Integrating Jupyter Notebooks Into Our Compute Environment @ SDCC

Updates & Plans

70 YEARS OF DISCOVERY

A CENTURY OF SERVICE

William Strecker-Kellogg
<willsk@bnl.gov>

HEPiX Spring 2019 UCSD

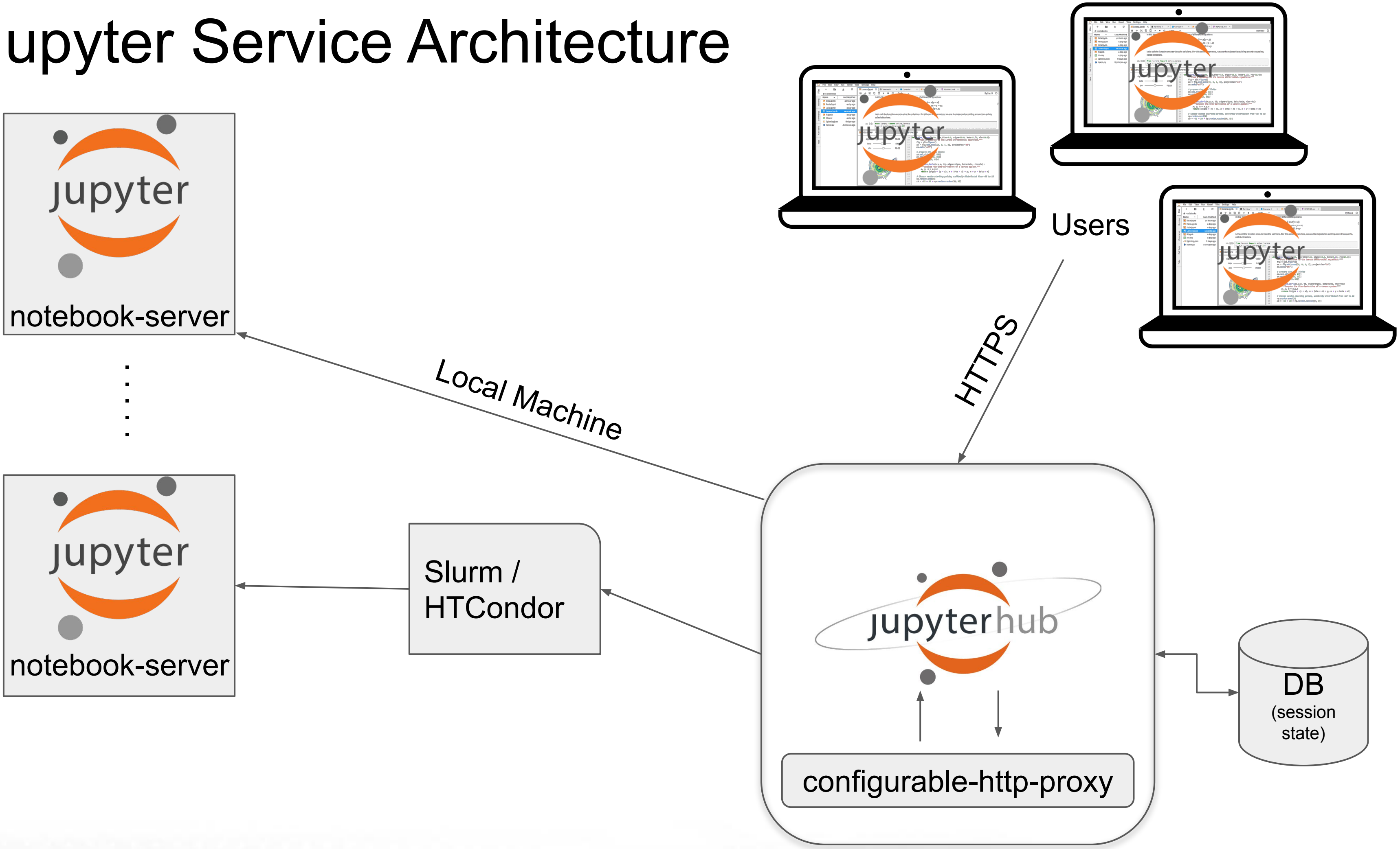
U.S. DEPARTMENT OF ENERGY

BROOKHAVEN NATIONAL LABORATORY

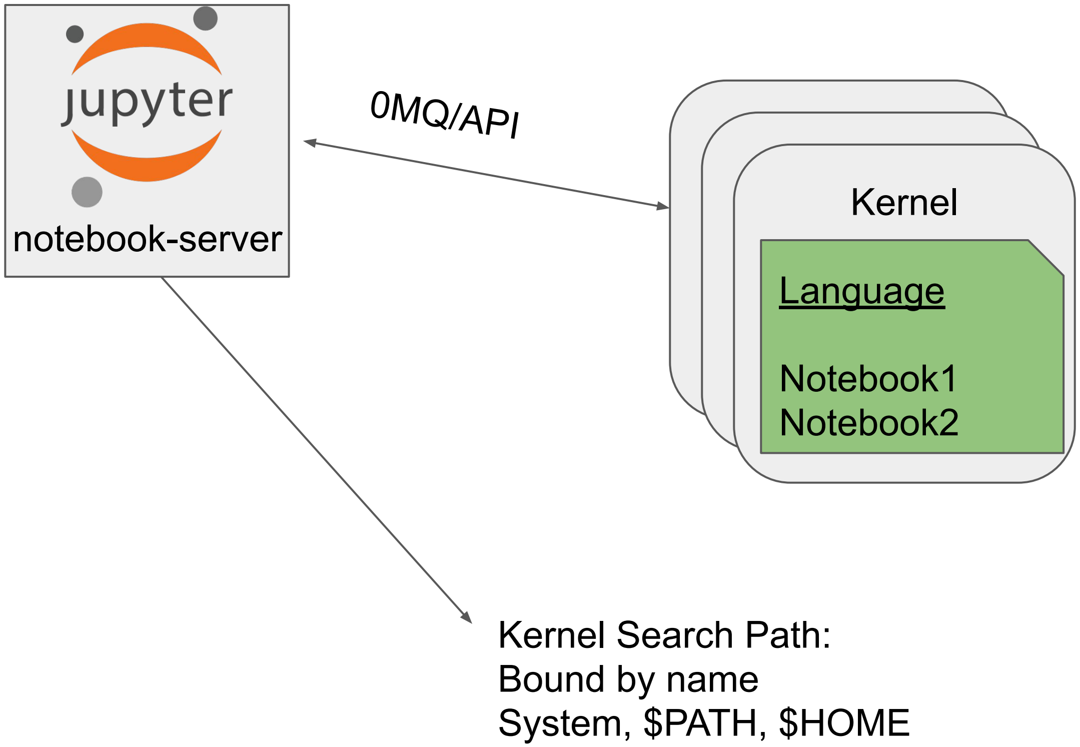
Shifting Analysis Methods

- Existing paradigm: all SSH-based
    - Split work between interactive and batch resources
    - Batch 100x-1000x size of interactive
        - Users sufficiently motivated to (learn to and) use batch systems
        - Intuitively understand workflow:
            - Develop, compile, test, small-scale run, data movement, all on interactive nodes
            - Workflow processing done on batch
- New paradigm: Jupyter
    - Expanding interactive toolset
    - Lower barrier of entry—both for learning curve and user-base
        - Learning curve
            - SSH, Shell, Batch Systems, etc... "steep" for some newer users
            - More problematic in domains newer to very-large-scale computing
                - Life Sci, Photon Sci, etc...
        - Userbase: supporting external users
            - From other domains, growing need to support external users

Jupyter Service Architecture



Jupyter Service Architecture

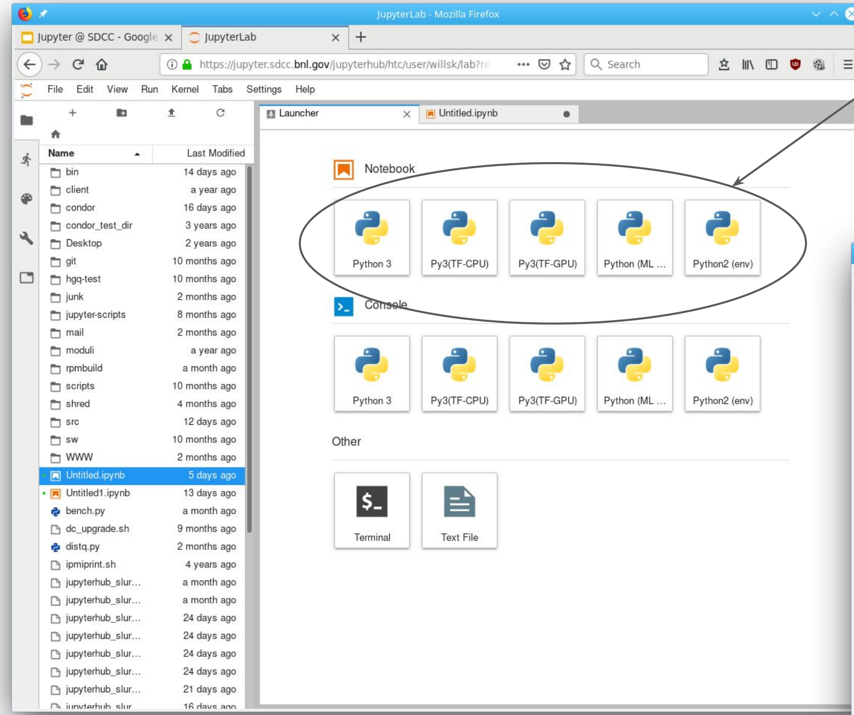


Kernel Search Path:
Bound by name
System, $PATH, $HOME

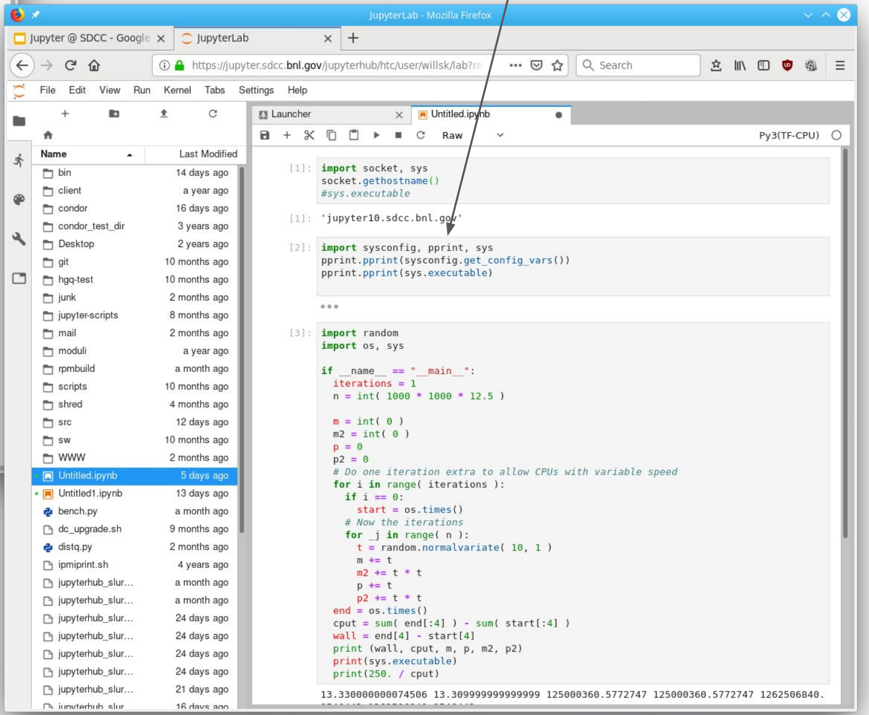
Jupyter Service UI

Kernels

Notebook Files



Name	Last Modified
bin	14 days ago
client	a year ago
condor	16 days ago
condor_test_dir	3 years ago
Desktop	2 years ago
git	10 months ago
hgq-test	10 months ago
junk	2 months ago
jupyter-scripts	8 months ago
mail	2 months ago
moduli	a year ago
rpmbuild	a month ago
scripts	10 months ago
shred	4 months ago
src	12 days ago
sw	10 months ago
WWW	2 months ago
Untitled.ipynb	5 days ago
Untitled1.ipynb	13 days ago
bench.py	a month ago
dc_upgrade.sh	9 months ago
distq.py	2 months ago
ipmiprint.sh	4 years ago
jupyterhub_slur…	a month ago
jupyterhub_slur…	a month ago
jupyterhub_slur…	24 days ago
jupyterhub_slur…	24 days ago
jupyterhub_slur…	24 days ago
jupyterhub_slur…	24 days ago
jupyterhub_slur…	21 days ago
jupyterhub_slur	16 days ago



Name	Last Modified
bin	14 days ago
client	a year ago
condor	16 days ago
condor_test_dir	3 years ago
Desktop	2 years ago
git	10 months ago
hgq-test	10 months ago
junk	2 months ago
jupyter-scripts	8 months ago
mail	2 months ago
moduli	a year ago
rpmbuild	a month ago
scripts	10 months ago
shred	4 months ago
src	12 days ago
sw	10 months ago
WWW	2 months ago
Untitled.ipynb	5 days ago
Untitled1.ipynb	13 days ago
bench.py	a month ago
dc_upgrade.sh	9 months ago
distq.py	2 months ago
ipmiprint.sh	4 years ago
jupyterhub_slur…	a month ago
jupyterhub_slur…	a month ago
jupyterhub_slur…	24 days ago
jupyterhub_slur…	24 days ago
jupyterhub_slur…	24 days ago
jupyterhub_slur…	24 days ago
jupyterhub_slur…	21 days ago
jupyterhub_slur	16 days ago

```
[1]: import socket, sys
     socket.gethostname()
     #sys.executable

[1]: 'jupyter10.sdcc.bnl.gov'

[2]: import sysconfig, pprint, sys
     pprint.pprint(sysconfig.get_config_vars())
     pprint.pprint(sys.executable)

     •••

[3]: import random
     import os, sys

     if __name__ == "__main__":
       iterations = 1
       n = int( 1000 * 1000 * 12.5 )

       m = int( 0 )
       m2 = int( 0 )
       p = 0
       p2 = 0
       # Do one iteration extra to allow CPUs with variable speed
       for i in range( iterations ):
         if i == 0:
           start = os.times()
         # Now the iterations
         for _j in range( n ):
           t = random.normalvariate( 10, 1 )
           m += t
           m2 += t * t
           p += t
           p2 += t * t
       end = os.times()
       cput = sum( end[:4] ) - sum( start[:4] )
       wall = end[4] - start[4]
       print (wall, cput, m, p, m2, p2)
       print(sys.executable)
       print(250. / cput)

     13.330000000074506 13.309999999999999 125000360.5772747 125000360.5772747 1262506840.
```

U.S. DEPARTMENT OF ENERGY

BROOKHAVEN NATIONAL LABORATORY

70 YEARS OF DISCOVERY
A CENTURY OF SERVICE

Jupyter at BNL: Recent Changes

- I took over leading the effort of bringing this service to production
  - Significant back-end engineering required: details to follow
- Common solution for HTC and HPC resource access
  - Significant challenges, try to leverage existing resources as much as possible
- Opening up access
  - Front-end authenticating proxy landing page
    - Choose running model (HTC / HPC, Local, Batch, etc...)
    - Get running instance appropriate to your security zone
      - Internal users get full access to our environment
      - External? Isolated in container, no POSIX FS access, storage by API/Token only
        - Scratch area? Home?
      - This is a radical departure from cyber-security norms!
- Automation
  - Puppetized proxy, hub, and configuration of everything

Production Architecture

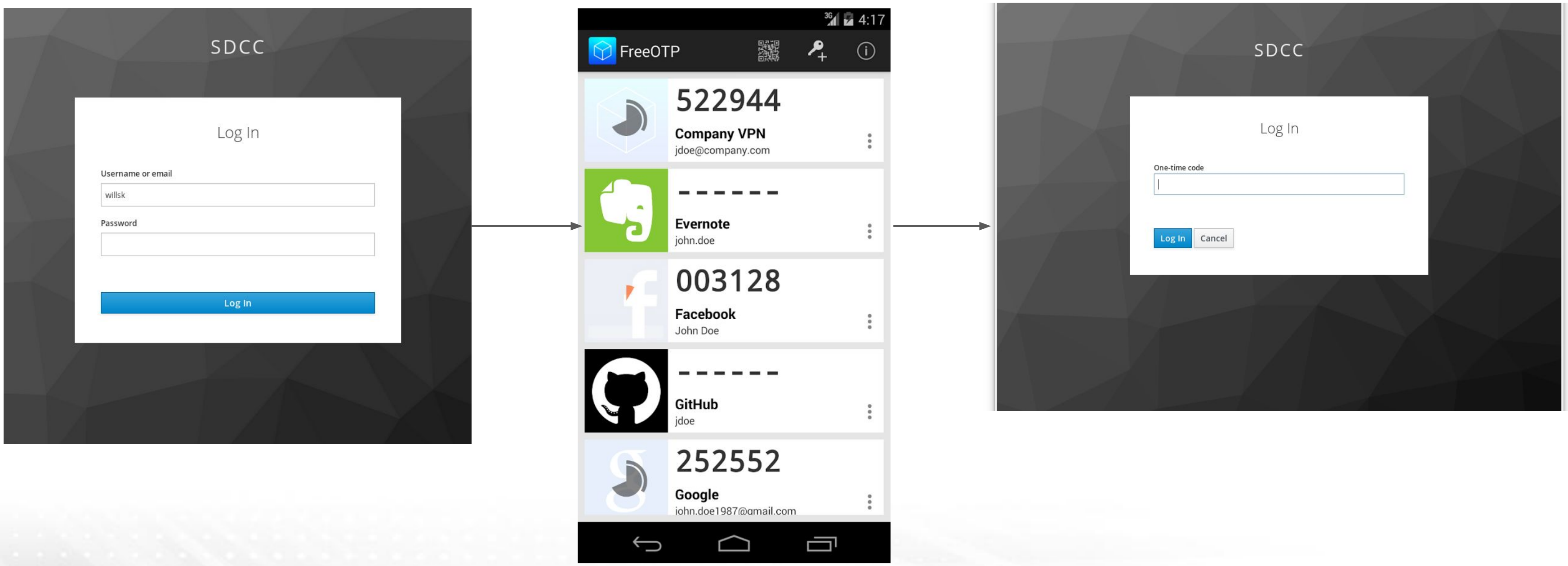
- Requirements:
  - Expose to the world via https://jupyter.sdcc.bnl.gov
    - Satisfy cyber-security constraints
      - A major footgun for cybersecurity!
  - Allow users to leverage *all* computational resources here they need
    - Implies enabling both HTC and HPC / GPU / ML workflows
  - Simplicity
    - Should be as close to a "familiar" environment as possible
- Design:
  - Insert authenticating proxy in front to decouple jupyterhub from cybersecurity requirements
  - Scale notebooks via load-balancing as well as via batch systems
  - Enable access to GPU nodes in a user-friendly way
    - UI Changes for Slurm
- Status:
  - Have proxy with multifactor auth in place
    - Uses SDCC accounts + Keycloak (for SSO) + FreeOTP Token (for MFA)

Cybersecurity Requirements: Multifactor

Using Keycloak (for SSO—slated to replace our Shibboleth)

Has built-in MFA via FreeOTP (QR Code + Google Auth app or own app)

Easy setup by scanning QR code first time



SDCC

Log In

Username or email: willsk

Password:

522944
Company VPN
jdoe@company.com

Evernote
john.doe

003128
Facebook
John Doe

GitHub
jdoe

252552
Google
john.doe1987@gmail.com

SDCC

Log In

One-time code:

Compute: Two modes, Two workflows

HPC & HTC (parallel vs interlinked and accelerator vs plain-cpu -- often falls along this same divide)

- High-throughput systems for big-data parallel processing
- High-performance systems for GPUs / MPI / accelerators

Dev/Test Cycle & Orchestration (working on code vs submitting workflows)

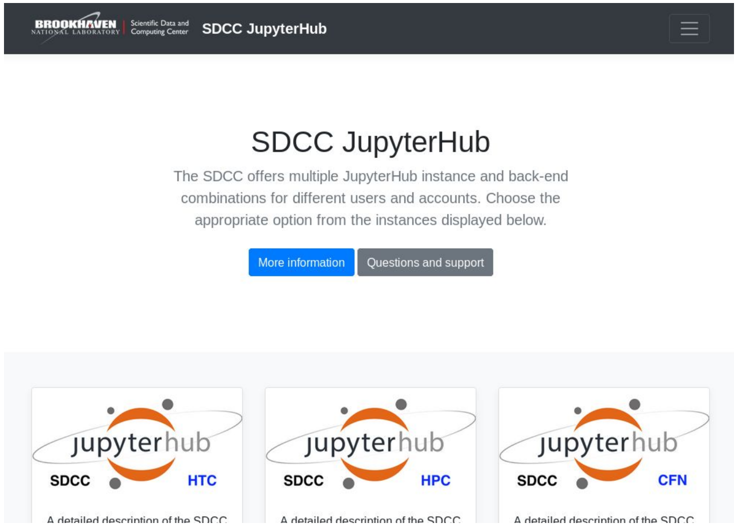
1. Job workflow management–make use of GUI to submit / manage batch flows
    a. Convenience and progress-tracking
    b. Some users dubious of value to tracking a 100k job workflow through a GUI
2. Direct development & testing on better hardware
    a. Using Slurm batch spawner to access GPUs
    b. Use condor spawner to access multicore / lots of memory

Jupyter Work Scheduling

- Interactive / batch divide: Let's apply the same paradigm here!
- For Orchestration: a small cluster of directly-launched jupyter instances
  - HTTP-level Load-balanced from frontend proxy
  - One each on IC and HTCondor shared pool
- For Develop and Test: Use existing batch systems
  - HTCondor and Slurm support running a jupyterlab session as a batch job
  - Containers can enter at the batch level to isolate external people
  - Or can be based on the choice of environment
  - Open questions:
    - Latency, cleanup, starvation

HTTP Frontend Configuration

- Authentication via Mellon plugin (for Keycloak)
- Subdivide URL space for different hub servers
  - */jupyterhub/$cluster* for HTC/HPC/others



SDCC JupyterHub

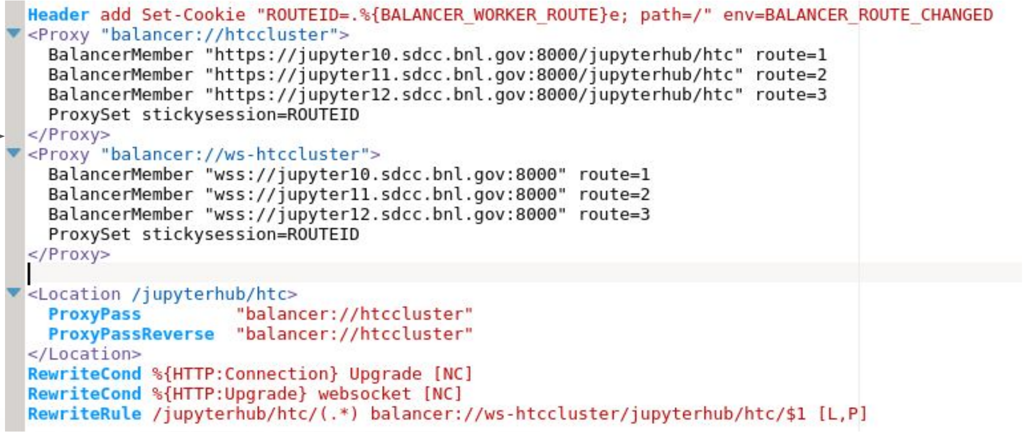
The SDCC offers multiple JupyterHub instance and back-end combinations for different users and accounts. Choose the appropriate option from the instances displayed below.

A detailed description of the SDCC

A detailed description of the SDCC

A detailed description of the SDCC

- Load-balancing configuration
  - Need cookie for sticky-sessions
  - Newest apache on RHEL7
    - Req. websockets support

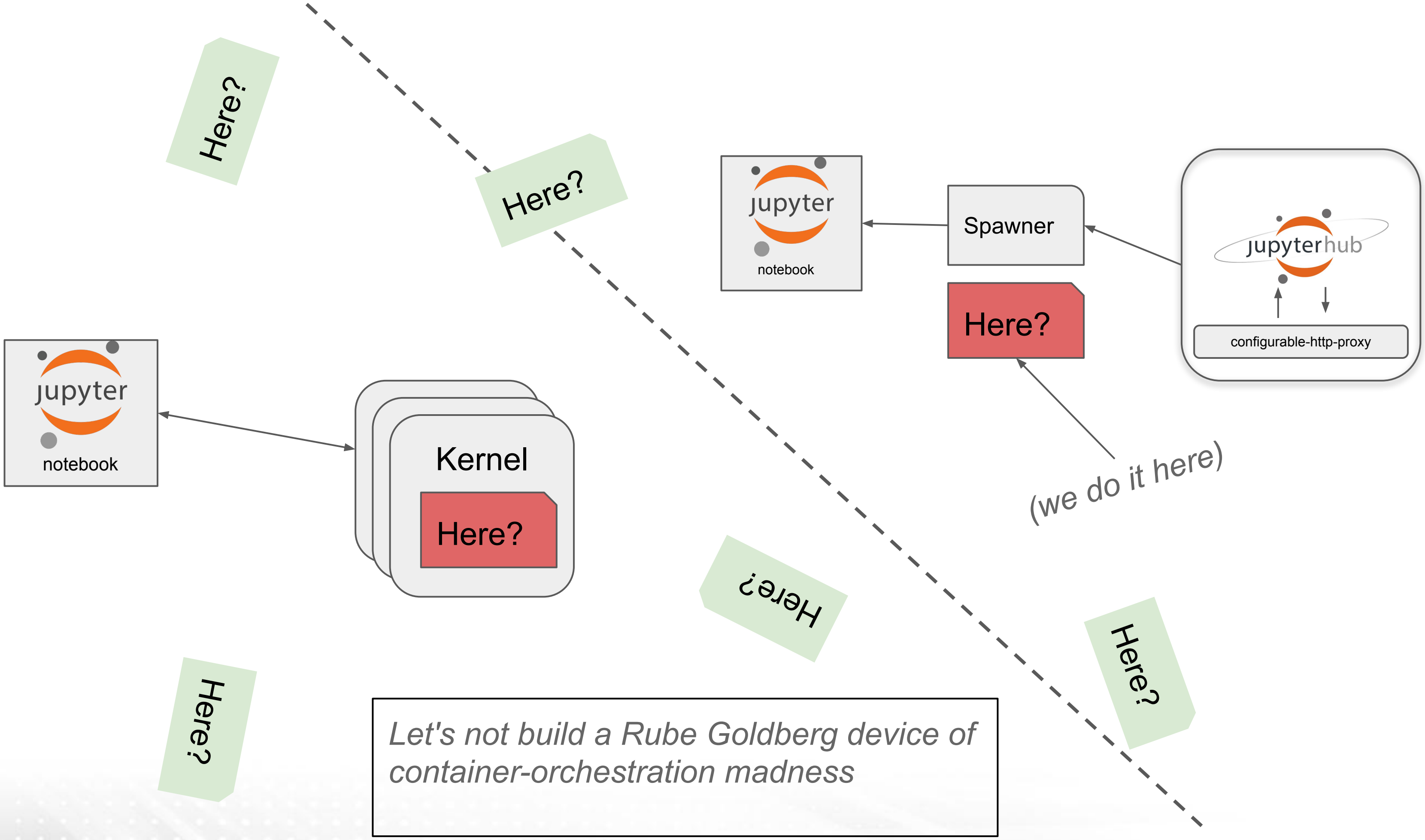


```
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED
<Proxy "balancer://htccluster">
  BalancerMember "https://jupyter10.sdcc.bnl.gov:8000/jupyterhub/htc" route=1
  BalancerMember "https://jupyter11.sdcc.bnl.gov:8000/jupyterhub/htc" route=2
  BalancerMember "https://jupyter12.sdcc.bnl.gov:8000/jupyterhub/htc" route=3
  ProxySet stickysession=ROUTEID
</Proxy>
<Proxy "balancer://ws-htccluster">
  BalancerMember "wss://jupyter10.sdcc.bnl.gov:8000" route=1
  BalancerMember "wss://jupyter11.sdcc.bnl.gov:8000" route=2
  BalancerMember "wss://jupyter12.sdcc.bnl.gov:8000" route=3
  ProxySet stickysession=ROUTEID
</Proxy>

<Location /jupyterhub/htc>
  ProxyPass          "balancer://htccluster"
  ProxyPassReverse   "balancer://htccluster"
</Location>
RewriteCond %{HTTP:Connection} Upgrade [NC]
RewriteCond %{HTTP:Upgrade} websocket [NC]
RewriteRule /jupyterhub/htc/(.*) balancer://ws-htccluster/jupyterhub/htc/$1 [L,P]
```

Experiment Environments / Containers

- (opinion) Containers solve the problem of different user environments, not the problem of scheduling / deploying units of compute
    - (biased statement) Many, many attempts at "scheduling" as an afterthought in a project, almost always done poorly! Just use a batch system
- When you get a jupyter session, what environment are you in?
- Many "layers" in which to inject a container, which one is not clear
- Create a "default" env by cloning our native farm-image
    - The one on the current farm nodes
- User choice at portal for which environment to start?
    - Local jupyter: spawn in container, access software in shared area
    - Batch spawner: batch system container layer to spawn
- No orchestration needed, but whose problem is setting up the environments?
    - Collaborative between admins and experiment software folks
    - There is work here for a software librarian

Containers?

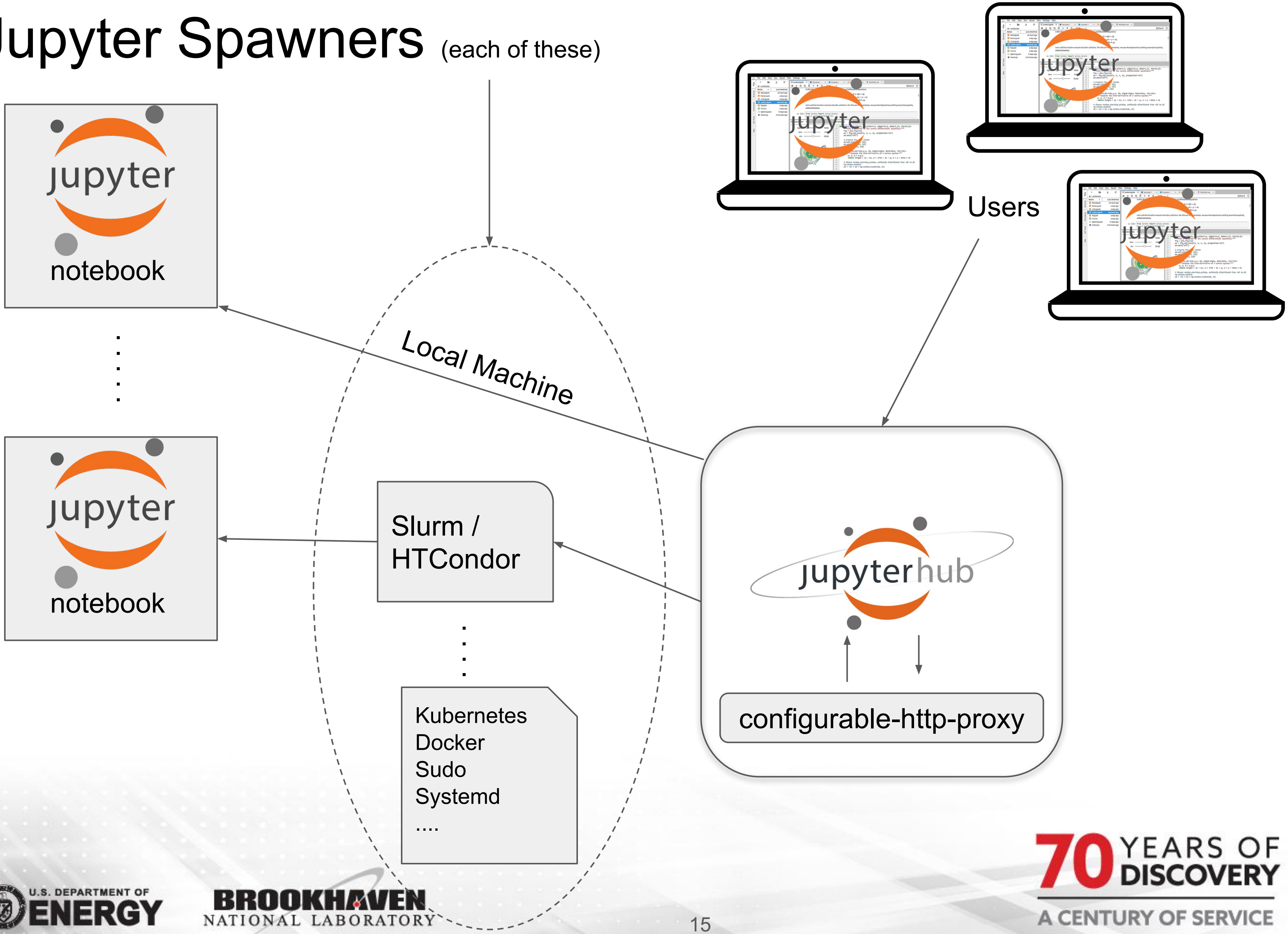


Let's not build a Rube Goldberg device of container-orchestration madness

Singularity

- At the spawner level: seems logical because it can be presented to the user as an option
  - And users can choose native to run their own containers via kernels themselves
- Distribution
  - CVMFS (not available on HPC)
  - Shared FS
  - Scalability? TBD
- Too Much Site-specific Customizations?
  - need to bind-mount everything in that we would need for shared FS (we have dozens)
  - Bare minimum of our GPFS + home directories
- We plan to provide a few generally useful containers but will delegate to software librarian / experiment computing people

Jupyter Spawners (each of these)



jupyter notebook

jupyter notebook

Local Machine

Slurm / HTCondor

Kubernetes
Docker
Sudo
Systemd
....

Users

jupyterhub

configurable-http-proxy

Dynamic Spawner Forms

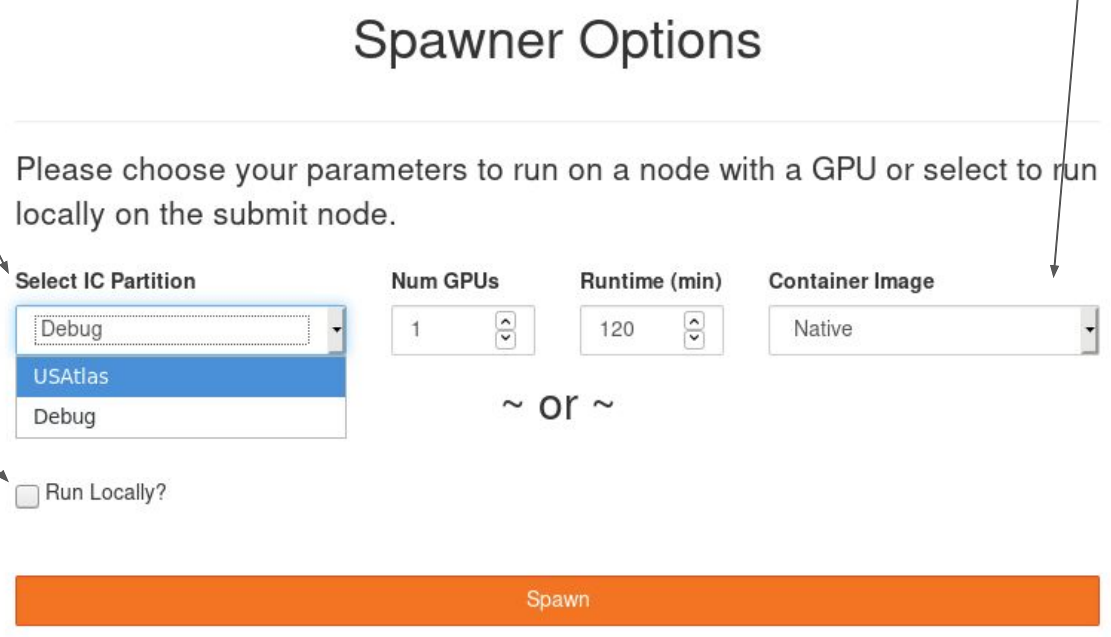
- Class object in JupyterHub
  - GET method presents a form, POST with formdata launches form
- Need to dynamically select a spawner from a form
  - See ProfileSpawner, presents a simple form
- Also need to be able to dynamically present a form based on user / backend

- I wrote a form-spawner which allows dynamic forms to be presented with some logic that determines which spawner to choose based on the form contents and which variables to pass through to each spawner backend
  - Form template file is dynamically configurable
  - Logic for parsing form input in subclass
  - Override set_class() method to return correct class

Dynamic Spawner Forms

Display only partitions that user has access to

JS to auto-select container image per-partition



Spawner Options

Please choose your parameters to run on a node with a GPU or select to run locally on the submit node.

Select IC Partition	**Num GPUs**	**Runtime (min)**	**Container Image**
Debug	1	120	Native

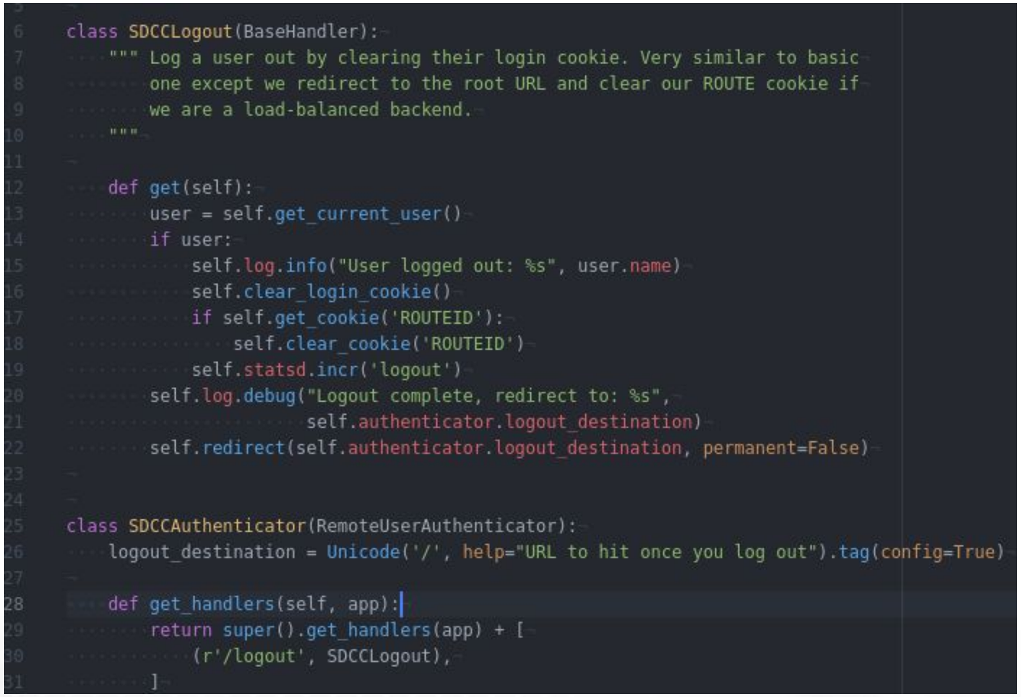
~ or ~

□ Run Locally?

Select here and will launch Local instead of Batch spawner

Logout Problem

- Delegated auth on front-end→remote-user auth on backend
- Default remote-user auth does not handle logout!
    - I modfied as below to ensure it works as well as clear load-balancer cookie:
    - `auth.logout_destination` is keycloak's logout-handler



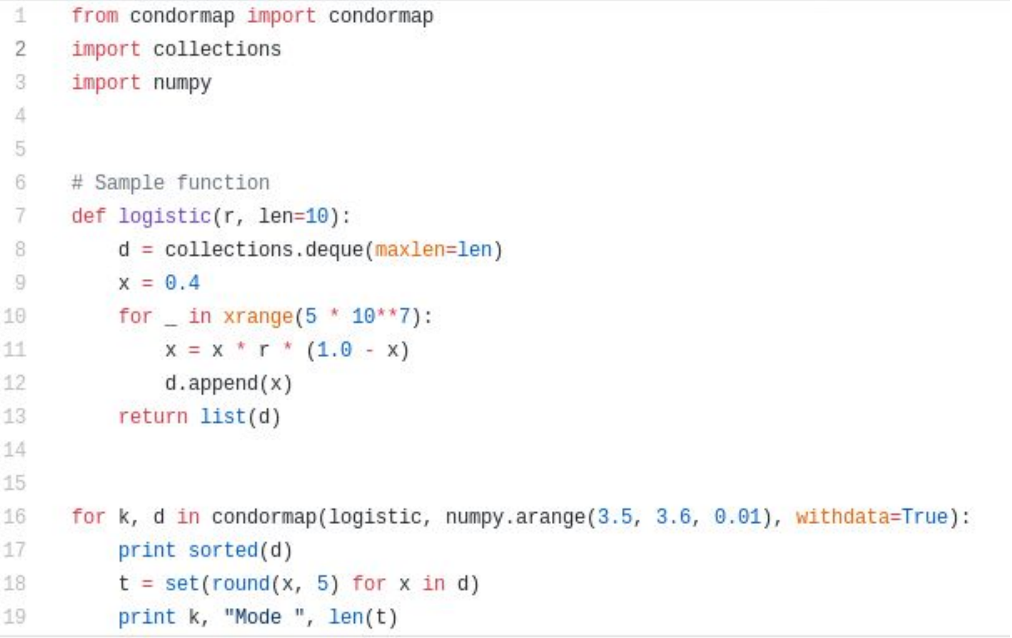
```python
class SDCCLogout(BaseHandler):
    """ Log a user out by clearing their login cookie. Very similar to basic
        one except we redirect to the root URL and clear our ROUTE cookie if
        we are a load-balanced backend.
    """

    def get(self):
        user = self.get_current_user()
        if user:
            self.log.info("User logged out: %s", user.name)
            self.clear_login_cookie()
            if self.get_cookie('ROUTEID'):
                self.clear_cookie('ROUTEID')
            self.statsd.incr('logout')
        self.log.debug("Logout complete, redirect to: %s",
                       self.authenticator.logout_destination)
        self.redirect(self.authenticator.logout_destination, permanent=False)


class SDCCAuthenticator(RemoteUserAuthenticator):
    logout_destination = Unicode('/', help="URL to hit once you log out").tag(config=True)

    def get_handlers(self, app):
        return super().get_handlers(app) + [
            (r'/logout', SDCCLogout),
        ]
```

Anticipated Problems

- Batch system latencies
  - Need UI / UX improvements for when you can't get a slot
  - Solved by providing over-subscribable load-balanced resources
- Redundant / Recoverable hub
  - When hub restarts, can we reconnect sessions?
  - What about sharing the database across hubs
    - Need to explore if this is possible
- Sticky LB Sessions
  - If you reconnect with same browser you get same backend via ROUTEID cookie
  - If you use a different browser, there is a (N-1)/N chance you get a different backend!
    - Could apache LB be integrated with shared DB session store?
      - This is probably beyond me!

Integrating Jupyter with Compute

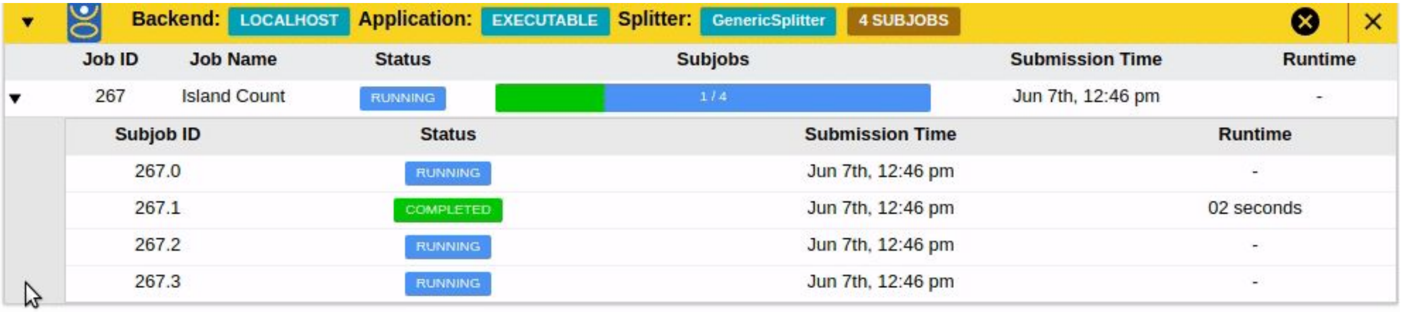
- How to make it easier to use compute from Jupyter?
  - Abstract away using a batch system
    - Experimental code I wrote
    - Dask / IPyParallel / Parsl etc...
- Goal: abstract away the fact that you are using a batch system at all
  - Either through trivial substitutes
    - map()→condormap()
  - Or (better) through cell "magics"
    - %slurm or equivalent
  - Or via nice pythonic decorators that submit to batch systems
    - Parsl



```python
from condormap import condormap
import collections
import numpy


# Sample function
def logistic(r, len=10):
    d = collections.deque(maxlen=len)
    x = 0.4
    for _ in xrange(5 * 10**7):
        x = x * r * (1.0 - x)
        d.append(x)
    return list(d)


for k, d in condormap(logistic, numpy.arange(3.5, 3.6, 0.01), withdata=True):
    print sorted(d)
    t = set(round(x, 5) for x in d)
    print k, "Mode ", len(t)
```

Integrating Jupyter with Compute

- Collaborating with Swan folks at CERN
  - Had a Google SoC student make a really nice JS UI for job tracking
  - Used Ganga integration
    - We are looking into how to integrate HTCondor with this or do something similar
    - Ganga "api" make some assumptions that are not good for HTC
      - Have a student intern part-time who may work on this



Backend: LOCALHOST Application: EXECUTABLE Splitter: GenericSplitter 4 SUBJOBS

Job ID	Job Name	Status	Subjobs	Submission Time	Runtime
267	Island Count	RUNNING	1 / 4	Jun 7th, 12:46 pm	-

Subjob ID	Status	Submission Time	Runtime
267.0	RUNNING	Jun 7th, 12:46 pm	-
267.1	COMPLETED	Jun 7th, 12:46 pm	02 seconds
267.2	RUNNING	Jun 7th, 12:46 pm	-
267.3	RUNNING	Jun 7th, 12:46 pm	-

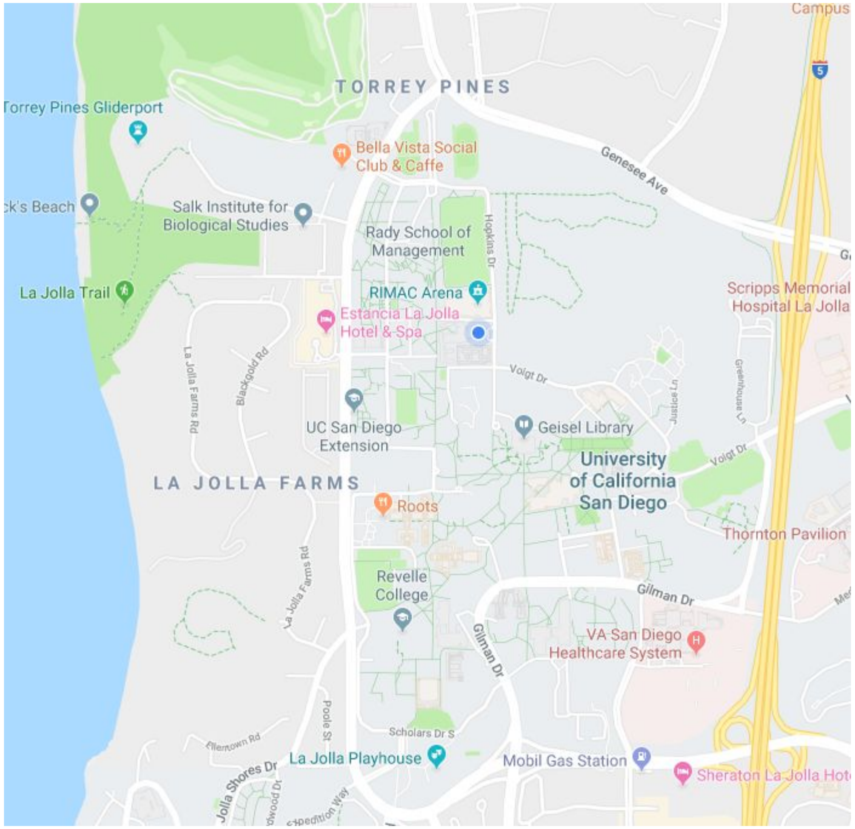
Status and Rollout

- Currently a beta-level of service / support
  - Our Keycloak infrastructure is in pre-production
  - Our jupyterhub service is new and not considered stable yet
  - If we change MFA providers, users may need to re-generate tokens in a few months
- High interest from Photon Sciences people at BNL
  - NSLS-II is very interested in using our jupyterhub
  - Center for Functional Nanomaterials (CFN) as well
    - Planning on really testing it next month
- Interest from USATLAS ML people
  - Ability to use IC cluster GPUs is key for beamline fast-online vs theory people
- Open question who will maintain library of kernels or notebooks and at what level
  - Each beamline / experiment?
  - Each facility?
  - Need to manage complexity here...

Ongoing Work / Open Questions

- Working towards notebook sync / share and multi-server management
  - BNLBox (owncloud), Git, what else?
- Notebook-Server Management
  - Kludgey interface today–want to add support for multiple servers in one hub instance
- Batch Interface Issues
  - Latency+usability; how to gracefully fail when no resources available
- Cleanup HTCondor BatchSpawner
  - Don't write API key in environment anyone can see...
- Need feedback on best way to integrate wildly different communities
  - Will a few /hub servers suffice? Don't want this to balloon out of control

Thank You! Questions?



HTCondor Live!

Thanks to Ofer Rind, Mizuki Karasawa, Jamal Irving, John DeStefano, Doug Benjamin, Costin Caramarcu and others for all your help!