JupyterLab at the Saint Petersburg State University

Andrey Erokhin and Andrey Zarochentsev

Saint Petersburg State University, 7/9 Universitetskaya Emb., 199034, St Petersburg, Russia

E-mail: st016885@student.spbu.ru, a.zarochentsev@spbu.ru

Abstract. This paper describes our experience using and extending JupyterLab. We started with a copy of CERN SWAN environment, but now our project evolves independently. A major difference is that we switched from classic Jupyter Notebook to JupyterLab, because our users are more insterested in text editor plus terminal workflow rather than in Notebook workflow. However, like in SWAN, we are still using CVMFS to load Jupyter kernels and other software packages, and EOS to store user home directories.

1. Introduction

Computational-heavy scientific research typically uses batch systems to run code on large clusters. But at least on some stages of scientific computing, the bottleneck isn't the computer processing time, but a scientist's efficiency. Because of this, interactive computation systems with immediate response to code input and data visualisation capabilities became popular.

One of the popular examples of such systems is an open-source project named Jupyter Notebook [1]. As the name suggests, it is a computational notebook — a special kind of software which allows mixing formatted text, equations, inline images and plots with programming code, which is executed by a so-called *kernel* and the results of the execution are also shown inline in the document. There are dosens of programming languages supported by Jupyter Notebook.

Jupyter Notebook is used as part of some other projects, for example, in CERN SWAN,[1] where Notebook was modified for tighter integration with CERNBox [2].

JupyterLab [3] project reuses Jupyter Notebook file format and kernel protocol, but provides a new "paned" web-interface.

2. Jupyter deployment

Jupyter Notebook and JupyterLab are single-user applications. JupyterHub project [4] provides multi-user support: it manages user authentication and spawning of their single-user Jupyter servers. JupyterHub is customizable: one can choose among multiple authentication and spawning methods.

To distribute single-user Jupyter servers over several physical computing nodes we use Kubernetes [5] container orchestrator. By default Kubernetes uses Docker [6] as the container runtime, so we ship Jupyter Notebook and JupyterLab in a Docker image. But Jupyter kernels (except for the default Python one) are not packed into the Jupyter image. Instead, like in SWAN, they are loaded from CVMFS [7] — a read-only filesystem used to deliver software in high-energy physics experiments.

[1] Service for Web based ANalysis https://swan.web.cern.ch/.

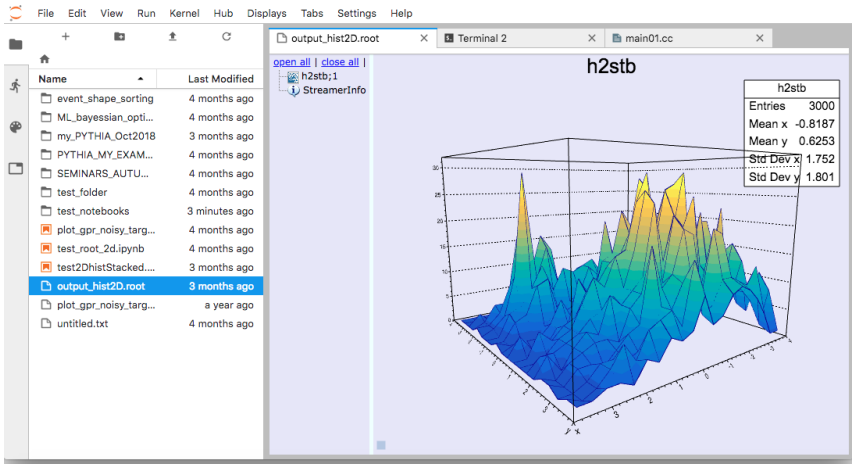


Figure 1. A ROOT file opened in JupyterLab.

The use of CVMFS allows one to keep Docker images smaller and makes it possible to download Jupyter kernels and other software on demand. CVMFS repositories are mounted as subdirectories under the `/cvmfs` directory. The `/cvmfs` directory is shared between containers through the host (Kubernetes node) on which they are running. The automounter daemon is running in the CVMFS container. It monitors accesses to subdirectories under `/cvmfs` and mounts requested repositories.

To store user home directories, EOS [8] is used. It is mounted to user containers and nodes similarly to how CVMFS is mounted, except that there is no automounter daemon involved and EOS is mounted statically.

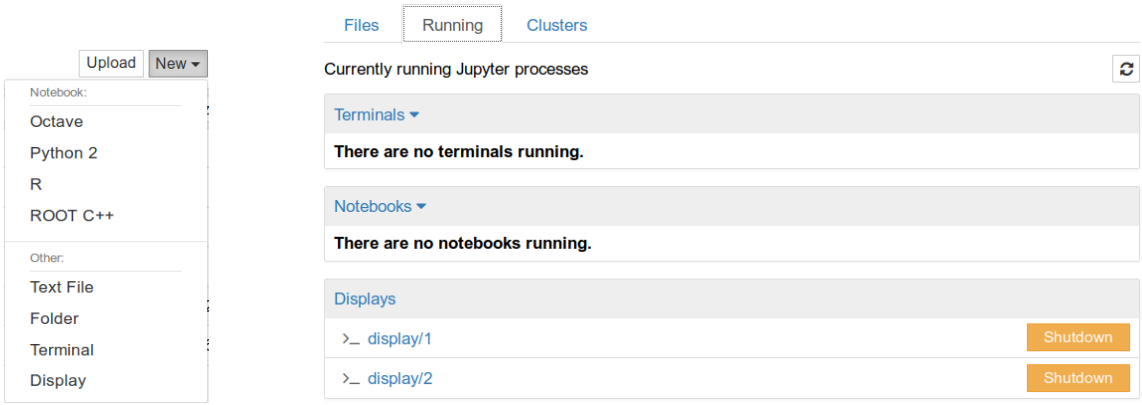
3. Integration of JSROOT

CERN ROOT framework [9] is widely used in high energy and nuclear physics experiments. It is implemented mostly in C++ [10] and supports C++ object serialization into files of a special format (ROOT files). The framework has a special component named ROOT Browser which enables one to inspect ROOT files. A JavaScript library JSROOT [11] makes it possible to show the content of ROOT files in a web browser.

We decided to integrate JSROOT into JupyterLab to provide users a way to view ROOT files not leaving the JupyterLab environment. Our JupyterLab extension was named "jupyterlab_jsroot" [12]. Figure 1 shows a ROOT file opened in JupyterLab.

JSROOT seems to be incompatible with the way extensions are bundled in JupyterLab. The solution is, instead of add the JSROOT code into the extension, to load it in HTML `iframe` and show a ROOT file contents there.

Mime Rendering extensions (extensions which define how to present data in specific format such as PNG images or, in our case, ROOT serialization format) in JupyterLab do not receive a file name to render, because data could come not from a file, but be embedded into a notebook. Instead, the file content in `base64` encoding is provided to a Mime Rendering extension. JSROOT, in turn, expects an URL to a file, so the `base64` representation of a file has to be converted to a binary array and a JavaScript Blob object has to be created from this array. Then a special kind of URL (object URL) representing the Blob object is provided to JSROOT.



(a) The "Display" button triggers spawn of a new Xpra session

(b) List of running Xpra sessions ("Displays")

Figure 2. Parts of the Jupyter Notebook interface modified by our extension.

4. Jupyter and Xpra

In Jupyter Notebook (or JupyterLab), a user can run notebooks, edit text files, use remote terminal. But sometimes possibility to run GUI applications is required. There is no built-in support for this in Jupyter. `nbnonvc` Jupyter Notebook extension [13], proxying a VNC session using `nbserverproxy` extension [14], is a project providing the possibility. However, instead of using `nbnonvc` extension, it was decided to implement own extension proxying Xpra [15]. The `nbserverproxy` extension[2] was forked and rewritten to increase flexibility which allowed, for example, to proxy Unix domain sockets.

Our extension consists of three logical parts:

(i) server-side code responsible for spawning and tracking Xpra sessions, providing API to spawn and control sessions, notifying about their state change;

(ii) Jupyter Notebook extension adding control buttons to this frontend (see Figure 2);

(iii) JupyterLab extension adding control buttons to it (see Figure 3).

Currently (May 2019), the JupyterLab panel showing running kernels and terminals is not extensible, so the list of running Xpra sessions is shown in the "Displays" submenu in the top JupyterLab menubar. (see top of the Figure 3, the "Displays" submenu is not open).

5. Summary

Web-based access to remotely deployed Jupyter liberates our users from the need to install software on their computers. JupyterLab provides a new interfrace, making some types of workflow easier than in classic Jupyter Notebook. Our "jupyterlab_jsroot" extension allows users to view ROOT files directly in JupyterLab. The Xpra integrating extension hugely enriches the Jupyter usage experience, enabling users to spawn and access remote desktop sessions.

Acknowledgments

This work is supported by the Russian Science Foundation, GRANT 17-72-20045.

[2] `nbserverproxy` was renamed to `jupyter-server-proxy` upstream.

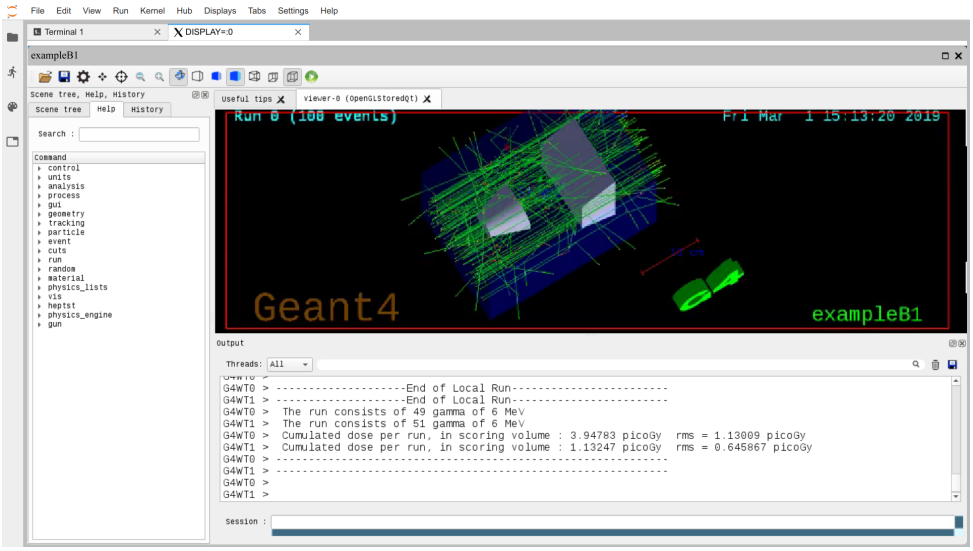


Figure 3. Geant4 [16] running remotely accessed through Xpra HTML5 client integrated into JupyterLab

References

[1] Jupyter Notebook, "Jupyter" [software], version 5.5.0. Available from `https://pypi.org/project/notebook/5.5.0/` [accessed 2019-05-16]

[2] González Labrador H 2016 CERNBox: Petabyte-Scale Cloud Synchronisation and Sharing Platform *Bachelor Thesis* `http://doi.org/10.5281/zenodo.58838`

[3] JupyterLab [software], version 0.35.4. Available from `https://pypi.org/project/jupyterlab/0.35.4/` [accessed 2019-05-24]

[4] JupyterHub [software], version 0.9.1. Available from `https://pypi.org/project/jupyterhub/0.9.1/` [accessed 2019-05-28]

[5] Kubernetes [software], version 1.12. Available from `https://kubernetes.io/` [accessed 2019-05-28]

[6] Docker.com, "Docker" [software], version 1.13.1. Available from `https://www.docker.com/` [accessed 2019-05-28]

[7] Jakob Blomer *et al* 2011 *J. Phys.: Conf. Ser.* **331** 042003

[8] AJ Peters *et al* 2015 *J. Phys.: Conf. Ser.* **664** 042042

[9] Rene Brun and Fons Rademakers 1997 *Nucl. Inst. & Meth. in Phys. Res.* A **389** 81–86. See also `http://root.cern.ch/`

[10] ISO/IEC 2017 *ISO/IEC 14882:2017(E) Programming Languages – C++* Geneva, Switzerland: International Organization for Standardization (ISO)

[11] Bertrand Bellenot and Sergey Linev 2015 *J. Phys.: Conf. Ser.* **664** 062033

[12] JupyterLab JSROOT project, "jupyterlab_jsroot" [software], version 0.0.2. Might be available from `https://gitlab.cern.ch/spbu-jupyterhub/jupyterlab_jsroot/tree/0.0.2` [accessed 2019-05-25]

[13] `nbnonvc` [software]. Available from `https://pypi.org/project/nbnovnc/` [accessed 2019-05-28]

[14] `nbserverproxy` [software]. Available from `https://pypi.org/project/nbserverproxy/` [accessed 2019-05-28]

[15] Xpra project, "Xpra" [software], version 2.5. Available from `https://xpra.org/` [accessed 2019-05-28]

[16] Geant4 project, "Geant" [software], version 10.4.2. Loaded from CVMFS.