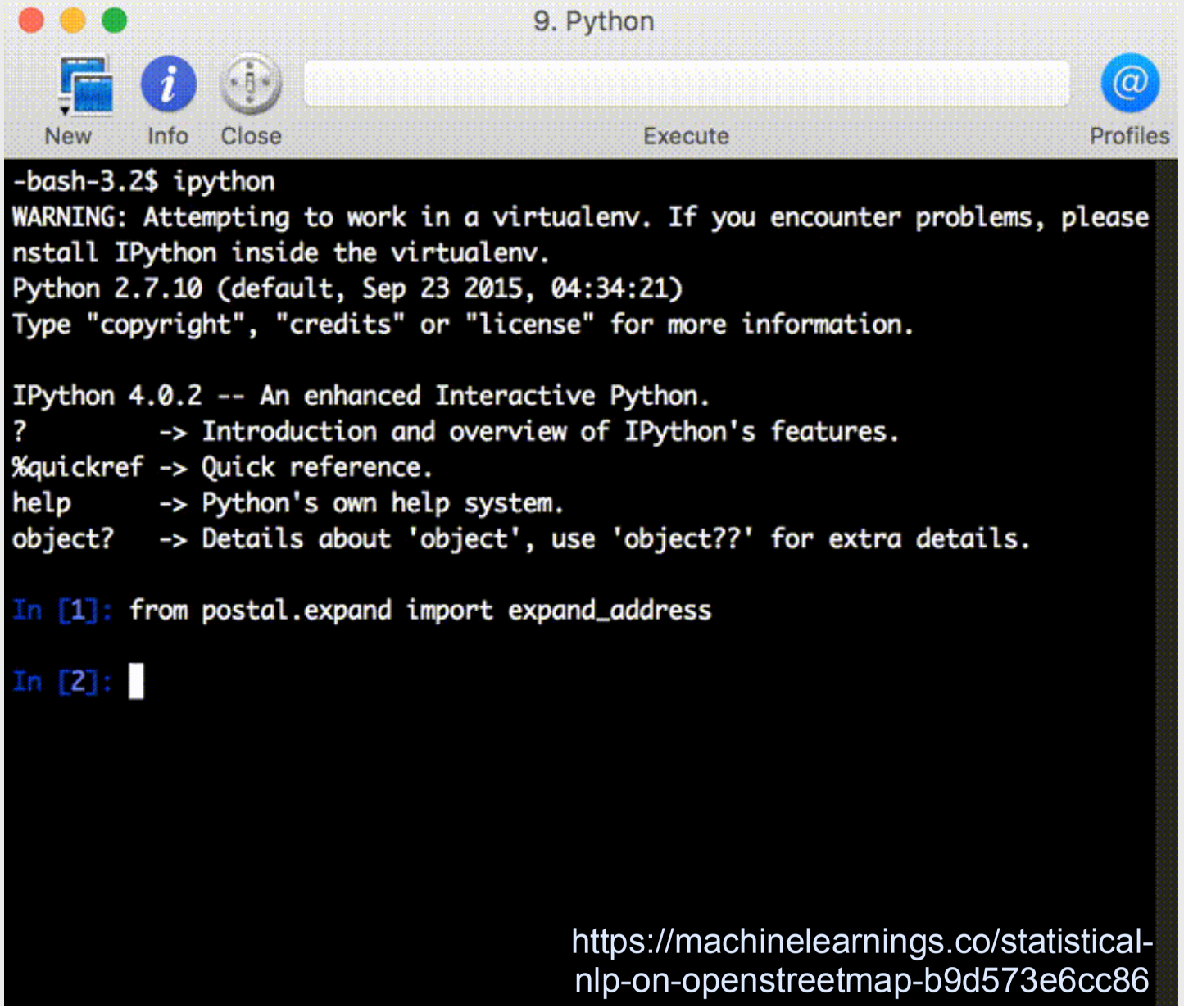
Reproducibility and Extensibility in Scientific Research

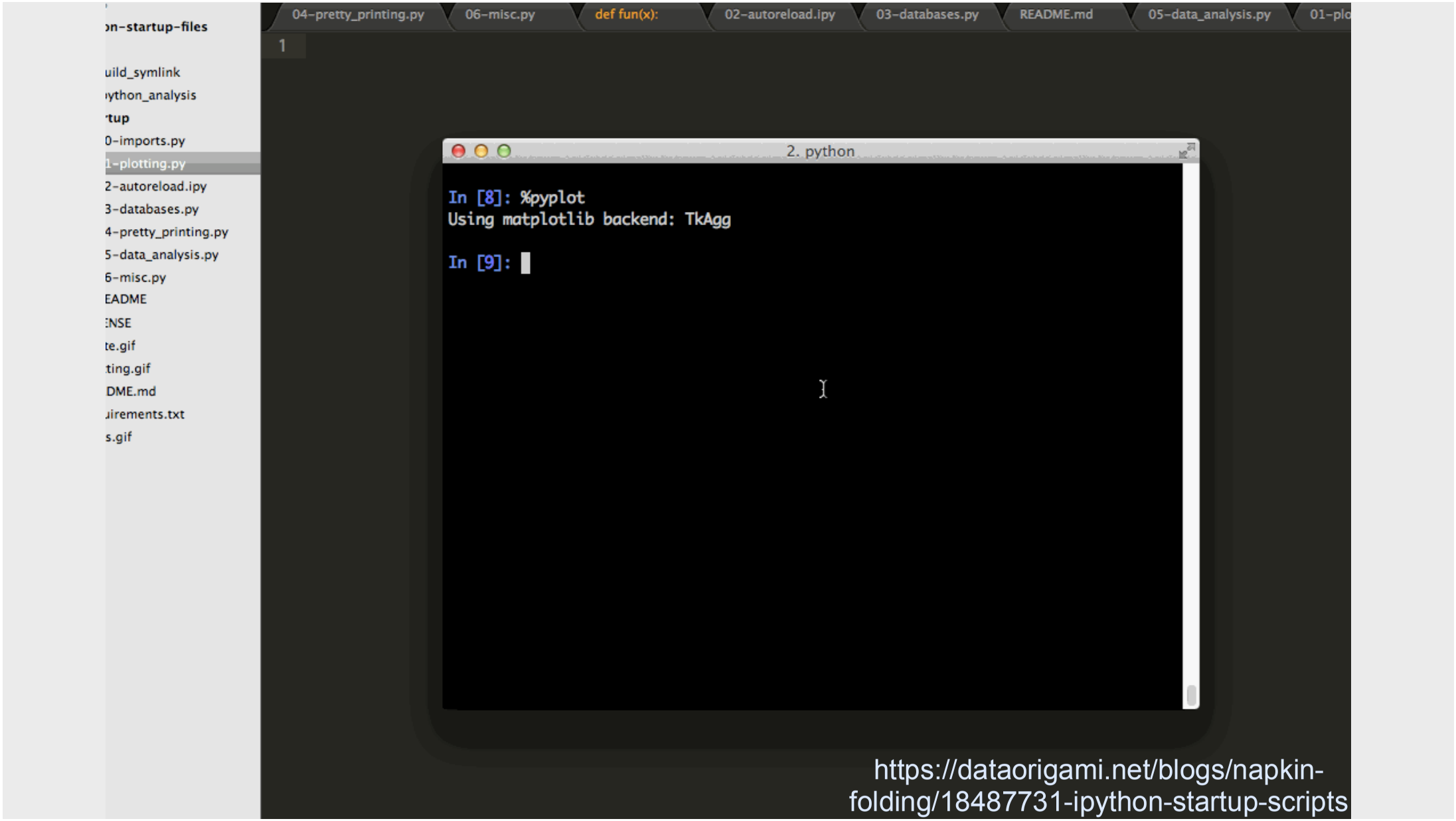
Jessica Forde
Project Jupyter
@projectjupyter
@mybinderteam

Overview

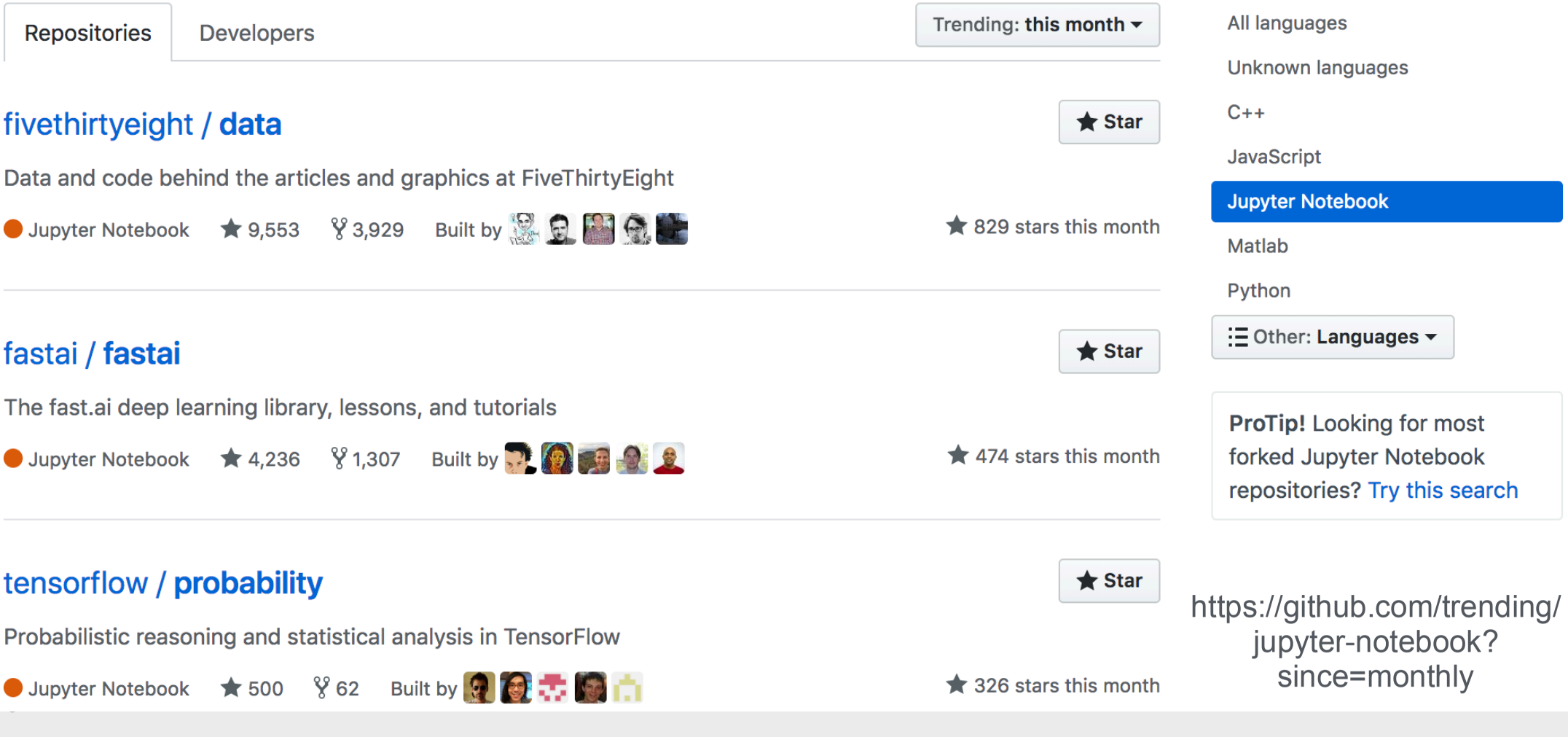
- Project Jupyter
- IPython
- Jupyter Notebook
- Architecture of JupyterHub
- The problem of reproducibility in science
- Repo2docker
- Binder
- Extending research via interactive computing

Who are we?

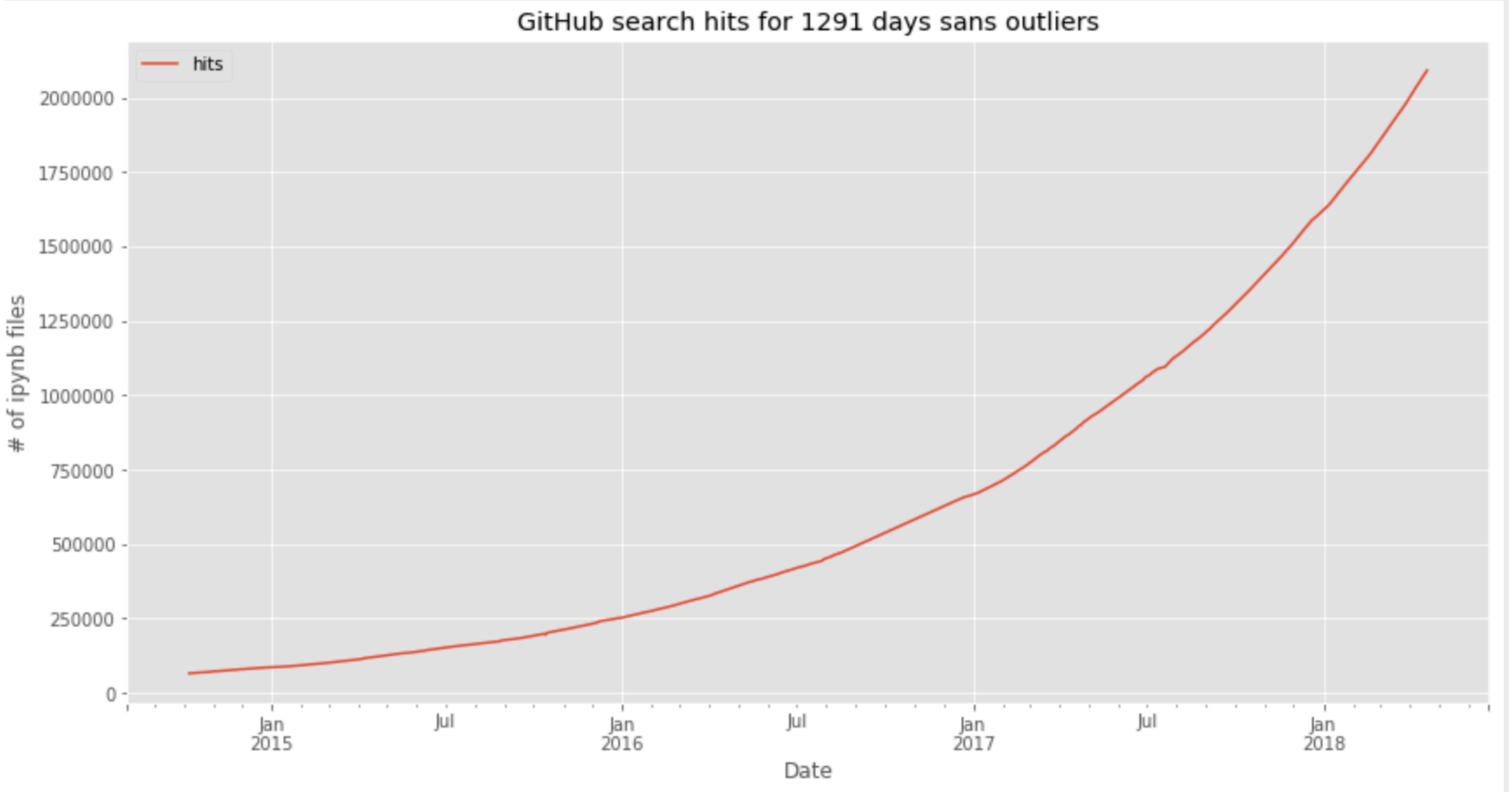

9. Python
New
Info
Close
Execute
Profiles
-bash-3.2$ ipython
WARNING: Attempting to work in a virtualenv. If you encounter problems, please
nstall IPython inside the virtualenv.
Python 2.7.10 (default, Sep 23 2015, 04:34:21)
Type "copyright", "credits" or "license" for more information.
IPython 4.0.2 -- 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.
In [1]: from postal.expand import expand_address
In [2]:
https://machinelearnings.co/statistical-
nlp-on-openstreetmap-b9d573e6cc86


on-startup-files
uild_symlink
ython_analysis
tup
0-imports.py
1-plotting.py
2-autoreload.ipy
3-databases.py
4-pretty_printing.py
5-data_analysis.py
6-misc.py
EADME
ENSE
te.gif
ting.gif
DME.md
irements.txt
s.gif
04-pretty_printing.py
06-misc.py
def fun(x):
02-autoreload.ipy
03-databases.py
README.md
05-data_analysis.py
01-plo
1
2. python
In [8]: %pyplot
Using matplotlib backend: TkAgg
In [9]:
https://dataorigami.net/blogs/napkin-folding/18487731-ipython-startup-scripts

~2.1 Million Notebooks on GitHub



https://github.com/trending/jupyter-notebook?since=monthly



https://mybinder.org/v2/gh/parente/nbestimate/master?filepath=estimate.src.ipynb


jupyter
Project Jupyter exists to develop open-source software, open-standards, and services
for interactive computing across dozens of programming languages.

Project Jupyter Mission

Project Jupyter

Project Jupyter exists to develop open-source software, open standards, and services for interactive and reproducible computing.

Jul 7, 2015 · 3 min read

GORDON AND BETTY MOORE FOUNDATION

ALFRED P. SLOAN FOUNDATION 1934

THE LEONA M. AND HARRY B. HELMSLEY CHARITABLE TRUST

New funding for Jupyter

We are pleased to announce that the Jupyter/IPython project has received $6M in funding from three organisations:

- The Leona M. and Harry B. Helmsley Charitable Trust
- The Gordon and Betty Moore Foundation
- The Alfred P. Sloan Foundation

The grant, which is being made both to the University of California, Berkeley and California Polytechnic State University, San Luis Obispo, will support the project for three years and includes new collaborations with the University of Southampton, and the Simula Research Lab in Norway.

Mission and Background

Project Jupyter's mission is to create open source tools for interactive scientific computing and data science in research, education and industry, with an emphasis on usability, collaboration and reproducibility.

This project is structured in a "3+1" format, with three main focus areas of research and development and one extra topic of ongoing work. The three focus areas are *Interactive Computing*, *Computational Narratives* and *Collaboration*. The problem of *Sustainability* will require ongoing attention but is conceptually distinct from the first three, as it doesn't focus on specific research questions or deliverables.

Newsjournal of the Society for Industrial and Applied Mathematics sinews.siam.org

siam news

Volume 51/ Issue 2
March 2018

Jupyter: Tools for the Life Cycle of a Computational Idea

By Min Ragan-Kelley, Carol Willing, and Jason Grout

Computation is increasingly becoming an integral part of science and education across disciplines. The life cycle of a computational idea typically involves interactive exploration and experimentation, as well as publication and communication of results. Reproducible computation demands open research tools, good software practices, and transparent documentation of research processes and results. Project Jupyter[1] is an open community that builds open-source software tools and protocols for the life cycle of a computational idea. Two core pieces of the project are an open protocol for interactive computation and an open document format with which to record and share computational ideas. The Jupyter Notebook application builds on these to provide a powerful, interactive, computational environment.

The Jupyter Notebook

What Is a Notebook? The Jupyter

The notebook document format is free, transparent, and understandable, in keeping with its aim to facilitate open and accessible science. It is stored as a single JSON-formatted text file, making it easy to manipulate and understand using standard programming tools, without the need for Jupyter software. The notebook file format is public,[2] and Jupyter software is open-source under the BSD license.

Many authors communicate using Jupyter Notebook. GitHub hosts 1.4 million notebooks, and some people have written entire books as collections of notebooks, such as Jake Vanderplas's *Python Data Science Handbook*.[3] Because notebook documents preserve their content structure and metadata, they are easily convertible to other formats, including plain scripts in the document's language of choice. This also makes them easy to integrate into publication pipelines via formats such as LaTeX, Markdown, and reStructuredText via Jupyter's conversion tool, nbconvert.[4]

Using Notebook Documents. The Jupyter Notebook server is a web-based application for interacting with notebook

cuting code cells in the same persistent kernel session, thus building and exploring a computational idea. The user can also create and interleave markdown cells to explain an idea using formatted text and mathematics (see Figure 1).

Users typically install the notebook application on their local computer, where it functions like a desktop application and works with local notebooks and data. Since it is a web application, the notebook server can also run on a remote computer and provide the notebook application via a browser, with no installation required on the user's machine

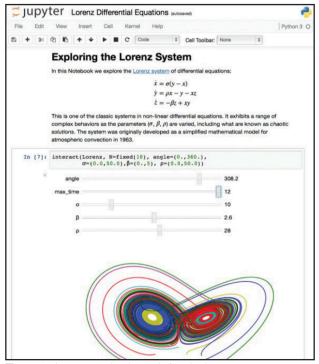


Figure 1. *Jupyter notebooks exploring the Lorenz system.*

output representations between front ends

We're not-for-profit

README.md

Project Jupyter Governance

The purpose of this repository is to formalize the governance process that the IPython/Jupyter project has used informally since its inception in 2001. This document clarifies how decisions are made and how the various elements of our community interact, including the relationship between open source collaborative development and work that may be funded by for-profit or non-profit entities.

Table of Contents

License of Governance Documents

https://github.com/jupyter/governance

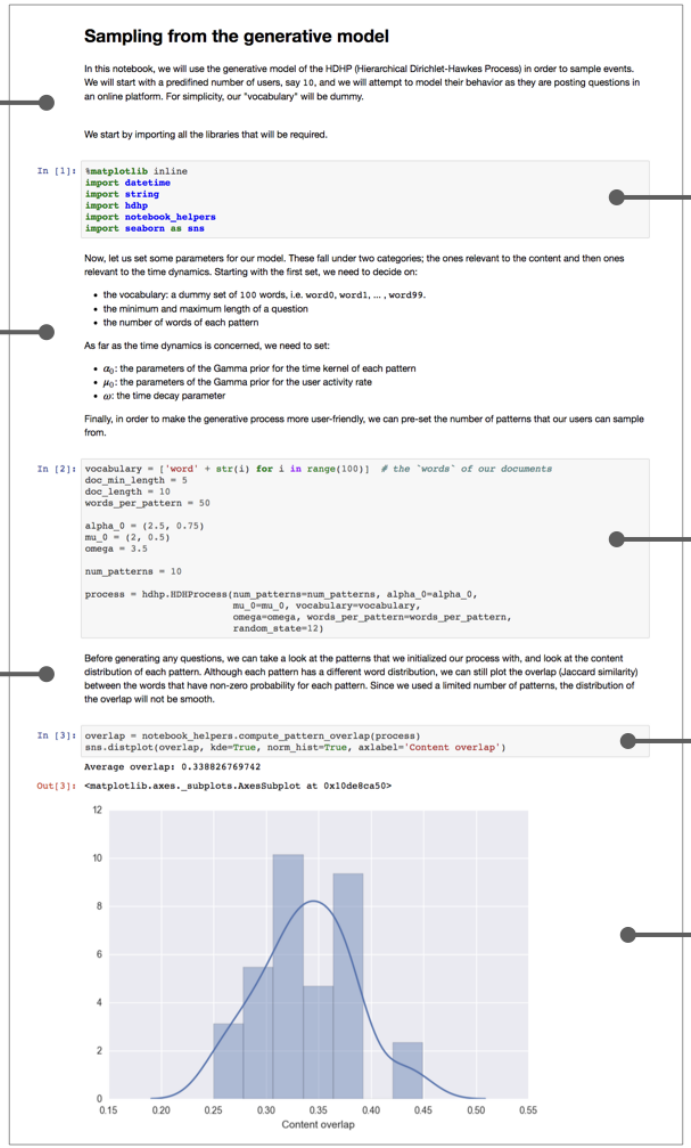
Jupyter Notebook

Narrative Text

Notebook title and introduction

Description of model parameters

Description of need to profile data



Sampling from the generative model

In this notebook, we will use the generative model of the HDHP (Hierarchical Dirichlet-Hawkes Process) in order to sample events. We will start with a predifined number of users, say `10`, and we will attempt to model their behavior as they are posting questions in an online platform. For simplicity, our "vocabulary" will be dummy.

We start by importing all the libraries that will be required.

In [1]:
```
%matplotlib inline
import datetime
import string
import hdhp
import notebook_helpers
import seaborn as sns
```

Now, let us set some parameters for our model. These fall under two categories; the ones relevant to the content and then ones relevant to the time dynamics. Starting with the first set, we need to decide on:

- the vocabulary: a dummy set of `100` words, i.e. `word0`, `word1`, ... , `word99`.
- the minimum and maximum length of a question
- the number of words of each pattern

As far as the time dynamics is concerned, we need to set:

- α_0: the parameters of the Gamma prior for the time kernel of each pattern
- μ_0: the parameters of the Gamma prior for the user activity rate
- ω: the time decay parameter

Finally, in order to make the generative process more user-friendly, we can pre-set the number of patterns that our users can sample from.

In [2]:
```
vocabulary = ['word' + str(i) for i in range(100)]  # the `words` of our documents
doc_min_length = 5
doc_length = 10
words_per_pattern = 50

alpha_0 = (2.5, 0.75)
mu_0 = (2, 0.5)
omega = 3.5

num_patterns = 10

process = hdhp.HDHProcess(num_patterns=num_patterns, alpha_0=alpha_0,
                          mu_0=mu_0, vocabulary=vocabulary,
                          omega=omega, words_per_pattern=words_per_pattern,
                          random_state=12)
```

Before generating any questions, we can take a look at the patterns that we initialized our process with, and look at the content distribution of each pattern. Although each pattern has a different word distribution, we can still plot the overlap (Jaccard similarity) between the words that have non-zero probability for each pattern. Since we used a limited number of patterns, the distribution of the overlap will not be smooth.

In [3]:
```
overlap = notebook_helpers.compute_pattern_overlap(process)
sns.distplot(overlap, kde=True, norm_hist=True, axlabel='Content overlap')
```

```
Average overlap: 0.338826769742
```

Out[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x10de8ca50>
```


Code and Visualizations

Importing external packages

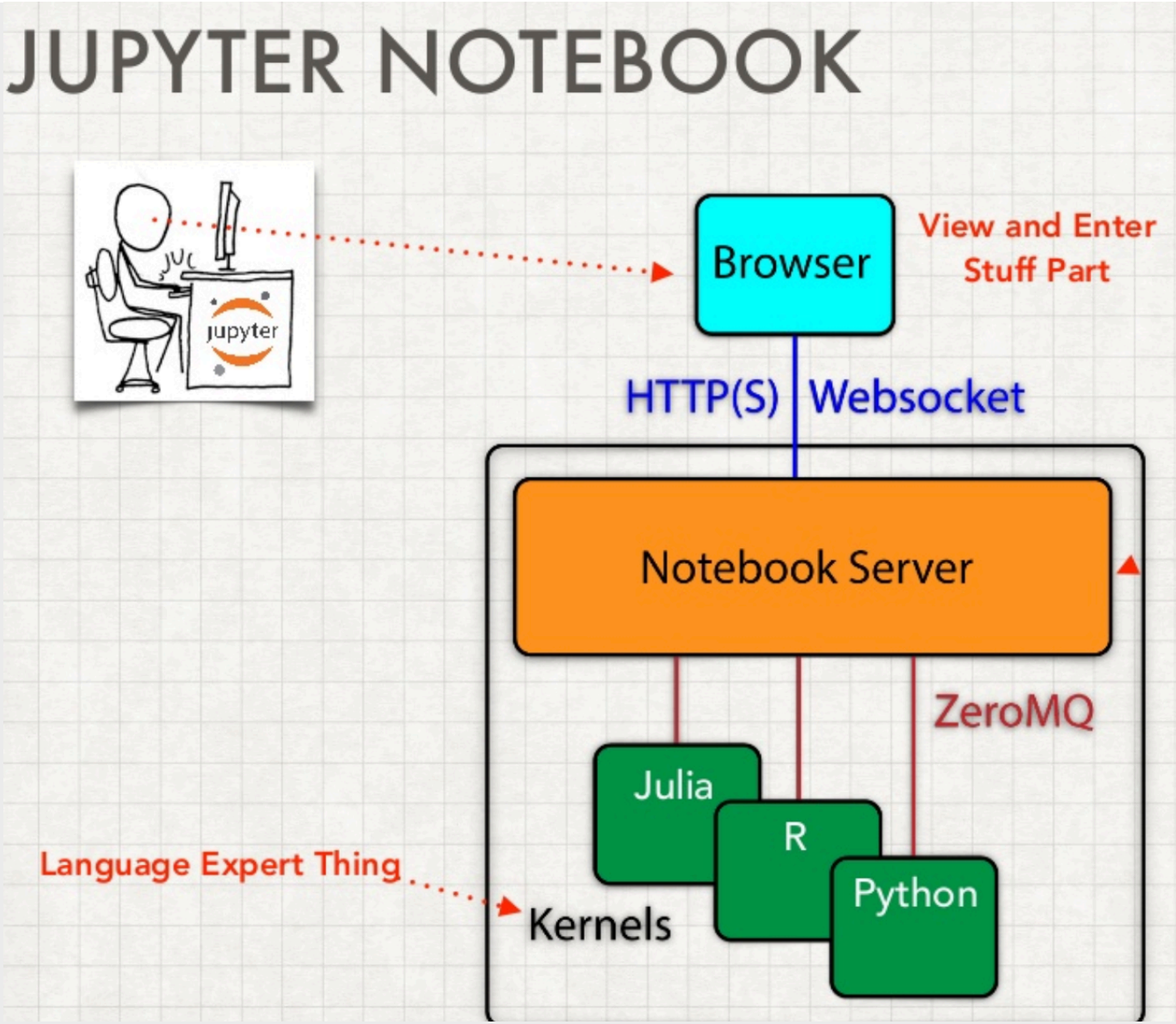
Implementation of parameters

Profile plotting code

Inline plot

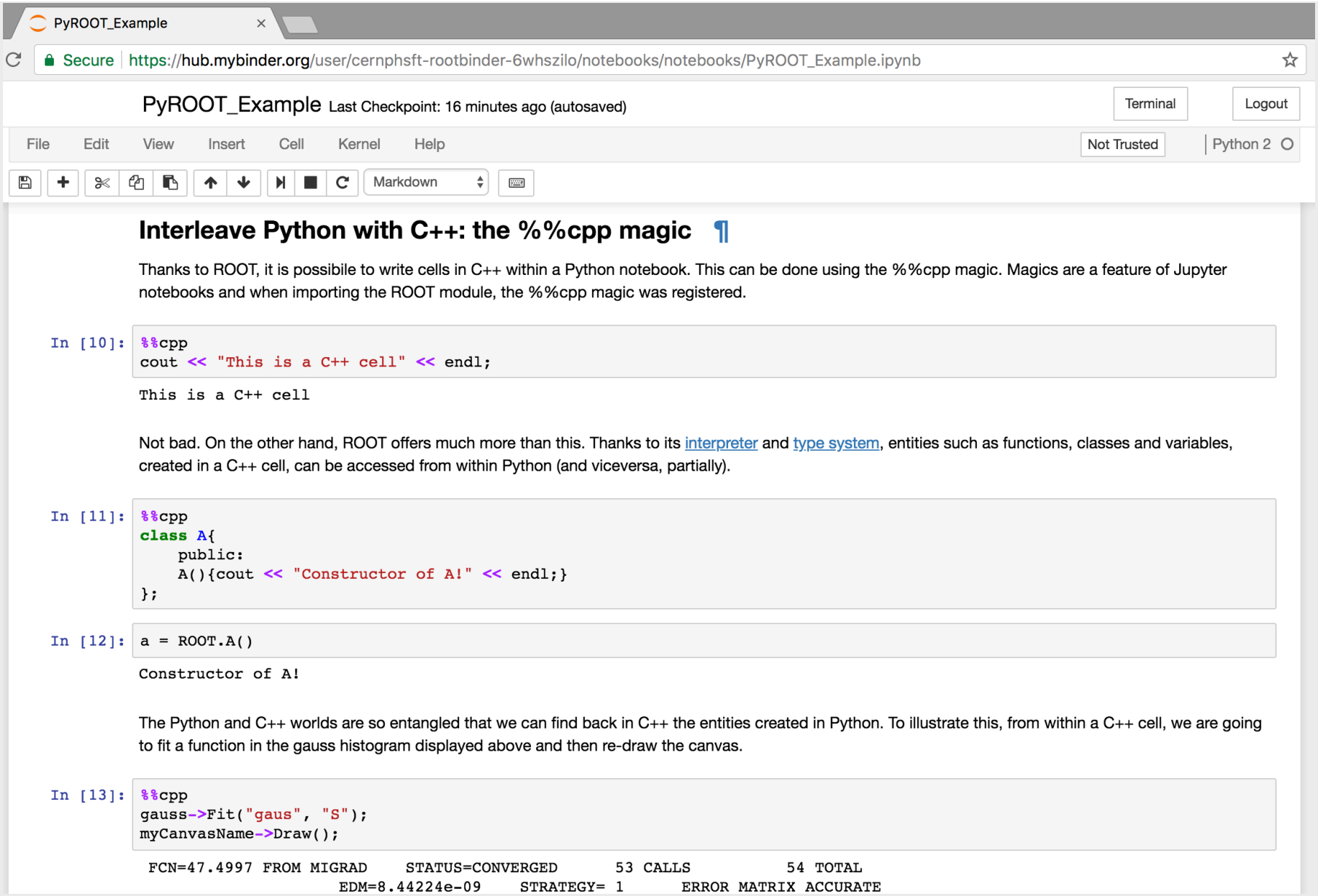
Rule et al., 2018

JUPYTER NOTEBOOK



Carol Willing

More than Python


PyROOT_Example
Secure https://hub.mybinder.org/user/cernphsft-rootbinder-6whszilo/notebooks/notebooks/PyROOT_Example.ipynb
PyROOT_Example Last Checkpoint: 16 minutes ago (autosaved)
Terminal
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 2
Markdown
Interleave Python with C++: the %%cpp magic ¶
Thanks to ROOT, it is possibile to write cells in C++ within a Python notebook. This can be done using the %%cpp magic. Magics are a feature of Jupyter notebooks and when importing the ROOT module, the %%cpp magic was registered.
In [10]:
%%cpp
cout << "This is a C++ cell" << endl;
This is a C++ cell
Not bad. On the other hand, ROOT offers much more than this. Thanks to its interpreter and type system, entities such as functions, classes and variables, created in a C++ cell, can be accessed from within Python (and viceversa, partially).
In [11]:
%%cpp
class A{
public:
A(){cout << "Constructor of A!" << endl;}
};
In [12]:
a = ROOT.A()
Constructor of A!
The Python and C++ worlds are so entangled that we can find back in C++ the entities created in Python. To illustrate this, from within a C++ cell, we are going to fit a function in the gauss histogram displayed above and then re-draw the canvas.
In [13]:
%%cpp
gauss->Fit("gaus", "S");
myCanvasName->Draw();
FCN=47.4997 FROM MIGRAD STATUS=CONVERGED 53 CALLS 54 TOTAL
EDM=8.44224e-09 STRATEGY= 1 ERROR MATRIX ACCURATE

HiggsFit

https://hub.mybinder.org/user/cernphsft-rootbinder-6whszilo/notebooks/notebooks/HiggsFit.ipynb

HiggsFit (autosaved)

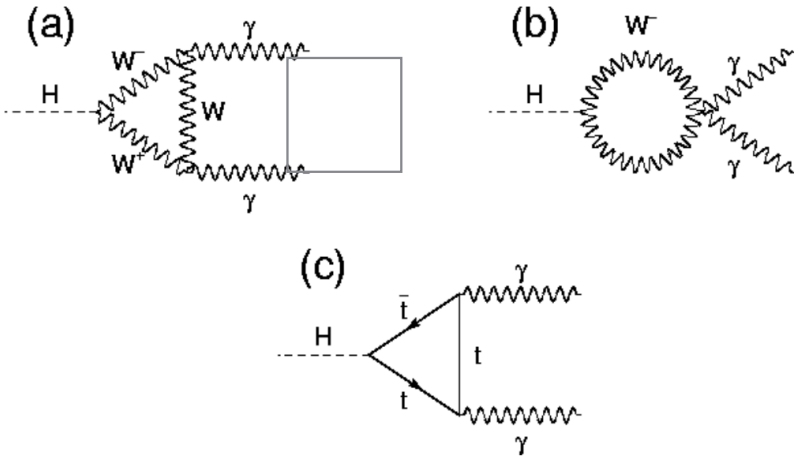
Terminal Logout

File Edit View Insert Cell Kernel Help Not Trusted ROOT C++

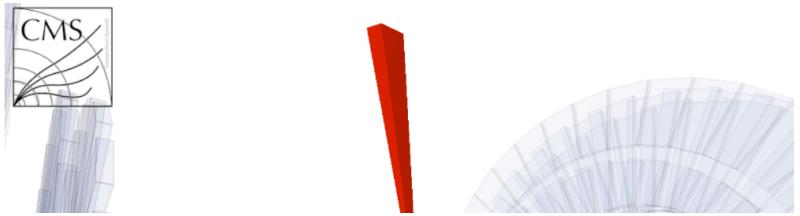
Code

Higgs decay to two photons

The Standard Model predicted the decay of the Higgs bosons into photons. The process is depicted by the diagrams below:



At the Large Hadron Collider, this process has been measured. This figure shows how an Higgs boson decay looks in the CMS detector:



HiggsFit

https://hub.mybinder.org/user/cernphsft-rootbinder-6whszilo/notebooks/notebooks/HiggsFit.ipynb

HiggsFit (autosaved)

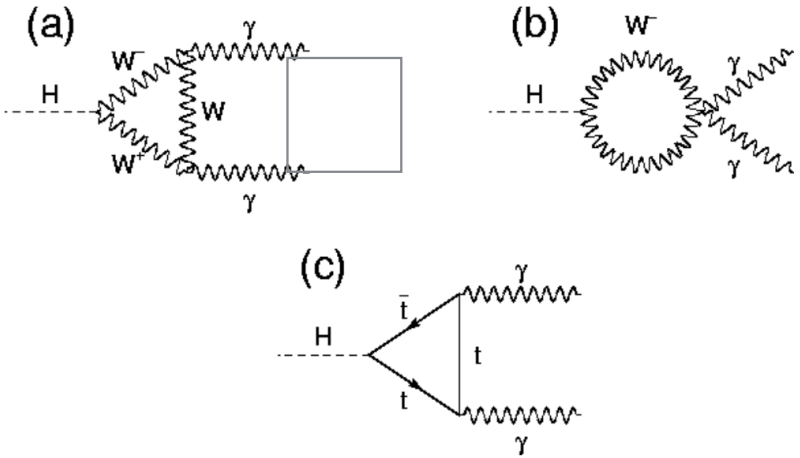
Terminal Logout

File Edit View Insert Cell Kernel Help Not Trusted ROOT C++

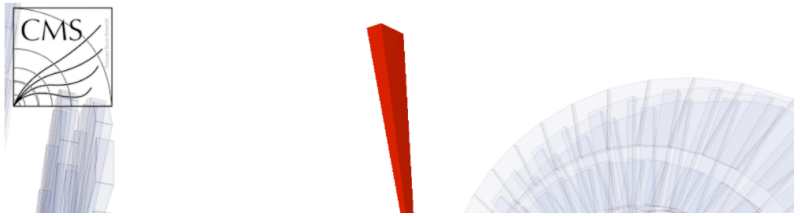
Code

Higgs decay to two photons

The Standard Model predicted the decay of the Higgs bosons into photons. The process is depicted by the diagrams below:



At the Large Hadron Collider, this process has been measured. This figure shows how an Higgs boson decay looks in the CMS detector:



In [ ]:
```
import (
    "fmt"
    "time"
)
```

In [ ]:
```
// sum calculates the sum of two integers
func sum(x, y int) int {
    return x + y
}
```

In [ ]:
```
a, b := 3, 4
```

In [ ]:
```
fmt.sp("sum(%d, %d) = %d", a, b, sum(a, b))
```

In [ ]:
```
start := time.Now()
defer func() {
    end := time.Now()
    fmt.Println("Interrupted an infinite loop after",
                end.Sub(start))
}()
for {}
```

Yu Watanabe

Jupyter kernels

Kernel Zero is IPython, which you can get through ipykernel, and is still a dependency of jupyter. The IPython kernel can be thought of as a reference implementation, as CPython is for Python.

Here is a list of available kernels. If you are writing your own kernel, feel free to add it to the table!

Name	Jupyter/IPython Version	Language(s) Version	3rd party dependencies	Exar Notel
Coarray-Fortran	Jupyter 4.0	Fortran 2008/2015	GFortran >= 7.1, OpenCoarrays, MPICH >= 3.2	Demo Binde demo
sparkmagic	Jupyter >=4.0	Pyspark (Python 2 & 3), Spark (Scala), SparkR (R)	Livy	Noteb Docke Image

https://github.com/jupyter/jupyter/wiki/Jupyter-kernels

Creating new Jupyter kernels

Making kernels for Jupyter in the documentation.

Simple example kernel

IHaskell creator blog post

Testing kernels against message specification (work in progress)

Tool to test a kernel against specification (work in progress)

https://github.com/jupyter/jupyter/wiki/Jupyter-kernels

More than Notebooks

jupyterhub

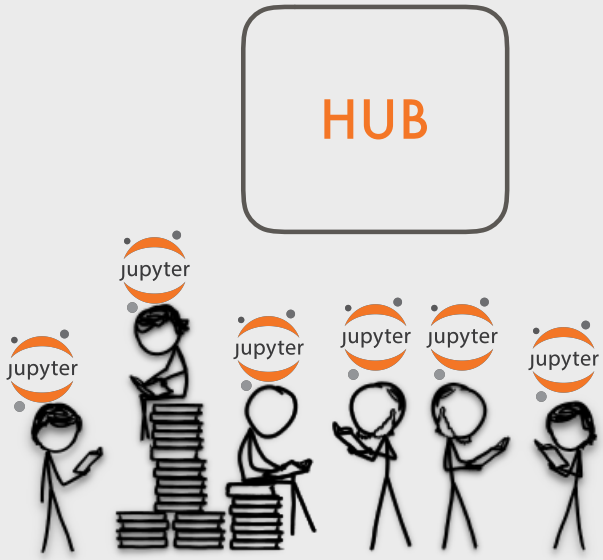


photo: xkcd, Jupyter, C. Willing

- JupyterHub provides a single user Jupyter Notebook server for each person in a group
- Similar to SWAN

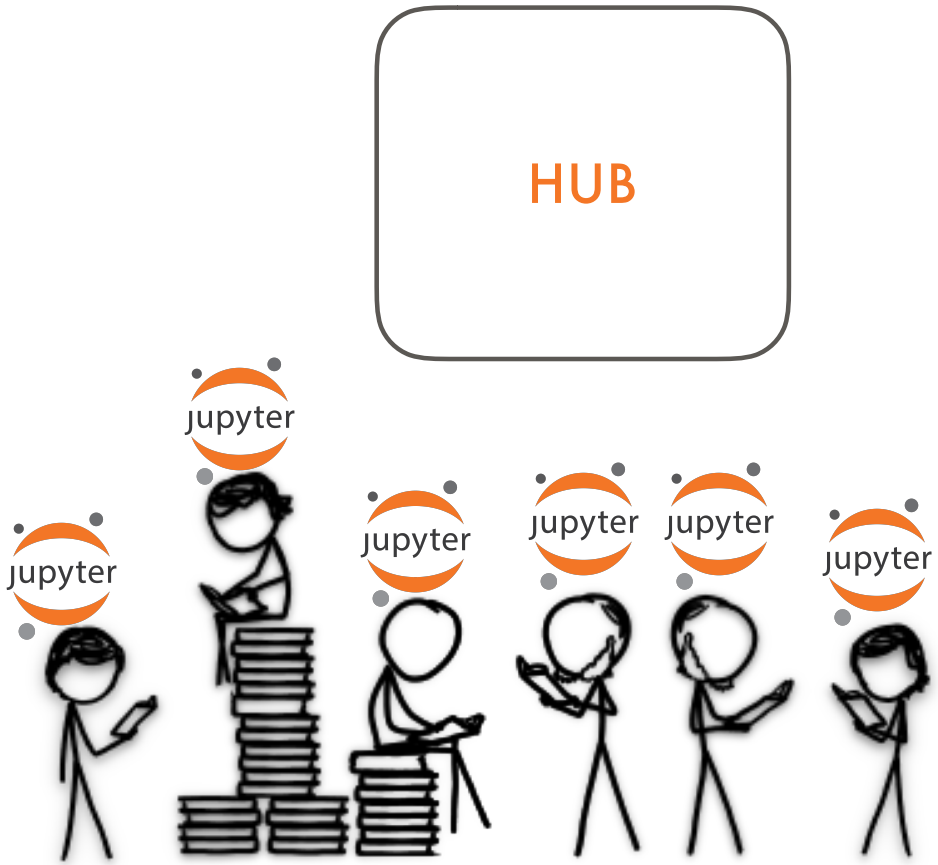
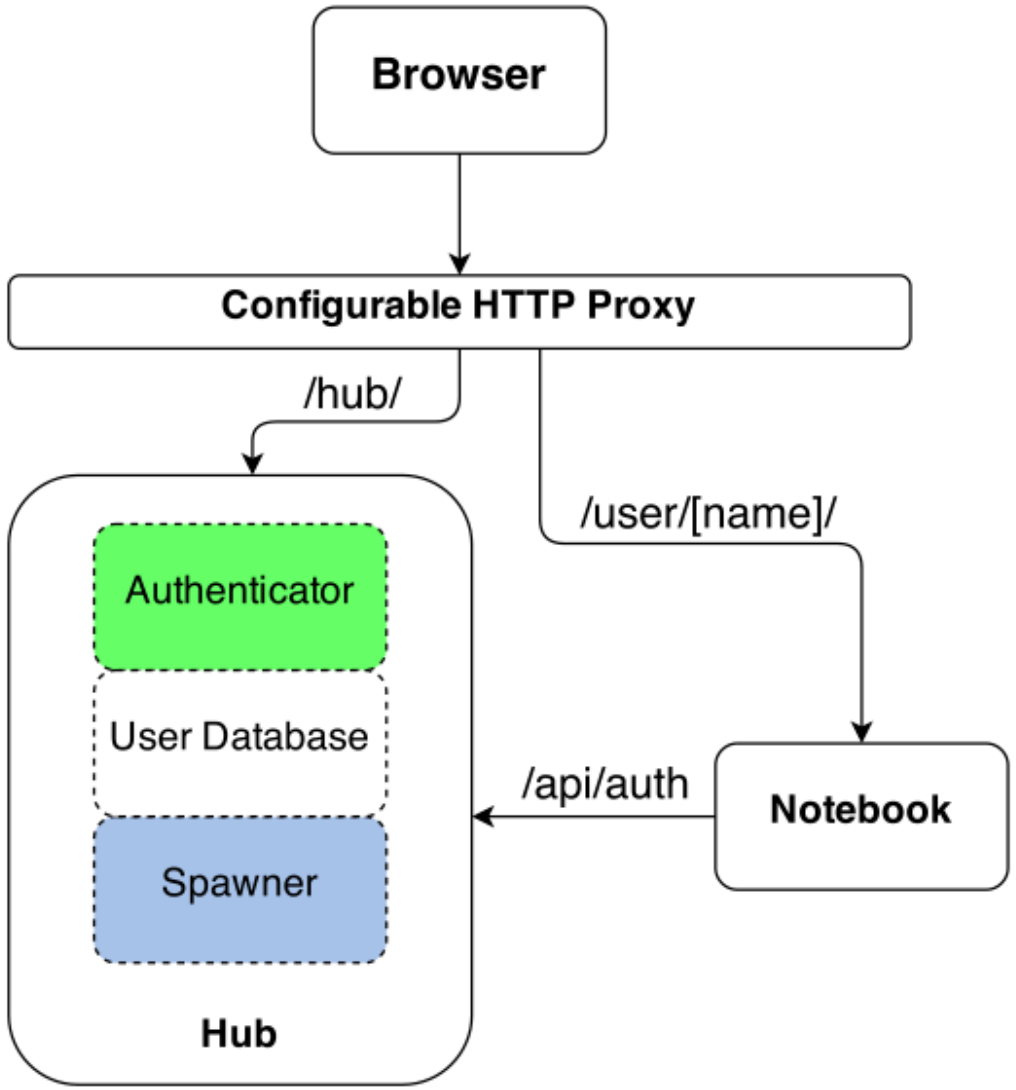


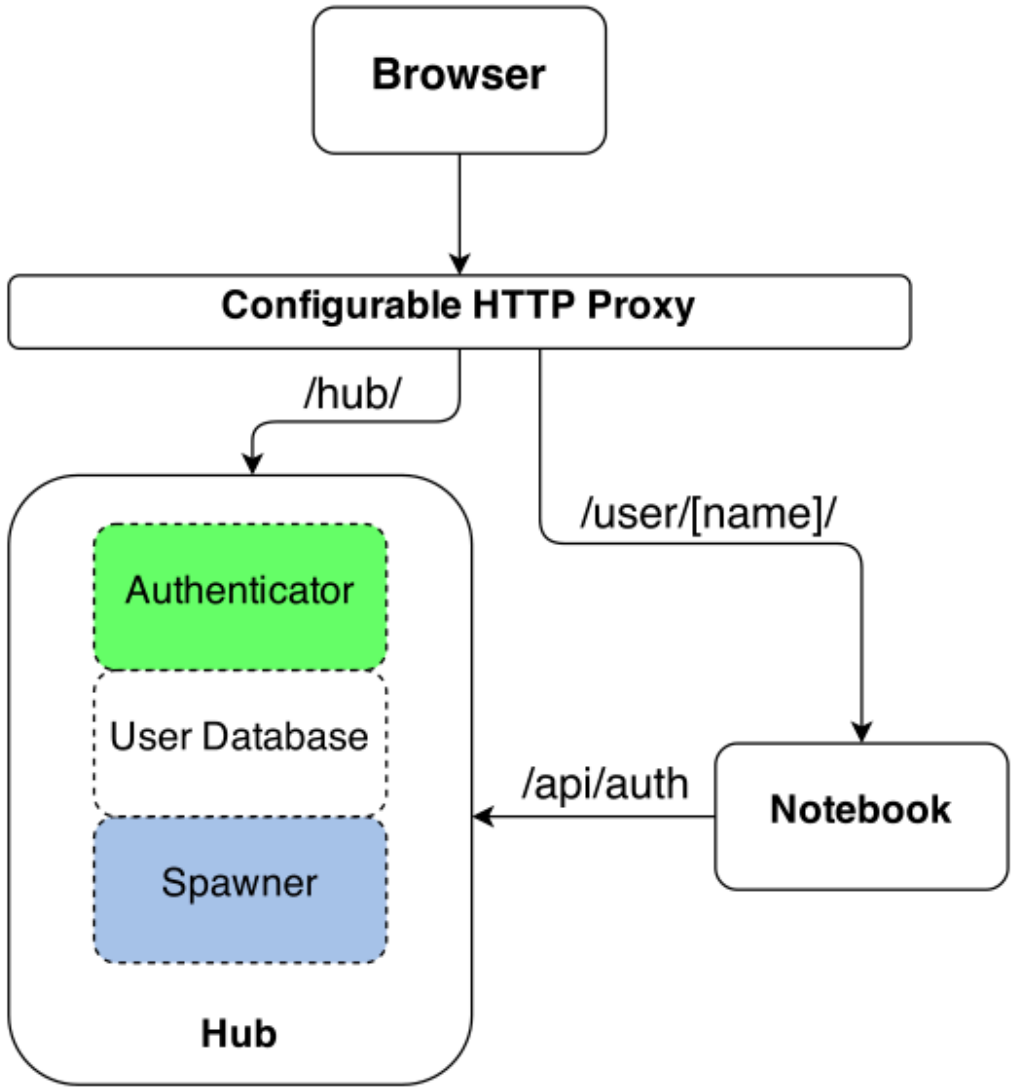
photo: xkcd, Jupyter, C. Willing

- **Hub**: manages user accounts, authentication, and coordinates Single User Notebook Servers using a Spawner
- **Proxy**: routes HTTP requests to the Hub and Single User Notebook Servers.
- **Spawner**: starts a **single-user notebook servers** when a user logs in



https://jupyterhub.readthedocs.io

- Hub launches a proxy
- Proxy forwards all requests to Hub by default
- Hub handles login, and spawns single-user servers on demand
- Hub configures proxy to forward url prefixes to the single-user notebook servers



https://jupyterhub.readthedocs.io

Kubernetes as Spawner

Next »

Zero to JupyterHub with Kubernetes

A tutorial to help install and manage JupyterHub with Kubernetes

Quick search

Zero to JupyterHub

JupyterHub is a tool that allows you to quickly utilize cloud computing infrastructure to manage a hub that enables users to interact remotely with a computing environment that you specify. JupyterHub offers a useful way to standardize the computing environment of a group of people (e.g., for a class of students or an analytics team), as well as allowing people to access the hub remotely.

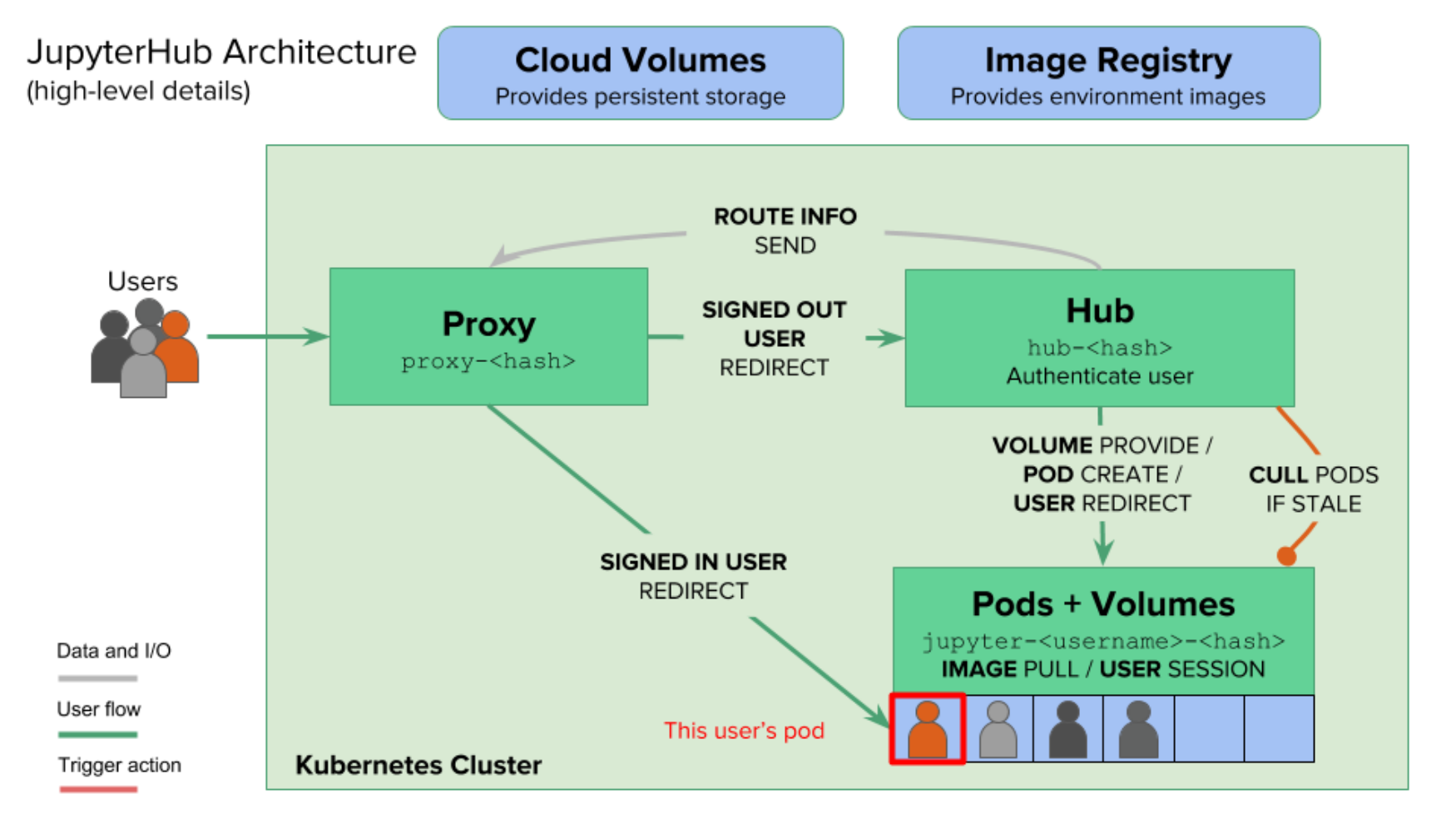
This growing collection of information will help you set up your own JupyterHub instance. It is in an early stage, so the information and tools may change quickly. If you see anything that is incorrect or have any questions, feel free to reach out at the issues page.

Creating your JupyterHub

This tutorial starts from "step zero" and walks through how to install and configure a complete JupyterHub deployment in the cloud. Using Kubernetes and the JupyterHub Helm chart provides sensible defaults for an initial deployment.

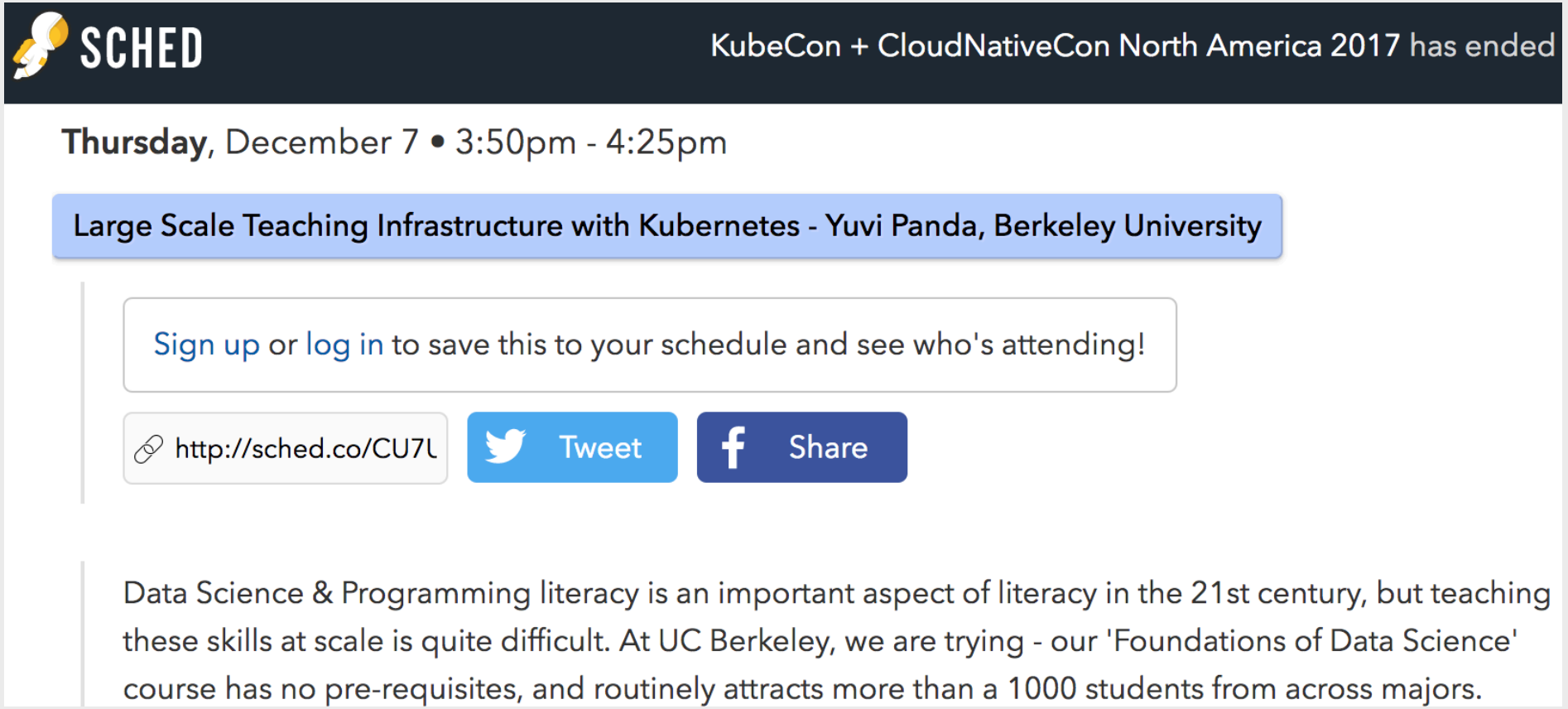
https://zero-to-jupyterhub.readthedocs.io

Kubernetes as Spawner



https://zero-to-jupyterhub.readthedocs.io

Scaling JupyterHub



https://kccncna17.sched.com/event/CU7U/large-scale-teaching-infrastructure-with-kubernetes-yuvi-panda-berkeley-university

Scaling JupyterHub

PAWS

Phabricator project: **#paws**

PAWS: A Web Shell (PAWS) is a Jupyter notebooks deployment that has been customized to make interacting with Wikimedia wikis easier. It allows users to create and share documents that contain live code, visualizations such as graphs, rich text, etc. The user created notebooks are a powerful tool that enables data analysis and scientific research, and also transforms the way in which programmers write code - by enabling an exploratory environment with a quick feedback loop, and a low barrier for entry through it's easy to use graphical interface.

Sign in with your wiki account and tada!

Contents [hide]

https://wikitech.wikimedia.org/wiki/PAWS

Scaling JupyterHub

Pangeo: JupyterHub, Dask, and XArray on the Cloud

This work is supported by Anaconda Inc, the NSF EarthCube program, and UC Berkeley BIDS

A few weeks ago a few of us stood up pangeo.pydata.org, an experimental deployment of JupyterHub, Dask, and XArray on Google Container Engine (GKE) to support atmospheric and oceanographic data analysis on large datasets. This follows on recent work to deploy Dask and XArray for the same workloads on super computers. This system is a proof of concept that has taught us a great deal about how to move forward. This blogpost briefly describes the problem, the system, then describes the collaboration, and finally discusses a number of challenges that we'll be working on in coming months.

The Problem

Atmospheric and oceanographic sciences collect (with satellites) and generate (with simulations) large datasets that they would like to analyze with distributed systems. Libraries like Dask and XArray already solve this problem computationally if scientists have their own clusters, but we seek to expand access by deploying on cloud-based systems. We build a system to which people can log in, get Jupyter Notebooks, and launch Dask clusters without much hassle. We hope that this increases access, and connects more scientists with more cloud-based datasets.

http://matthewrocklin.com/blog/work/2018/01/22/pangeo-2

Kubeflow

The Kubeflow project is dedicated to making **deployments** of machine learning (ML) workflows on Kubernetes simple, portable and scalable. Our goal is **not** to recreate other services, but to provide a straightforward way to deploy best-of-breed open-source systems **for ML** to diverse infrastructures. Anywhere you are running Kubernetes, you should be able to run Kubeflow.

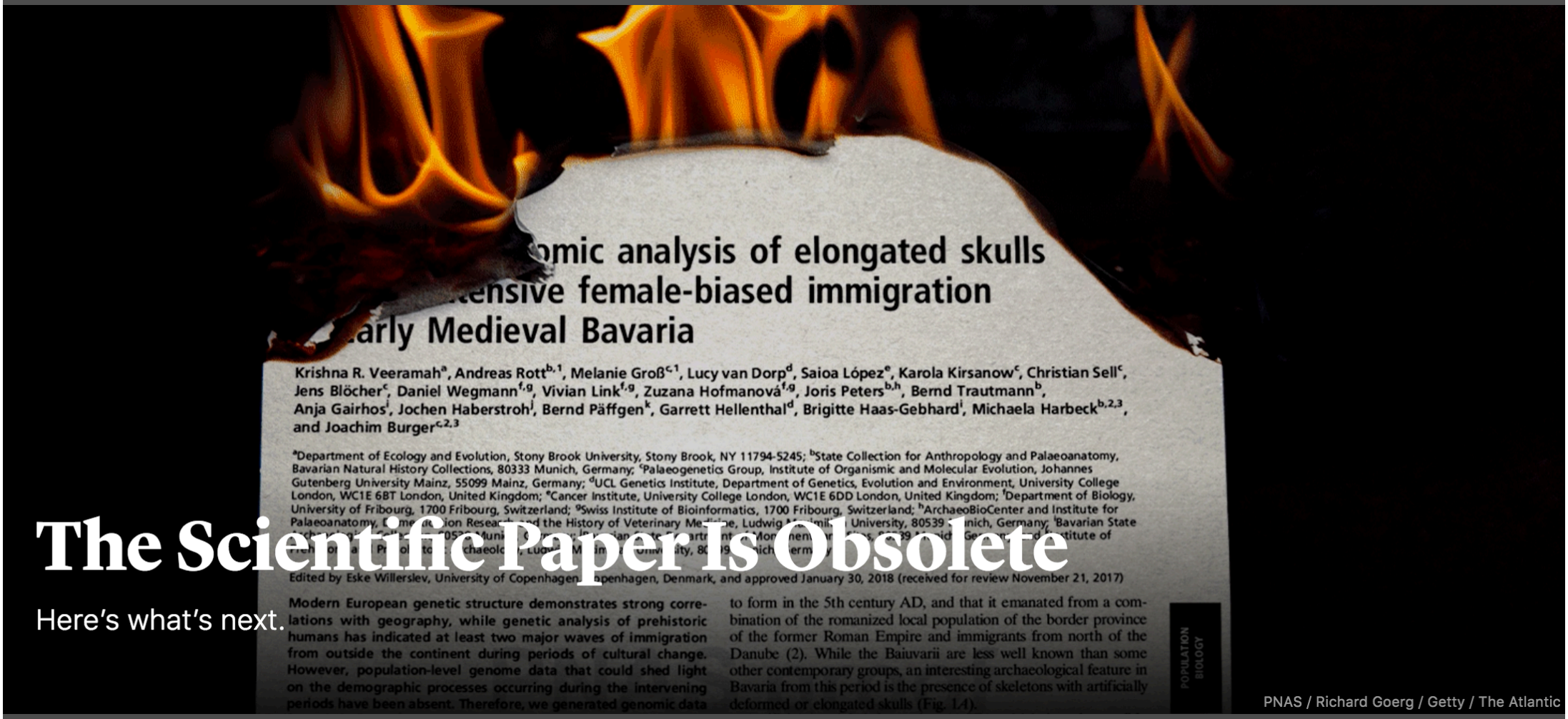
This repository contains the manifests for creating:

- A **JupyterHub** to create and manage interactive Jupyter notebooks. Project Jupyter is a non-profit, open-source project to support interactive data science and scientific computing across all programming languages.
- A **TensorFlow Training Controller** that can be configured to use either CPUs or GPUs and dynamically adjusted to the size of a cluster with a single setting
- A **TensorFlow Serving** container to export trained TensorFlow models to Kubernetes

This document details the steps needed to run the Kubeflow project in any environment in which Kubernetes runs.

https://github.com/kubeflow/kubeflow

Reproducibility



The Scientific Paper Is Obsolete

Here's what's next.

PNAS / Richard Goerg / Getty / The Atlantic

Atlantic Monthly, April 5 2018

The more sophisticated science becomes, the harder it is to communicate results. Papers today are longer than ever and full of jargon and symbols. They depend on chains of computer programs that generate data, and clean up data, and plot data, and run statistical models on data. These programs tend to be both so sloppily written and so central to the results that it's contributed to a replication crisis, or put another way, a failure of the paper to perform its most basic task: to report what you've actually discovered, clearly enough that someone else can discover it for themselves.

- James Somers

Reproducibility:

Producing similar results
with the same data

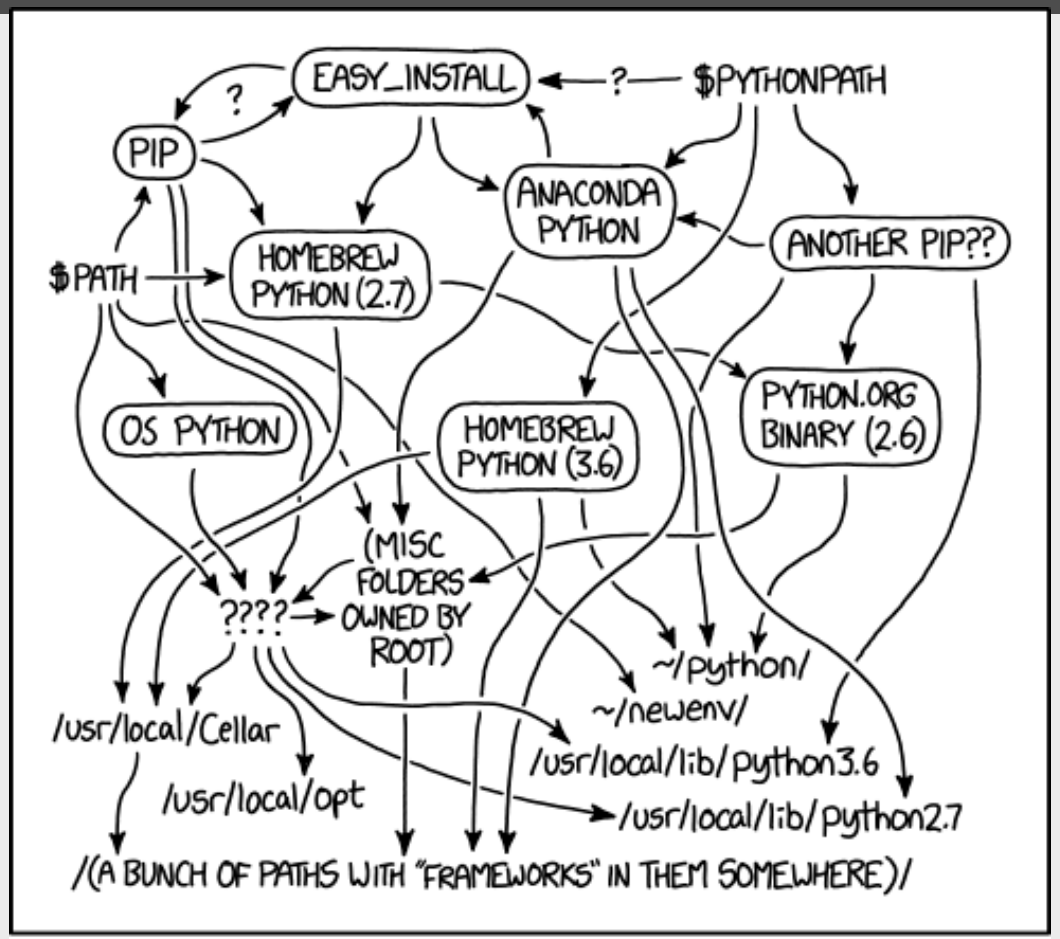
Reproducibility:

A software problem

Technical Solutions

- GitHub
- Open Science Framework
- CodaLab
- RunMyCode
- Research Journals

This week in xkcd



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

https://xkcd.com/1987/

Reproducible scientific software pipelines

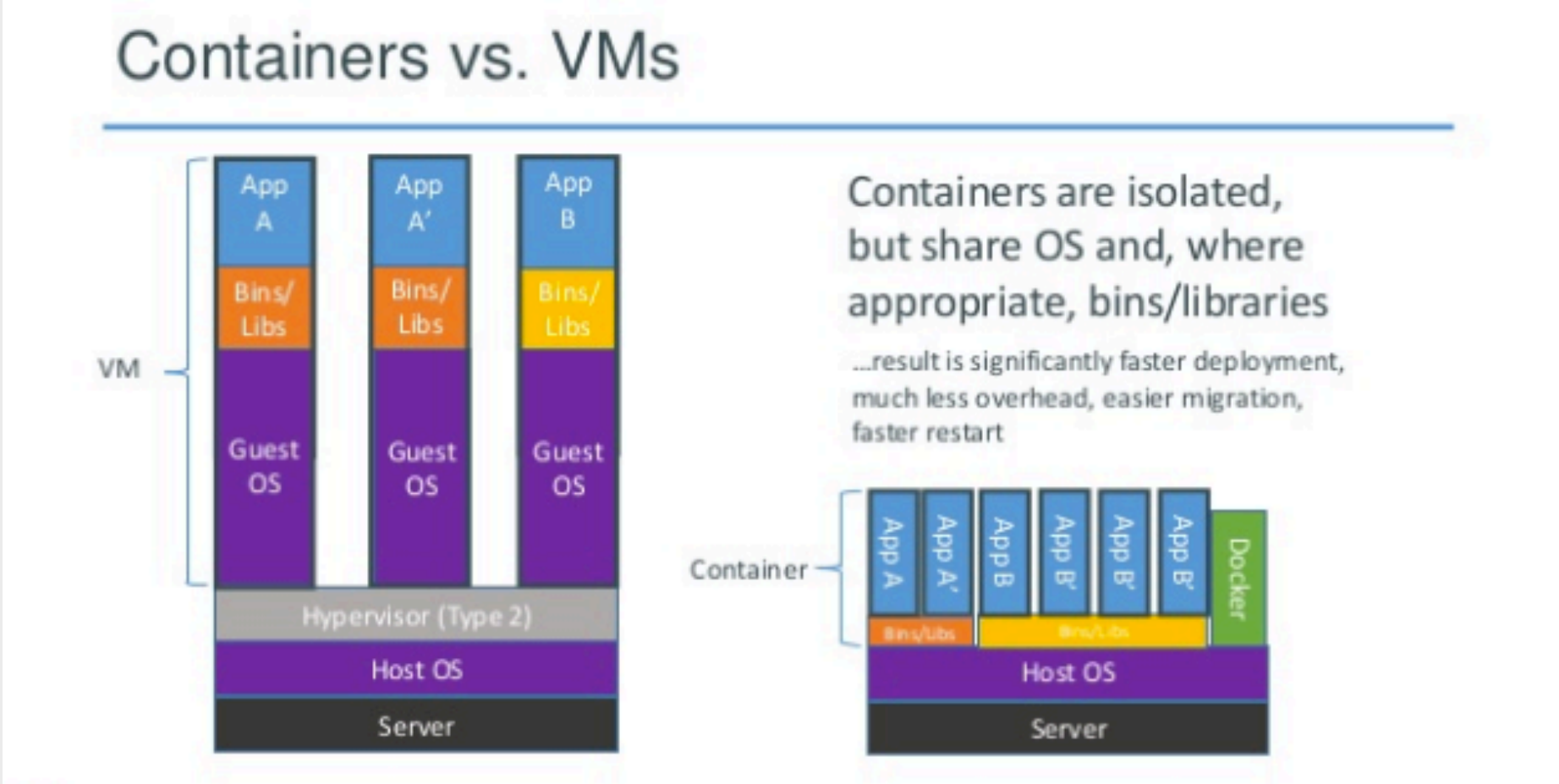
- Repository would contain:
  - Data
  - Dependencies
  - Hyperparameters
  - Scripts to run the jobs on similar hardware
  - Analysis code

Reproducible scientific software pipelines

- Repository would contain:
  - Data
  - Dependencies
  - Hyperparameters
  - Scripts to run the jobs on similar hardware
  - Analysis code

No mention of OS or lower-level software

Docker for reproducible software



source: Docker

Dockerfiles from GitHub Repos

jupyter-repo2docker

build passing docs passing

jupyter-repo2docker takes as input a repository source, such as a GitHub repo. It then builds, runs, and/or pushes Docker images built from that source.

See the repo2docker documentation for more information.

Pre-requisites

1. Docker to build & run the repositories. The community edition is recommended.
2. Python 3.4+.

Supported on Linux and macOS. See documentation note about Windows support.

https://github.com/jupyerhub/repo2docker

Using repo2docker

Note that Docker needs to be running on your machine for this to work.

Example:

```
jupyter-repo2docker https://github.com/norvig/pytudes
```

After building (it might take a while!), it should output in your terminal something like:

```
    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://0.0.0.0:36511/?token=f94f8fabb92e22f5bfab116c382b4707fc2cade56ad1ace0
```

https://github.com/jupyerhub/repo2docker

repo2docker as reproducibility unit-test

- repo2docker looks for configuration files to determine how to build the docker image
- Typically describe dependencies and other instructions to create the environment
- configs in either root or folder called `binder`

https://github.com/jupyerhub/repo2docker

Supported config files

- **Dockerfile**: full environment setup
- **environment.yml**: conda
- **requirements.txt**: pip
- **REQUIRE**: Julia
- **apt.txt**: Debian packages
- **postBuild**: custom install script
- **runtime.txt**: Python runtime

https://github.com/jupyerhub/repo2docker

Example repo

Branch: master ▾ **GAN_tutorial / environment.yml**

 mickypaganini Update environment.yml

1 contributor

9 lines (8 sloc) | 139 Bytes

```
1  name: binder
2  dependencies:
3    - pytorch
4    - torchvision
5    - matplotlib==2.0.2
6    - numpy==1.14.1
7    - ipython==6.2.1
8    - scikit-learn==0.19.1
```

Branch: master ▾ **GAN_tutorial / runtime.txt**

 **mickypaganini** Create runtime.txt

1 contributor

2 lines (1 sloc) | 11 Bytes

```
1  python-2.7
```

https://github.com/mickypaganini/GAN_tutorial