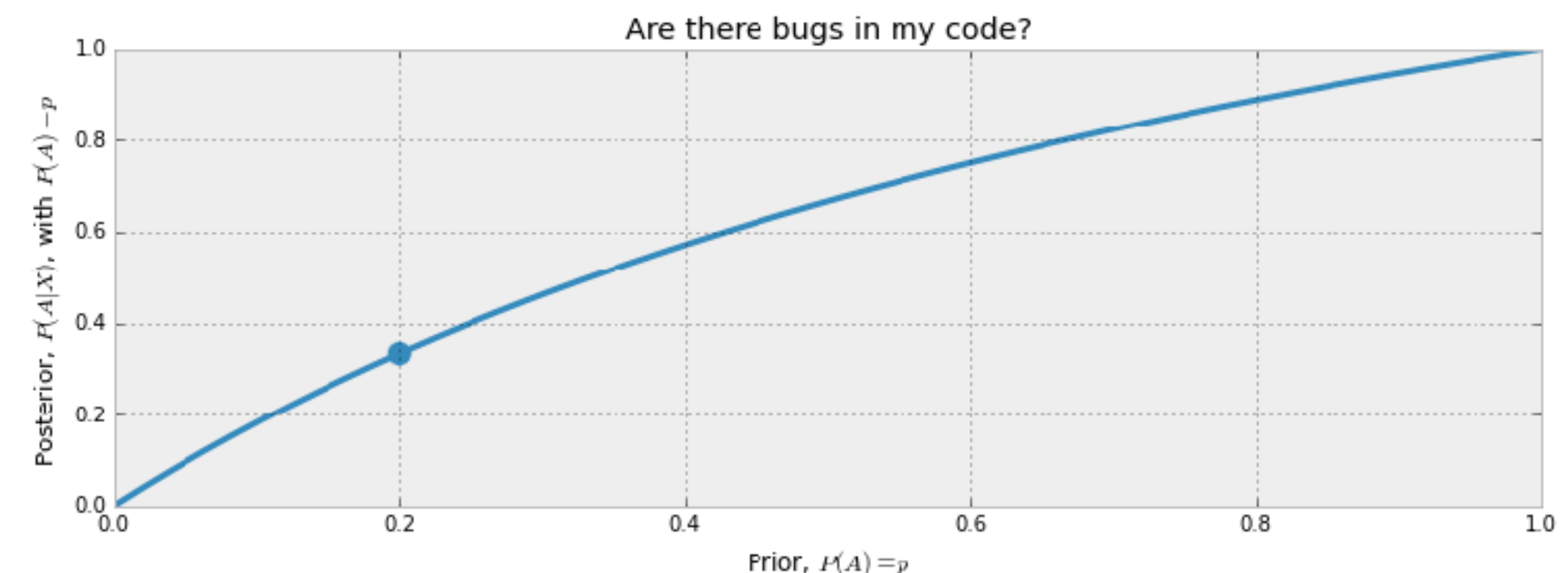
We have already computed  $P(X|A)$  above. On the other hand,  $P(X| \sim A)$  is subjective: our code can pass tests but still have a bug in it, though the probability there is a bug present is reduced. Note this is dependent on the number of tests performed, the degree of complication in the tests, etc. Let's be conservative and assign  $P(X| \sim A) = 0.5$ . Then

$$P(A|X) = \frac{1 \cdot p}{1 \cdot p + 0.5(1 - p)}$$
$$= \frac{2p}{1 + p}$$

This is the posterior probability. What does it look like as a function of our prior,  $p \in [0, 1]$ ?

```
figsize(12.5, 4)
p = np.linspace(0, 1, 50)
plt.plot(p, 2 * p / (1 + p), color="#348ABD", lw=3)
# plt.fill_between(p, 2*p/(1+p), alpha=.5, facecolor=["#A60628"])
plt.scatter(0.2, 2 * (0.2) / 1.2, s=140, c="#348ABD")
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.xlabel("Prior, $P(A) = p$")
plt.ylabel("Posterior, $P(A|X)$, with $P(A) = p$")
plt.title("Are there bugs in my code?")
```

<matplotlib.text.Text at 0x1051de650>





# Jupyter Protocol: REP\*L

any mime-type output

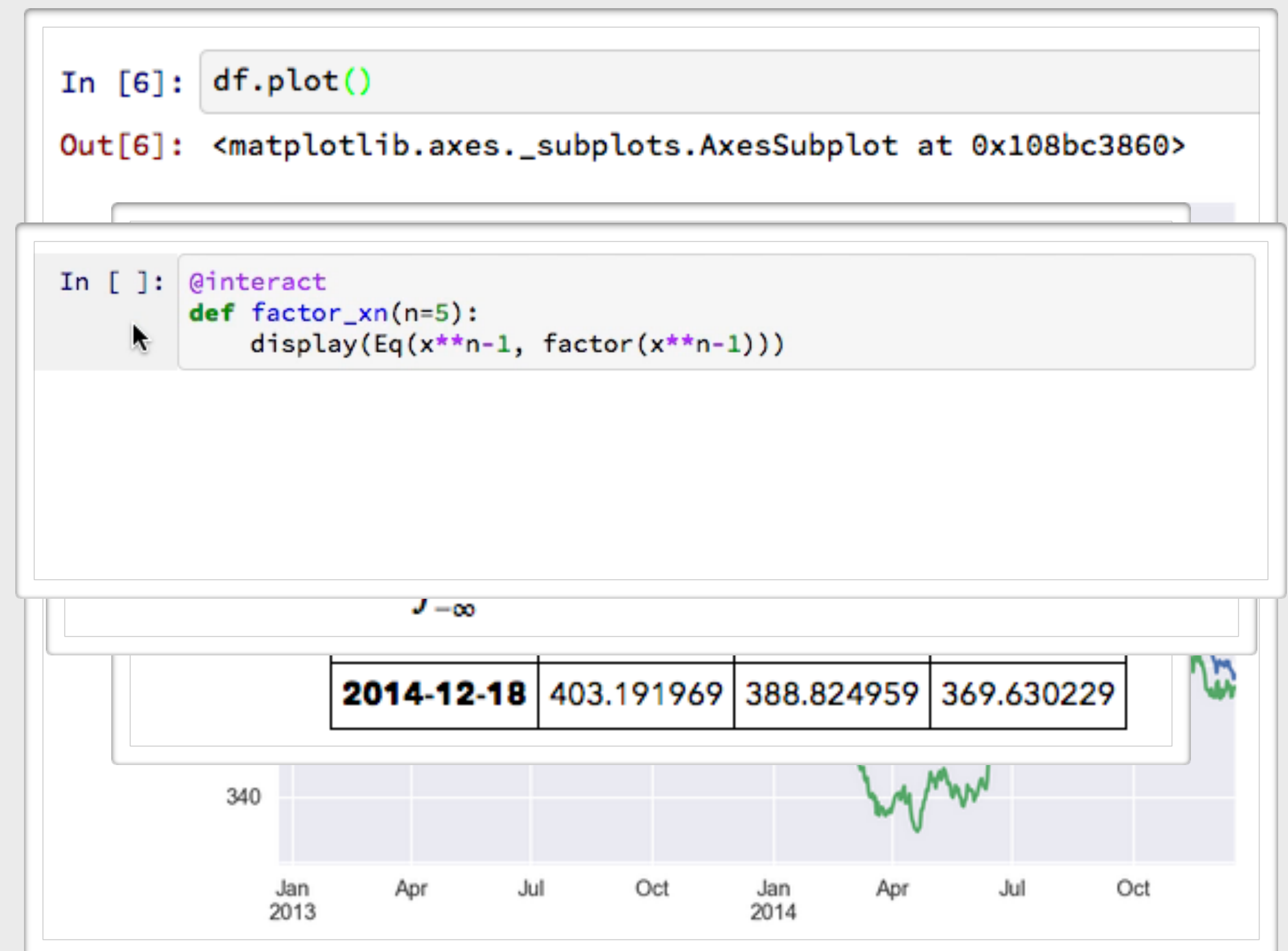
text

svg, png, jpeg

latex, pdf

html, javascript

interactive widgets



# Jupyter Protocol: Language Agnostic

 **Scala**



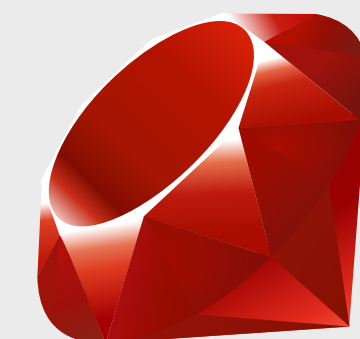
**julia**



**Spark**

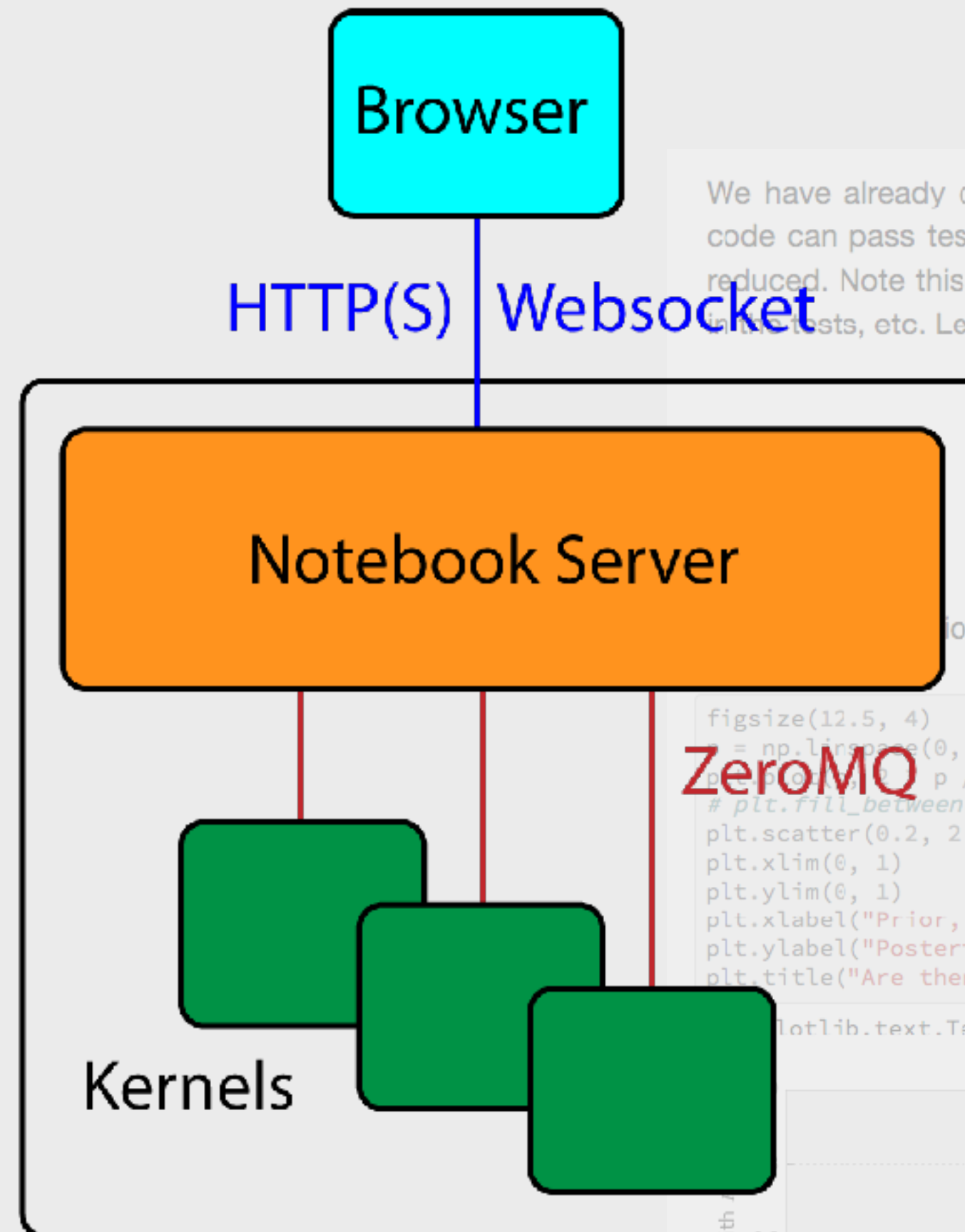


 **python**<sup>TM</sup>



# Web Application

## Jupyter Notebooks



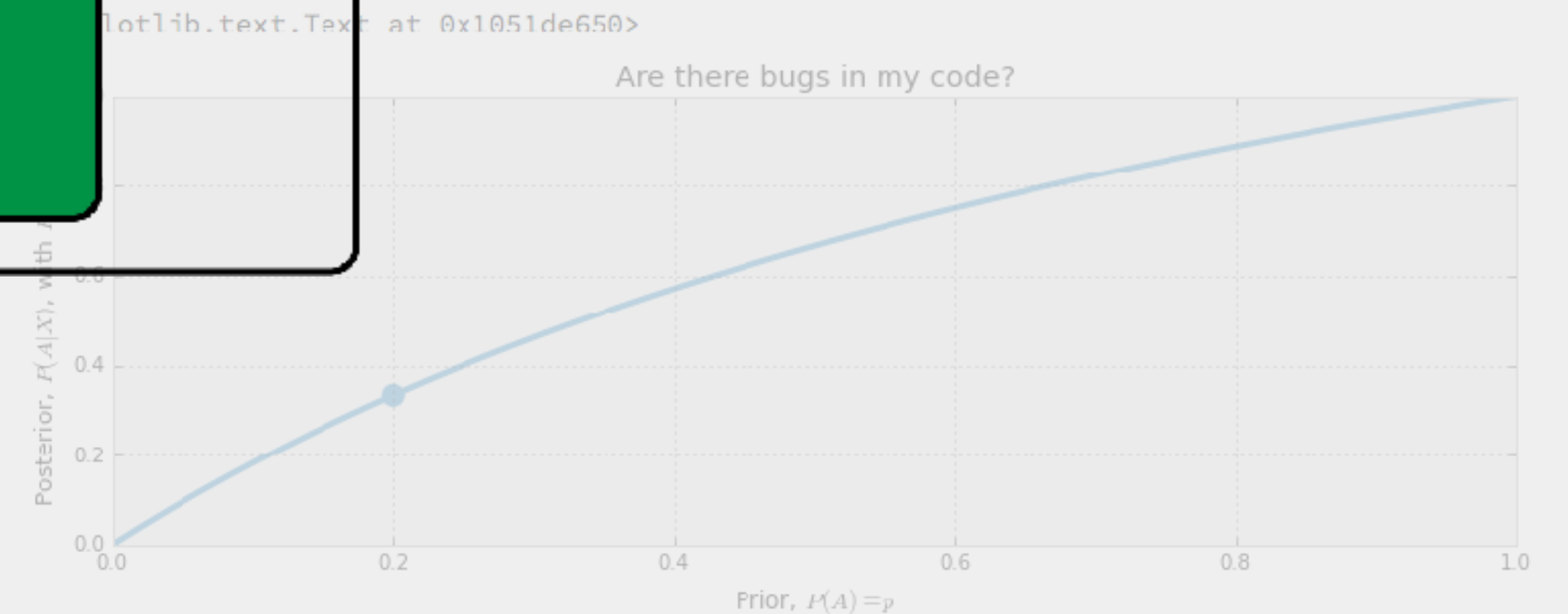
We have already computed  $P(X|A)$  above. On the other hand,  $P(X| \sim A)$  is subjective: our code can pass tests but still have a bug in it, though the probability there is a bug present is reduced. Note this is dependent on the number of tests performed, the degree of complication of the tests, etc. Let's be conservative and assign  $P(X| \sim A) = 0.5$ . Then

$$P(A|X) = \frac{1 \cdot p}{1 \cdot p + 0.5(1 - p)}$$

$$= \frac{2p}{1 + p}$$

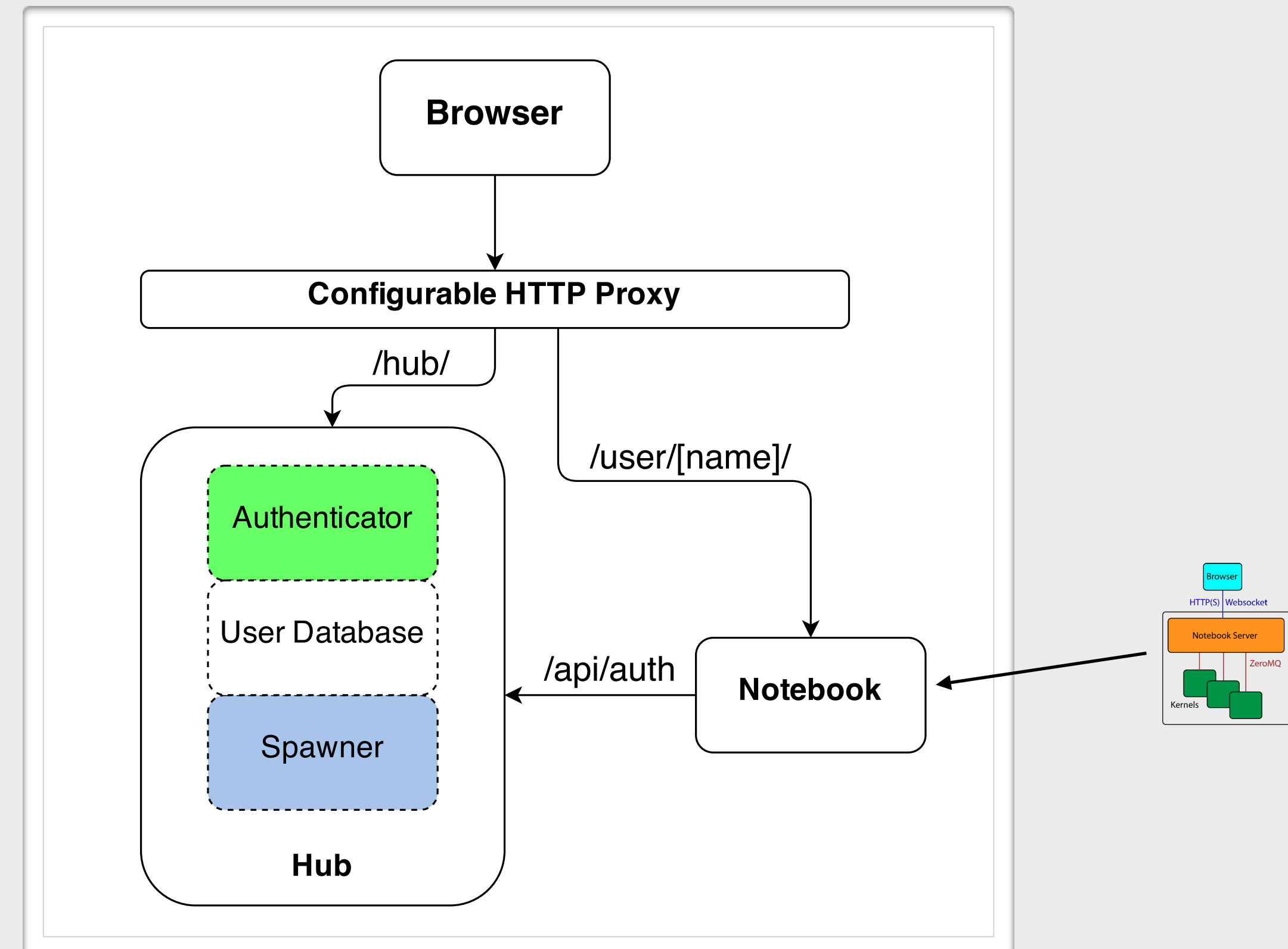
for probability. What does it look like as a function of our prior,  $p \in [0, 1]$ ?

```
figsize(12.5, 4)
p = np.linspace(0, 1, 50)
plt.plot(p, p / (1 + p), color="#348ABD", lw=3)
# plt.fill_between(p, 2*p/(1+p), alpha=.5, facecolor=["#A60628"])
plt.scatter(0.2, 2 * (0.2) / 1.2, s=140, c="#348ABD")
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.xlabel("Prior, $P(A) = p$")
plt.ylabel("Posterior, $P(A|X)$, with $P(A) = p$")
plt.title("Are there bugs in my code?")
```



# Authenticated Notebook Service

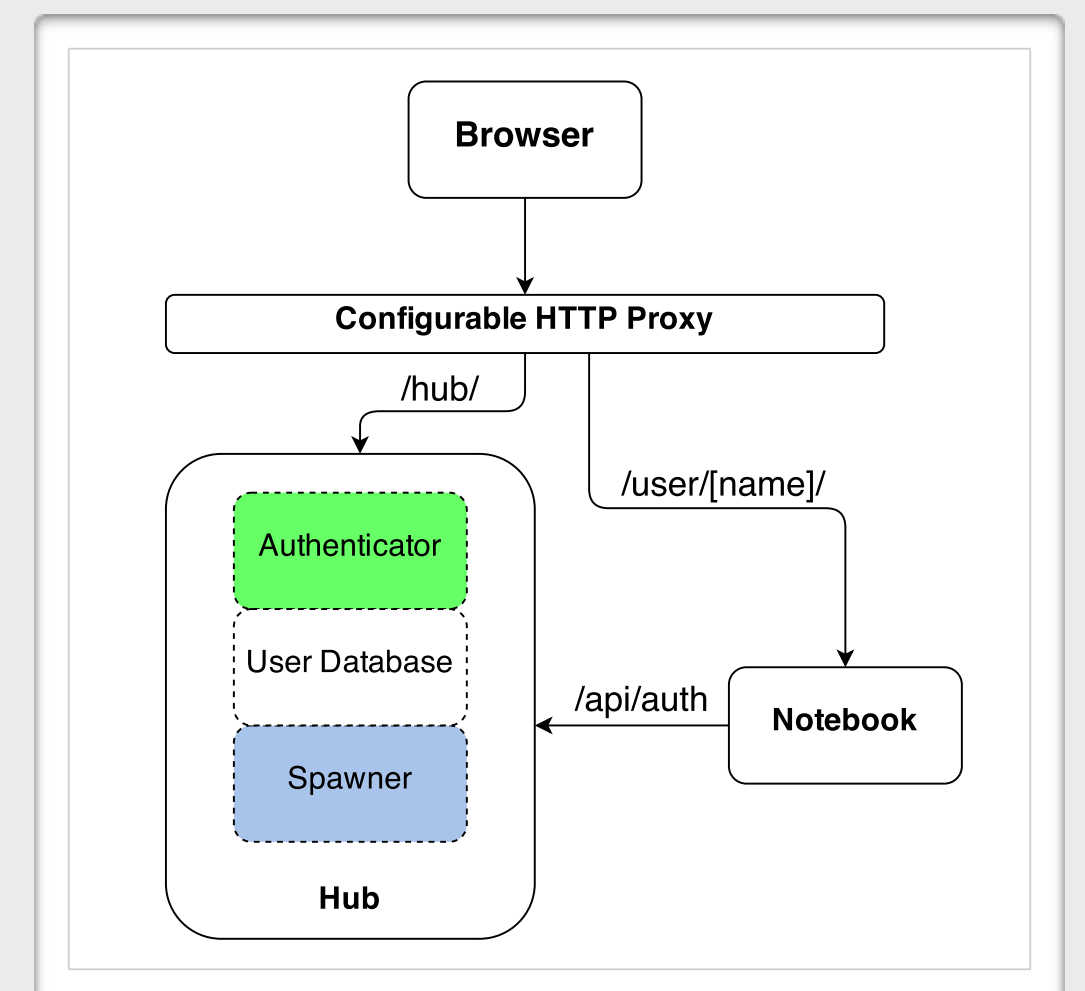
JupyterHub





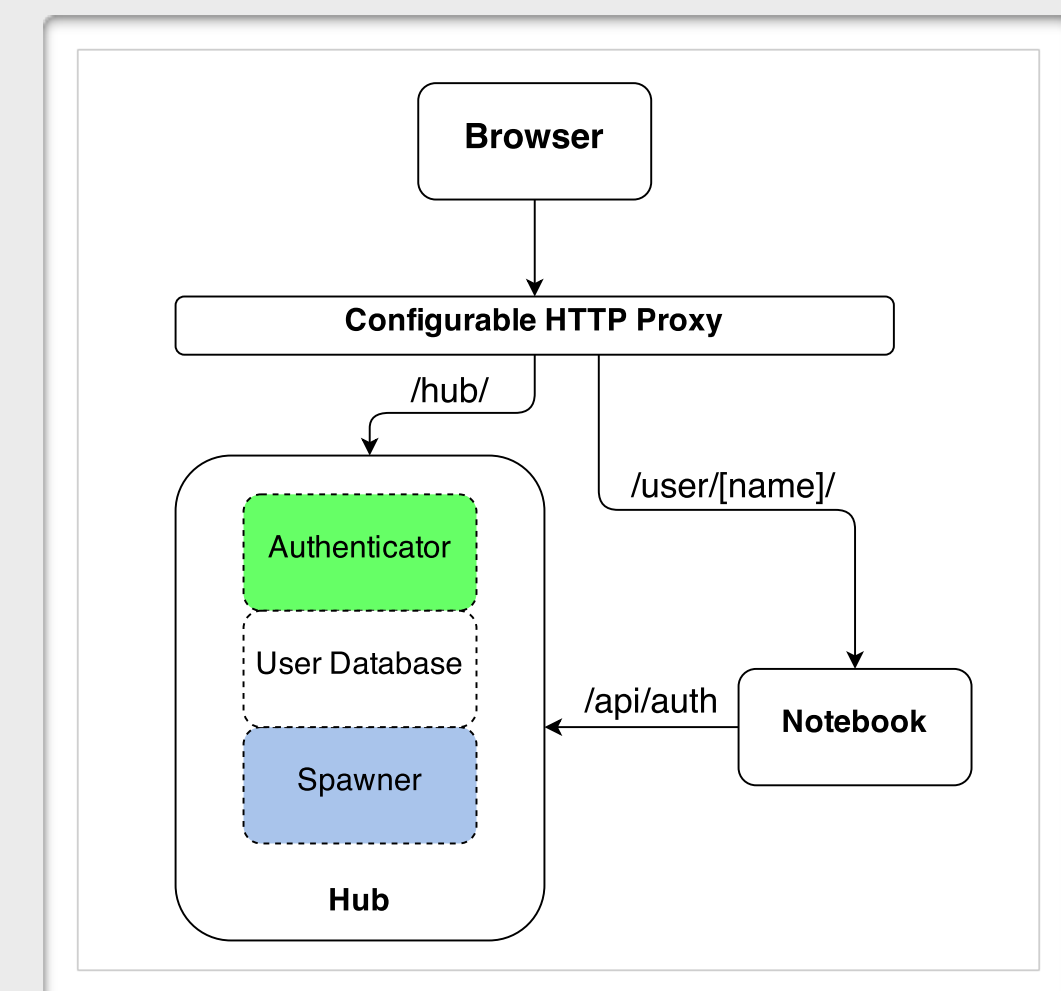
# JupyterHub

- **Proxy** in front (configurable-http-proxy)
- **Authenticator** handles authentication
- **Spawner** launches single-user servers
- **Services** for integrating with the Hub
- **Hub** connects everything



# JupyterHub

- Allows Jupyter notebooks to be the access point for your cluster
- Integrates with existing authentication
- Can spawn notebook servers via PBS/SLURM/Kubernetes/etc.
- in use at institutions:  
UC Berkeley, NERSC, XSEDE,  
Compute Canada,  
San Diego Supercomputing  
Center, Bryn Mawr, more



# IPython and Dask

Two libraries for interactive Data Science



# IPython Parallel

- **Direct** multiplexing API to remote namespaces
- Interactive **MPI** Simulations
- Facilitated debugging or steering traditional MPI
- Supports simple task-farming as well
- Focused on **interactivity, debugging**
- concurrent.futures-based **Executor** API



# Dask Distributed

- **Graph** evaluation engine
- High-level APIs for **collections, data frames, arrays**
- Lazy graph evaluation - only compute what you need
- Detailed profiling information
- Data-source support for S3, HDFS, files, HDF5, memory, etc.
- concurrent.futures-based **Executor** API





Demo



# Takeaways

- **Jupyter** is a web-based application for interactive computing, widely used by data scientists
- **JupyterHub** has the building blocks for institutional deployments, enabling shell-free access to compute
- **IPython Parallel** enables interactive steering, development, and debugging of traditional parallel tasks, including **MPI** jobs
- **Dask Distributed** provides high-level APIs for distributed algorithms and data structures, with analogies to Apache Spark





Interactive HPC

with Jupyter, IPython, and Dask

SOS 21

Min Ragan-Kelley  
Simula Research Lab