

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



Welcome to JupyterHub @ the Fermilab Elastic Analysis Facility

Use your Fermi SERVICES domain credentials to log in

If you have an existing environment and want to run it as a notebook, go to EAF [BinderHub](#)

EAF is in beta testing phase. This is the point where we need your help:

- Please note that GPU availability is on a first come, first serve basis. If you request a notebook with a GPU and it times out, please try again later.
- Inactive/Idle notebooks will be automatically stopped after 8 hours
- To report your feedback please visit the following [GitHub issue](#), open as a safe feedback space.
- If you uncover a security issue, please report it privately by emailing eaf-admins@fnal.gov
- If you find any other regressions, please open an issue in the [EAF GitHub repository](#)
- If you don't find any issues, we also appreciate positive input. Make sure to add the successful update on the [feedback space](#).

Sign in

Username:

Password:

Sign in

This is a Federal computer (and/or it is directly connected to a Fermilab local network system) that is the property of the United States Government. It is for authorized use only. Users (authorized or unauthorized) have no explicit or implicit expectation of privacy. Any or all uses of this system and all files on this system may be intercepted, monitored, recorded, copied, audited, inspected, and disclosed to authorized site, Department of Energy and law enforcement personnel, as well as authorized officials of other agencies, both domestic and foreign. By

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

The screenshot shows the JupyterLab interface with the Launcher view active. The top navigation bar includes 'File', 'Edit', 'View', 'Run', 'Kernel', 'Tabs', 'Settings', and 'Help'. The left sidebar shows a file explorer with a search bar and a list of files and folders. The main area displays the Launcher view with several options: 'Notebook', 'Console', and 'Other'. The 'Notebook' section has two 'Python 3 (ipykernel)' options. The 'Other' section has 'Terminal', 'Text File', 'Markdown File', 'Python File', and 'Show Contextual Help' options. Annotations with arrows point to the '+' button in the top left, the 'Notebook' section, and the 'Python 3 (ipykernel)' options. A text box in the center says 'Click here and then select one of these to start a new session'. Another text box at the bottom left says 'Or navigate to and open a notebook (*.ipynb)'. The bottom status bar shows 'Simple', '18', '0', and 'Launcher'.

Click here and then select one of these to start a new session

Or navigate to and open a notebook (*.ipynb)

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`