jupyter

Deploying JupyterHub for students and researchers

JupyterCon 2017

Min Ragan-Kelley, Simula
Carol Willing, Cal Poly
Yuvi Panda, UC Berkeley
Ryan Lovett, UC Berkeley

Set Up

```
git clone \
    https://github.com/jupyterhub/jupyterhub-tutorial \
    /srv/jupyterhub
```

Tutorial logistics

9:00 - 9:15	Welcome
9:15 - 10:30	JupyterHub
10:30 - 11:00	Morning break
11:00 - 12:20	JupyterHub and Kubernetes
12:20 - 12:30	Wrap up

Tutorial logistics

- Gitter channel for this tutorial

  https://gitter.im/jupyterhub-tutorial/Lobby

- Post-its

  questions and assistance

  done with task and doing well

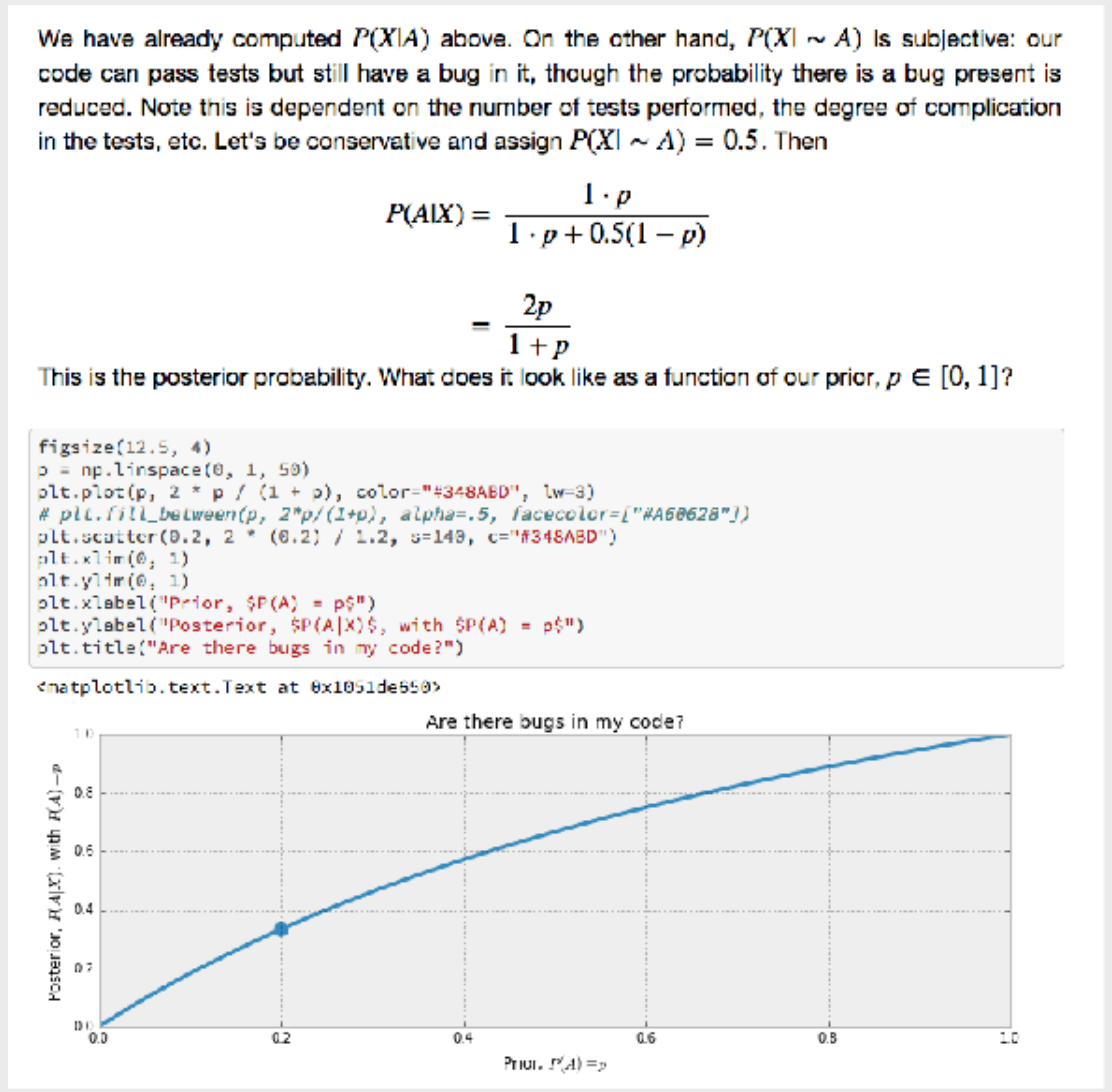
- Tutorial materials

  https://github.com/jupyterhub/jupyterhub-tutorial

What are Jupyter and JupyterHub?

What is a Notebook?

- Document
- Environment
- **Web app**



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, $p \in [0, 1]$?

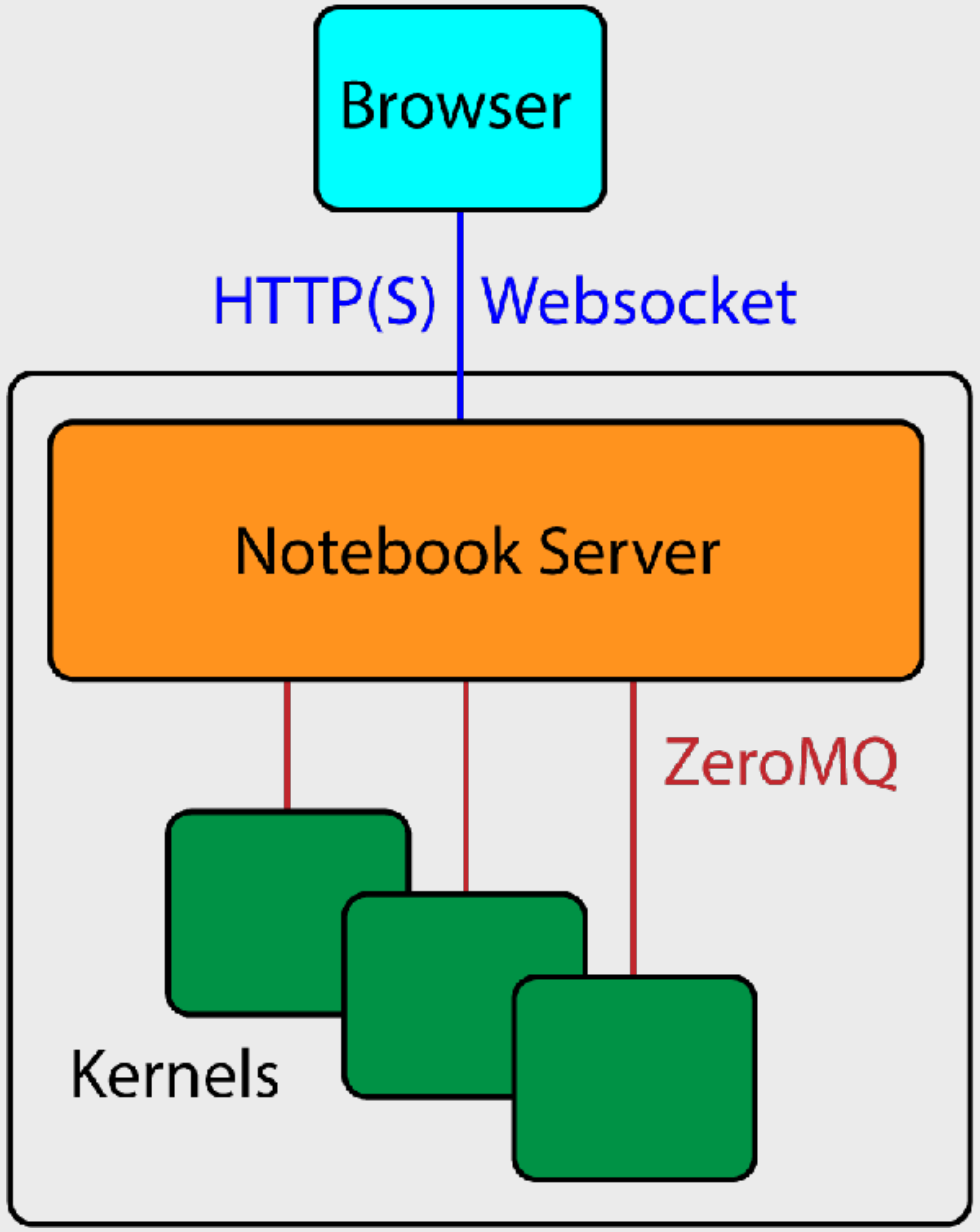
```
figsize(12.5, 4)
p = np.linspace(0, 1, 50)
plt.plot(p, 2 * p / (1 + p), color="#348ABD", lw=3)
# plt.fill_between(p, 2*p/(1+p), alpha=.5, facecolor=["#A60628"])
plt.scatter(0.2, 2 * (0.2) / 1.2, s=140, c="#348ABD")
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.xlabel("Prior, $P(A) = p$")
plt.ylabel("Posterior,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

```
<matplotlib.text.Text at 0x1051de550>
```

Are there bugs in my code?
Posterior, $P(A|X)$, with $P(A) = p$
Prior, $P(A) = p$
0.0
0.2
0.4
0.6
0.8
1.0

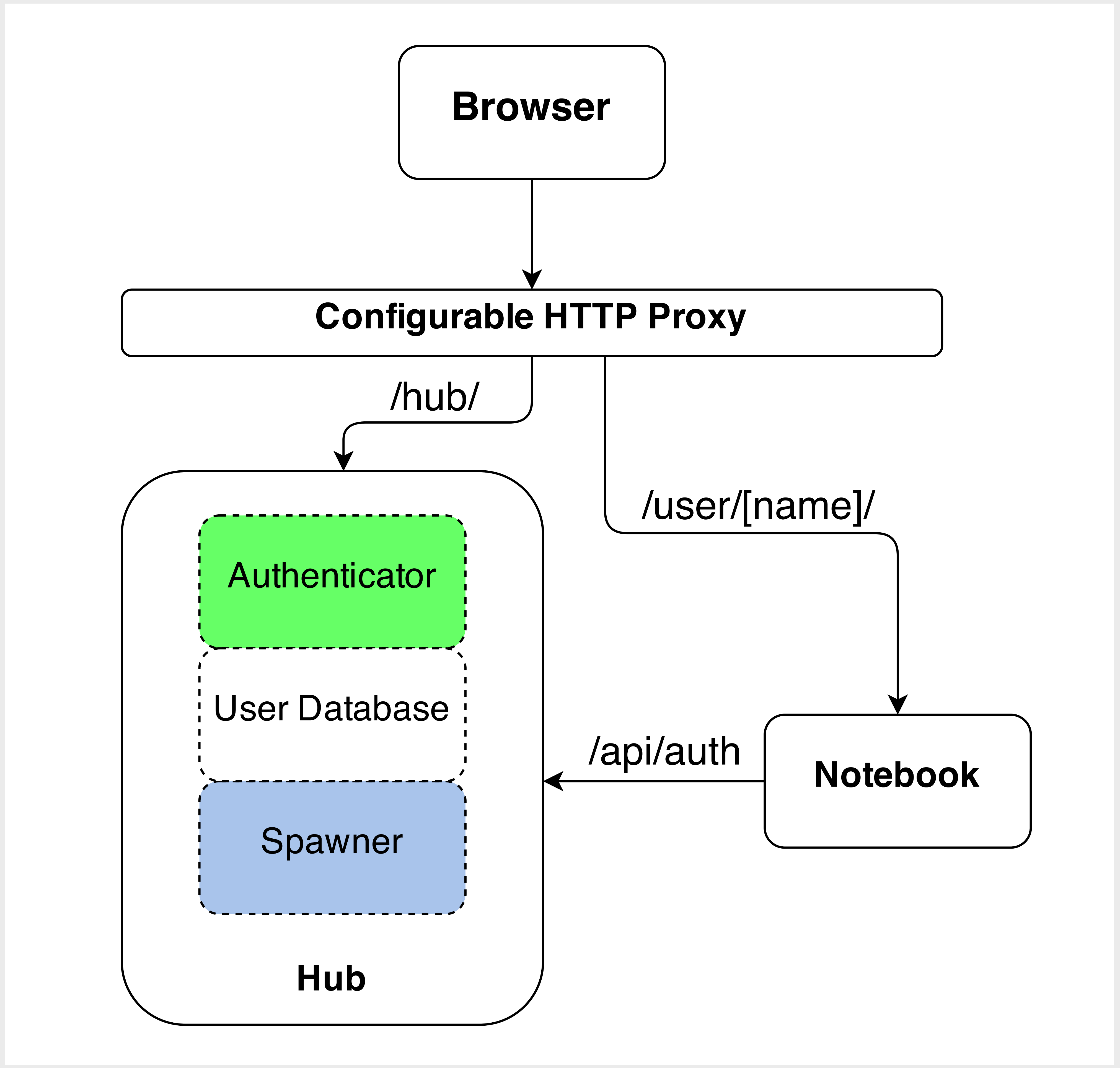
https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers

What is a Notebook Server?



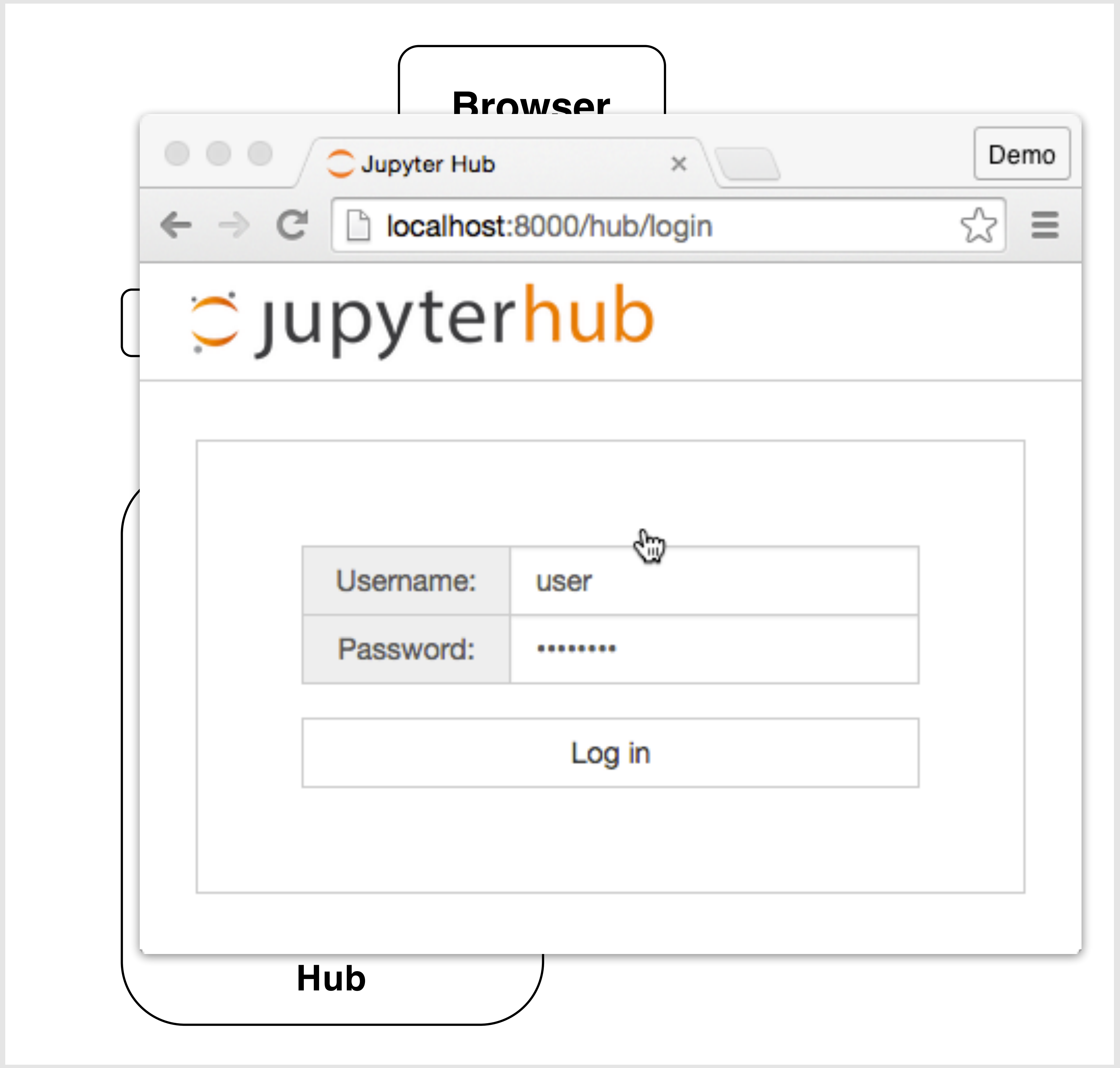
jupyterhub

- Manages authentication
- Spawns single-user servers on-demand
- Each user gets a complete notebook server



jupyterhub

- Initial request is handled by Hub
- User authenticates via form / OAuth
- Spawner starts single-user server
- Hub notifies Proxy
- Redirects user to `/user/[name]`
- Single-user Server verifies auth with Hub



Username: user
Password: ••••••••

Installation

Photo taken by Matthew Bowers

Installation (as administrator)

conda:

```
conda install -c conda-forge jupyterhub
conda install notebook
```

pip, npm:

```
python3 -m pip install jupyterhub
npm install -g configurable-http-proxy
```

test:

```
jupyterhub -h
configurable-http-proxy -h
```

Installation (this repo)

```
conda env create -f environment.yml
source activate jupyterhub-tutorial
```

Installation Caveats

JupyterHub installation must be **readable+executable** by all users*

This is often not the case for envs, so be careful

*when using local users

Plug: conda-forge

Community-managed conda packages.

https://conda-forge.github.io

```
conda config --add channels conda-forge
```

Installation: Spawner and Single User

https://docs.docker.com/engine/installation

```
pip install dockerspawner
docker pull jupyterhub/singleuser
```

JupyterHub Defaults

- Authentication: PAM (local users, passwords)
- Spawning: Local users
- Hub must run as root

Aside: SSL

- JupyterHub is an authenticated service - users login. That should **never** happen over plain HTTP.
- For testing, we can generate self-signed certificates:

```
openssl req -x509 -nodes -days 365 -newkey rsa:1024 \
  -keyout jupyterhub.key -out jupyterhub.crt
```

Note: Safari will not connect websockets to untrusted (self-signed) certs

SSL: Let’s Encrypt

- https://letsencrypt.org/getting-started/
- Free SSL for any domain

```
wget https://dl.eff.org/certbot-auto
chmod a+x certbot-auto
./certbot-auto certonly --standalone -d mydomain.tld

key:  /etc/letsencrypt/live/mydomain.tld/privkey.pem
cert: /etc/letsencrypt/live/mydomain.tld/fullchain.pem
```

Configuration

Photo taken by Matthew Bowers

Start configuring JupyterHub

```
jupyterhub --generate-config
```

```
c.JupyterHub.ssl_key = 'jupyterhub.key'
c.JupyterHub.ssl_cert = 'jupyterhub.crt'
c.JupyterHub.port = 443
```

jupyterhub_config.py

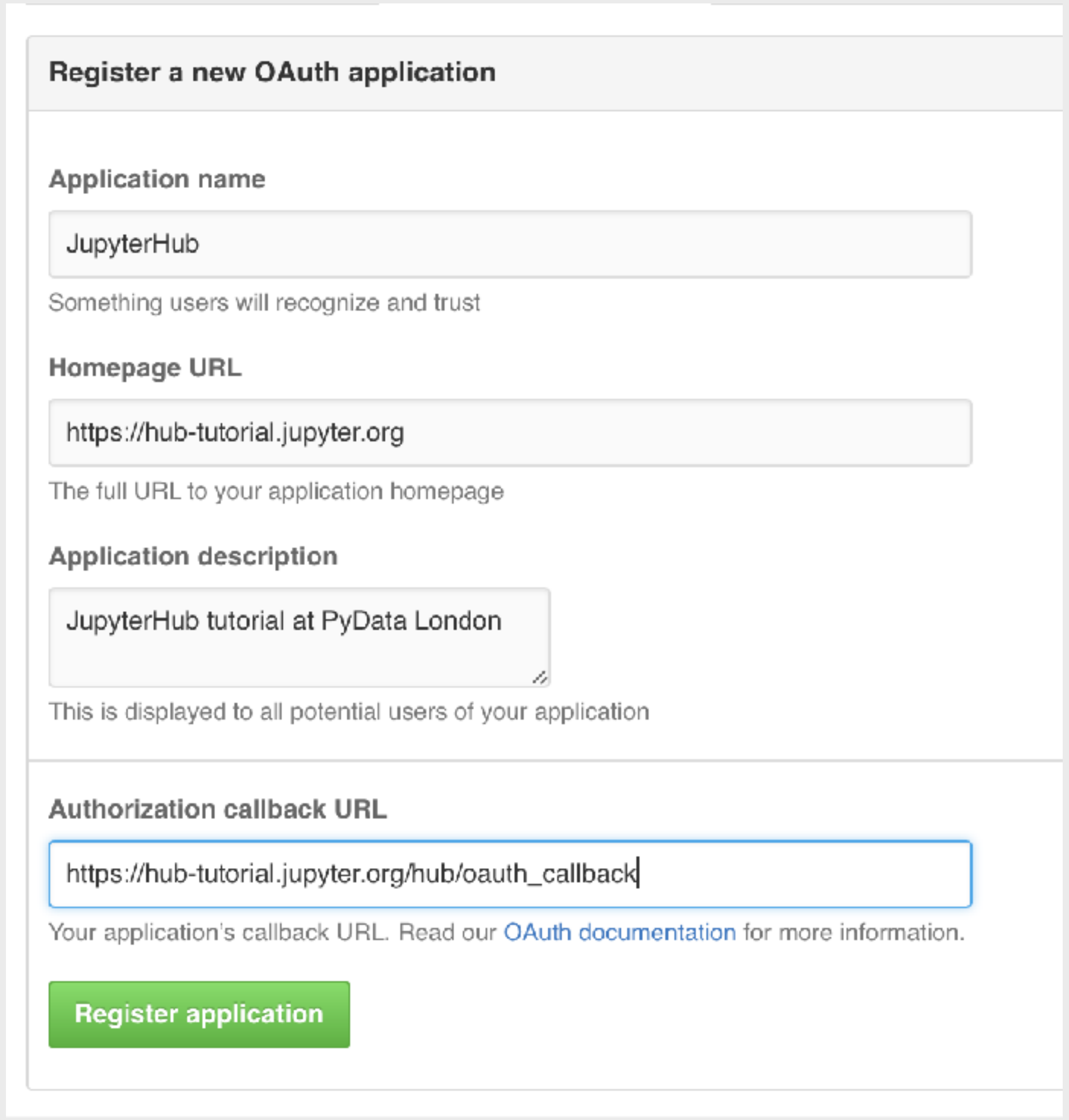
Installing language kernels for all users

```
conda create -n py2 python=2 ipykernel
conda run -n py2 -- ipython kernel install

jupyter kernelspec list
```

Using GitHub OAuth

https://github.com/settings/applications/new



Register a new OAuth application

Application name

JupyterHub

Something users will recognize and trust

Homepage URL

https://hub-tutorial.jupyter.org

The full URL to your application homepage

Application description

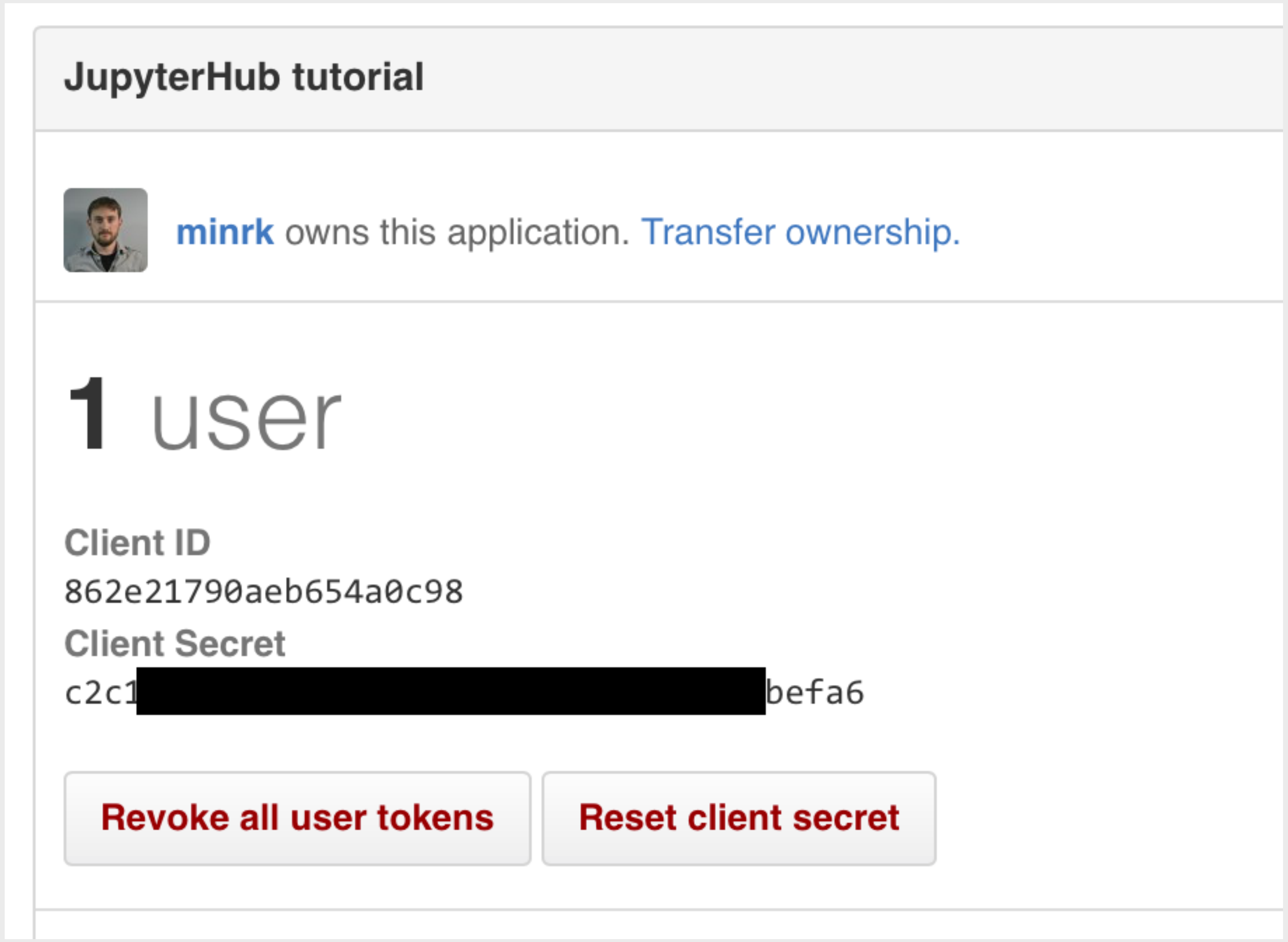
JupyterHub tutorial at PyData London

This is displayed to all potential users of your application

Authorization callback URL

https://hub-tutorial.jupyter.org/hub/oauth_callback

Your application's callback URL. Read our OAuth documentation for more information.



JupyterHub tutorial

minrk owns this application. Transfer ownership.

1 user

Client ID

```
862e21790aeb654a0c98
```

Client Secret

```
c2c1[redacted]befa6
```

Using GitHub OAuth

File: ./env

```
export GITHUB_CLIENT_ID=from_github
export GITHUB_CLIENT_SECRET=from_github
export OAUTH_CALLBACK_URL=https://YOURDOMAIN/hub/oauth_callback
```

```
source ./env
```

Using GitHub OAuth

We need to install: OAuthenticator

```
python3 -m pip install oauthenticator
```

Config file: `jupyterhub_config.py`

```
from oauthenticator.github import LocalGitHubOAuthenticator
c.JupyterHub.authenticator_class = LocalGitHubOAuthenticator
c.LocalGitHubOAuthenticator.create_system_users = True
```

Specifying Users

By default, any user that successfully authenticates is allowed to use the Hub.

This is appropriate for shared workstations with PAM Auth, but probably not GitHub:

```
# set of users allowed to use the Hub
c.Authenticator.whitelist = {'minrk', 'takluyver'}

# set of users who can administer the Hub itself
c.Authenticator.admin_users = {'minrk'}
```

Custom Authenticators

Photo taken by Matthew Bowers

Using DockerSpawner

We need DockerSpawner:

```
python3 -m pip install dockerspawner netifaces
docker pull jupyterhub/singleuser
```

In jupyterhub_config.py:

```python
from oauthenticator.github import GitHubOAuthenticator
c.JupyterHub.authenticator_class = GitHubOAuthenticator

from dockerspawner import DockerSpawner
c.JupyterHub.spawner_class = DockerSpawner
```

Using DockerSpawner

```python
from dockerspawner import DockerSpawner
c.JupyterHub.spawner_class = DockerSpawner

# The Hub's API listens on localhost by default,
# but docker containers can't see that.
# Tell the Hub to listen on its docker network:
import netifaces
docker0 = netifaces.ifaddresses('docker0')
docker0_ipv4 = docker0[netifaces.AF_INET][0]
c.JupyterHub.hub_ip = docker0_ipv4['addr']
```

Using DockerSpawner

- There is *loads* to configure with Docker
- Networking configuration
- Data volumes
- `DockerSpawner.image = ‘jupyter/scipy-notebook:8f56e3c47fec’`

Customizing Spawners

Photo taken by Matthew Bowers

JupyterHub with supervisor

apt-get install supervisor

```bash
#!/usr/bin/env bash
# /srv/jupyterhub/launch.sh
set -e
source ./env
exec jupyterhub $@
```

```ini
# /etc/supervisor/conf.d/jupyterhub.conf
[program:jupyterhub]
command=bash launch.sh
directory=/srv/jupyterhub
autostart=true
autorestart=true
startretries=3
exitcodes=0,2
stopsignal=TERM
redirect_stderr=true
stdout_logfile=/var/log/jupyterhub.log
stdout_logfile_maxbytes=1MB
stdout_logfile_backups=10
user=root
```

Reference Deployments

https://github.com/jupyterhub/helm-chart

Helm, KubeSpawner, Hub in container

https://github.com/jupyterhub/jupyterhub-deploy-docker

docker-compose, DockerSpawner, Hub in Docker

https://github.com/jupyterhub/jupyterhub-deploy-teaching

ansible, no docker, nbgrader

Docker Deployment

- Docker Compose: https://docs.docker.com/compose/install/
- git clone https://github.com/jupyterhub/jupyterhub-deploy-docker
- Setup the basics (creates volumes, network, pulls images):

```
make build
```

Docker Deployment

- `mkdir secrets`
- Copy SSL key, cert to:
  - secrets/jupyterhub.cert (cert)
  - secrets/jupyterhub.key (key)

Docker Deployment

```
Make userlist:

   minrk admin
   willingc admin
   yuvipanda
   ryanlovett
```

Docker Deployment

Launch: 🚀

```
docker-compose up
```

Optimizations and Best Practices

Photo taken by Matthew Bowers

Optimizations and best practices

- **Always use SSL!**
- Use postgres for the Hub database
- Put nginx in front of the proxy
- Run cull-idle-servers service to prune resources
- Global configuration in `/etc/jupyter` and `/etc/ipython`
- **Back up your user data!!!**

When to use JupyterHub

- A class where students can do homework (nbgrader)
- A short-lived workshop, especially if installation is hard
- A research group with a shared workstation or small cluster
- On-site computing resources for researchers and analysts at an institution

When ***not*** to use JupyterHub

- JupyterHub is Authenticated and Persistent
- It takes work to keep it going
- SageMathCloud is *hosted* and provides realtime-collaboration
- A growing number of cloud providers support notebook services (Google, Microsoft, etc.)

JupyterHub API

Photo taken by Matthew Bowers

End of JupyterHub Fundamentals

Break - 30 min
See you back here at 11:00 am.

Up next:
JupyterHub and Kubernetes

Attributions and recognition

A huge thank you to the Project Jupyter team and community.
Your hard work and passion makes this all possible.

GORDON AND BETTY MOORE FOUNDATION

Alfred P. Sloan FOUNDATION

THE LEONA M. AND HARRY B. HELMSLEY CHARITABLE TRUST

Thank you

Thank you

[try.jupyter.org]

[www.jupyter.org]

[ipython.org]

[numfocus.org]