Jupyter

JupyterHub

JupyterCon 2017

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

[FYI for tutorial planning] Description

Description

JupyterHub, a multiuser server for Jupyter notebooks, enables you to offer a notebook server to everyone in a group—which is particularly useful when teaching a course, as students no longer need to install software on their laptops. Min Ragan-Kelley, Carol Willing, Yuvi Panda, and Ryan Lovett get you started deploying and customizing JupyterHub for your needs.

Min, Carol, Yuvi, and Ryan explore JupyterHub's architecture and how to assemble a deployment exactly the way you want it. They then walk you through a complete deployment of JupyterHub with Kubernetes following best practices learned from the campus-wide deployment at UC Berkeley, including integration with campus authentication (via Google OAuth), status monitoring and data collection with statsd, and automating as much of the deployment and maintenance process as possible. Along the way, Min, Carol, Yuvi, and Ryan demonstrate how development, testing, and production systems can be easily managed to increase reliability and scaling of the deployment and how to customize your components (authenticators, spawners) and provide examples of services that can be managed by or interact with the hub and its users. You'll then create a simple, functioning JupyterHub deployment of your own.

Topics include:

- Authenticators that allow JupyterHub to integrate with any existing authentication system, such as GitHub or Google OAuth, PAM, and LDAP
- Spawners, which are used to run notebook servers on any system for spawning processes, such as Docker, Kubernetes, or local processes
- Proxies
- Services

[FYI for tutorial planning] Outline

Outline

JupyterHub overview

- Getting started with JupyterHub
- Composing a deployment
  - What parts can I choose from? (Authenticator, spawner, proxy, single-user server image (if applicable))
  - How can I put them together?
- Composing the simplest possible working deployment
- Swapping out various parts from the simplest deployment
  - Using and configuring a different authenticator
  - Using and configuring a different spawner

Putting together a deployment of JupyterHub with Kubernetes

- Background: Getting started with Kubernetes
- Basic concepts
  - A mental model for working with Kubernetes
  - Pods, deployments, and services
  - Persistent volumes and cloud providers
  - Where to learn more (lecture series, books, wonderful tutorials, etc.)
- Safely deploying JupyterHub by using standard release engineering practices
  - Deploying and upgrading a JupyterHub with helm
- Building your own user images
- How to debug your deployment when things inevitably go wrong

Appendix: More customization

- Subclassing in jupyterhub_config.py for fun and profit
- Writing your own authenticator
- Writing your own spawner

Set Up

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

What is Jupyter and JupyterHub?

What is a Notebook?
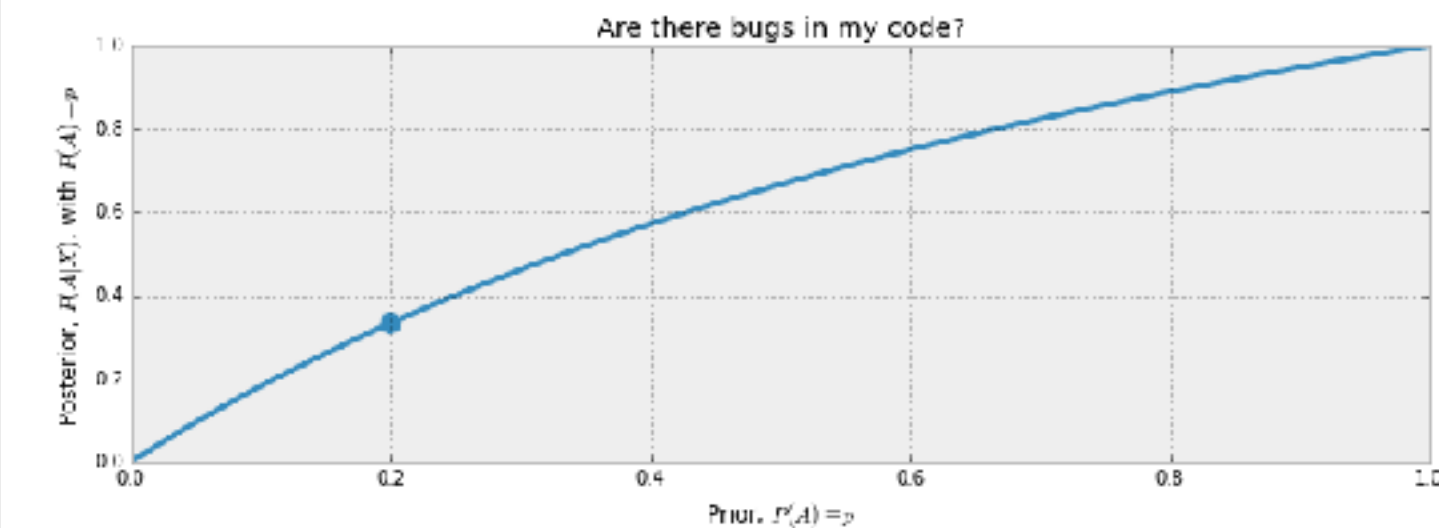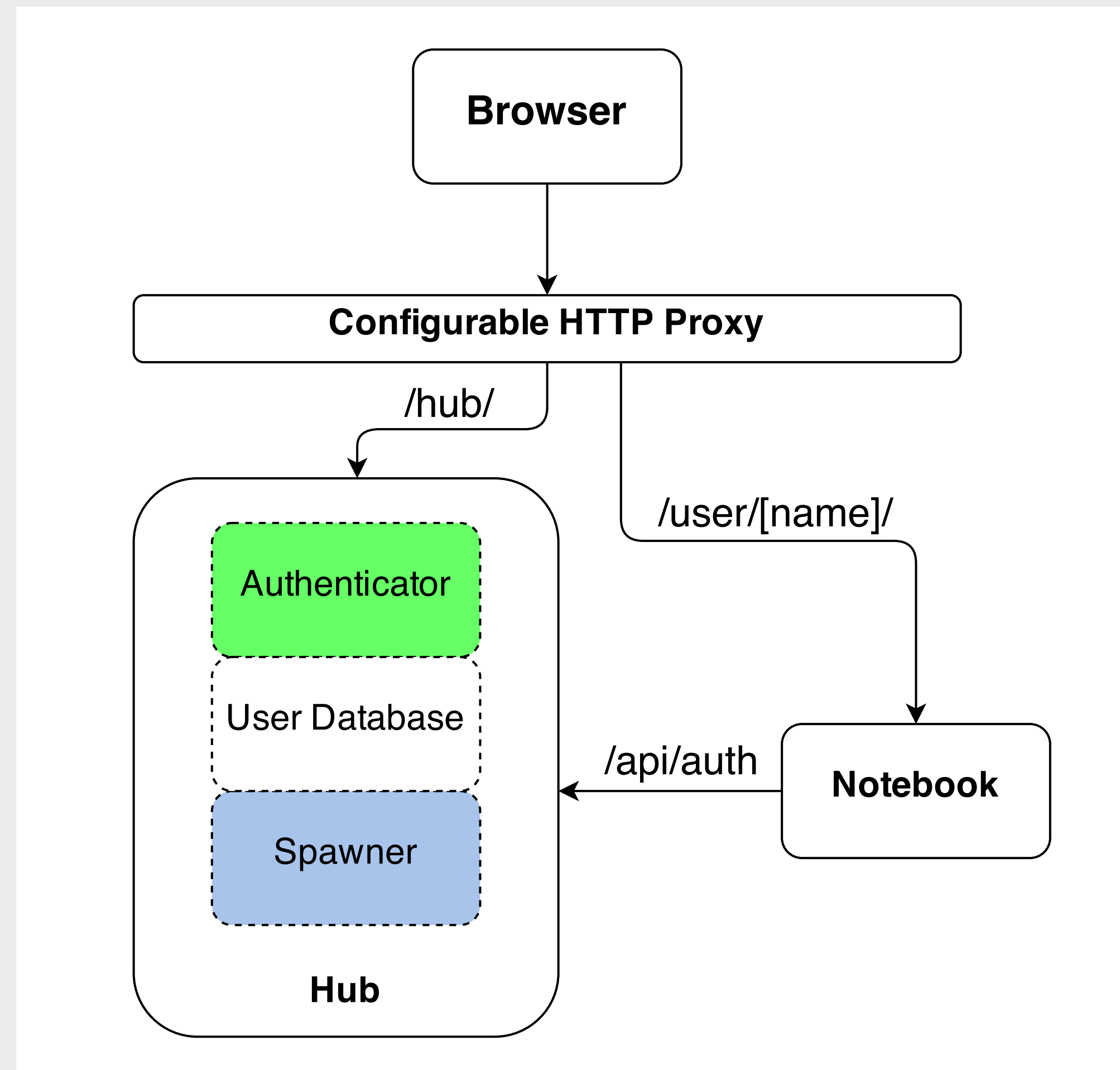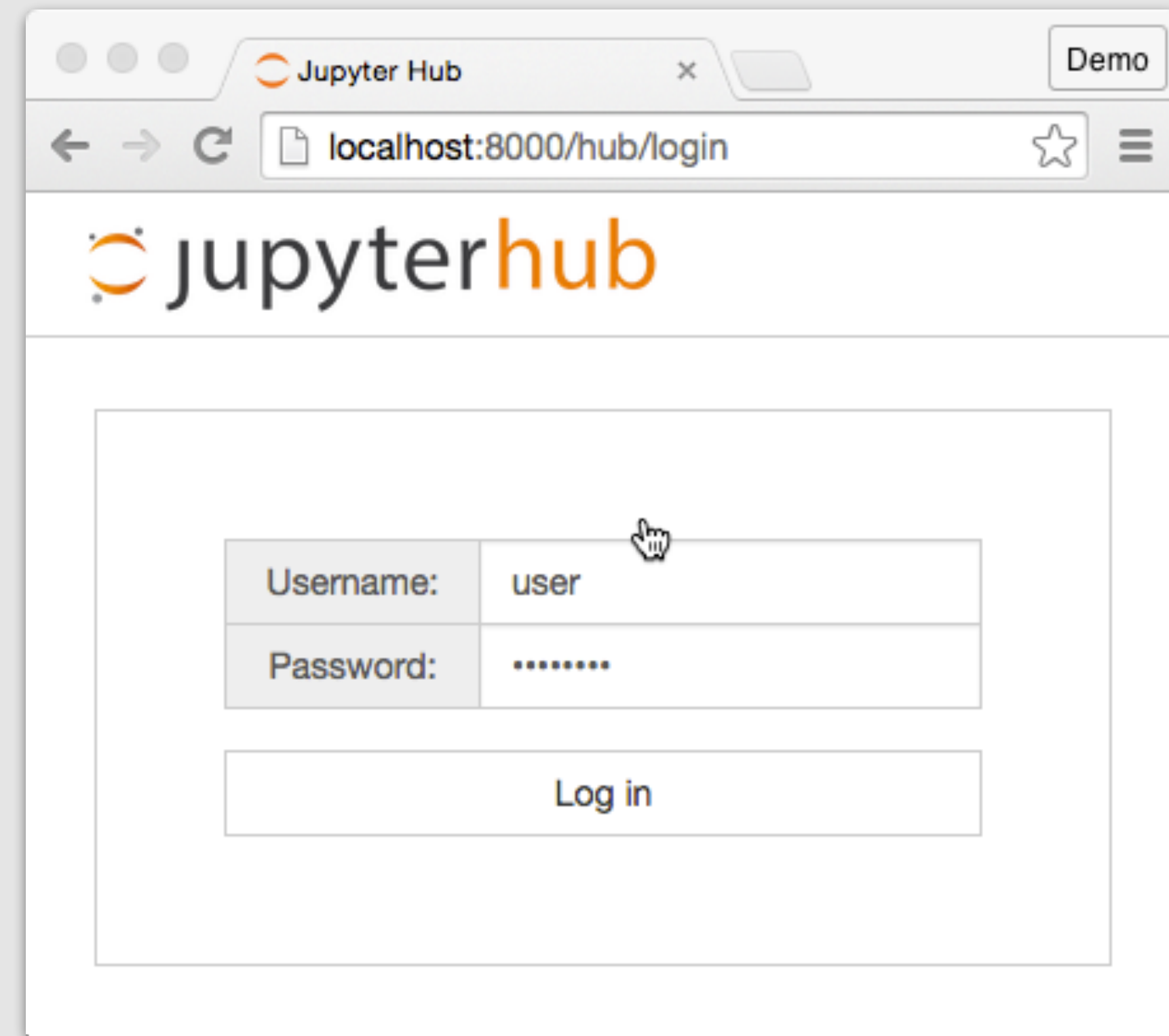

- Document

- Environment

- **Web app**

What is a Notebook Server?
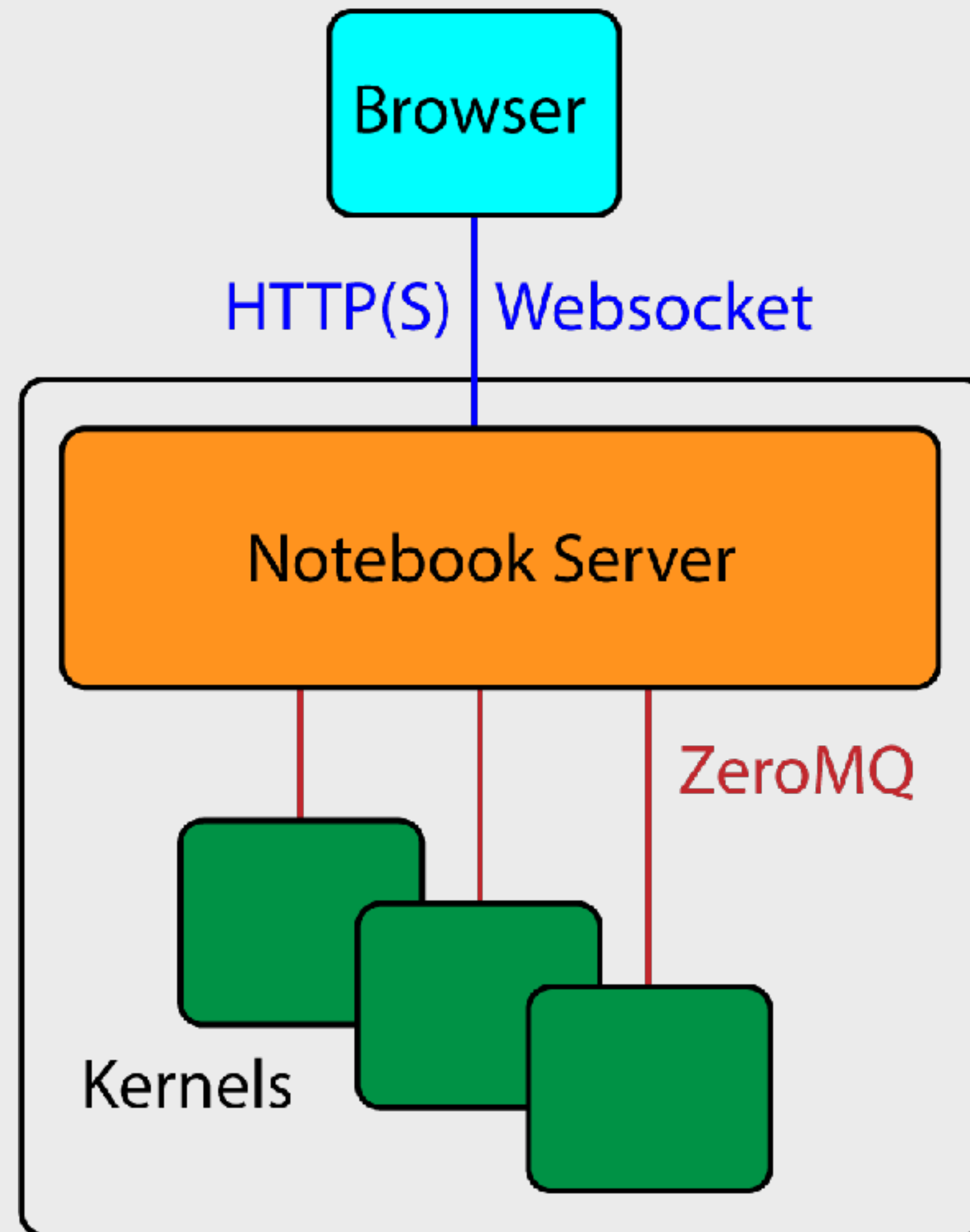
Jupyterhub

- Manages authentication

- Spawns single-user servers on-demand

- Each user gets a complete notebook server

Browser

Configurable HTTP Proxy

/hub/

/user/[name]/

Authenticator

User Database

/api/auth

Notebook

Spawner

Hub

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

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
```

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

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
```

Configure file: `jupyterhub_config.py`

```python
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.container_image = 'jupyterhub/singleuser'`

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

- tmpnb: anonymous, ephemeral notebooks

- binder: tmpnb + GitHub repos

- SageMathCloud is *hosted* and provides realtime-collaboration

JupyterHub API

Photo taken by Matthew Bowers

End of JupyterHub Fundamentals

Break
See you back in 15 minutes.

Up next: JupyterHub and Kubernetes

JupyterHub with Kubernetes

Putting together a deployment of JupyterHub with Kubernetes

- Background: Getting started with Kubernetes
- Basic concepts
  - A mental model for working with Kubernetes
  - Pods, deployments, and services
  - Persistent volumes and cloud providers
  - Where to learn more (lecture series, books, wonderful tutorials, etc.)
- Safely deploying JupyterHub by using standard release engineering practices
  - Deploying and upgrading a JupyterHub with helm
- Building your own user images
- How to debug your deployment when things inevitably go wrong

End of JupyterHub with Kubernetes

Break
See you back in 5 minutes.

Up next:
More customization
(if time permits)

More customization

Appendix: More customization

- Subclassing in jupyterhub_config.py for fun and profit
- Writing your own authenticator
- Writing your own spawner

End of
More Customization

Attributions and recognition

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

Thank you

Thank you

try.jupyter.org

www.jupyter.org

ipython.org

numfocus.org