JUPYTERHUB AT THE ESS. AN INTERACTIVE PYTHON COMPUTING ENVIRONMENT FOR SCIENTISTS AND ENGINEERS

L. Fernández, R. Andersson, H. Hagenrud, T. Korhonen, E. Laface,
European Spallation Source, ERIC, Lund, Sweden
B. Zupanc, Cosylab, Ljubljana, Slovenia

Abstract

The European Spallation Source [1] will be the world's most powerful neutron source, once its construction is finished. In order to design, build and operate this complex machine many dif ferent software components and frameworks will be needed. One of those is Jupyterhub, a scripting environment for data analysis, scientific computing and physics simulations. Jupyterhub is a multiuser version of the IPython notebook (Jupyter) [2] that can be deployed in a centralized server; provides centralized authentication, centralized deployment, promotes collaboration and provides access to the most advanced libraries for data cleaning and transformation, simulation and statistics. At the Integrated Controls System Division a customized version of Jupyterhub was deployed, providing sandboxed environments to users using Docker [3] containers. Among other characteristics of this installation we can find: clustering, load balancing, A/B testing, Amazon Web Services integration, nbviewer and OpenXAL integration.

INTRODUCTION

During commissioning and operation of the ESS accelerator, physicists and engineers will need to develop a big amount of scripts. Many of those scripts will involve data analysis and in many cases they will make use of physics simulators and emulators of physical devices. The Integrated Controls System Division (ICS) at ESS is making an effort in standardization of the development of such scripts. The goal is to provide a common approach, ESS wide, regarding the development of scripts and the development of data analysis software.

The standardization of the scripting platform will bring many benefits to ESS, such as:

- Keep control on the quality of the software produced by physicists and engineers. The use of a common framework and the use of a centralized storage for scripts will facilitate the integration and the testing of the software produced. Most of the tests will be static analysis of the code and completely automated. ICS will guarantee as well that scripts follow code conventions among other parameters.
- Avoid the proliferation of scripting languages. It will be impossible for ESS to maintain software written in any possible language.
- Setup of a centralized repository for scripts. Having a central storage system for scripts, will make it possible to backup all the software and also to facilitate the integration with the Git version control system provided by ICS.
- It will also bring the possibility of creating a shared space where users can publish their scripts and make them public to the rest of the ESS users and in-kind collaborators.

The scripting environment selected by ICS was Jupyter, in particular the Jupyterhub solution was the one finally deployed. Some languages will be officially supported among others: Python, R and Julia.

JUPYTER AND JUPYTERHUB

Jupyter is an open source initiative for interactive data science and scientific computing. Jupyter is language agnostic and provides support for many different scripting languages. Jupyter Notebook is the tool selected by ICS for scripting standardization. Jupyter Notebook is a web application that will let the user create documents containing code and documentation, such as: equations, plots, videos and text. Among the different domains where Jupyter Notebook has successfully been used is worth to mention: data cleaning, data transformation, numerical simulation, statistical modelling and machine learning. Figure 1 shows an example of a notebook using OpenXAL.

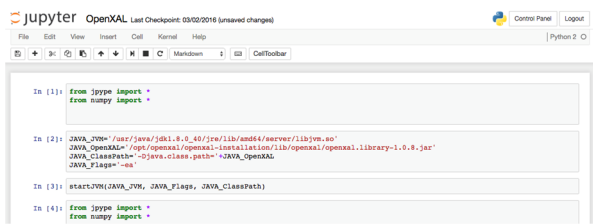

Figure 1: Screenshot of a Jupyter notebook

The use of Jupyter Notebook at ESS will bring the possibility of keeping code and documentation together in the same file. This will be extremely interesting for the scripts used in commissioning and developed not only by ESS staff but also in-kind contributors. It will also be possible to organize effectively all the scripts developed, so they can be easily searched and also audited.

Jupyter and Jupyter Notebook can be run in standalone mode in the user's machine. But, ICS wants to provide that service from a centralized server hosted at the ESS premises. The benefits of such centralization are:

- Provide automatic backups of the code. The user does not have to be worried about the backup strategy; this process will be taken in charge by ICS backing all the notebooks on a daily basis.
- Provide access to the different ICS services and databases. Access to all the ICS ecosystem will be pro-

vided from the central server where the scripts are running. ICS ecosystem involves: naming service, CCDB, access to EPICS, access to online model, and some other small databases and services.

- A centralized place for running scripts makes also easier the process of sharing such scripts among colleagues.
- Having all the scripts running from the same place facilities the monitoring of the software and the management of the scripts. Clean up and some other managerial tasks can be easily automated.

ICS provided in 2015 a custom centralized infrastructure for Jupyter notebooks. This infrastructure was recently moved and ported to the community-supported project Jupyterhub. Jupyterhub is a multiuser version of the Jupyter Notebook designed for centralized deployments. Jupyterhub is a very active project right now and many institutes and universities are intensively collaborating and deploying this solution. The UC Berkeley deployed an extremely interesting solution [4] based on Jupyterhub. ICS built its own setup based on the UC Berkeley approach.

ICS JUPYTERHUB INFRASTRUCTURE

The ICS Jupyterhub is built on top of the infrastructure provided by the IT Division of the ESS. Figure 2 shows an overview of the ICS infrastructure for Jupyterhub. The close collaboration between IT and ICS has been proved to be a key component to the success of this project. The main components of this infrastructure are:

- NGINX proxy server. Running in a Docker container, the main goal of this proxy server is to provide a stable web location to all the users of Jupyterhub independently on where the Jupyterhub server is actually running. ICS provides 2 instances of Jupyterhub running in 2 separate servers. Currently one is used for testing new features and the other one for production. It is intended to have 2 stable instances of Jupyterhub, one in operation and one used as a failover instance. In that case NGINX will manage this failover mechanism as well.
- Jupytherhub servers. Jupyterhub server contains different components [5]. One of them is the Spawner, that is the component in charge of creating the user's jupyter server, that is the environment where the scripts are actually running. These notebook servers are self-contained environments where the users can run their notebooks. ICS uses a modified version of the Swarm spawner [6]. This spawner creates user's notebook servers in Docker containers in different nodes. Creating an insulated environment lets users run the scripts without being affected by other users and without affecting some other user's environments. The Swarm spawner will also manage the creation of containers in different nodes avoiding the overload of such nodes.
- NFS Server and clients. All the notebooks are stored in a centralized filesystem. This filesystem is served by NFS to the different notebook servers running in Docker containers. Having all the notebooks located in a specific server allows us to properly back them up. ICS also provides a separate NFS filesystem used as collaborative area, where notebooks can be published and shared.
- Owncloud. The notebooks, private and shared, are also accessible through an Owncloud [7] server instance. Owncloud is a world-leading solution for sharing files. Users can access their notebooks (and also the shared ones) using the owncloud web interface from any place in the world and from any type of device (laptop, tablet, smartphone). This gives the users flexibility in the way they can access their data.
- Local Docker notebook server. ICS also provides a Docker container for local development that is a copy of the container used by the spawner in the Jupyterhub context. The reason for this container is to provide a local environment that can be installed in the user's machine when the access to the jupyterhub server is not available (see Fig. 3). This becomes extremely useful in combination with the owncloud client. A user can use a local installation and yet use the

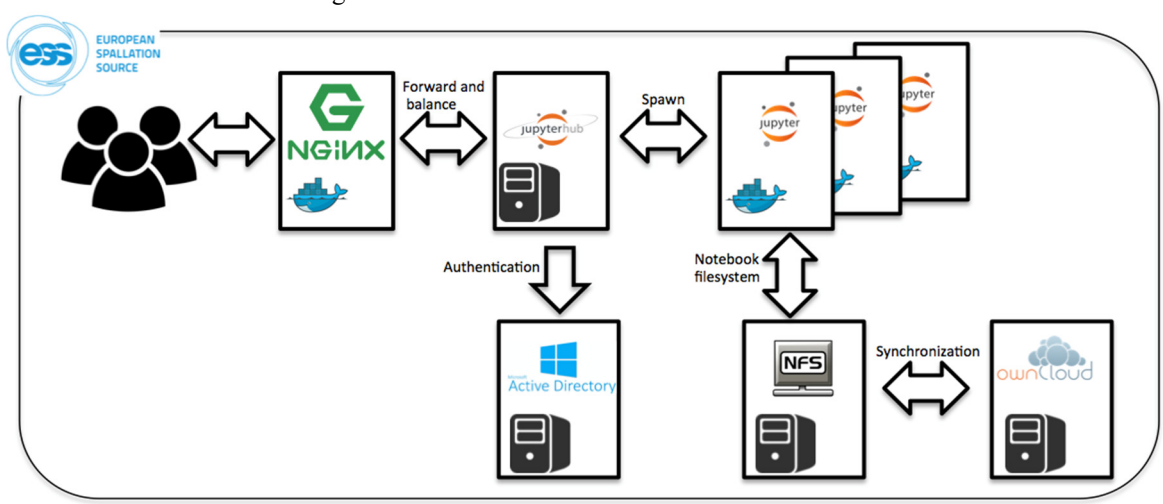

Figure 2: ESS setup for the different components of Jupyterhub

ISBN 978-3-95450-147-2

playbooks that are stored in the ESS central Jupyterhub. The Owncloud client will synchronize local changes with the ESS server ones the connection to the internet is re-established.

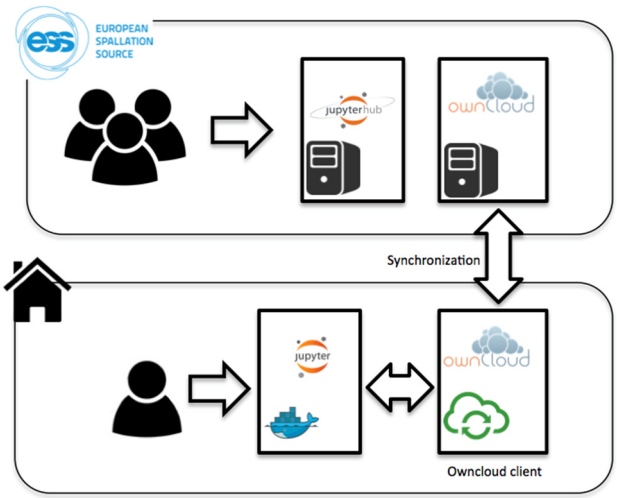

Figure 3: Infrastructure to work without connection to the Jupyterhub instance hosted at ESS

Even if local installation and local development is possible, ICS always encourages development in the centralized Jupyterhub environment. This guarantees access to all the ICS resources.

FUTURE WORK

It is in the plan of ICS to extend the capabilities of the currently installed Jupyterhub environment, among the new features we want to emphasise:

- Use of the Nbviewer [8] project to publish notebooks. Nbviewer is able to render notebooks into HTML pages and other formats. The goal is to setup a web environment where ESS scientists could publish their work.
- Integration of the ICS version control system (Git) in Jupyterhub.
- Integration of parallel computation. Heavy computation and physics simulations may need the use of parallel computing.
- Modification of the Jupyterhub spawner in order to let users decide what Jupyter server container should run. Instead of having a single Docker container used by all the users, they may be able to select the container that better fits their needs.
- Add support for high availability. ICS needs to guarantee a high level of reliability and availability. The infrastructure of Jupyterhub needs to be redesigned to support those levels.
- Monitoring of the whole Jupyterhub infrastructure. It is planned to monitor the infrastructure in 2 levels:
  - Server and service monitoring: integration with PRTG [9] system. This is the monitoring system already used by IT at ESS.
  - Monitoring of logs using Logstash, Elasticsearch and Kibana. This will let us produce statistics, create dashboards that will display important information about the status of Jupyterhub and its components, as well as produce and trigger alarms if something goes wrong.
- Create a physics accelerator related community of Jupyter users. ICS is very interested in finding collaborations with other institutes in order to work on the customization of Jupyter and Jupyterhub for the physics accelerator domain.

REFERENCES

[1] European Spallation Source ESS AB web site: https://europeanspallationsource.se/

[2] Jupyter home page: http://jupyter.org/

[3] Docker web site: https://www.docker.com/

[4] UC Berkeley setup for Jupyterhub: https://developer.rackspace.com/blog/deploying-jupyterhub-for-education/

[5]Description of the different components of Jupyterhub: https://github.com/jupyterhub/jupyterhub

[6] Collections of spawners available in Jupyterhub: https://github.com/jupyterhub/jupyterhub/wiki/Spawners

[7] Owncloud Wikipedia : https://en.wikipedia.org/wiki/OwnCloud

[8] NBViewer project site: https://github.com/jupyter/nbviewer

[9] PRTG web site: https://www.paessler.com/prtg