Jupyter and dunerun

DUNE SW Architecture

David Adams

BNL

March 18, 2022

Introduction (1)

Motivation

- I have been studying the TPC data from the many DUNE prototypes
- Typically run dataprep to generate plots
  - Event displays, noise vs. channel, signal strength, …
  - Run jobs at FNAL because that is where the data resides
- The examine and share the plots
  - Copy plots back to my laptop and/or
  - Post at https://internal.dunescience.org/people/dladams
    - Requires dune docdb password
- I want to enable others to easily produce these plots
  - Now procedure is very cumbersome:
    - Set up (and often build) DUNE release,
    - Install and set up my analysis packages,
    - Issue commands to process the data,
    - View plots
  - How can we make this easier? →

Introduction (2)

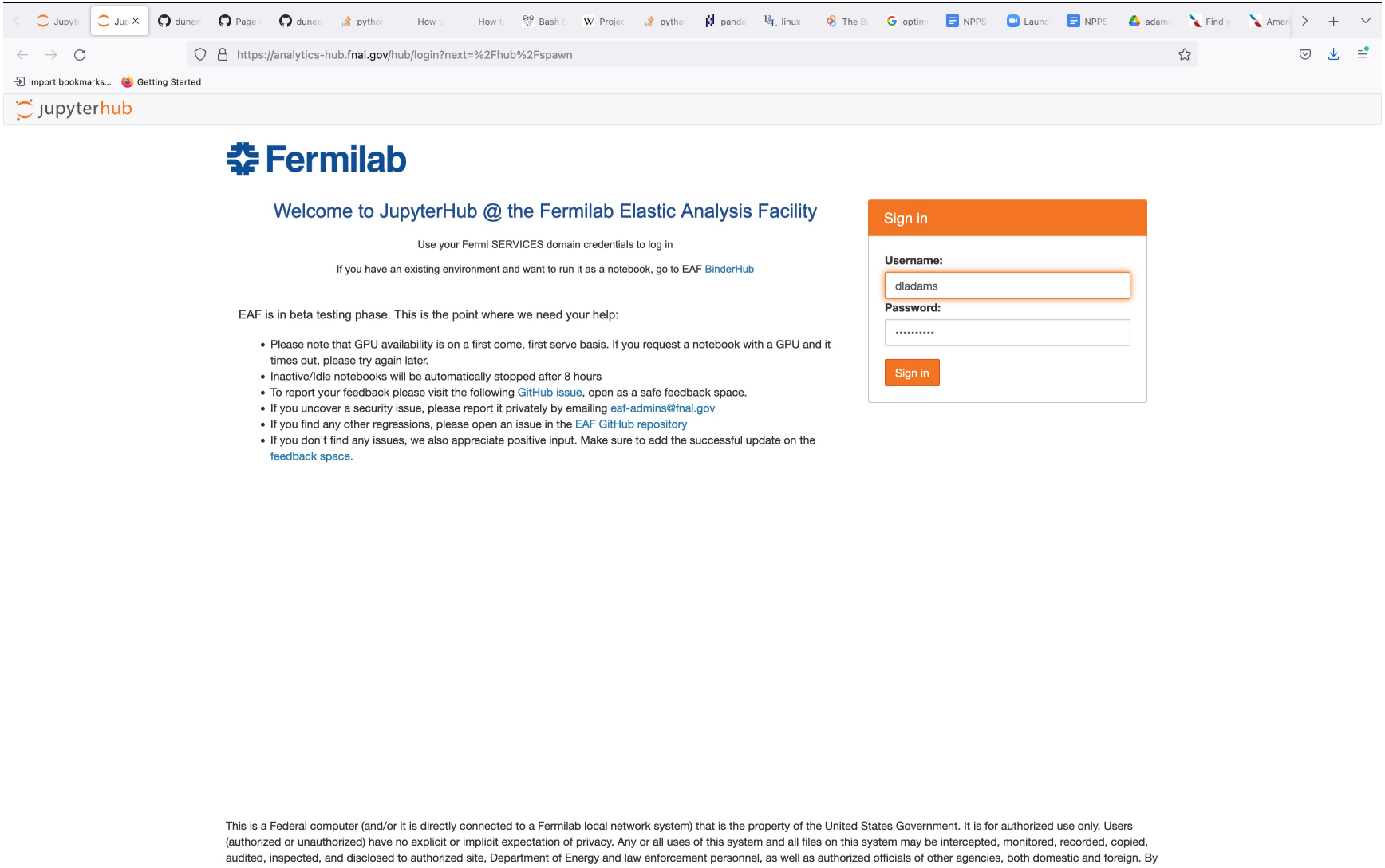
One approach is a Jupyter notebook

- Interactively run a *notebook* on a JupyterLab web service
  - Resulting plots can both appear on the page and be written to local files
  - Easy for user to edit (e.g., change run or event number) and remake plots
  - Service can run close to data with results appearing in laptop browser
- Natural first choice for programming language is Python
  - Somewhat familiar to HEP community
  - Interactive python provides the required inline plotting capability
  - Look at other choices like Root-C++ later

Where to find such a service?

- FNAL provides the EAF "Elastic analysis facility"
  - https://analytics-hub.fnal.gov
  - OS is SL7 (like dunegpvm)
  - User can access code on cvmfs (DUNE SW) and data with xrootd
  - Cannot (yet?) see disks mounted on dunegpvm
    - So I cannot share my build with you

EAF login page



Kernels

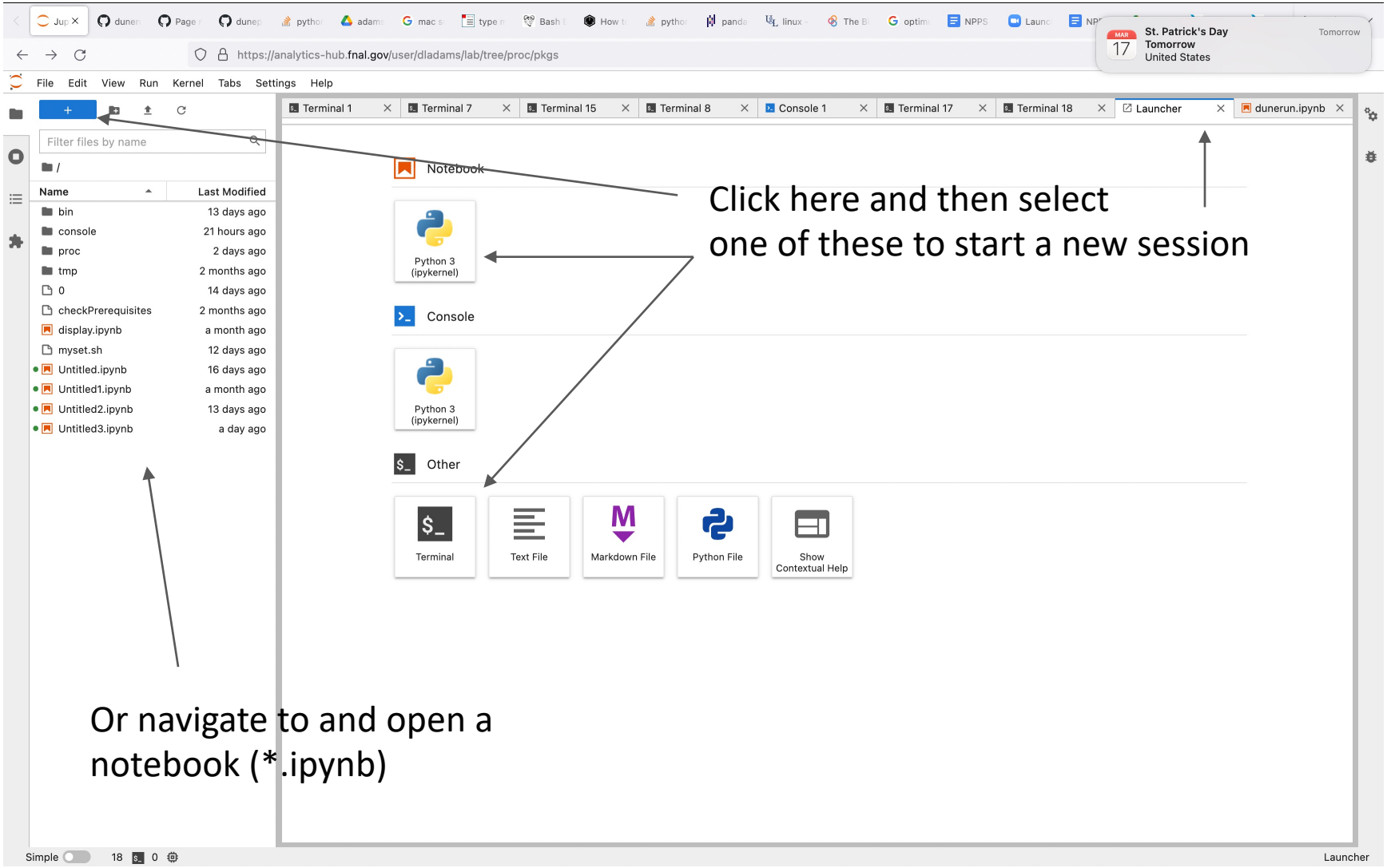
Jupyter server provides a selection of Kernels

- Shell: Python-notebook, Python-console, Terminal, …
- Extensible:
  - Add more shells (Julia, R, Root, …)
  - Add env setup, e.g. set up a DUNE release
- Current EAF suite shown on following page
- DUNE might want to provide shells with recent releases
  - Dune commands (e.g. lar …) immediately available
  - Python version would be that of the release
  - DUNE classes with dicts easily accessible in a Python (or Root) shell

For now I focus on using the Python-notebook kernel

- Recent version of python
- User copies (probably github clone) a notebook of interest

EAF kernels


https://analytics-hub.fnal.gov/user/dladams/lab/tree/proc/pkgs
File Edit View Run Kernel Tabs Settings Help
Filter files by name
/
Name
Last Modified
bin 13 days ago
console 21 hours ago
proc 2 days ago
tmp 2 months ago
0 14 days ago
checkPrerequisites 2 months ago
display.ipynb a month ago
myset.sh 12 days ago
Untitled.ipynb 16 days ago
Untitled1.ipynb a month ago
Untitled2.ipynb 13 days ago
Untitled3.ipynb a day ago
Terminal 1
Terminal 7
Terminal 15
Terminal 8
Console 1
Terminal 17
Terminal 18
Launcher
dunerun.ipynb
Notebook
Python 3 (ipykernel)
Console
Python 3 (ipykernel)
Other
Terminal
Text File
Markdown File
Python File
Show Contextual Help
St. Patrick's Day
Tomorrow
United States
Tomorrow
Click here and then select one of these to start a new session
Or navigate to and open a notebook (*.ipynb)
Simple
18
0
Launcher

My DUNE external packages

I have created many packages to aid my analyses

- In github (many but not all are for DUNE)
- Mostly follow same convention for build/install
  - Clone git repository
  - Run script *build* in the top level directory
    - Installation uses these env variables
      - » DUNE_INSTALL_DIR – base installation directory
      - » DUNE_BUILD_ DIR – base build directory
    - Actual install/build appends directory with package name to these
      - » Possible to defeat that with another variable
    - If variables include %VERSION%, it is replaced with $DUNESW_VERSION
      - » E.g. v09_45_00_00
      - » Maybe should also add %QUAL%

Today I will talk about two of these

- dunerun – Help set up and used DUNE env from shell and python
- duneproc – Help to run DUNE processing

dunerun

Motivation for dunerun

- Enable easy set up of DUNE SW releases
  - From Linux (SL7, bash) command line
  - From Python, in particular IPython and Jupyter
- So we can create Jupyter notebooks to process data
  - Combine setup, processing and viewing of results on one page
  - Provides convenient means to
    - Obtain the results
    - Document what was done
    - Enable others to reproduce them

duneproc

I have maintained duneproc for many years

- Built on code that evolved in my user area years before that
- Fcl, scripts and code
- For many low-level analyses
  - See the plots I have shown in DRA and other places
- For many prototype detectors
- Many features (too many?)
  - Script duneproc runs lar jobs
    - Data input from user-define datasets
    - Sequence of fcl file names
      - » Can be real files or generated automatically to set parameters, e.g. event numbers, detector regions, plot limits and much more
      - » Run directory constructed from the sequence
  - Facilities for building dataset (not same as SAM)

Demos

Better to show rather than talk

- If you are interested, please try the following sequence
- Let me know if you have any problems

Sequence

- Connect to analysis server, e.g. EAF
- Install dunerun: see the top of the dunerune README
  - Open a terminal for this
- Notebooks
  - dunerun/dunerun
  - dunerun/dunedata
  - duneproc/dqm

Demo today

- I won't repeat all the above
- Show results and maybe repeat running the notebooks

Notebooks and beyond

Notebooks can be useful

- Documentation and demonstration of code and data exploration
- Easy for users to make modifications and rerun
- WYSIWYG make it easy to create a demo

But same features can be limitations for serious analysis

- Don’t want to clutter display with now-familiar code
- Don’t need or want to rerun the whole sequence to look at next event
- For complex code, prefer text editor to WYSIWYG

Develop analysis tools

- Migrate code from notebook to analysis tool
- Call the tool from notebook or console
- This was done with the proxy code dunerun.DuneProxy