Sophon: an Extensible Platform for Collaborative Research

Stefano Pigozzi
ste.pigozzi@gmail.com
University of Modena and Reggio Emilia
Modena, Italy, Italy

Francesco Faenza
francesco.faenza@unimore.it
University of Modena and Reggio Emilia
Modena, Italy, Italy

Claudia Canali
claudia.canali@unimore.it
University of Modena and Reggio Emilia
Modena, Italy, Italy

ABSTRACT

In the last few years, the web-based interactive computational environment called Jupyter notebook has been gaining more and more popularity as a platform for collaborative research and data analysis, becoming a de-facto standard among researchers. In this paper we present a first implementation of Sophon, an extensible web platform for collaborative research based on JupyterLab. Our aim is to extend the functionality of JupyterLab and improve its usability by integrating it with Django. In the Sophon project, we integrate the deployment of dockerized JupyterLab instances into a Django web server, creating an extensible, versatile and secure environment, while also being easy to use for researchers of different disciplines.

CCS CONCEPTS

• **Human-centered computing** → **Collaborative and social computing systems and tools**; Synchronous editors; • **Software and its engineering** → Collaboration in software development; • **Applied computing** → Education.

KEYWORDS

Notebook, Jupyter, Collaborative Research, Extensibility

ACM Reference Format:
Stefano Pigozzi, Francesco Faenza, and Claudia Canali. 2022. Sophon: an Extensible Platform for Collaborative Research. In *Practice and Experience in Advanced Research Computing (PEARC '22), July 10–14, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3491418.3535163

1 INTRODUCTION

Computational notebooks allow users to execute their code and show result of its execution on the same document, marking an essential step towards explainability of research output [9], hindered sometimes by poor code quality and practices, which impact the reproducibility of the results [15]. Project Jupyter is a non-profit organization whose goal is to provide "free software, open standards, and web services for interactive computing across all programming languages.". Their leading open-source software, JupyterLab [12], the successor of Jupyter Notebook, is a browser-based tool that brings together code, text, charts, and rich media, widely used in the field of research to provide the results and the details of research processes. Its usage has seen a consistent spike in adoption in the last years, and it is now considered a de-facto standard [11].

From a research institute's point of view, the research workflow consists of many other important aspects besides the main elements related to data analysis and results. As we experienced during inter-disciplinary research projects, such as [6, 7], critical elements for the success of a platform for collaborative research are related to the capability to support an easy and productive collaboration between researchers and a high usability for scientists who are not experts of system administration. With the Sophon project, we try to pose the foundations for a new and more extensible platform for collaborative research based on JupyterLab, providing the extra features responding to the needs of our research group, such as usability, extensibility and controlled access to the various resources, thus taking steps to allow members of inter-disciplinary research groups with little knowledge of system administration to easily use powerful cloud collaborative research tools. To this end, we integrate the deployment of dockerized JupyterLab instances into a Django [2] web server, creating an extensible, versatile, and secure environment for all researchers. In this paper, we present the proposed Sophon platform, focusing the description of the Sophon client-server architecture, the interaction between the main modules and the principal offered and planned features of the platform.

The rest of the paper is organized as follows. Section 2 discusses some related works. Section 3 presents the Sophon collaborative platform, and Section 4 summarizes the conclusions.

2 RELATED WORK

Approaching this project, we first evaluated the already available Jupyter Hub [13], a multi-user version of the notebook designed for companies, classrooms and research labs. The platform already provided the possibility to share access to notebooks and, additionally, many features related to scalability, flexibility and portability.

Since more fine-tuning was needed, we decided to opt for the aforementioned approach to link Jupyter to a Django server. The main drives in adopting the Django web framework were its built-in concept of modularity, its very active and vast community and its soft learning curve [1]. For the same reasons, the Center for Integrated Cyberinfrastructure Research at Indiana University's Pervasive Technology Institute used the Django web framework to develop a frontend for Apache Airavata, simplifying the steps of their specific workflow. Their idea was to expose the capabilities of Airavata in a simplified manner other than allowing customization through Django framework modularity.

In the research community, especially amongst groups using cyber environments, it is common to try to abstract away the complexities of programming and computer science from peers with

PEARC '22, July 10–14, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9161-0/22/07.
https://doi.org/10.1145/3491418.3535163

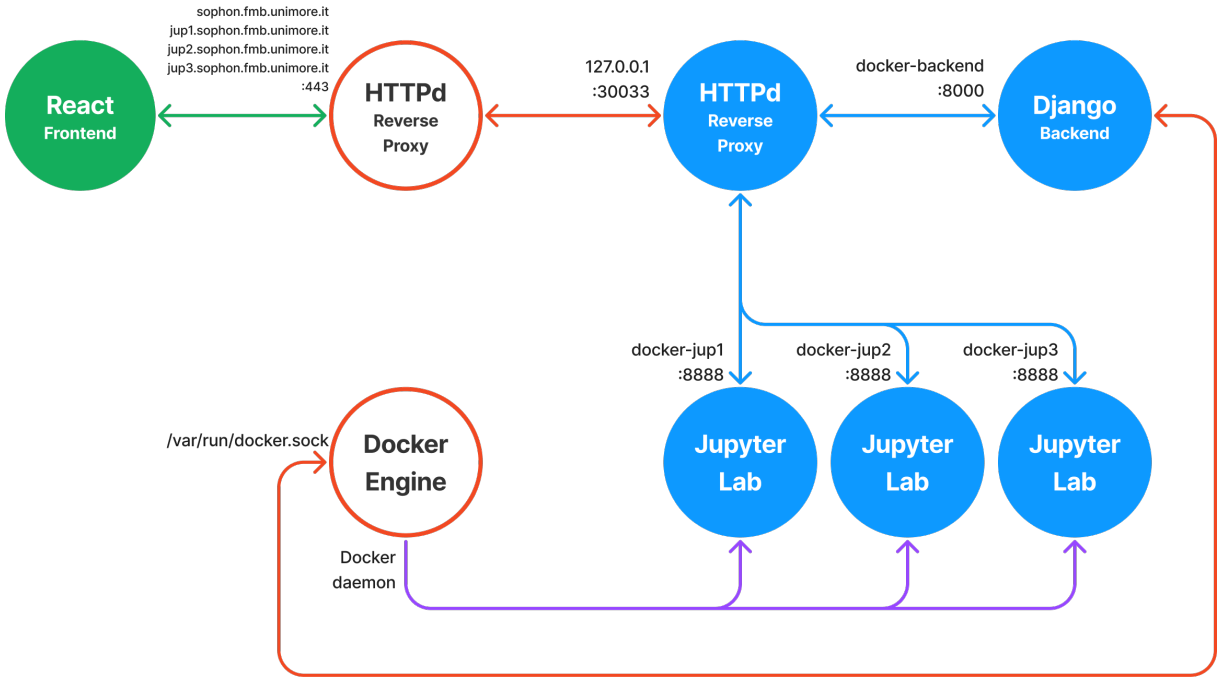

Figure 1: The modules of Sophon and their interactions.

no experience in the field, and to try to integrate platforms, such as JupyterLab, in the pre-existent infrastructure. It is the case of the Texas Advanced Computing Center [5], which successfully integrated Jupyter Hub into their ecosystem, providing advanced computational capabilities to their University researcher. Another case of integration with a pre-existing platform is the Galaxy-Jupyter integration [8]. Biologists commonly use Galaxy, a web-based platform for reproducible computational analysis, and the integration with Jupyter allows them to try new tools without an initial set of programming or scripting skills. While these studies integrate Jupyter in very domain-specific platforms, in this paper we proposed its integration in another general-purpose framework as Django, thus creating a more flexible and versatile instrument that may help researchers working on different disciplines. Moreover, the popularity of Django may help the diffusion and the adoption of the proposed platform.

3 THE SOPHON PLATFORM

In this section, we describe Sophon, a web platform for collaborative research that we developed. We focus on the provided features, and on how these features were implemented. The source code of the project is available on GitHub[1].

3.1 Client-server Architecture

The Sophon platform is based on a client-server architecture, with a Django REST Framework [4] web server acting as a machine-accessible interface, and a React [10] single-page application using the methods of the Django API to display a user-accessible interface.

We decided to use Django for the Sophon backend module because it is widely adopted among the research groups of the Department of Engineering 'Enzo Ferrari' of the University of Modena and Reggio Emilia. Additionally, it has the advantage of being a very popular framework with many plugins and online resources. For the frontend, we decided to use the React framework, because it allows for a smooth user experience in a custom user interface accessible from any user device.

3.2 Jupyter Host

The primary goal of Sophon is to act as a remote host for collaborative computational notebooks, automating the setup of shared research environments.

Sophon can currently launch complete web-accessible Jupyter environments contained in Docker [3] images. This is achieved by having the platform access the Docker Engine management socket, and having it create and destroy specially configured containers that provide their service on the TCP port 8888. To preserve user data between creations and destructions of containers, a uniquely named volume is generated for each container, which is then bound to the data directory used in Jupyter. The port is later locally exposed to a random available port of the host system, allowing an Apache HTTP Server to access it. After the port is exposed, Sophon communicates to the Apache HTTP Server the name and the port of the started container, so it may begin reverse-proxying secure requests incoming from the `NAME.domain.example` subdomain. Finally, Sophon lets the users know the URL where their notebook is available at, allowing them to begin editing.

[1] https://github.com/Steffo99/sophon

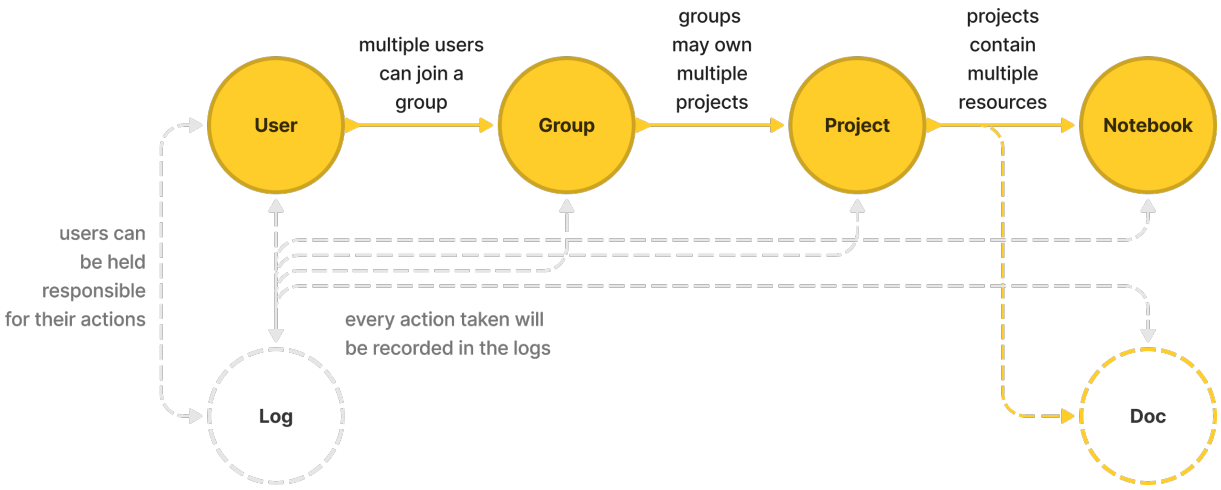

Figure 2: The entities of Sophon and the relations among them. Entities represented with dashed lines are planned, but have not been implemented yet.

3.3 Interactions between modules

The interaction between the Sophon modules is shown in Figure 1. The user mainly interacts with Sophon through the frontend module, a React single-page application that sends HTTP requests to the proxy module. The proxy module inspects those requests, and uses their `Origin` HTTP header to forward them to either the backend module or one of the created Jupyter environments. The backend module responds to the requests it receives by returning data to the frontend and optionally by performing technical operations, such as starting and stopping Jupyter containers.

3.4 Simultaneous Jupyter editing

The Jupyter Docker images included with Sophon enable by default the experimental *Real Time Collaboration* [14] Jupyter feature, which allows multiple users to work simultaneously on the same notebook file. With the RTC feature enabled, edits on a cell by an user are automatically synchronized to all others, instead of prompting everyone to reload the file, allowing for a smoother collaboration experience.

3.5 Groups and projects

Sophon organizes its Jupyter notebooks in *research projects*, collections of items related to a specific research. Users are instead organized in *research groups*, collections of users working together on certain projects. Each project belongs to a research group, and has its own title, rich text description, and a visibility setting of either *public*, visible only to registered *internal* Sophon users, or *private* and visible only to the members of its group. Like projects, groups have their own title and description, and can be set to either allow any registered user to join the group, or to restrict access only to specific users chosen by the group creator. This structure is used by many similar "hub" websites, such as GitLab and GitHub, and should provide sufficient versatility for most of the users' use-cases, since it is similar to how academic groups operate in Italy: each department (30-100 people) has its own computing infrastructure, which is shared between small groups (2-20 people) formed by researchers which focus on a single specific field. Figure 2 illustrates how users, groups, projects and notebooks interact with each other.

3.6 Notebook security

To prevent harmful data leaks, Sophon forbids unauthorized access to Jupyter notebooks. Every time a Jupyter Docker image is run, Sophon securely generates a random token, which is passed to the created Docker container as the `JUPYTER_TOKEN` environment variable, and to the user interface of authorized users via an API response. When the `JUPYTER_TOKEN` variable is set, Jupyter prevents access to anybody who does not know its value, thus allowing only access to authorized users; additionally, the user interface automatically performs the authentication step on the Jupyter notebook on behalf of the user by making use of the `?token=` query string parameter.

3.7 Extensibility

Sophon was developed from the start to be easily configurable and extensible, so that the system administrators can tweak it based on their and their department's needs.

Environment variables. By adding functionality to Django's settings system, we were able to allow configuration of the backend via environment variables, so that they can be easily set when running Sophon via Docker. Configurable options include selecting the HTTP origins from which authenticated API requests can come from, choosing a database engine from the ones supported by Django, changing the default authentication system, and tweaking how Sophon uses Docker.

Django apps. Sophon's functionality can be changed via Django's pluggable apps system: the abstractions of groups, projects and notebooks can be used and modified by external apps which may provide new features based on them. Additionally, the builtin apps can be selectively disabled, in case an administrator intends to use

only certain features, such as the group-project structure without enabling the Jupyter notebooks.

Docker images. Sophon can be modified to allow starting different Docker images of applications providing a web interface, effectively providing Docker-containers-as-a-service. Currently, all images require for their web service to be exposed on port 8888, and for authentication to be verified via the previously mentioned token system.

3.8 Future ambitions

Our ambition is to disseminate the platform beyond our immediate collaborators. To this aim, we will use Sophon in multidisciplinary groups, in the context of national and international projects and of university doctorate programmes. Moreover, we plan to develop some Sophon extensions; in this subsection we describe the planned features and the benefits the users would gain.

Documents. A new abstraction, called *document*, could be created, which would allow users to attach long-form text posts to their groups or projects and make them available to the public or to a superset of Sophon users. This would provide a secure channel for official communication from research groups to their audience, sponsors or fellow researchers. Additionally, a comment system could be implemented on top of these documents, allowing for two-way communication and gathering of immediate feedback. Depending on the features implemented on these documents, this might range from trivial to quite challenging to design and implement. Figure 2 shows the hypothetical relationship between documents ("doc") and the rest of Sophon's entities.

Full Text Search. A full text search system could be implemented on top of all abstractions and on top of the notebook's contents, allowing users to find to easier find the resource they were looking for. Accessing the contents of a notebook for indexing could be a challenge, as they are located on a Docker volume that is separate and isolated from the one of the Django backend.

Audit log. Audit logging could be setup on a platform or on an abstraction level, tracing all actions executed on the platform or abstraction respectively. This would allow users to identify those responsible of destructive operations, such as project deletions, or the leaking of sensitive data. A pre-made Django app could be used for this purpose, or a new one could be created specifically, depending on how fine-grained the control on the log should be. Figure 2 makes an example of how logs would interact with the other entities in Sophon.

Federation. Sophon instances could be configured to federate with each other, allowing for controlled interactions between users of separate Sophon installations. Since cross-website federation is a relatively new concept, it may be a complicated feature to design, however, its benefits would be significant to all users, especially if Sophon was to gain some popularity.

4 CONCLUSIONS

JupyterLab has emerged as the actual standard notebook for data scientists. It offers many functions, successfully supporting researchers in communicating their research output. This paper focused on our project of extending its functionality and usability via an integration with Django. The integration provided the ability to fit JupyterLab in the pre-existent research workflow, handling authentication, research groups collaboration, and research project related information, and additionally the extensibility provided by Django's modularity. All the code related to the project is published on the code-sharing platform GitHub to create a community able to provide new suggestions and eventually contribute to the project development. It is worth to underline that the implementation presented in this paper and the code made available is just a first step towards the design and the implementation of flexible and complete platform for collaborative research: as described in the previous section, several technical improvements may be added to the project.

REFERENCES

[1] Marcus Christie, Suresh Marru, Eroma Abeysinghe, Dimuthu Upeksha, Sudhakar Pamidighantam, Stephen Paul Adithela, Eldho Mathulla, Aarushi Bisht, Shivam Rastogi, and Marlon Pierce. 2020. *An Extensible Django-Based Web Portal for Apache Airavata.* Association for Computing Machinery, New York, NY, USA, 160–167.

[2] Django Software Foundation. 2022. *Django Project.* Retrieved May 26, 2022 from https://web.archive.org/web/20220526005242/https://www.djangoproject.com/

[3] Docker Inc. 2022. *Home - Docker.* Retrieved May 27, 2022 from https://web.archive.org/web/20220527151152/https://www.docker.com/

[4] Encode OSS Ltd. 2022. *Django REST Framework.* Retrieved May 26, 2022 from https://web.archive.org/web/20220526005411/https://www.django-rest-framework.org/

[5] Joe Stubbs et al. 2020. *Integrating Jupyter into Research Computing Ecosystems: Challenges and Successes in Architecting JupyterHub for Collaborative Research Computing Ecosystems.* Association for Computing Machinery, New York, NY, USA, 91–98.

[6] Francesco Faenza, Claudia Canali, and Antonella Carbonaro. 2021. ICT Extracurricular Activities: The "Digital Girls" Case Study for the Development of Human Capital. In *Research and Innovation Forum 2021*, Anna Visvizi, Orlando Troisi, and Kawther Saeedi (Eds.). Springer International Publishing, Cham, 193–205.

[7] Francesco Faenza, Claudia Canali, Michele Colajanni, and Antonella Carbonaro. 2021. The Digital Girls Response to Pandemic: Impacts of in Presence and Online Extracurricular Activities on Girls Future Academic Choices. *Education Sciences* 11, 11 (2021). https://doi.org/10.3390/educsci11110715

[8] Björn A. Grüning, Eric Rasche, Boris Rebolledo-Jaramillo, Carl Eberhard, Torsten Houwaart, John Chilton, Nate Coraor, Rolf Backofen, James Taylor, and Anton Nekrutenko. 2017. Jupyter and Galaxy: Easing entry barriers into complex data analyses for biomedical researchers. *PLOS Computational Biology* 13, 5 (05 2017), 1–10.

[9] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Perez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. 2016. Jupyter Notebooks – a publishing format for reproducible computational workflows.

[10] Meta Platforms Inc. 2022. *A JavaScript library for building user interfaces.* Retrieved May 26, 2022 from https://web.archive.org/web/20220526005901/https://reactjs.org/

[11] Jeffrey M. Perkel. 2018. Why Jupyter is data scientists' computational notebook of choice. *Nature* 563 (November 2018), 145–146. 7732.

[12] Project Jupyter. 2022. *Jupyter Lab Documentation.* Retrieved May 26, 2022 from https://web.archive.org/web/20220526005009/https://jupyterlab.readthedocs.io/en/stable/

[13] Project Jupyter. 2022. *JupyterHub.* Retrieved May 27, 2022 from https://web.archive.org/web/20220527150340/https://jupyter.org/hub

[14] Project Jupyter. 2022. *Real Time Collaboration.* Retrieved May 26, 2022 from https://web.archive.org/web/20220526010155/https://jupyterlab.readthedocs.io/en/stable/user/rtc.html

[15] Jiawei Wang, Li Li, and Andreas Zeller. 2020. Better Code, Better Sharing: On the Need of Analyzing Jupyter Notebooks. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results* (Seoul, South Korea) *(ICSE-NIER '20)*. Association for Computing Machinery, New York, NY, USA, 53–56.