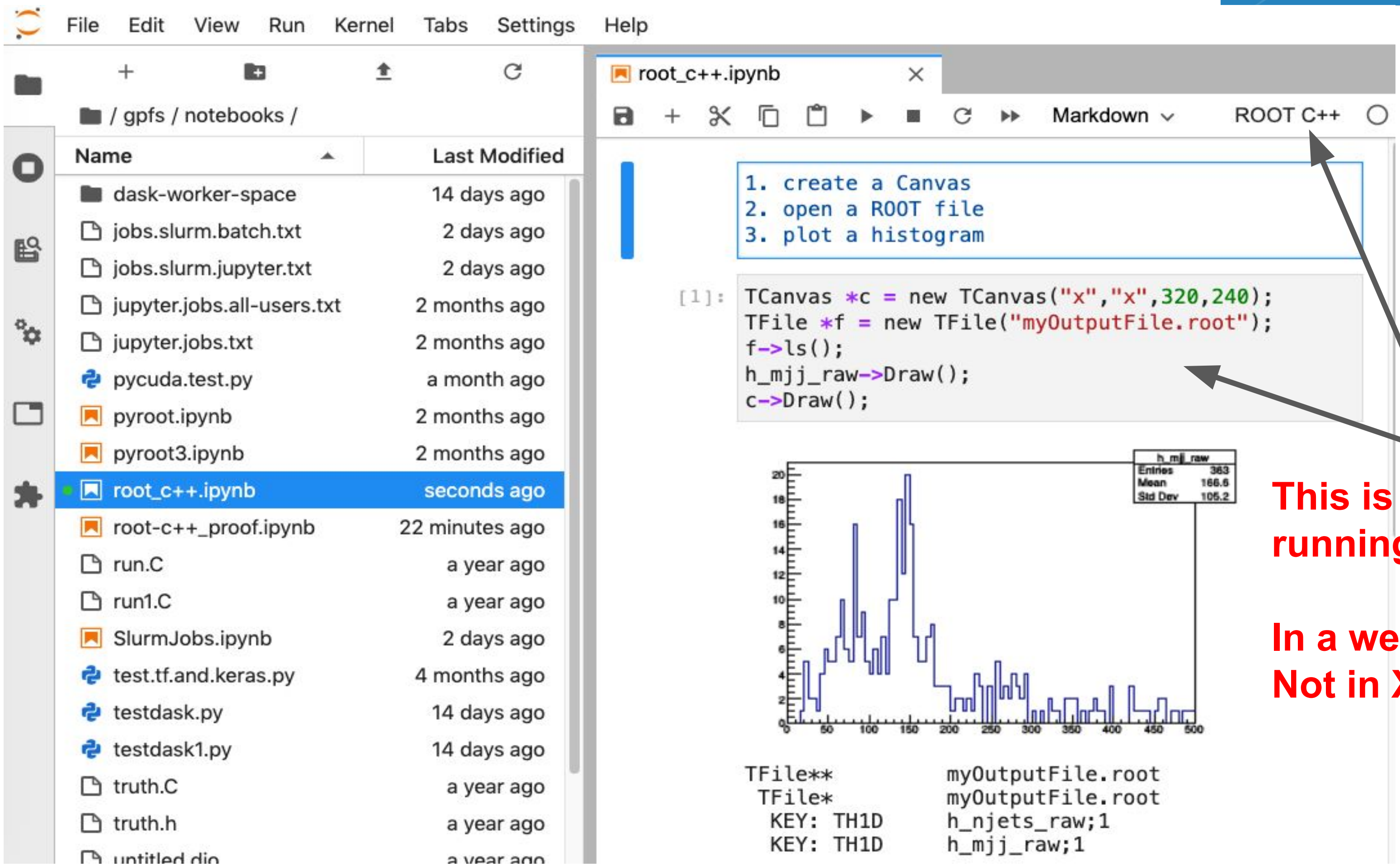
Jupyter at BNL and SLAC Analysis Facilities

Doug Benjamin
William Strecker-Kellogg
Wei Yang

OSG All-Hands Meeting 2020 / OSG and US LHC, Sep 4, 2020



Name	Last Modified
dask-worker-space	14 days ago
jobs.slurm.batch.txt	2 days ago
jobs.slurm.jupyter.txt	2 days ago
jupyter.jobs.all-users.txt	2 months ago
jupyter.jobs.txt	2 months ago
pycuda.test.py	a month ago
pyroot.ipynb	2 months ago
pyroot3.ipynb	2 months ago
root_c++.ipynb	seconds ago
root-c++_proof.ipynb	22 minutes ago
run.C	a year ago
run1.C	a year ago
SlurmJobs.ipynb	2 days ago
test.tf.and.keras.py	4 months ago
testdask.py	14 days ago
testdask1.py	14 days ago
truth.C	a year ago
truth.h	a year ago
untitled.dio	a year ago

```
1. create a Canvas
2. open a ROOT file
3. plot a histogram
```

```
TCanvas *c = new TCanvas("x","x",320,240);
TFile *f = new TFile("myOutputFile.root");
f->ls();
h_mjj_raw->Draw();
c->Draw();
```

```
TFile**         myOutputFile.root
 TFile*         myOutputFile.root
  KEY: TH1D     h_njets_raw;1
  KEY: TH1D     h_mjj_raw;1
```

This is Jupyter, running ROOT C++

In a web browser
Not in X-window

Why Jupyter at US ATLAS Analysis Facilities

- It can replace the traditional ROOT over X-windows
  - Smoothly handle display latency (so we don’t have to run NoMachine or FastX)
  - User can close the web browser and later reconnect again
  - Notebook capability (no need to save your plots somewhere and the forget what are they)
- More importantly: a bridge to world of python and modern data science tools
  - Numpy, matplotlib, GPU/ML etc.
- Jupyter should support interactive data analysis for two types of users:
  - I just want to get my analysis done (ROOT based data analysis)
    - ROOT C++, PyROOT, uproot, integrated with ATLAS analysis releases
    - Currently python2.7
  - I want to explore data science, GPU/ML, etc. in my analysis
    - GPU support and ML packages
    - python3

Standardize Jupyter environment at US ATLAS AFs

- Flat the learning curve: common look and feel:
  - A common set of Jupyter kernels, Terminal, Markdown, etc.
  - “Interactive login Terminal”
    - Like a ssh session: vi/editor, gcc/g++/gdb, ssh, openssl/Grid tools, batch tools
- PyROOT and ROOT C++ Kernels
  - For ROOT based analysis (and include uproot)
  - Integrate ATLAS AnalysisBase (thus python2.7) - PyROOT included in AnalysisBase
- Python3 kernels
  - Include PyROOT and uproot as well, but there kernels are:
  - **Data science and GPU/ML oriented**
  - Include key ML packages such as Tensorflow/Keras
    - Packages that are not easy for users to install and tune by themselves.
  - In the meantime, allow users to install additional packages (via PIP, etc.)
    - This help reducing the number of python packages AFs have to maintain

Challenge of deployment

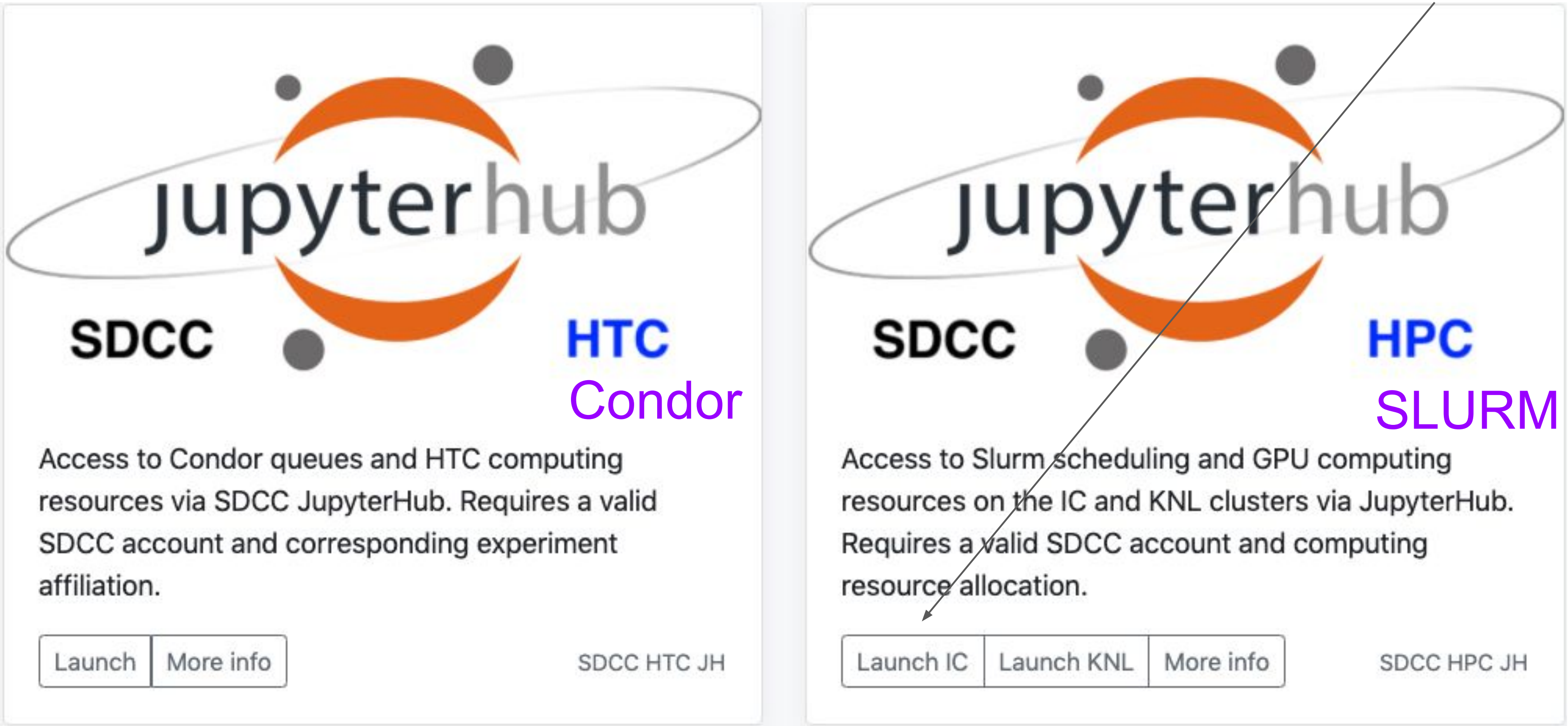
- Neither BNL and SLAC AFs have significant dedicated hardware for Jupyter
  - Part of a much large JupyterLab or batch deployment
  - However, BNL jupyterlab jobs have priority on the USATLAS HPC hardware
- JupyterLab runtime environments provided differently at BNL and SLAC
  - BNL uses virtualenv, frontend + 3 backends (Condor Cluster, IC cluster/GPU, KNL cluster)
  - SLAC uses Singularity container, frontend (open ondemand) + a GPU cluster
- So we choose to have a common set of Jupyter kernels
  - BNL provides cvmfs space to store relevant kernels and software packages.
  - PyROOT and ROOT C++ kernels are easy to stardalize, in JupyterLab runtime envorinment
    - AnalysisBase provides PyROOT and Python2.7
  - Python3 kernels
    - JupyterLab runtime environments provides
      - Python3, uproot, Tensorflow, numpy, matplotlib, DASK, etc.
      - ROOT 6.22 in CVMFS (to support PyROOT)
    - CUDA library from compute node environment (BNL) and container (SLAC)

Containers

- The software environments can be integrated within Singularity containers
  - Singularity image is in use at SLAC
  - Experimenting at BNL (currently virtualenv)
  - The images can be used anywhere
- Containers are published alongside the Jupyter kernels CVMFS
  - Hosted by BNL SDCC
- Jupyter spawners at both sites provide UIs that allow users to choose which container-image to launch in
  - Works on both interactive and batch-launched Jupyter sessions
  - User-provided containers are easy to implement and integrate.
- No direct Docker support is planned
  - However Singularity can pull Docker images

https://jupyter.sdcc.bnl.gov

Nvidia GPUs

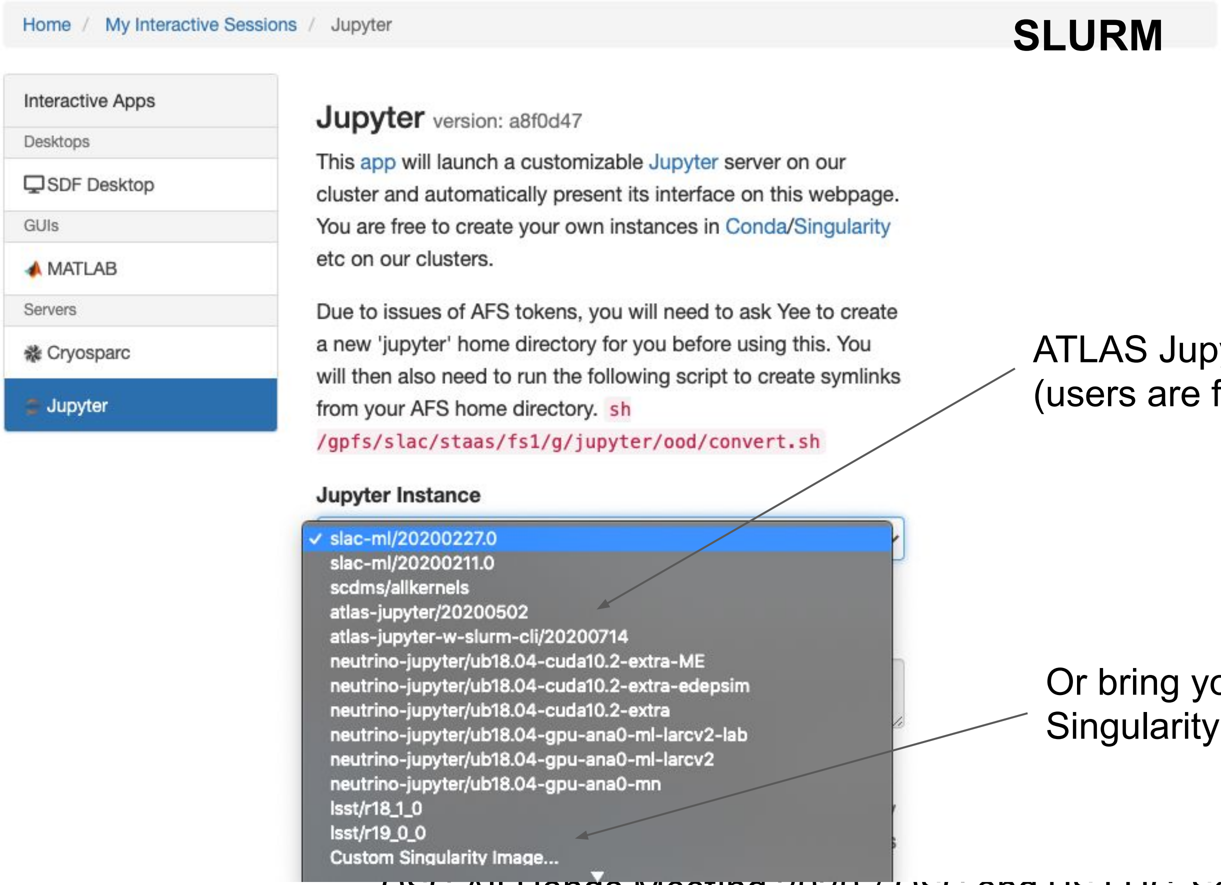


Access to Condor queues and HTC computing resources via SDCC JupyterHub. Requires a valid SDCC account and corresponding experiment affiliation.

Access to Slurm scheduling and GPU computing resources on the IC and KNL clusters via JupyterHub. Requires a valid SDCC account and computing resource allocation.

https://sdf.slac.stanford.edu

Shared GPU clusters managed by SLURM



Jupyter version: a8f0d47

This app will launch a customizable Jupyter server on our cluster and automatically present its interface on this webpage. You are free to create your own instances in Conda/Singularity etc on our clusters.

Due to issues of AFS tokens, you will need to ask Yee to create a new 'jupyter' home directory for you before using this. You will then also need to run the following script to create symlinks from your AFS home directory. `sh /gpfs/slac/staas/fs1/g/jupyter/ood/convert.sh`

Jupyter Instance

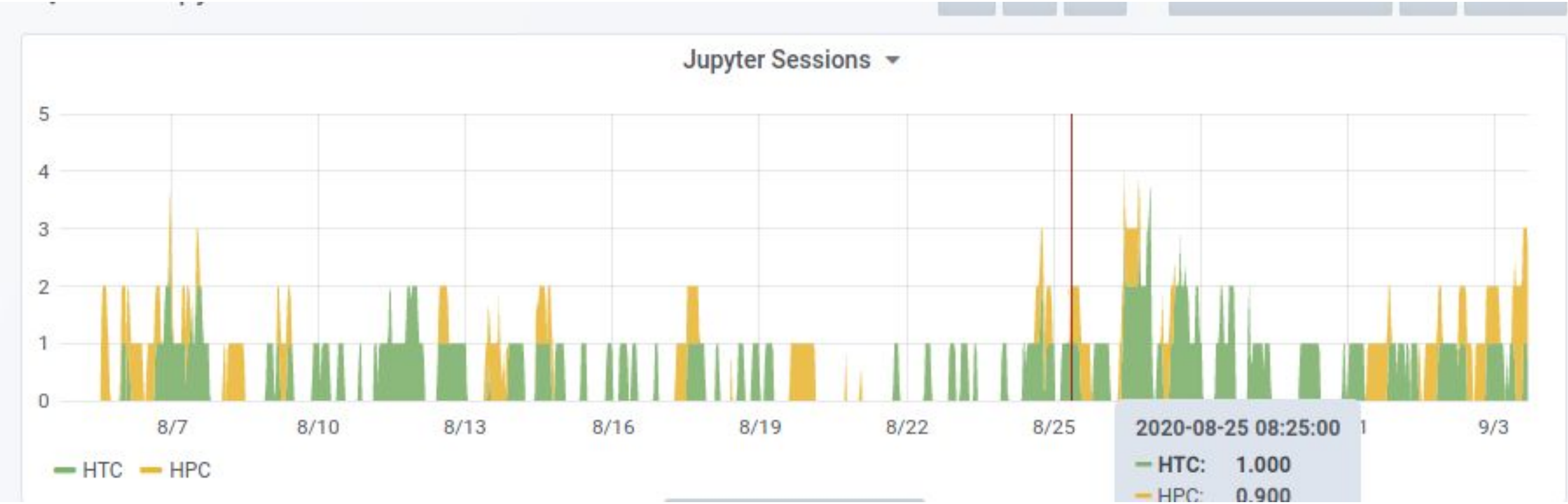
- ✓ slac-ml/20200227.0
- slac-ml/20200211.0
- scdms/allkernels
- atlas-jupyter/20200502
- atlas-jupyter-w-slurm-cli/20200714
- neutrino-jupyter/ub18.04-cuda10.2-extra-ME
- neutrino-jupyter/ub18.04-cuda10.2-extra-edepsim
- neutrino-jupyter/ub18.04-cuda10.2-extra
- neutrino-jupyter/ub18.04-gpu-ana0-ml-larcv2-lab
- neutrino-jupyter/ub18.04-gpu-ana0-ml-larcv2
- neutrino-jupyter/ub18.04-gpu-ana0-mn
- lsst/r18_1_0
- lsst/r19_0_0
- Custom Singularity Image...

ATLAS Jupyter environments
(users are free to use other Jupyter image)

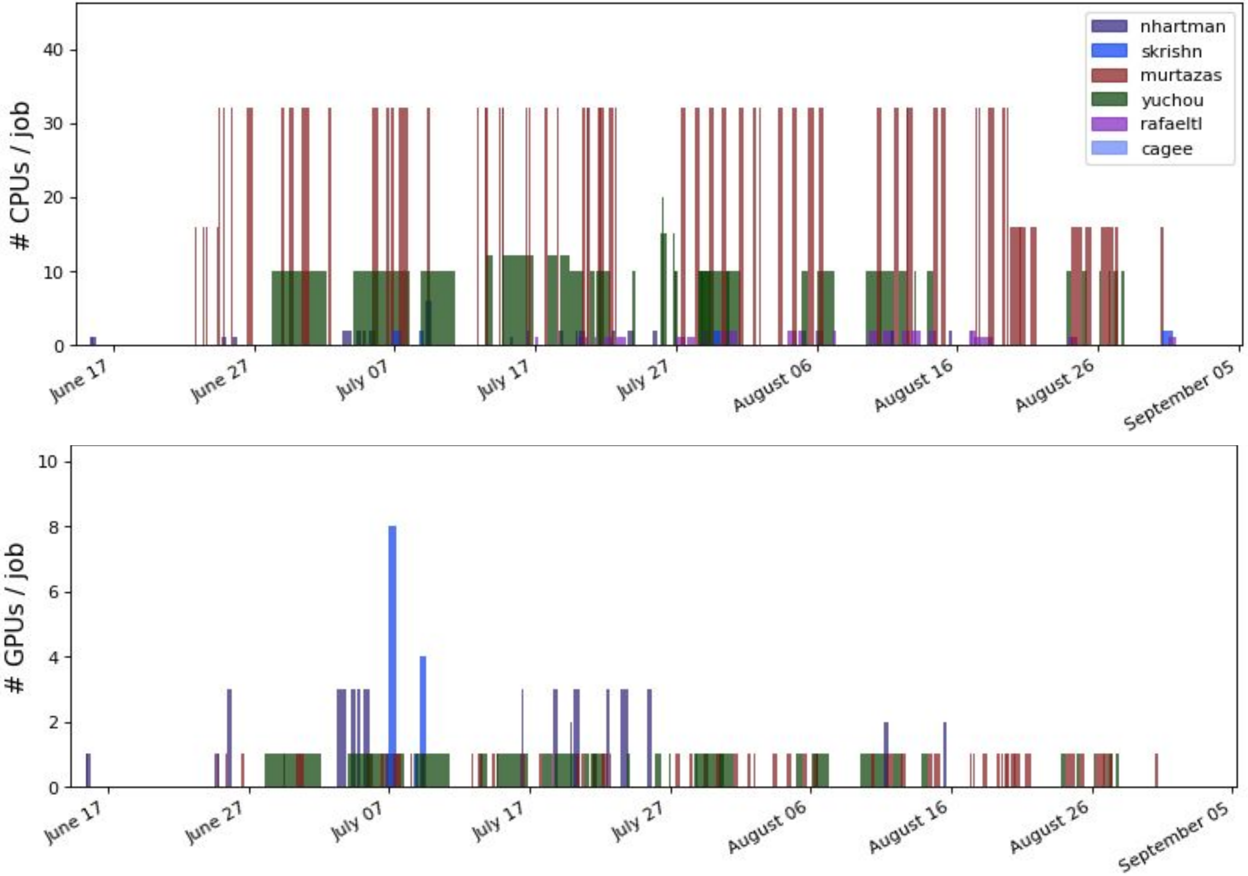
Or bring your own Jupyter environment in Singularity or Conda

Jupyter Sessions at BNL

- More non-GPU jobs than GPU jobs
  - Much analysis done on a few shared jupyter-interactive nodes (HTC)
  - Real CPU usage is quite light for most jupyter work we've seen



ATLAS users' usage pattern at SLAC



From SLAC SLURM accounting data:

- ATLAS Jupyter users tend to ask for a lot of CPUs, and moderate # of GPUs
- Some users really uses it as long, interactive sessions: 4 days

Documentation

https://usatlas.github.io/tier3docs/

- Entry point to public documents for US ATLAS AFs
  - Not just JupyterLab but also account procedures, tutorials, etc.
- Document for BNL and SLAC JupyterLab
- Examples on how to read read xAOD in PyROOT, how to use uproot, etc.
- Examples on how to add additional packages by users themselves
  - E.g. PyCUDA, DASK

Moving forward

- Can easily extrapolates our JupyterLab environment for other experiments
  - Technology choices are minimal and very standard in wider community (cvmfs, JupyterLab, etc...)
  - Kernels do not depend on ATLAS environment to run
- We have a different AF at Univ of Chicago.
  - Offers ML platform to all ATLAS users
  - Lots of experience there to learn
- Mitigate the risk associated with standardization
  - Standardization: after a while, we tend to narrowly focus on just a few “useful” things.
  - Encourage AFs diverse a bit so that we have better chance to catch up with the fast moving python/data science/ML world.
- User feedback is important for our standardized JupyterLab to be successful
  - It is hard to get feedback