An Overview of Jupyter Notebooks at SDCC (with examples)

Ofer Rind

SDCC Technology Meeting

September 27, 2018

70 YEARS OF DISCOVERY

A CENTURY OF SERVICE

U.S. DEPARTMENT OF ENERGY

BROOKHAVEN NATIONAL LABORATORY

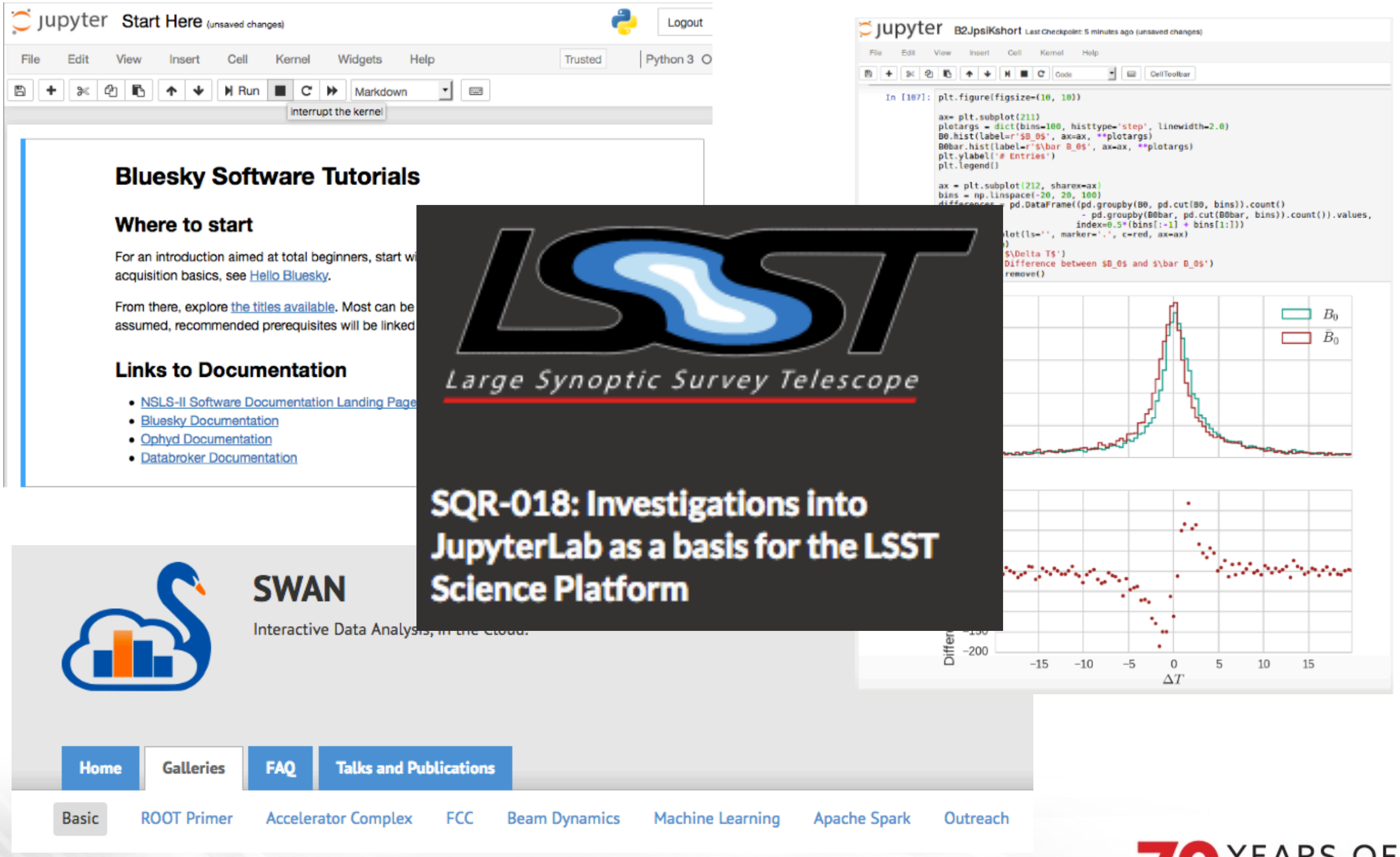
Data Analysis As A Service

- Jupyter Notebooks (iPython)
    - Provide a flexible, standardized, platform independent interface through a web browser
    - Well-suited for *interactive* analysis
    - No local software to install
    - Many language extensions (kernels and tools available
    - Easy to share, reproduce, document results and create tutorials

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

- From the facility point of view: Can we layer this service atop existing resources, without building a new dedicated infrastructure, such as a specialized cluster? (cf. CERN Swan)

Growing in Popularity Across Scientific Community



Bluesky Software Tutorials

Where to start

For an introduction aimed at total beginners, start wi
acquisition basics, see Hello Bluesky.

From there, explore the titles available. Most can be
assumed, recommended prerequisites will be linked

Links to Documentation

- NSLS-II Software Documentation Landing Page
- Bluesky Documentation
- Ophyd Documentation
- Databroker Documentation

jupyter
B2JpsiKshort Last Checkpoint: 5 minutes ago (unsaved changes)
File
Edit
View
Insert
Cell
Kernel
Help
Code
CellToolbar

```
In [107]: plt.figure(figsize=(10, 10))

          ax= plt.subplot(211)
          plotargs = dict(bins=100, histtype='step', linewidth=2.0)
          B0.hist(label=r'$B_0$', ax=ax, **plotargs)
          B0bar.hist(label=r'$\bar B_0$', ax=ax, **plotargs)
          plt.ylabel('# Entries')
          plt.legend()

          ax = plt.subplot(212, sharex=ax)
          bins = np.linspace(-20, 20, 100)
          differences = pd.DataFrame((pd.groupby(B0, pd.cut(B0, bins)).count()
                                      - pd.groupby(B0bar, pd.cut(B0bar, bins)).count()).values,
                                     index=0.5*(bins[:-1] + bins[1:]))
                          lot(ls='', marker='.', c=red, ax=ax)
                          )
                          $\Delta T$')
                          Difference between $B_0$ and $\bar B_0$')
                          remove()
```

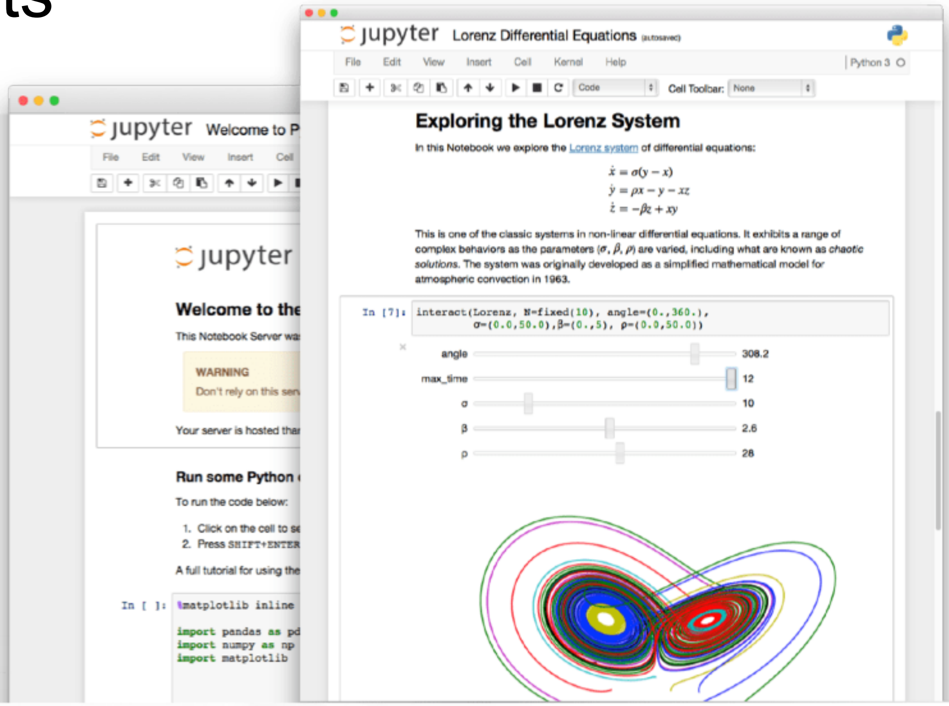
SQR-018: Investigations into JupyterLab as a basis for the LSST Science Platform

SWAN

Interactive Data Analysis in the Cloud.

Some terminology

- **Jupyter notebook**: web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results
- **Jupyterlab**: next-generation web-based user interface


jupyter Welcome to P
File Edit View Insert Cell
jupyter
jupyter Lorenz Differential Equations (autosaved)
File Edit View Insert Cell Kernel Help
Python 3 ○
Code
Cell Toolbar: None

Welcome to the

This Notebook Server wa

WARNING

Don't rely on this serv

Your server is hosted thar

Run some Python

To run the code below:

1. Click on the cell to se
2. Press `SHIFT+ENTER`

A full tutorial for using the

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

Exploring the Lorenz System

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

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

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

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

angle 308.2

max_time 12

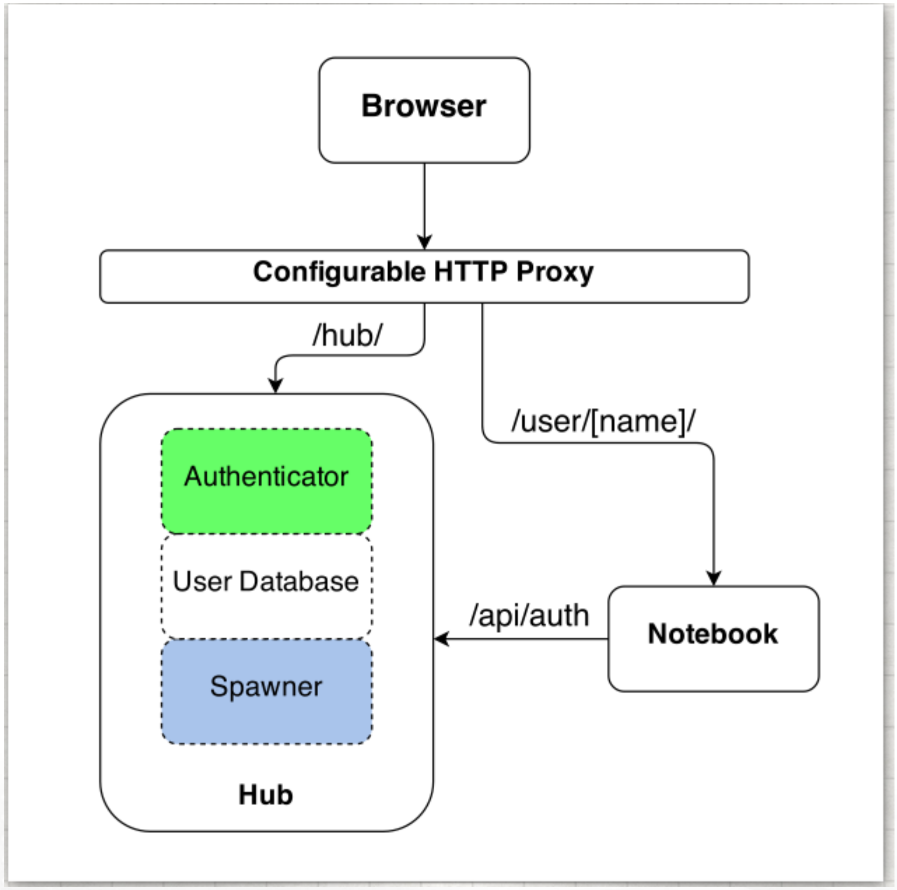
σ 10

β 2.6

ρ 28

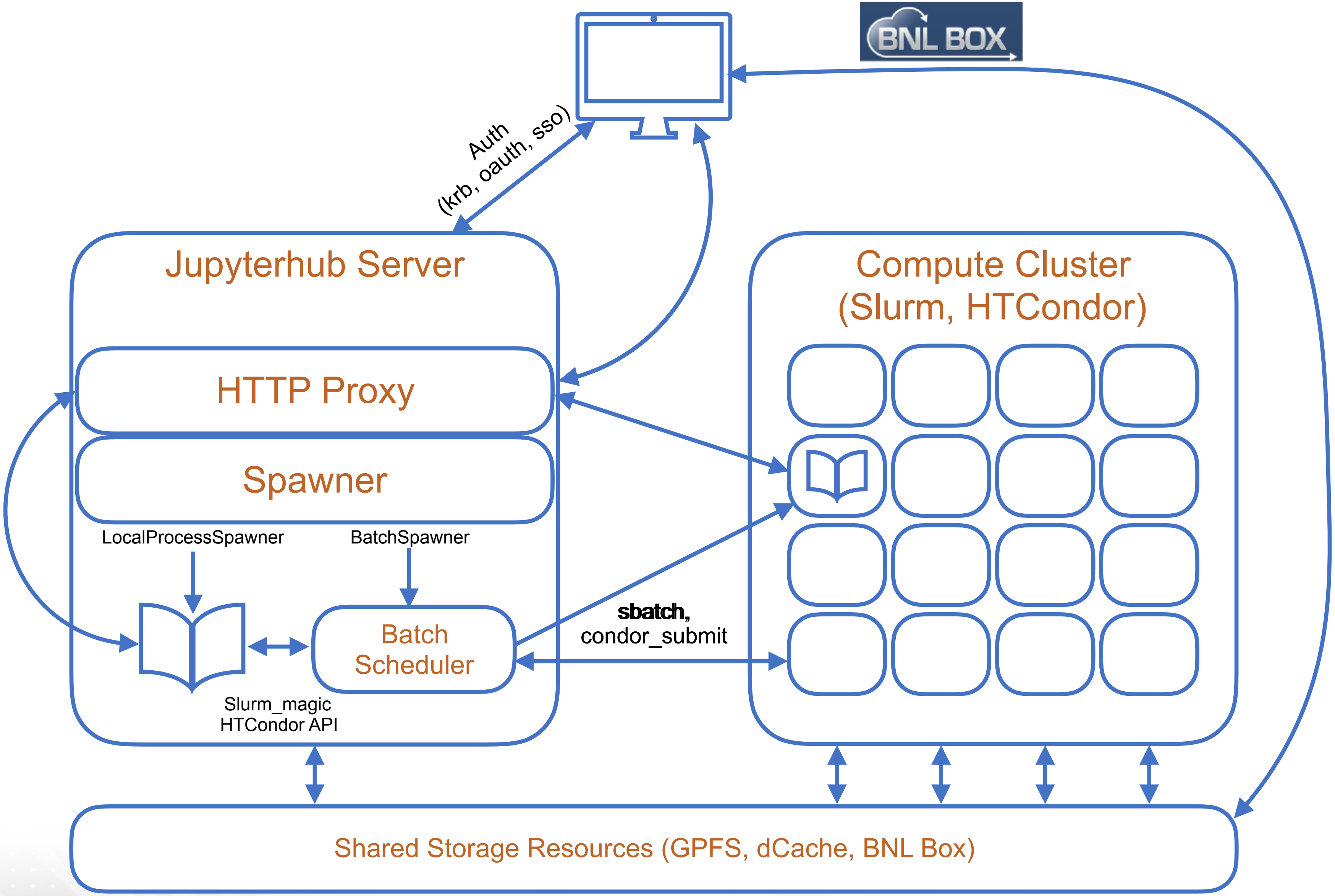
Some terminology

- **Jupyterhub**: multi-user hub, spawns, manages, and proxies multiple instances of the single-user Jupyter notebook server



Current Setup at BNL

- Three Jupyterhub servers deployed on RHEV
  - Jupyter01: Primary HTC development environment with access to HTCondor queue
  - Jupyter02: Primary HPC development environment with access to IC via Slurm
  - Jupyter03: Currently testing OAuth/Keycloak implementation
- Access via ssh tunnel through firewall to Jupyterhub https proxy
- Kerberos auth to Jupyterhub server - use your SDCC account
  - Transparent setup leverages PAM stack
- Implemented a couple of experiment-specific custom kernels


BNL BOX
Auth
(krb, oauth, sso)
Jupyterhub Server
HTTP Proxy
Spawner
LocalProcessSpawner
BatchSpawner
Batch
Scheduler
Slurm_magic
HTCondor API
sbatch,
condor_submit
Compute Cluster
(Slurm, HTCondor)
Shared Storage Resources (GPFS, dCache, BNL Box)

Examples

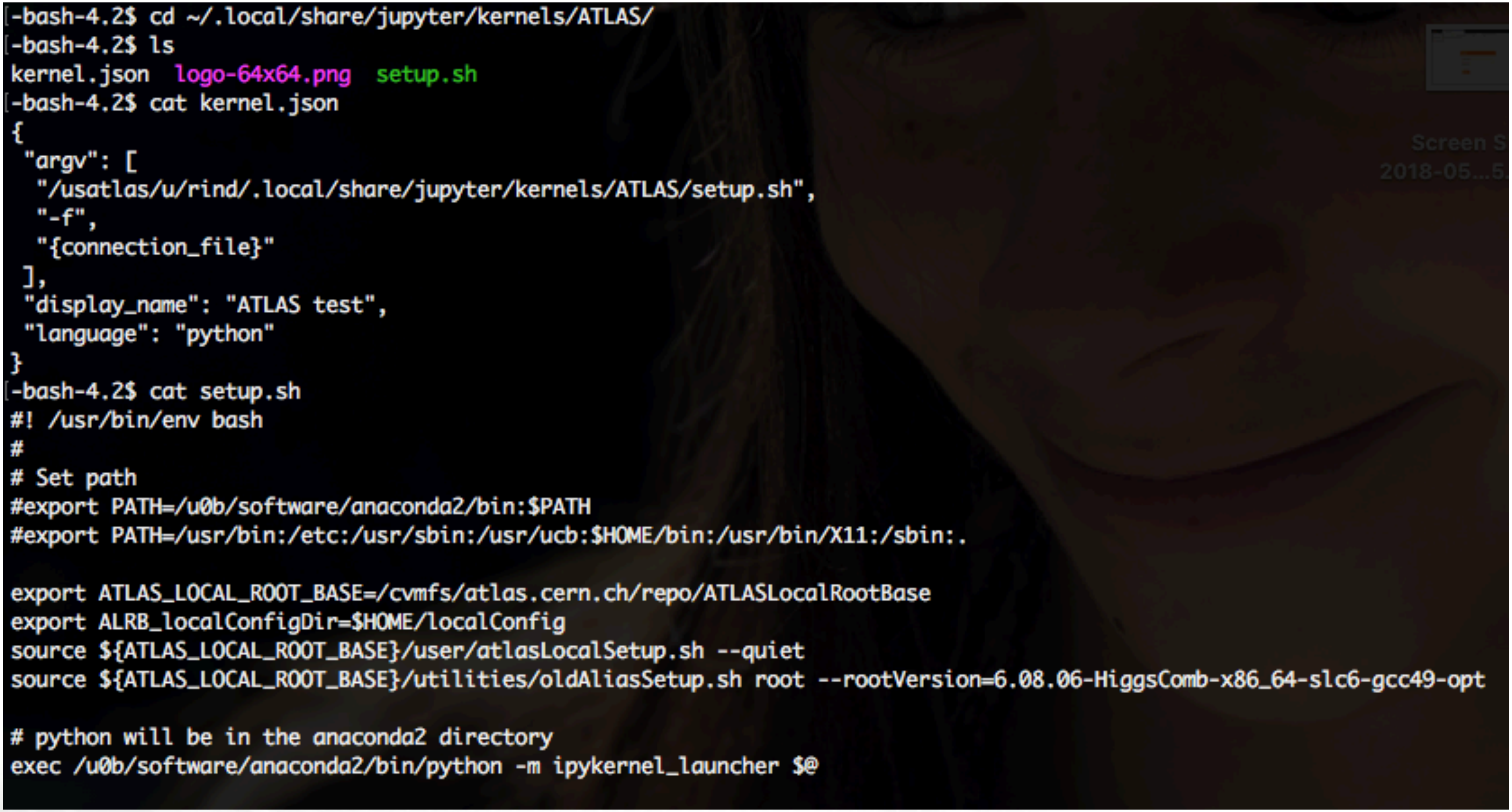
- Slurm - Higgs Test (Tensorflow) - IC python environment
- Belle-II (Bilas Pal) - Custom python3 kernel with BASF2 environment
- ATLAS GridScan (Viviana Cavaliere) - Custom python2 kernel with ATLAS environment
- ATLAS ML (JianCong Zeng) - Python3 kernel, with scikit-hep uproot and ML packages
- HTCondor API - Kristy Li/Will Strecker-Kellogg

Questions For Discussion

- Are people currently using Jupyter notebooks? How are they being used?
- Are there use cases at the SDCC, including at scale?
- What services would users like SDCC to provide?

Extra Slides

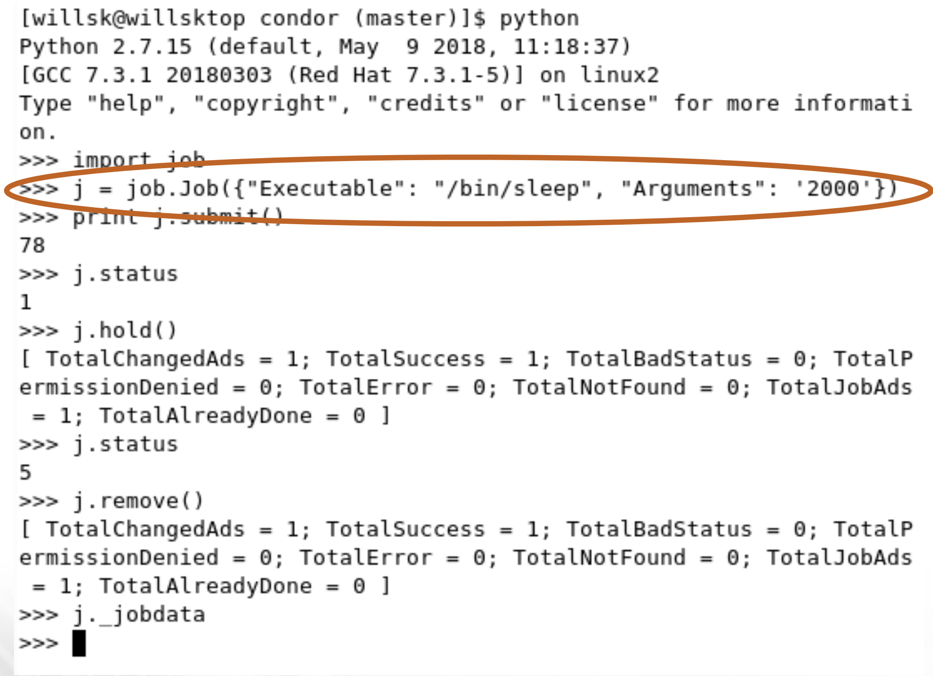
Example: Creating a Local Kernel Environment



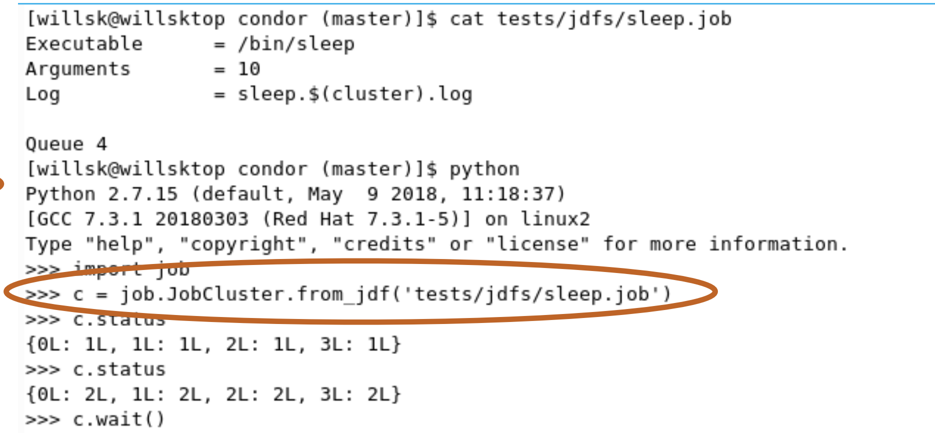
```
-bash-4.2$ cd ~/.local/share/jupyter/kernels/ATLAS/
-bash-4.2$ ls
kernel.json  logo-64x64.png  setup.sh
-bash-4.2$ cat kernel.json
{
 "argv": [
  "/usatlas/u/rind/.local/share/jupyter/kernels/ATLAS/setup.sh",
  "-f",
  "{connection_file}"
 ],
 "display_name": "ATLAS test",
 "language": "python"
}
-bash-4.2$ cat setup.sh
#! /usr/bin/env bash
#
# Set path
#export PATH=/u0b/software/anaconda2/bin:$PATH
#export PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/usr/bin/X11:/sbin:.

export ATLAS_LOCAL_ROOT_BASE=/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase
export ALRB_localConfigDir=$HOME/localConfig
source ${ATLAS_LOCAL_ROOT_BASE}/user/atlasLocalSetup.sh --quiet
source ${ATLAS_LOCAL_ROOT_BASE}/utilities/oldAliasSetup.sh root --rootVersion=6.08.06-HiggsComb-x86_64-slc6-gcc49-opt

# python will be in the anaconda2 directory
exec /u0b/software/anaconda2/bin/python -m ipykernel_launcher $@
```

HTCondor API

- Provide access to distributed computing through familiar APIs (python’s threading, multiprocessing, asyncio, etc…)
    - “I’d like to submit and manage a job or cluster of jobs”



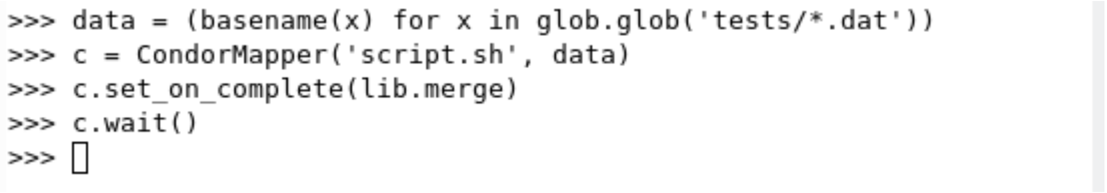
```
[willsk@willsktop condor (master)]$ python
Python 2.7.15 (default, May  9 2018, 11:18:37)
[GCC 7.3.1 20180303 (Red Hat 7.3.1-5)] on linux2
Type "help", "copyright", "credits" or "license" for more informati
on.
>>> import job
>>> j = job.Job({"Executable": "/bin/sleep", "Arguments": '2000'})
>>> print j.submit()
78
>>> j.status
1
>>> j.hold()
[ TotalChangedAds = 1; TotalSuccess = 1; TotalBadStatus = 0; TotalP
ermissionDenied = 0; TotalError = 0; TotalNotFound = 0; TotalJobAds
 = 1; TotalAlreadyDone = 0 ]
>>> j.status
5
>>> j.remove()
[ TotalChangedAds = 1; TotalSuccess = 1; TotalBadStatus = 0; TotalP
ermissionDenied = 0; TotalError = 0; TotalNotFound = 0; TotalJobAds
 = 1; TotalAlreadyDone = 0 ]
>>> j._jobdata
>>> 
```



```
[willsk@willsktop condor (master)]$ cat tests/jdfs/sleep.job
Executable      = /bin/sleep
Arguments       = 10
Log             = sleep.$(cluster).log

Queue 4
[willsk@willsktop condor (master)]$ python
Python 2.7.15 (default, May  9 2018, 11:18:37)
[GCC 7.3.1 20180303 (Red Hat 7.3.1-5)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import job
>>> c = job.JobCluster.from_jdf('tests/jdfs/sleep.job')
>>> c.status
{0L: 1L, 1L: 1L, 2L: 1L, 3L: 1L}
>>> c.status
{0L: 2L, 1L: 2L, 2L: 2L, 3L: 2L}
>>> c.wait()
```

HTCondor API

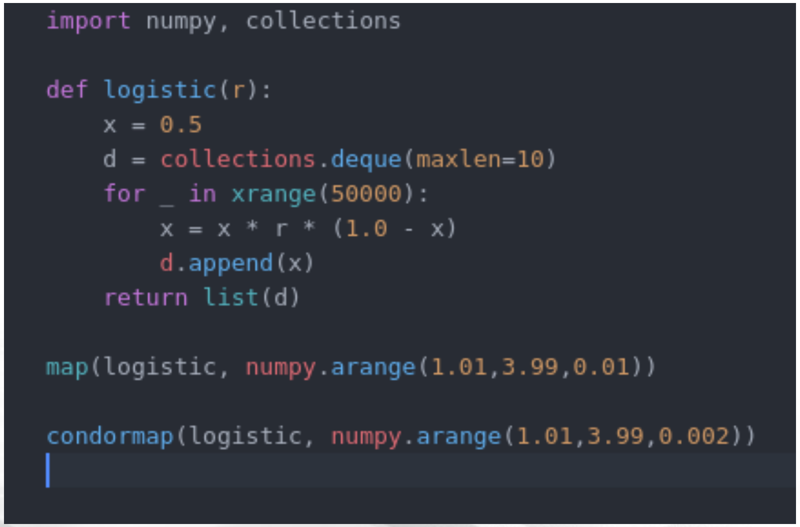
- At a higher level, abstract away the batch job layer
  - I’d like to run over a dataset



```
>>> data = (basename(x) for x in glob.glob('tests/*.dat'))
>>> c = CondorMapper('script.sh', data)
>>> c.set_on_complete(lib.merge)
>>> c.wait()
>>> 
```

- Serialize function, ship off to jobs, serialize output, gather…

- Early stage of development - see Will Strecker-Kellogg for details
- Also looking at dask-jobqueue



```
import numpy, collections

def logistic(r):
    x = 0.5
    d = collections.deque(maxlen=10)
    for _ in xrange(50000):
        x = x * r * (1.0 - x)
        d.append(x)
    return list(d)

map(logistic, numpy.arange(1.01,3.99,0.01))

condormap(logistic, numpy.arange(1.01,3.99,0.002))
```