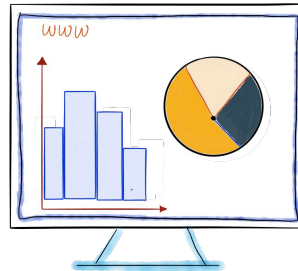


ESCAPE: a view of RUCIO + JupyterLab + ATLAS Open Data integration



Arturo Sánchez Pineda - LAPP

6th April 2021

Overview

An attempt to summarise the activities relative to *a* integration and consolidation of the Data Lake via RUCIO and a friendly web-based UI like JupyterLab, and the efforts to consolidate those in a single entity (container).

And how ATLAS Open Data is used as a Test for such technology and integrations.

In this case, the target audience refers to scientists & advanced users looking for data to perform or reproduce an analysis.

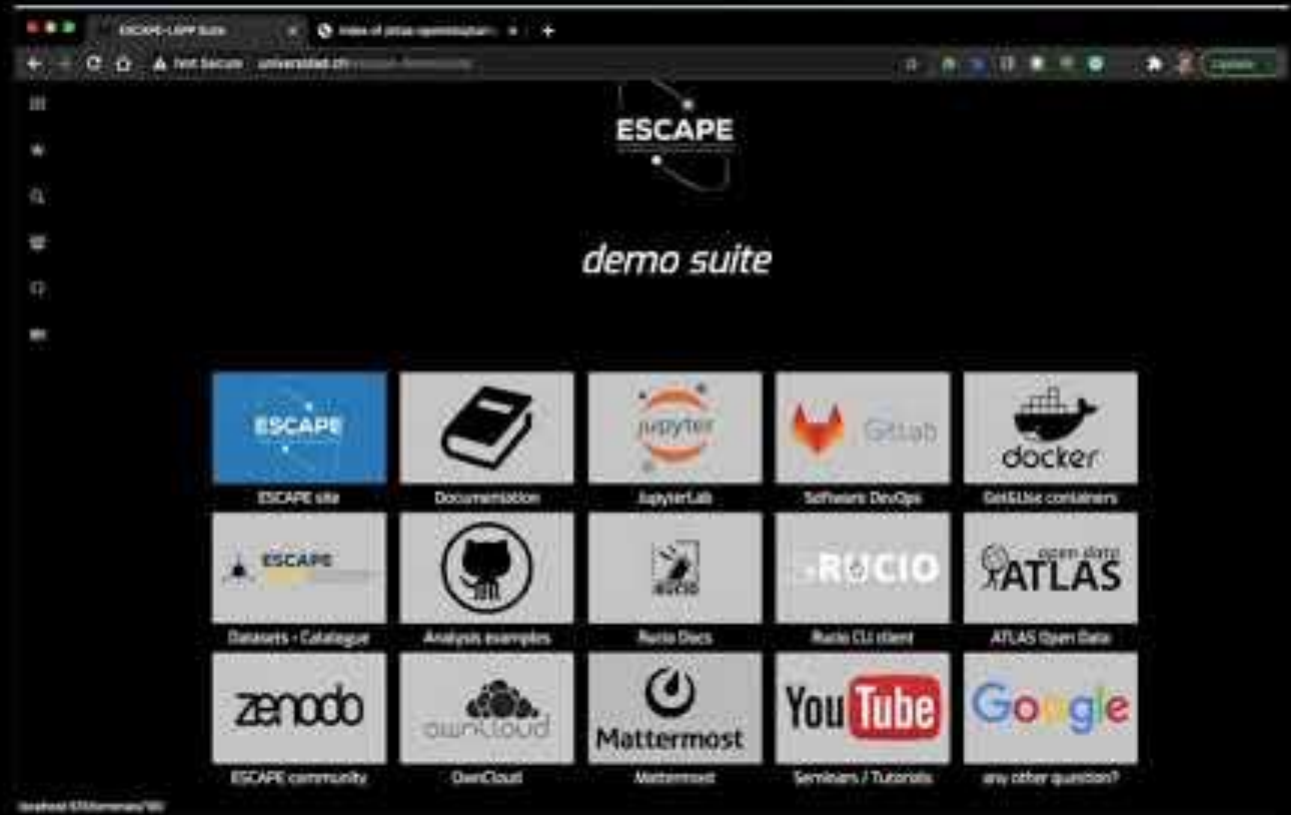
They are/should be aware of the RUCIO as a service, but enjoying the UI and features of a tool as JupyterLab.

My job focuses in integrating and testing the developments of many experts.

RUCIO & JupyterLab (container)

The RUCIO CLI client

(a 90 sec video, mainly
for new users)

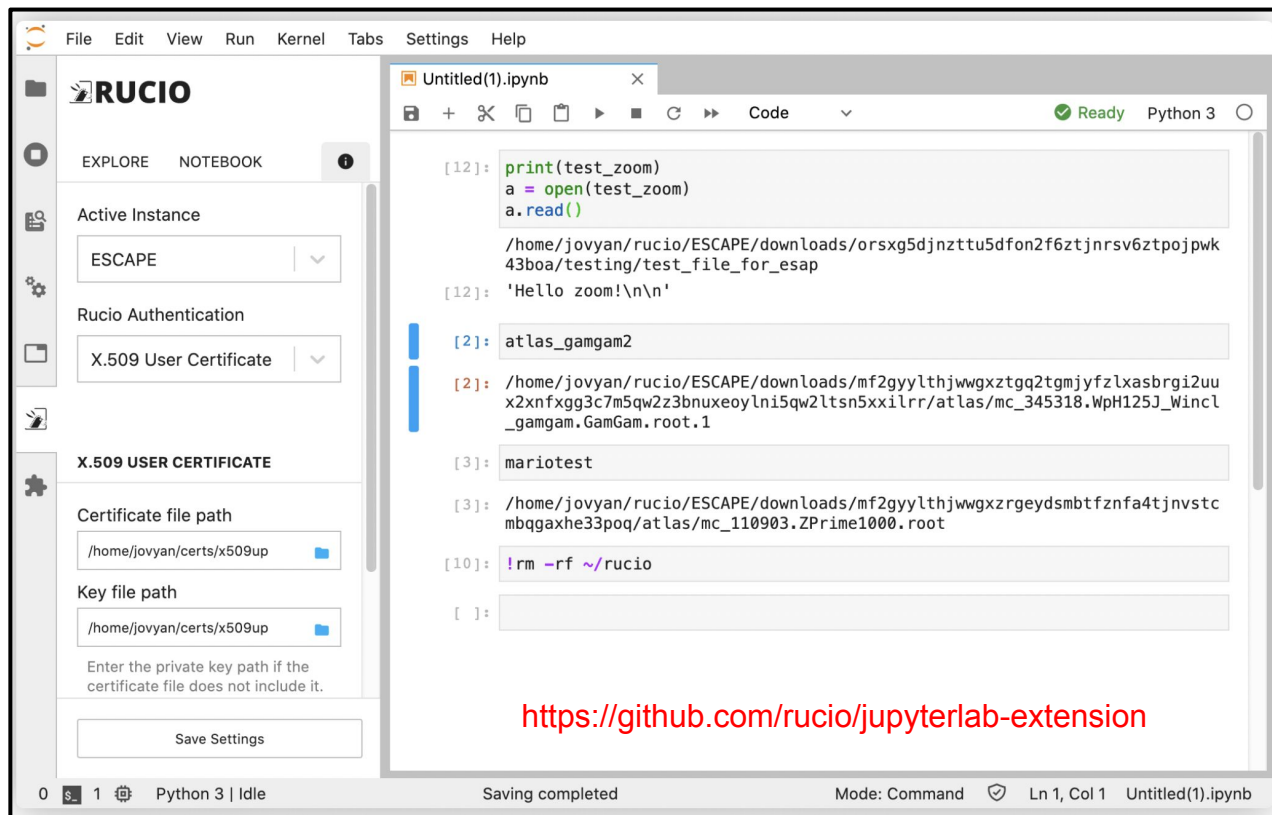


The RUCIO extension for JupyterLab

The JupyterLab RUCIO extension allows to authenticate and interact with the datasets from the web UI.

Making much easier the exploration and analysis of samples in the Data Lake infrastructure.

Slides in the backup for credits!



The screenshot displays the JupyterLab interface with the RUCIO extension installed. On the left, the RUCIO sidebar is visible, showing the 'EXPLORE' and 'NOTEBOOK' tabs. The 'Active Instance' is set to 'ESCAPE', and the 'Rucio Authentication' is configured with 'X.509 User Certificate'. The 'X.509 USER CERTIFICATE' section shows the 'Certificate file path' and 'Key file path' both set to '/home/jovyan/certs/x509up'. A 'Save Settings' button is at the bottom of this section.

The main notebook area shows a Python 3 kernel with the following code and output:

```
[12]: print(test_zoom)
a = open(test_zoom)
a.read()

/home/jovyan/rucio/ESCAPE/downloads/orsxg5djnzttu5dfon2f6ztjnrvs6ztpojpwk43boa/testing/test_file_for_esap

[12]: 'Hello zoom!\n\n'
```

Below the code, there are three output cells:

```
[2]: atlas_gangam2

[2]: /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjwwgxztgq2tgmjyflxasbrgi2uux2xnfxgg3c7m5qw2z3bnuxeoylni5qw2ltsn5xxilrr/atlas/mc_345318.WpH125J_Wincl_gangam.GamGam.root.1

[3]: mariotest

[3]: /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjwwgxzrgeydsmbtfzfnfa4tjnvstcmbqgaxhe33poq/atlas/mc_110903.ZPrime1000.root

[10]: !rm -rf ~/rucio

[ ]:
```

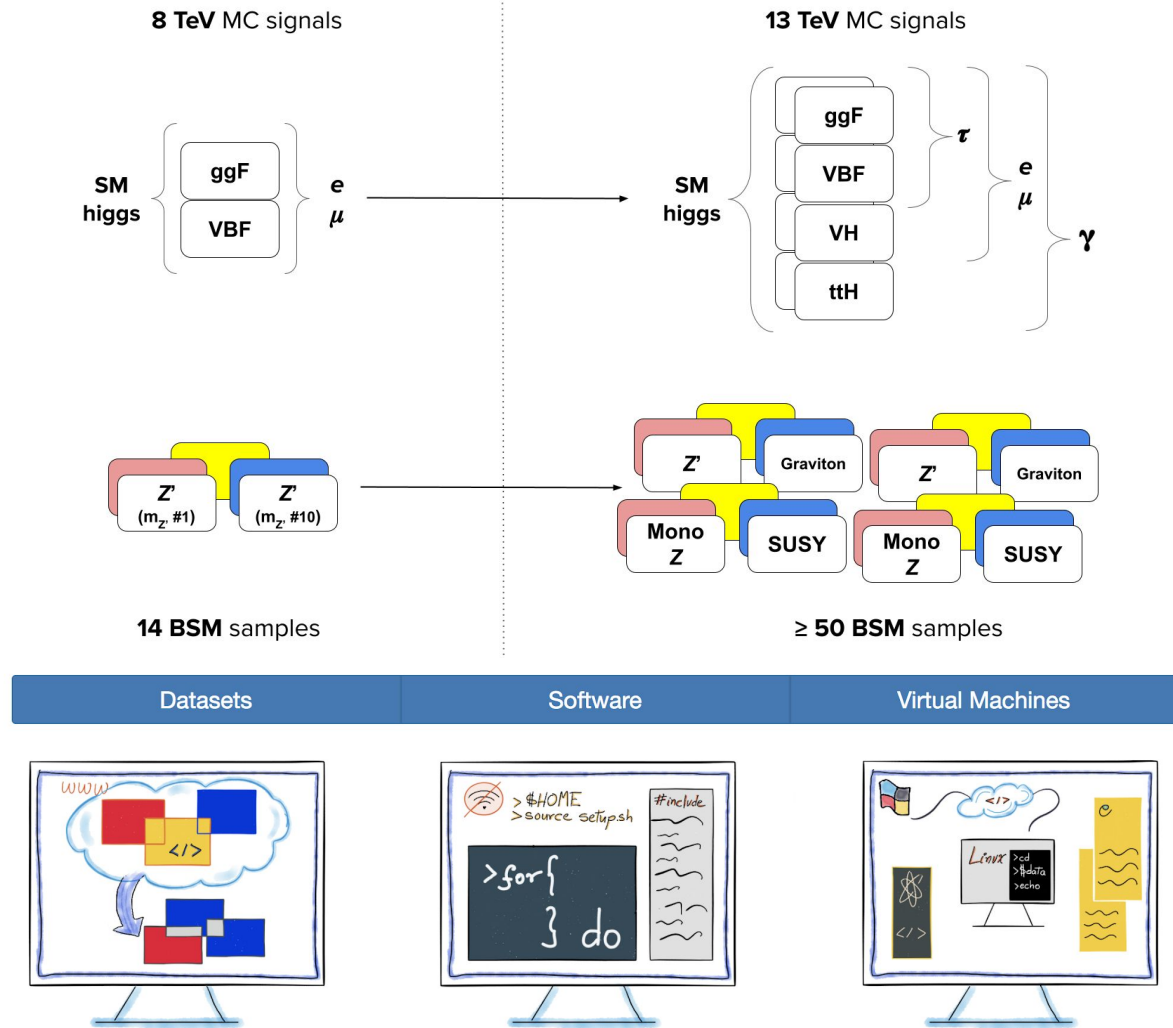
At the bottom of the notebook, the URL <https://github.com/rucio/jupyterlab-extension> is displayed in red text.

ATLAS

Open Data

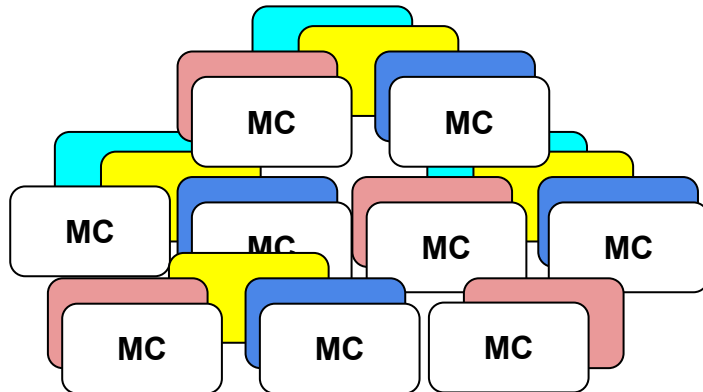
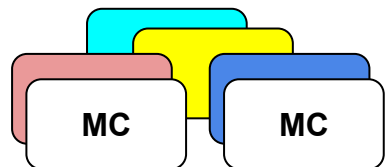
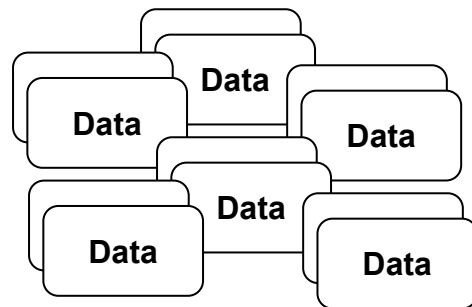
We release and deploy the resources on the Internet. In a nutshell, they are a series of

- Data samples in ROOT n-tuple format
- Framework software and **Jupyter Notebooks** in Python and C++ (kernels) to analyse the samples and produce physics analysis
- JavaScript (JS) applications to produce cut-and-count analysis
- Linux-based Virtual Machines with ROOT-CERN analysis framework and other multiple DevOps tools
- [GitHub](#) & [GitLab](#) repositories
- Web-based documentation sites to present the resources, activities and projects that can be performed



8 TeV release

13 TeV release

1 fb⁻¹10 fb⁻¹

44 samples

940 samples

1+7

Collections
based in
final states

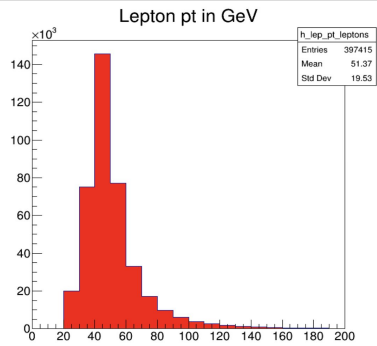

```

ATLAS_OpenData_06-cpp_simple_cut_and_count_analysis_example (unsaved changes)
File Edit View Insert Cell Kernel Help | ROOT Prompt
<< % of total evdotal >> << endlendl;
Total # events = 7500000. Events to run = 750000 corresponding to 10% of total events:

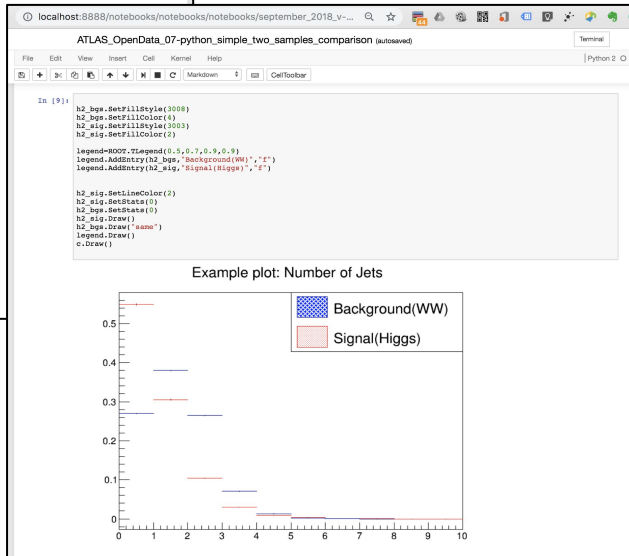
In [10]: for (i=0; i<events_to_run; i++)
{
  nbytes = dataset->GetEntry(i);
  if(lepton_n=1) // Number of leptons in the events has to be at least 2
  {
    if(lepton_type[0] == lepton_type[1]) //Leptons of the same family, i.e. 2 electrons or 2 muons (those are the t
    {
      if(lepton_charge[0] != lepton_charge[1]) // The two selected leptons must have opposite charge
      {
        float lepton_pt_inGeV = lepton_pt[0]/1000.; // The default value in the root file is in MeV, so, we div
        h lep_pt leptons->Fill(lepton_pt_inGeV);
      }
    }
  }
}

In [11]: TCanvas *ca = new TCanvas("ca","ca",10,10,700,700);
h lep_pt leptons->Draw();
ca->Draw();

```



Use
ROOTJS



Jupyter notebooks can run ROOT commands

- We produce a series of examples for training on the usage of the notebooks, reading of the samples and plotting simple analysis.
- The notebooks use both the Python and the C++ ROOT kernel to produce results that can be adjusted by teachers and trainers.
- Also ROOT-independent exercises complement the collection of examples, using upROOT.
- The notebooks can read the samples directly from the Internet (using https protocol) or run locally if present in the machine.

ATLAS Open Data datasets in the Datalake

- ROOT yet need to be importable from a notebook
 - It is deployed for testing in [DockerHub](#)
- Add more datasets to the Datalake
 - All the 13 TeV and 8 TeV ATLAS Open Data samples
 - 16 datasets → 940 samples (ROOT files)
 - < 200 GB
 - Scope used: **ATLAS_OD_EDU** (for **ATLAS Open Data for EDU**cation)
 - Source of the datasets:
<http://opendata.atlas.cern/samples-13tev/> & <http://opendata.atlas.cern/samples-8tev/>
 - Another [set of 10 ROOT files](#) to come (dedicated Jet MC samples) → 1 dataset, ~21 GB.

The pieces together

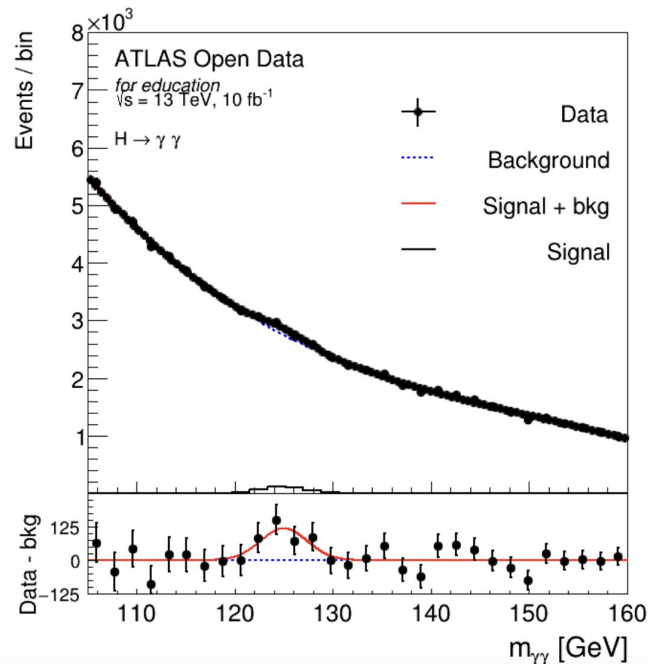
ATLAS Jupyter Notebooks and
JupyterLab RUCIO extension

The ATLAS Open Data as a test field

```
In [8]: for i in range(len(hyy_0)):
        print(i, hyy_0[i])

0 /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjxsqz1smnuxgzk7jb4xs
1 /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjxsqz1smnuxgzk7jb4xs
2 /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjxsqz1smnuxgzk7jb4xs
3 /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjxsqz1smnuxgzk7jb4xs
4 /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjxsqz1smnuxgzk7jb4xs
5 /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjxsqz1smnuxgzk7jb4xs
6 /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjxsqz1smnuxgzk7jb4xs
7 /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjxsqz1smnuxgzk7jb4xs
8 /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjxsqz1smnuxgzk7jb4xs
```

```
In [10]: show_image('histograms/hist_mYY_bin1.png')
```



Once the Open Data datasets are registered in RUCIO they can be downloaded and read, using the JupyterLab extension, including search features

Example in nbviewer.jupyter.org

ATLAS Open Data → C++ examples framework

To run C++ analyses

More computational-complex
particle physics analysis
examples using the existing
publicly available data

More in [Opendata.atlas.cern -
documentation 13 TeV - physics](https://opendata.atlas.cern/documentation/13TeV-physics)

Also use PROOF, adding a
parallel component to the
examples.



SM Higgs boson production in the $H \rightarrow ZZ$ decay channel in the four-lepton final state

Physics analysis examples

Overview of physics analysis examples

Brief introduction to the physics of the Higgs boson

SM W-boson production in the single-lepton final state

Single-top-quark production in the single-lepton final state

Top-quark pair production in the single-lepton final state

SM Z-boson production in the two-lepton final state

SM Higgs boson production in the $H \rightarrow WW$ decay channel in the two-lepton final state

Search for supersymmetric particles in the two-lepton final state

SM diboson production in the three-lepton final state

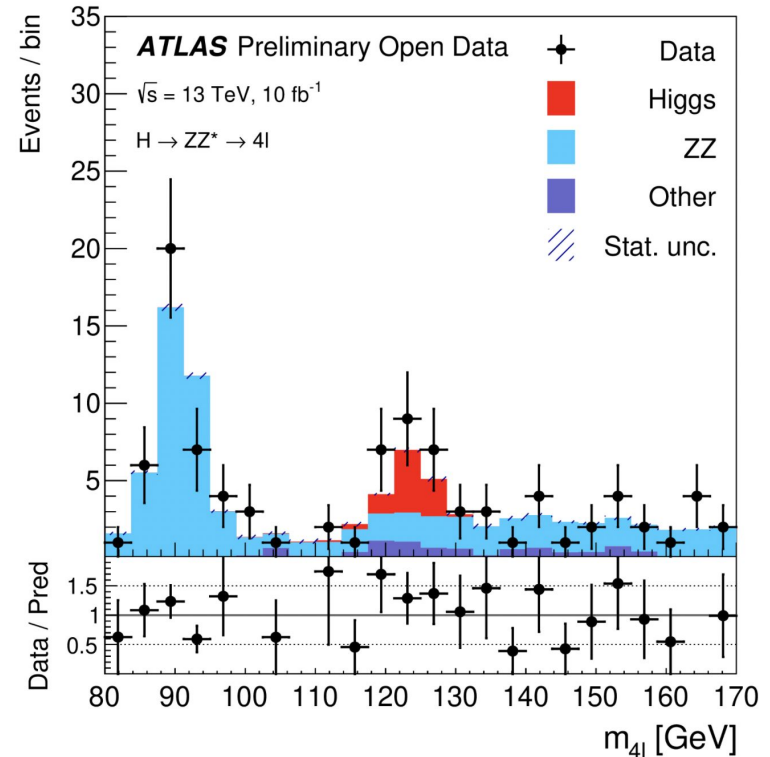
SM ZZ diboson production in the four-lepton final state

SM Higgs boson production in the $H \rightarrow ZZ$ decay channel in the four-lepton final state

SM Z-boson production in the two-tau-lepton final state

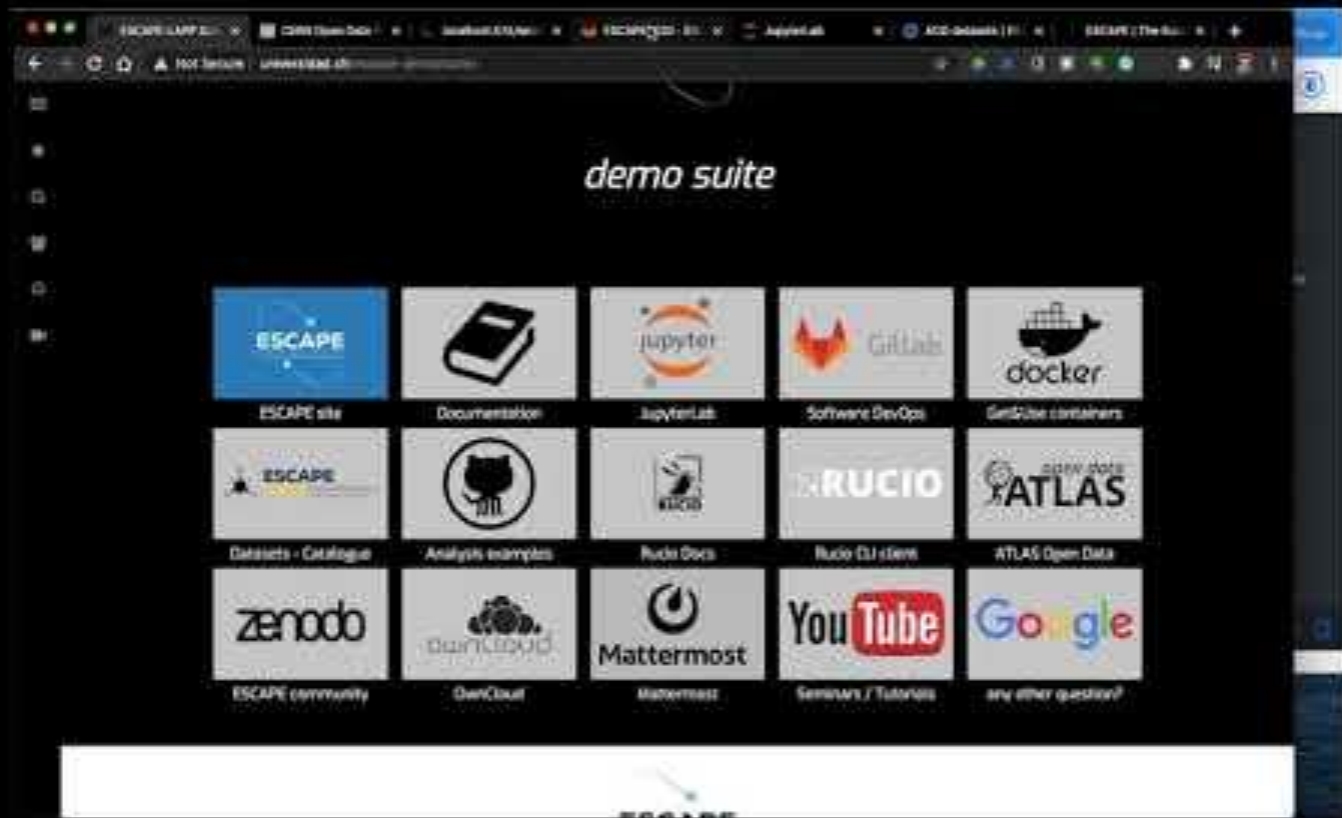
Search for BSM $Z' \rightarrow tt$ in the single-lepton boosted final state

SM Higgs boson production in the $H \rightarrow yy$ decay channel in the



Ongoing
developments
with
**JupyterLab &
RUCIO**
extension

(a 150 sec video)

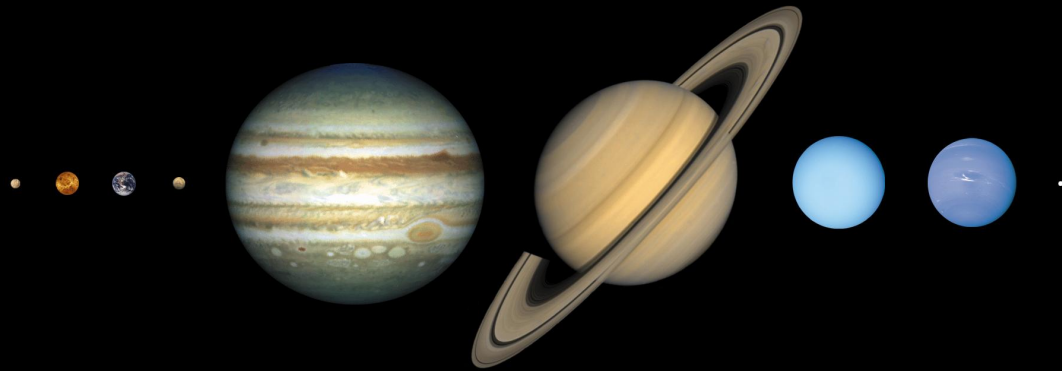


More tools to finish to
integrate in the container, like
more kernels, PROOF, CVMFS

Idea:

**A collection of
containers**

A supported container *system*



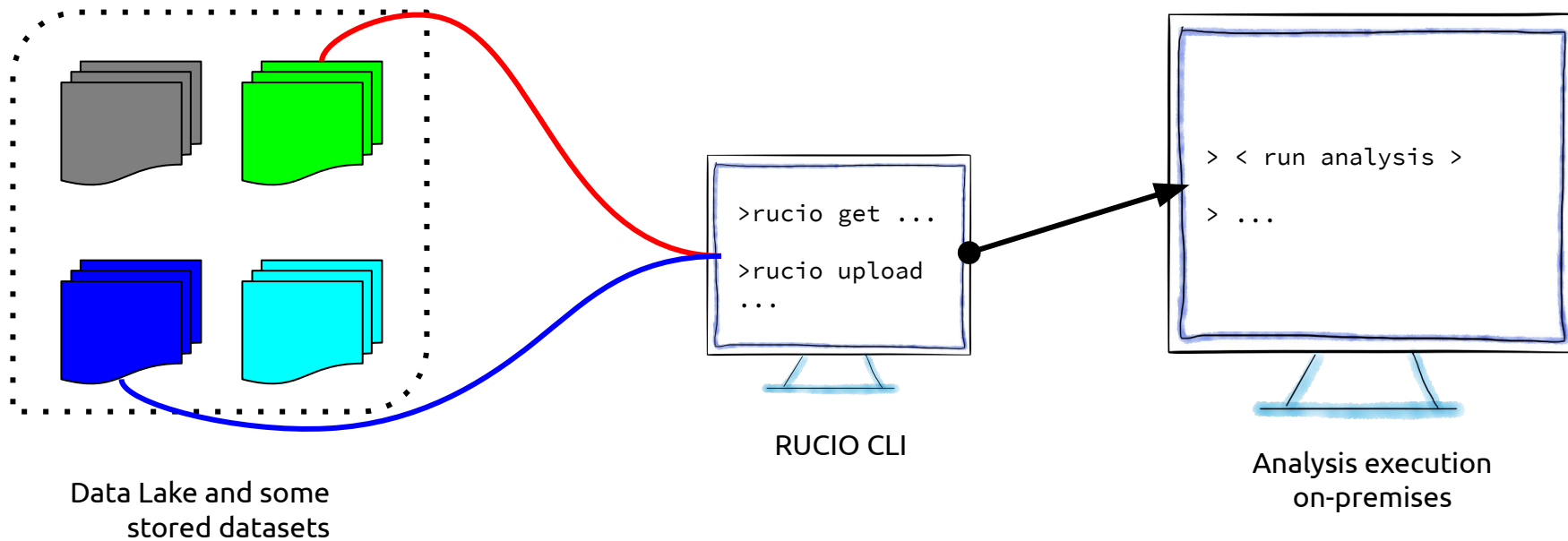
A collection of supported containers will allow maximising the reach of the target audiences while keeping a realistic objective in term of human capital for the creation, maintenance and user's support

- **Mercury** → it is the collection's base container. It has a minimal setup, including JupyterLab and rucio.
- **Venus** → a "hotter" version of Mercury, with standard DevOps software tools.
- **Earth** → The most popular container. Including a series of common HEP tools that *most* users have requested.
- **Mars** → A dedicated HEP container, slimming version of the Earth.
- **Jupyter** → the largest container. It has all the tools. For who want to have it "all".
- **Saturn** → Some experiment's custom version. Same with **Uranus**, **Neptune**?

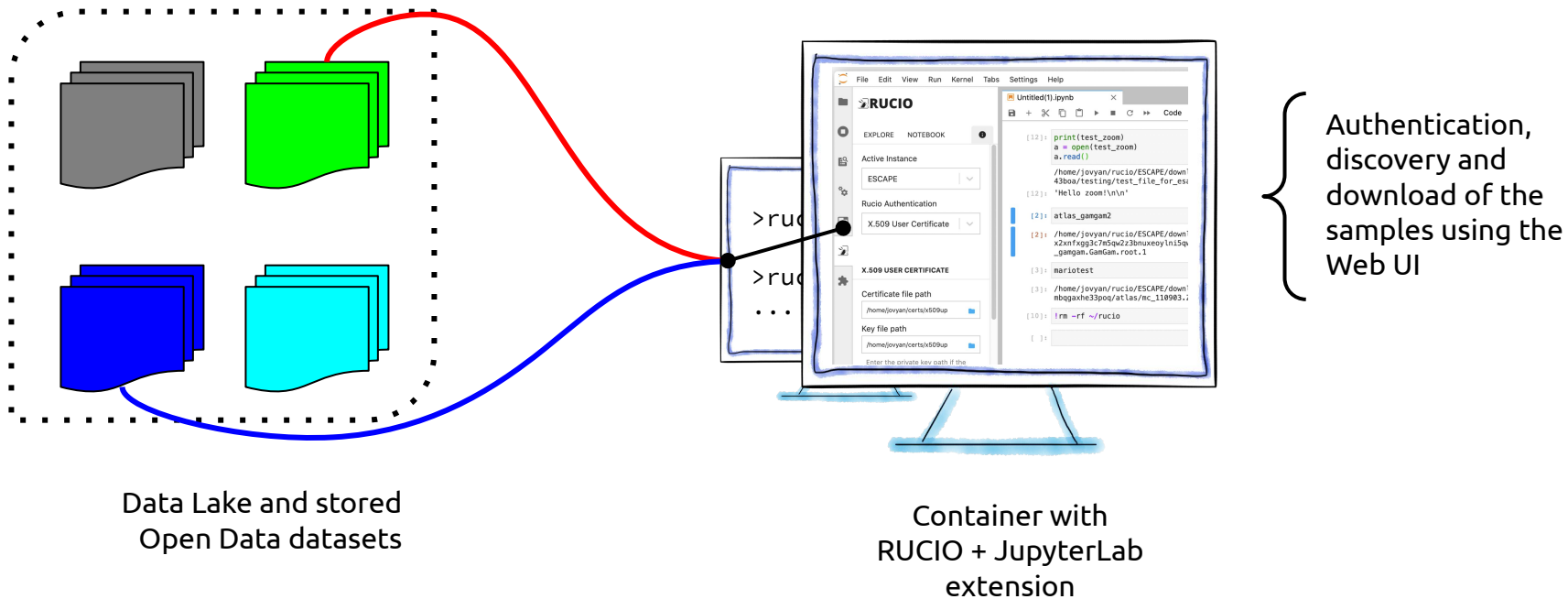
Why the Solar System?

it forces a fix and manageable number :)

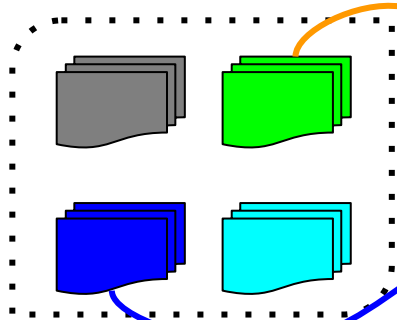
Recap



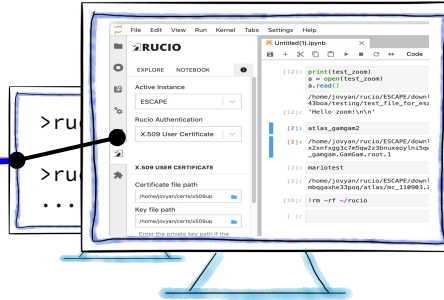
CLI interaction with samples



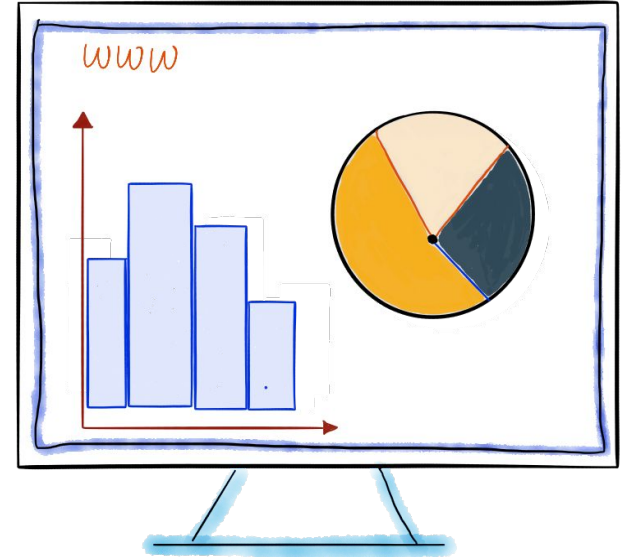
RUCIO+JupyterLab (container) interaction for users



Data Lake and stored
ATLAS Open Data datasets



RUCIO + JupyterLab with ATLAS
open data notebooks for testing



Analysis code, results and
visualisation

A view of the service

Summary

The job now is the testing, consolidation and use of the mentioned resources in a consistent way that resembles a single service + analysis of real experimental data.

More complex data access to be explored, and also workflows.

Backup

The traditional JupyterLab UI

A well-known tool for all of us (data analysis and visualisation) is the Jupyter notebook.

JupyterLab is a suite of tools and features that allow interacting with multiple elements in a single view. And do the computation, of course.

The screenshot displays the JupyterLab interface. On the left, a sidebar shows a file browser with a table of notebooks and files:

Name	Last Modified
Data.ipynb	an hour ago
Fasta.ipynb	a day ago
Julia.ipynb	a day ago
Lorenz.ipynb	seconds ago
R.ipynb	a day ago
iris.csv	a day ago
lightning.json	9 days ago
lorenz.py	3 minutes ago

The main area shows a notebook with the following content:

In this Notebook we explore the Lorenz system of differential equations:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy \end{aligned}$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

In [4]: `from lorenz import solve_lorenz`
`t, x_t = solve_lorenz(N=10)`

The Output View shows a 3D plot of the Lorenz attractor, a complex, swirling trajectory in a 3D space. The plot is rendered with a color gradient from blue to yellow.

Below the plot, there are three interactive sliders for parameters:

- sigma: 10.00
- beta: 2.67
- rho: 28.00

The bottom right pane shows the code for the `lorenz.py` file:

```

9 def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
10     """Plot a solution to the Lorenz differential equations."""
11     fig = plt.figure()
12     ax = fig.add_axes([0, 0, 1, 1], projection='3d')
13     ax.axis('off')
14
15     # prepare the axes limits
16     ax.set_xlim((-25, 25))
17     ax.set_ylim((-35, 35))
18     ax.set_zlim((5, 55))
19
20     def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
21         """Compute the time-derivative of a Lorenz system."""
22         x, y, z = x_y_z
23         return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]
24
25     # Choose random starting points, uniformly distributed from -15 to 15
26     np.random.seed(1)
27     x0 = -15 + 30 * np.random.random((N, 3))
28

```

The JupyterLab RUCIO plugin

In 2020 at CERN, Muhammad Aditya Hilmy created a JupyterLab extension that allows the proper authentication (login/pass or certificate) and access to the datasets in the Data Lake using RUCIO.

More on how it looks like in one of [Muhammad's presentations](#)

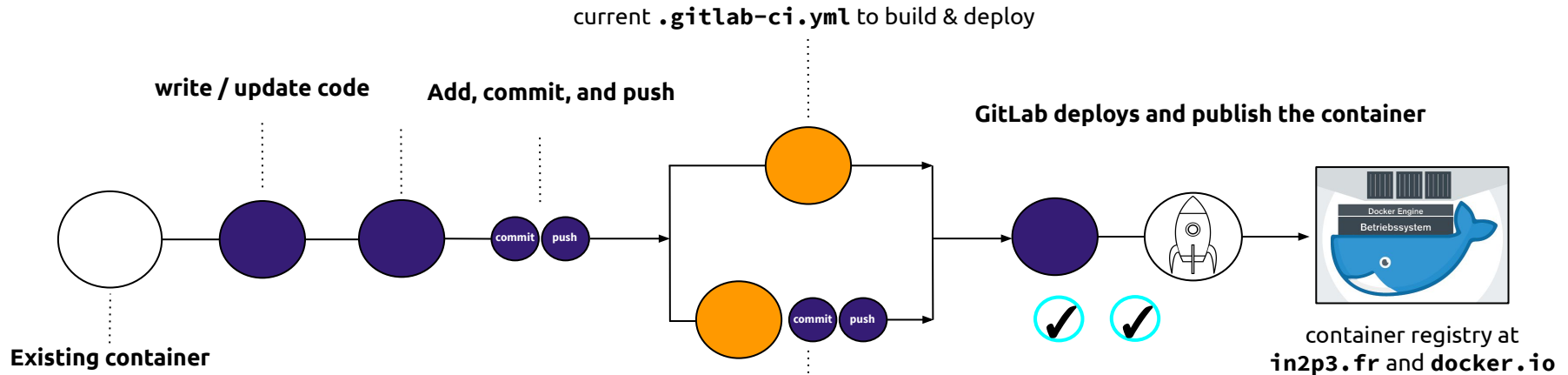
The main idea is to deliver an easy and transparent way to access, download and use datasets replicated in the Data Lake.

It hides all the complexity on that access and allows a seamless usage of the data in a Jupyter notebook analysis.

Work at LAPP

- What I understood until now - any missing info or mistake is mine

A very first view to the current container



updates goes in the `.gitlab-ci.yml`

Container CI / CD

- The series of resources is package in a single container
- The CI setup automatically handles the publication of the container

Frédéric &
Berkay's job

Several developments and deployment already in place

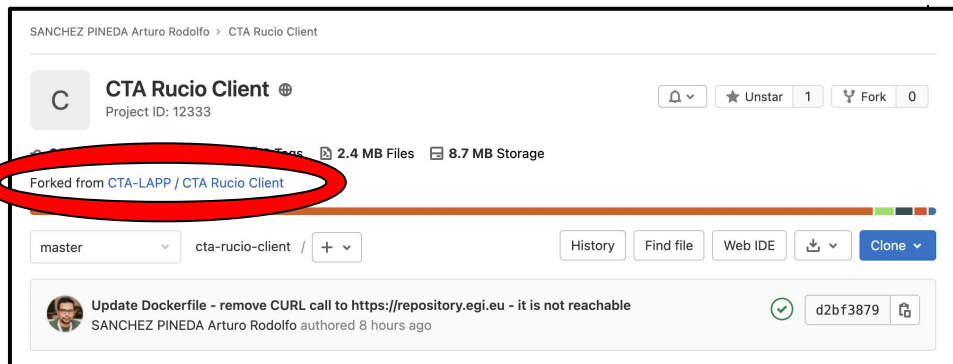
- The compendium of resources includes the current rucio client + JupyterLab + RUCIO extension, proxy & authentication, ...

A very first view to the current container

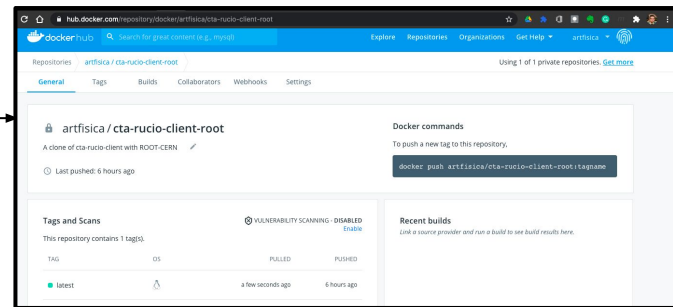
current `.gitlab-ci.yml` to build & deploy

write / update code

Add, commit, and push



GitLab deploys and publish the container



updates goes in the `.gitlab-ci.yml`

Container CI / CD

- The series of resources is package in a single container
- The CI setup automatically handles the publication of the container

Arturo profiting
from Frédéric
& Berkay's job

Several tools and updates added

- Mainly ROOT + some dependencies and extra tools...
- Jupyter conf file to handle the usage of the rucio extension (Muhammad feedback, see later)
- From JupyterLab-3 the widgets are installed using ipywidgets instead of labextension

The screenshot shows the RUCIO web interface on the left and a terminal window on the right. The terminal displays the output of a C++ program named HyyAnalysis.h. The program prompts for options to run data and MC samples, and whether to use PROOF for faster execution. The output shows that 700,000 events were analyzed in total.

```

HyyAnalysis.h Output_HyyAnalysis      main_HyyAnalysis
sh-4.2$ vim main_HyyAnalysis.C
sh-4.2$ ./run.sh
Which option should I run?
Options are:
0 = run all data and MC one after another
1 = run data only (can be run in parallel)
2 = run MC samples only (can be run in parallel)
0
Option is 0
Should I use PROOF? (will make things faster)
Options are:
0 = NO
1 = YES
0
PROOF option is 0
starting ROOT
Info in <TUnixSystem::ACLi>: crea
yAnalysis_C.so
Info in <TUnixSystem::ACLi>: crea
ysis_C.so
Starting analysis with process opt
Analysed a total of: 50000 events
Analysed a total of: 100000 events
Analysed a total of: 150000 events
Analysed a total of: 200000 events
Analysed a total of: 250000 events
Analysed a total of: 300000 events
Analysed a total of: 350000 events
Analysed a total of: 400000 events
Analysed a total of: 450000 events
Analysed a total of: 500000 events
Analysed a total of: 550000 events
Analysed a total of: 600000 events
Analysed a total of: 650000 events
Analysed a total of: 700000 events

```

The C++ code also runs in the container

Caveats: this example reads the samples from the internet, but changing one code-line allows to usage of the same samples from the notebook above.

