

# ROOT Tutorial

HASCO Summer School 2021

Antonio Sidoti

[Antonio.sidoti@bo.infn.it](mailto:Antonio.sidoti@bo.infn.it)

*Istituto Nazionale Fisica Nucleare – Sezione di Bologna*



# Tutorial

For the hands on/tutorial we will use jupyter notebooks.

Two options to avoid overloading the cloud infrastructure:

- ▶ Those who **do not** have a CERN account: Use the Goettingen jupyter cloud <https://jupyter-cloud.gwdg.de/>
- ▶ Those who **have a CERN account**: the SWAN Service at CERN <http://swan.cern.ch/>

A web-based interactive computing platform that combines code, equations, text and visualisations.



Many supported languages: Python, Haskell, Julia... One generally speaks about a “kernel” for a specific language

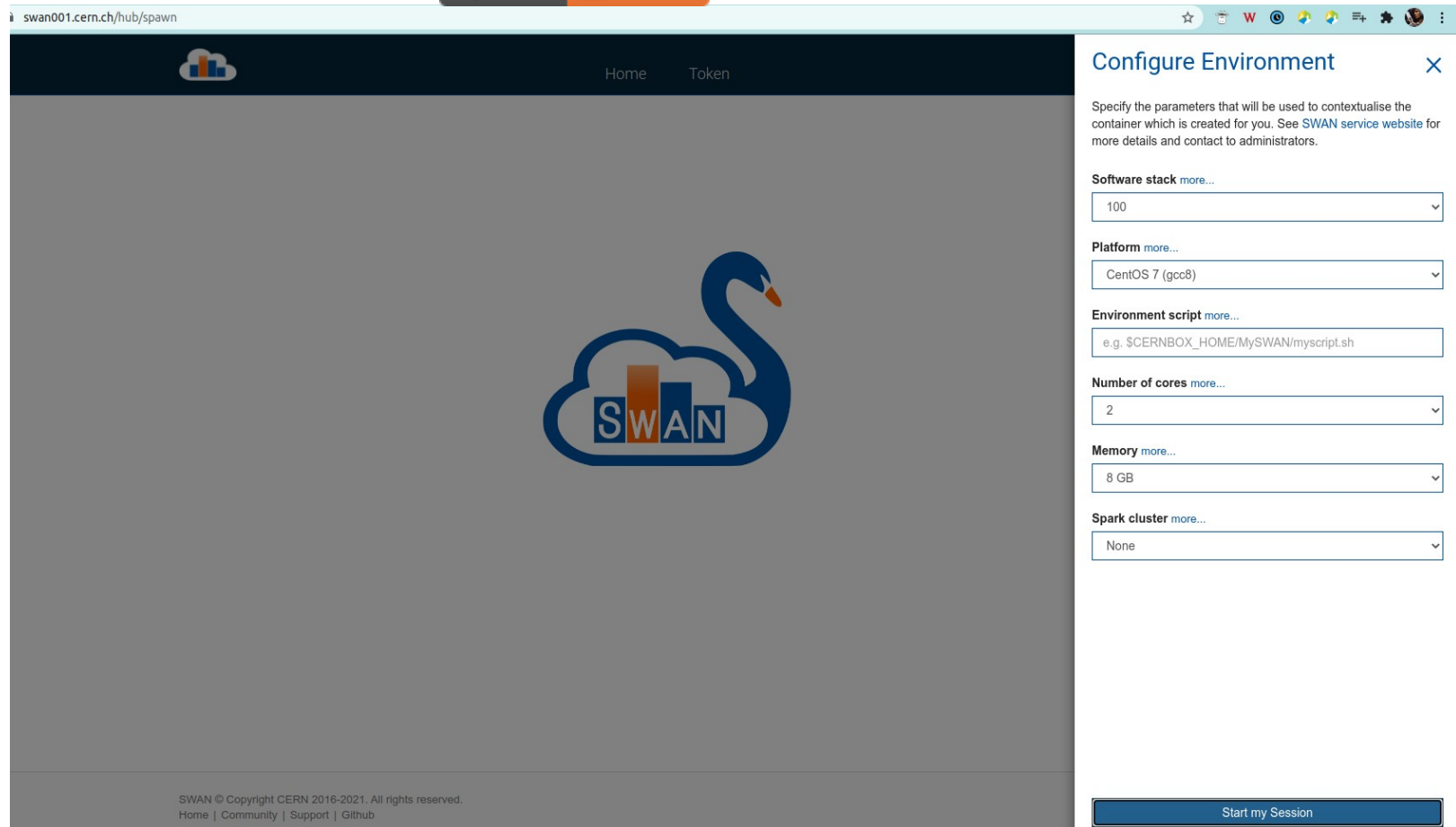
In a nutshell: an “interactive shell opened within the browser”

# Swan

From gitlab: <https://gitlab.cern.ch/chscheul/hasco-2021-root-update>

Click on Swan button

Open in  SWAN



The screenshot shows a web browser window with the URL `swan001.cern.ch/hub/spawn`. The main content area features a large blue swan logo with a bar chart inside its neck, and the word "SWAN" in white letters on a blue background. A dark navigation bar at the top contains a home icon and the text "Home" and "Token".

On the right side, a "Configure Environment" panel is open, containing the following configuration options:

- Software stack** more...: 100
- Platform** more...: CentOS 7 (gcc8)
- Environment script** more...: e.g. \$CERNBOX\_HOME/MySWAN/myscript.sh
- Number of cores** more...: 2
- Memory** more...: 8 GB
- Spark cluster** more...: None

A "Start my Session" button is located at the bottom right of the configuration panel.

At the bottom of the page, the footer text reads: "SWAN © Copyright CERN 2016-2021. All rights reserved. Home | Community | Support | Github".

# Goettingen cloud

<https://jupyter-cloud.gwdg.de/>

You should have already imported files and notebooks.

Otherwise open PYTHON(ROOT) launcher

And load gitlab repo: <https://gitlab.cern.ch/chscheul/hasco-2021-root-update>

The screenshot shows a JupyterLab interface. On the left is a file browser for the directory `/ hasco-2021-hands-on-sessions / root_session /`. It lists several files and folders, with `Exercise_Root_1.ipynb` selected. On the right is a notebook editor with the following content:

## Tutorial Part 1

**Goal** Open a root file and inspect its contents (trees, variables, etc) Quickly plot some variables. Histograms

First import ROOT stuff in your session

```
[ ]: import ROOT
      from ROOT import TCanvas, TF1
```

some black magic to show canvas you've produced

```
[ ]: %jsroot on
```

Open the root file

```
[ ]: f = ROOT.TFile.Open("../data/root_session/mc_147771.Zmumu.root")
```

Now you can inspect it. What is the content of this file? Usually you can visualize it with a TBrowser and you will see something like that.

Below the notebook is a window titled "ROOT Object Browser" showing a file tree with `root` and `PROOF Sessions`.

# Before you start

Make sure you've copied the data files in your home directory.

If not:

On swan run the script:  
copy\_swan.ipynb

On getting cloud the script:  
<https://gitlab.cern.ch/chscheul/hasco-2021-hands-on-sessions/-/blob/master/setup/setup.sh>

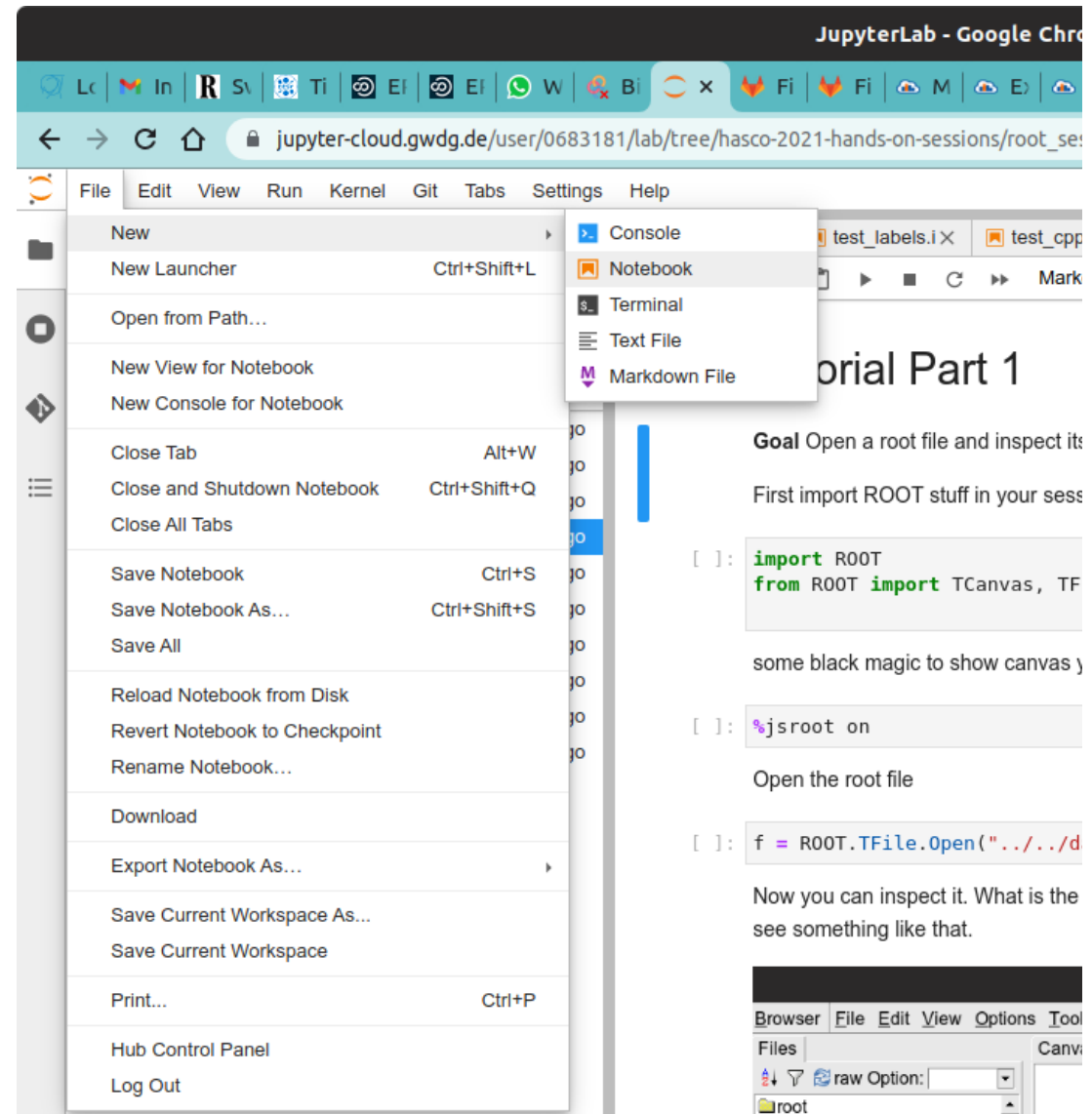
The best way to do the tutorial is to open a new notebook and proceed step by step

Important:

Set the boolean variable if you are working on swan

```
import ROOT
from ROOT import TCanvas, TF1

on_swan = True
```



The screenshot shows the JupyterLab interface in a Google Chrome browser. The 'File' menu is open, and 'New Notebook' is selected. The notebook content includes the following code:

```
[ ]: import ROOT
      from ROOT import TCanvas, TF1

      some black magic to show canvas y

[ ]: %jsroot on

      Open the root file

[ ]: f = ROOT.TFile.Open(".././d

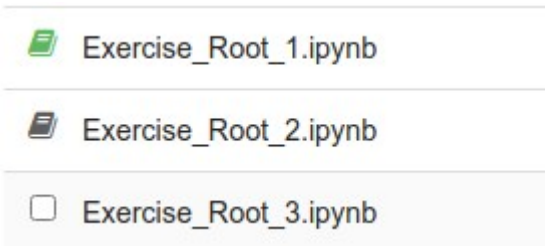
      Now you can inspect it. What is the
      see something like that.
```

- Exercise\_Root\_1.ipynb 18 minutes ago
- Exercise\_Root\_2\_solution.ipynb 3 days ago
- Exercise\_Root\_2.ipynb 3 days ago
- Exercise\_Root\_3.ipynb 3 days ago

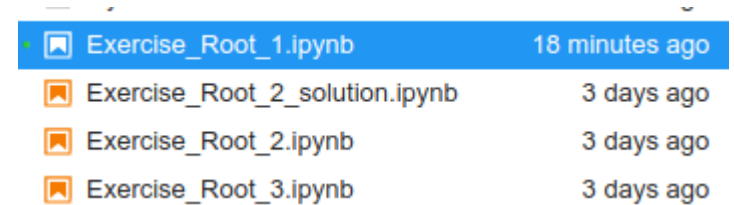
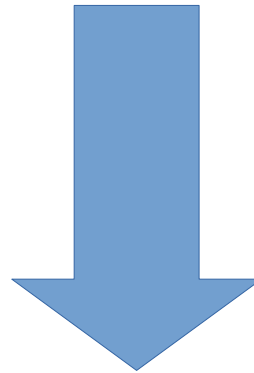
# Tutorial Parts

Three different notebooks. 1 → 2 → 3

For testing purposes use as input the reduced file size



Swan




Goettingen cloud





# Reference

In a browser

ROOT  Functionalities Control Panel Logout

File Edit View Insert Cell Kernel Help Python 2

  Code Cell Toolbar: None

# Welcome to the Notebook Technology

This is a markdown cell. You can add LaTeX code:  $\sum_{n=-\infty}^{\infty} |x(n)|^2$

Text & Formulas



Home × +

localhost:8888/tree

ROOT Functionalities Control Panel Logout

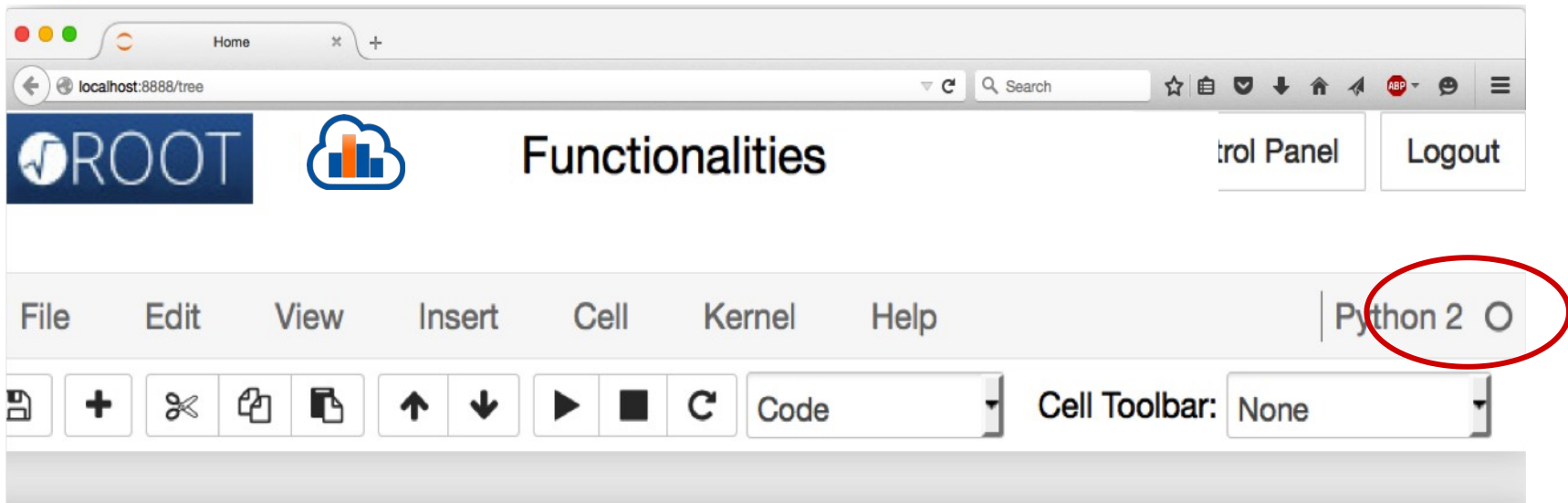
File Edit View Insert Cell Kernel Help Python 2

+ ✂ ↑ ↓ ▶ ■ ↻ Code Cell Toolbar: None

# Welcome to the Notebook Technology

This is a markdown cell. You can add LaTeX code:  $\sum_{n=-\infty}^{\infty} |x(n)|^2$

```
In [1]: def thisFunction():
        return 42
```



# Welcome to the Notebook Technology

This is a markdown cell. You can add LaTeX code:  $\sum_{n=-\infty}^{\infty} |x(n)|^2$

```
In [1]: def thisFunction():  
        return 42
```

Code

A *Python* notebook

The image shows a web browser window displaying a Jupyter Notebook. The browser's address bar shows 'localhost:8888/tree'. The notebook's title is 'Notebook Functionalities'. In the top right corner, there are buttons for 'Control Panel' and 'Logout'. The notebook's menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help', with 'Python 2' selected. Below the menu bar is a toolbar with icons for saving, adding, deleting, copying, pasting, undo, redo, and running. The main content area displays a large heading 'Welcome to the Notebook Technology'. Below the heading is a text block: 'This is a markdown cell. You can add LaTeX code:  $\sum_{n=-\infty}^{\infty} |x(n)|^2$ '. There are two code cells. The first cell contains the code: 

```
In [1]: def thisFunction():  
        return 42
```

. The second cell contains the code: 

```
In [2]: thisFunction()
```

. Below the second cell, the output is shown: 

```
Out[2]: 42
```

. A yellow button with the text 'Code' is positioned to the right of the code cells.

```
In [1]: def thisFunction():  
        return 42
```

```
In [2]: thisFunction()
```

```
Out[2]: 42
```

```
In [3]: %%bash  
        curl rootasdemo.web.cern.ch/rootasdemo/SaaSfee.jpg \  
        > SF.jpg
```



Invoke shell commands ...

Shell commands

```
In [1]: def thisFunction():  
        return 42
```

```
In [2]: thisFunction()
```

```
Out[2]: 42
```

```
In [3]: %%bash  
        curl rootaasdemo.web.cern.ch/rootaasdemo/SaaSFee.jpg \  
> SF.jpg
```

```
% Total      % Received % Xferd  Average Speed   Time  
Time        Time Current                Dload  Upload   Total  
Spent       Left  Speed  
100 128k 100 128k  0    0 2731k      0  --:--:--  
--:--:-- --:--:-- 2787k
```

... and get their output

Shell commands

```
In [1]: def thisFunction():  
        return 42
```

```
In [2]: thisFunction()
```

```
Out[2]: 42
```

```
In [3]: %%bash  
curl rootasdemo.web.cern.ch/rootasdemo/SaaSFee.jpg \  
> SF.jpg
```

```
% Total      % Received % Xferd  Average Speed   Time  
Time        Time Current                Dload  Upload   Total  
Spent       Left  Speed  
100 128k 100 128k  0    0 2731k      0  --:--:--  
--:--:-- --:--:-- 2787k
```

```
In [4]: from IPython.display import Image  
Image(filename='./SF.jpg',width=225)
```

```
In [1]: def thisFunction():  
        return 42
```

```
In [2]: thisFunction()
```

```
Out[2]: 42
```

```
In [3]: %%bash  
curl rootasdemo.web.cern.ch/rootasdemo/SaaSFee.jpg \  
> SF.jpg
```

```
% Total      % Received % Xferd  Average Speed   Time  
Time        Time Current                Dload  Upload   Total  
Spent       Left  Speed  
100 128k 100 128k  0      0 2731k      0  --:--:--  
--:--:-- --:--:-- 2787k
```

```
In [4]: from IPython.display import Image  
Image(filename='./SF.jpg',width=225)
```

```
Out[4]:
```



Figures

In a browser

```
function():  
42
```

```
In [2]: thisFunction()
```

```
Out[2]: 42
```

Text &  
Formulas

```
In [3]: %%bash
```

```
curl rootasdemo.web.cern.ch/rootasdemo/SaaSFee.jpg \  
> SF.jpg
```

Code

% Total Time	% Re Time		ge Speed	Time
Spent	Left	Speed	Upload	Total
100 128k	100 128k	0 0	2731k	0 ---:---:--
---:---:--	---:---:--	2787k		

No excuse not to  
document your  
analysis !!

Shell commands

```
.display import Image  
me="./SF.jpg",width=225)
```

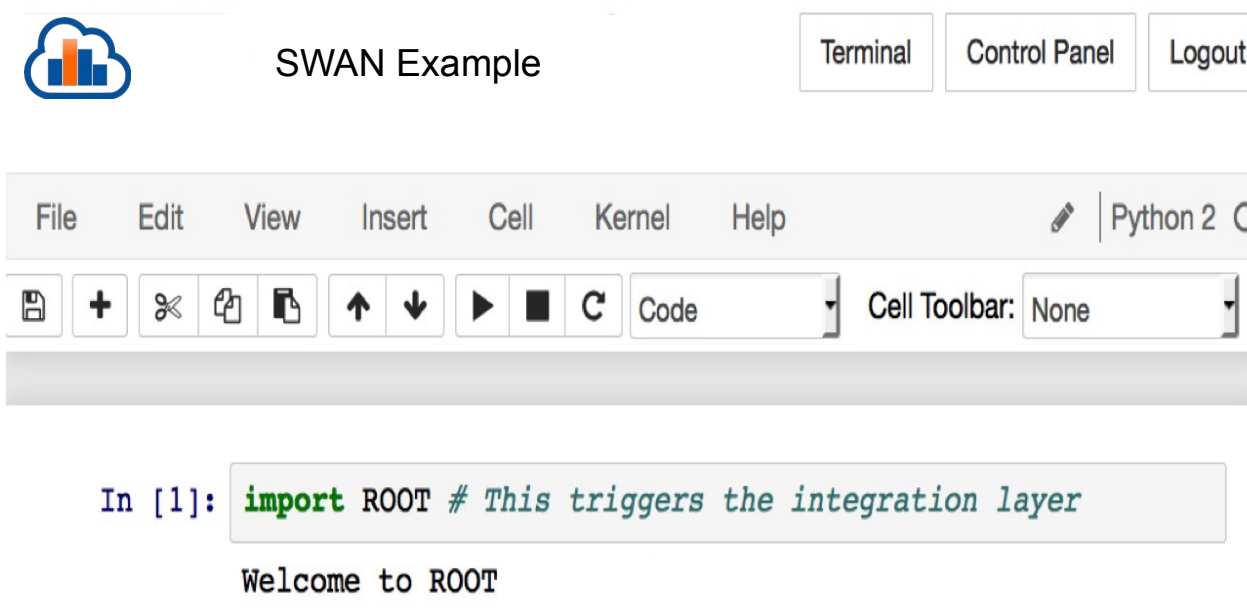
```
Out[4]:
```



Figures



# Some Goodies



The screenshot shows a Jupyter Notebook interface. At the top left is a logo with a cloud and a bar chart. The title "SWAN Example" is centered. On the top right are buttons for "Terminal", "Control Panel", and "Logout". Below the title bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help". To the right of the menu bar is a "Python 2" indicator. Below the menu bar is a toolbar with icons for save, add, cut, copy, paste, up, down, play, stop, and refresh. To the right of the toolbar is a "Code" dropdown menu and a "Cell Toolbar: None" dropdown menu. The main content area shows a code cell with the following text:

```
In [1]: import ROOT # This triggers the integration layer
```

Below the code cell, the output "Welcome to ROOT" is displayed.

# Some Goodies

The screenshot shows a Jupyter Notebook interface titled "SWAN Example". At the top right, there are buttons for "Terminal", "Control Panel", and "Logout". Below the title bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help". A toolbar contains icons for file operations and execution. The notebook content shows two code cells. The first cell contains a Python code snippet: `In [1]: import ROOT # This triggers the integration layer`, followed by the output "Welcome to ROOT". The second cell contains a C++ code snippet: `In [ ]: %%cpp` followed by `auto myHisto = TH|`. A dropdown menu is visible below the second cell, listing the following options: TH1, TH1C, TH1D, TH1F, TH1I, TH1K, TH1S, TH2, TH2C, and TH2D. Two yellow callout boxes are overlaid on the image: one on the left that says "C++ Cells in Python Notebooks" and one on the right that says "ROOT Tab-Completion".

C++ Cells in  
Python  
Notebooks

ROOT Tab-  
Completion

# Some Goodies



SWAN Example

Terminal

Control Panel

L

Menu: Edit View Insert Cell Kernel Help Python

Toolbar: + ✂ 📄 📄 ⬆ ⬇ ▶ ■ ↻ Code Cell Toolbar: None

```
In [1]: import ROOT # This triggers the integration layer
```

Welcome to ROOT

```
In [2]: %%cpp  
auto myHisto = TH1F("h", "MyData;X;Y", 64, -4, 4); // C++11
```

I really wanted to show modern C++...

# Some Goodies



SWAN Example

Terminal

Control Panel

L

Menu: Edit View Insert Cell Kernel Help Python

Toolbar: +, ✂, 📄, 📄, ⬆, ⬆, ▶, ■, ↻, Code, Cell Toolbar: None

```
In [1]: import ROOT # This triggers the integration layer
```

Welcome to ROOT

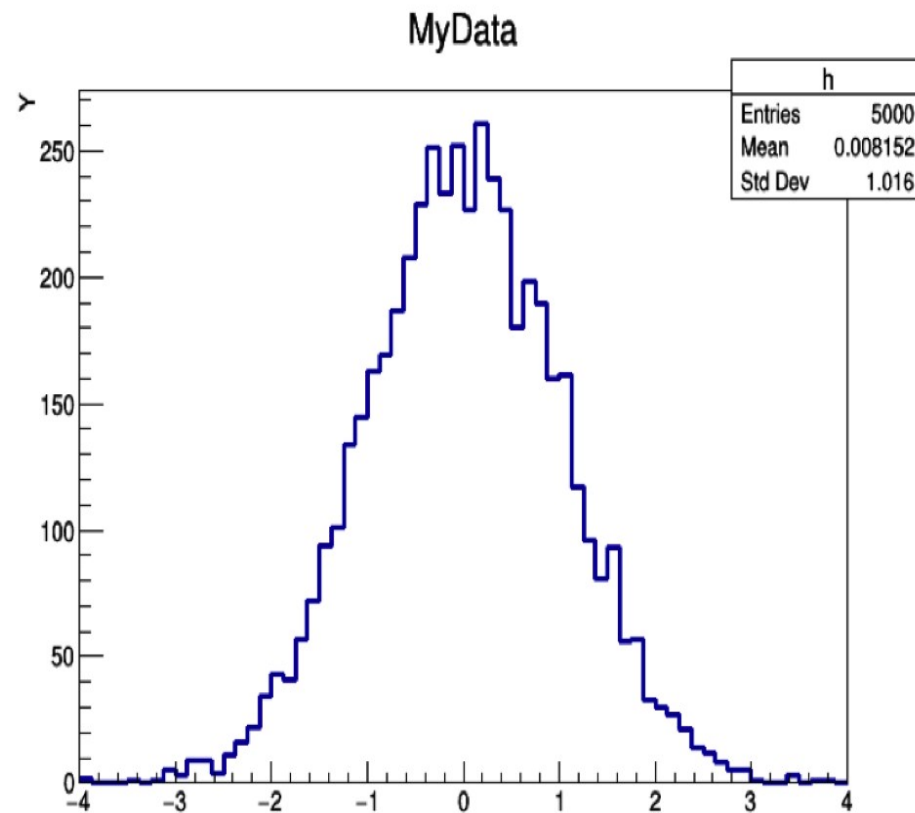
```
In [2]: %%cpp
auto myHisto = TH1F("h", "MyData;X;Y", 64, -4, 4); // C++11
```

```
In [3]: h = ROOT.myHisto # Find the variable back in Python!
h.FillRandom("gaus")
c = ROOT.TCanvas()
h.Draw()
c.Draw()
```

**Full Python-C++  
interoperability**

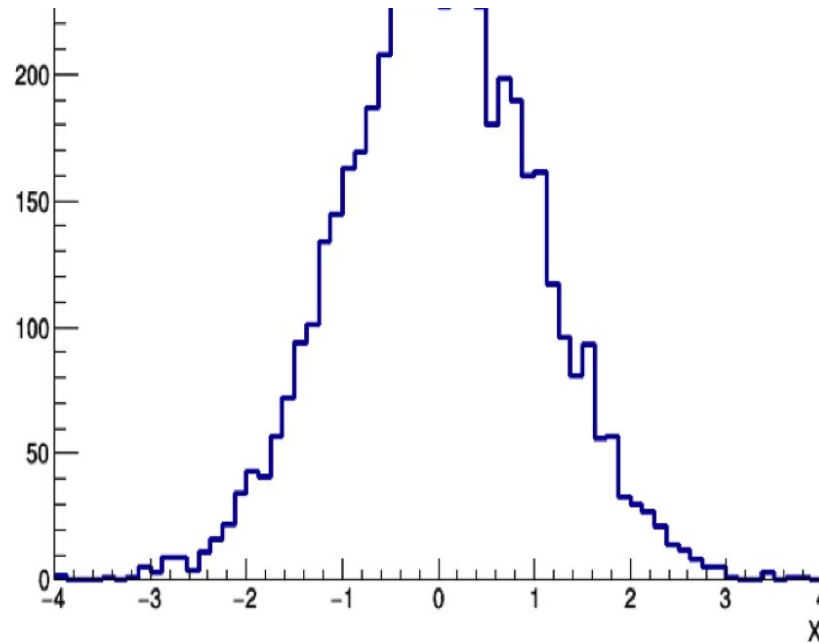
# Some Goodies

```
h.Draw()  
c.Draw()
```



Seamless display  
of graphics

# Some Goodies



```
In [4]: %%cpp -d
double myG(double* x, double* par){
    auto res = (x[0]-par[1])/par[2];
    auto e = -.5 * res * res;
    return par[0] * exp(e); // declare function
}
```

Syntax  
highlighting

# Some Goodies

```
In [4]: %%cpp -d
double myG(double* x, double* par){
    auto res = (x[0]-par[1])/par[2];
    auto e = -.5 * res * res;
    return par[0] * exp(e); // declare function
}
```

```
In [5]: f = ROOT.TF1("myGf",ROOT.myG,-5,5,3)
f.SetParameters(200,0,1);f.SetParNames("N","mu","sigma")
fr = ROOT.h.Fit(f,"S") # Capture printouts
```

# Some Goodies

```
In [4]: %%cpp -d
double myG(double* x, double* par){
  auto res = (x[0]-par[1])/par[2];
  auto e = -.5 * res * res;
  return par[0] * exp(e); // declare function
}
```

```
In [5]: f = ROOT.TF1("myGf",ROOT.myG,-5,5,3)
f.SetParameters(200,0,1);f.SetParNames("N","mu","sigma")
fr = ROOT.h.Fit(f,"S") # Capture printouts
```

```
FCN=47.4997 FROM MIGRAD      STATUS=CONVERGED      69 CALLS      70 TO
TAL
```

```
EDM=2.04372e-09      STRATEGY= 1      ERROR MATRIX ACC
```

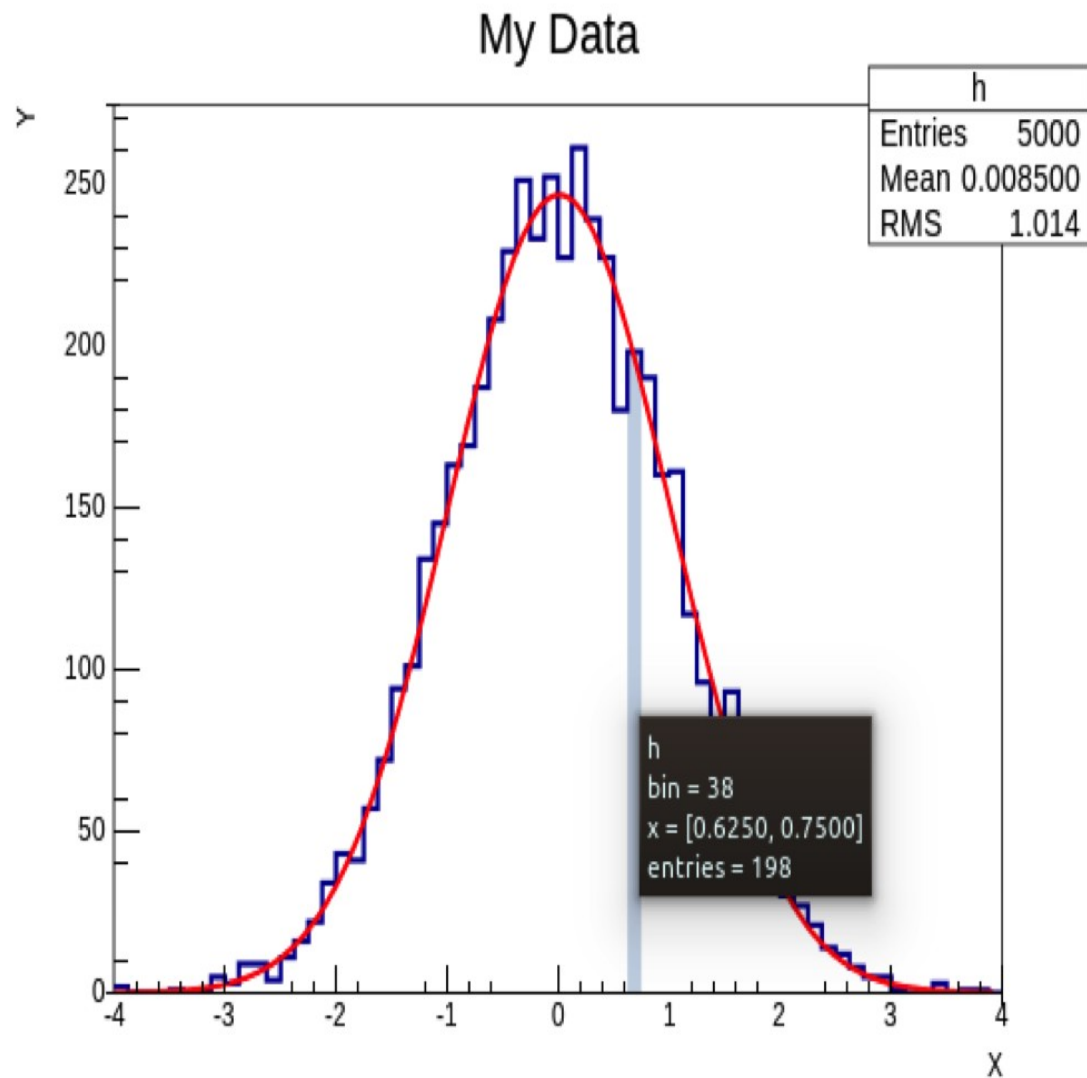
```
URATE
```

EXT	PARAMETER			STEP	FIRST
NO.	NAME	VALUE	ERROR	SIZE	DERIVATIVE
1	N	2.46469e+02	4.31493e+00	1.19092e-02	-5.38026e-06
2	mu	1.04793e-02	1.43576e-02	4.87640e-05	4.15093e-03
3	sigma	1.00316e+00	1.03818e-02	2.86307e-05	-2.55310e-04



# Some Goodies

```
%jsroot on
```



**JSROOT**  
Visualisation

# Use Notebooks at CERN

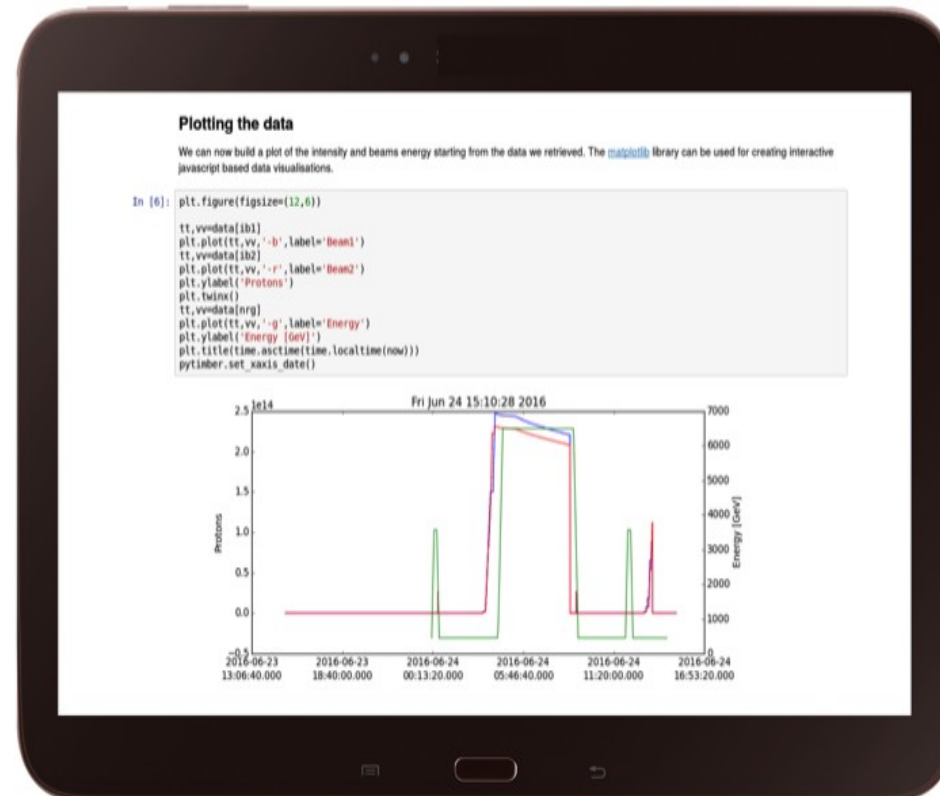
- ▶ **SWAN**: Service for **W**eb based **A**nalysis
- ▶ Get a CERNBox (if you don't have one)
  - Visit <https://cernbox.cern.ch>

<https://swan.cern.ch>

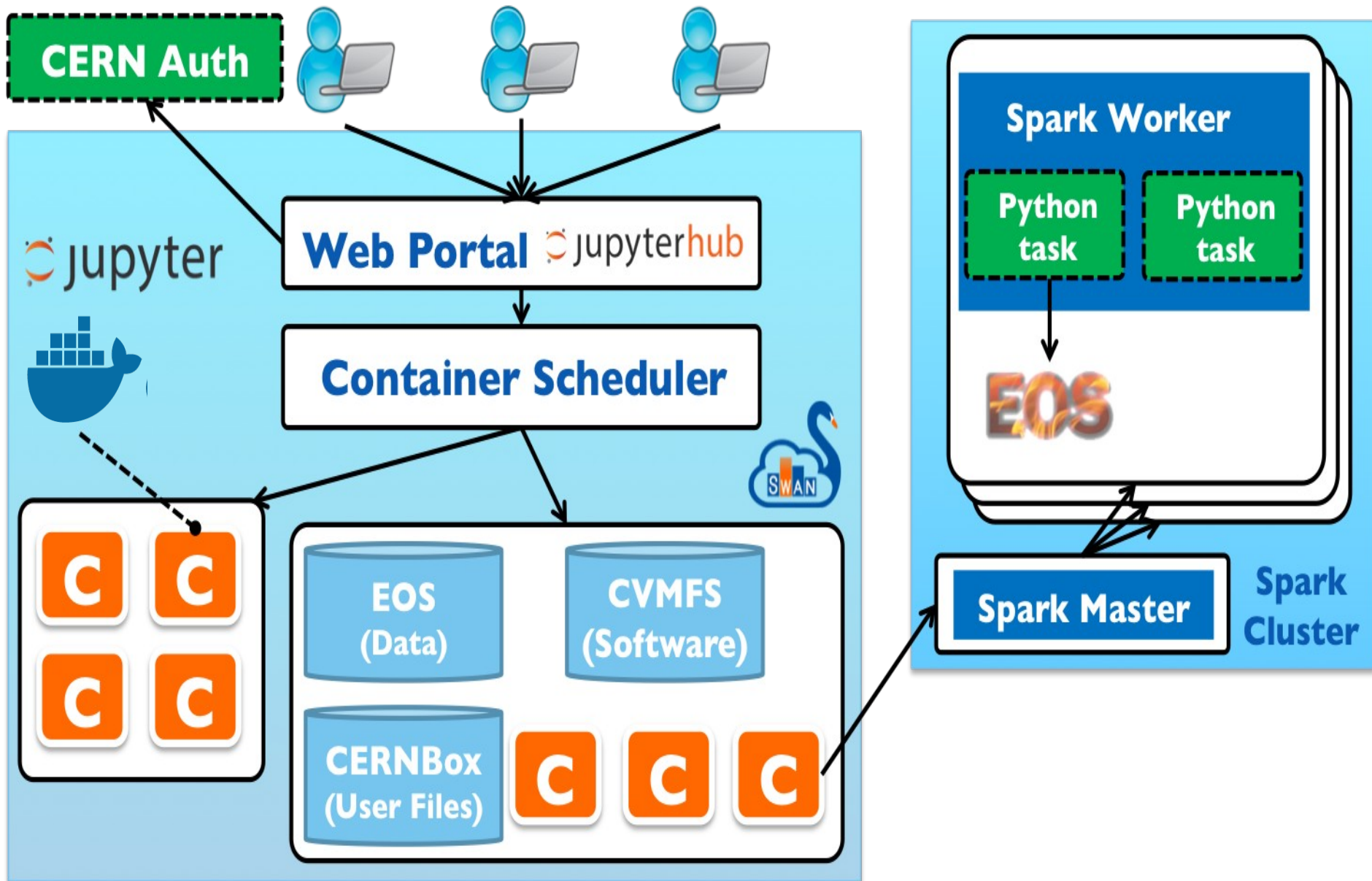


# The Original Idea: A Turn-Key System

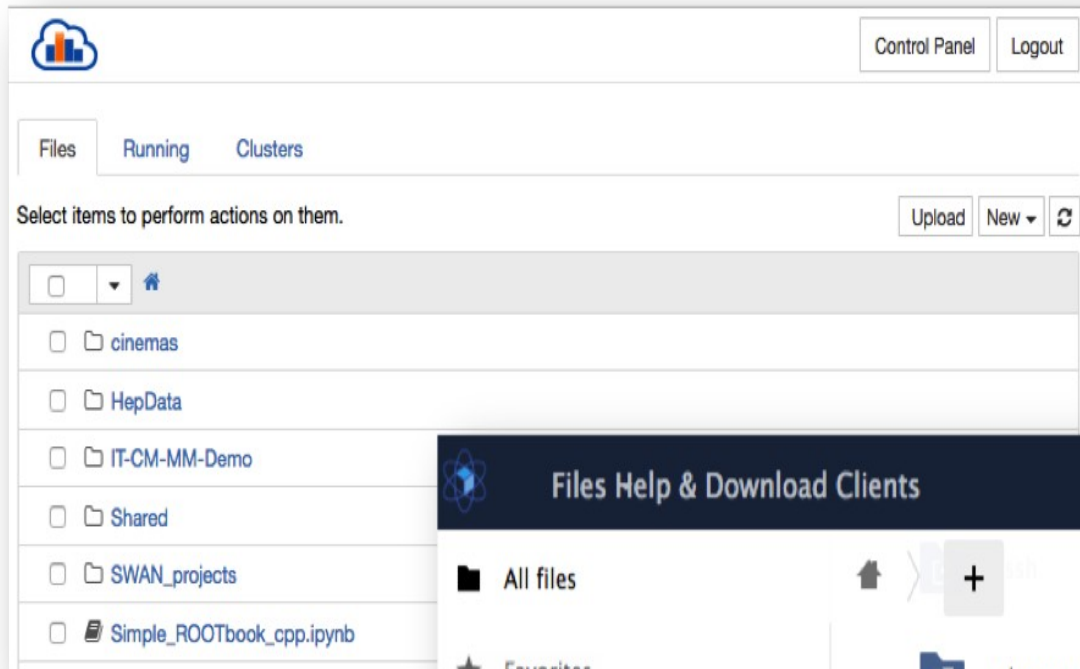
- ▶ A ready-to-use system for performing data analysis with **all the software on all the data we need. Only requirement: a web browser.**



# The Service in a Nutshell

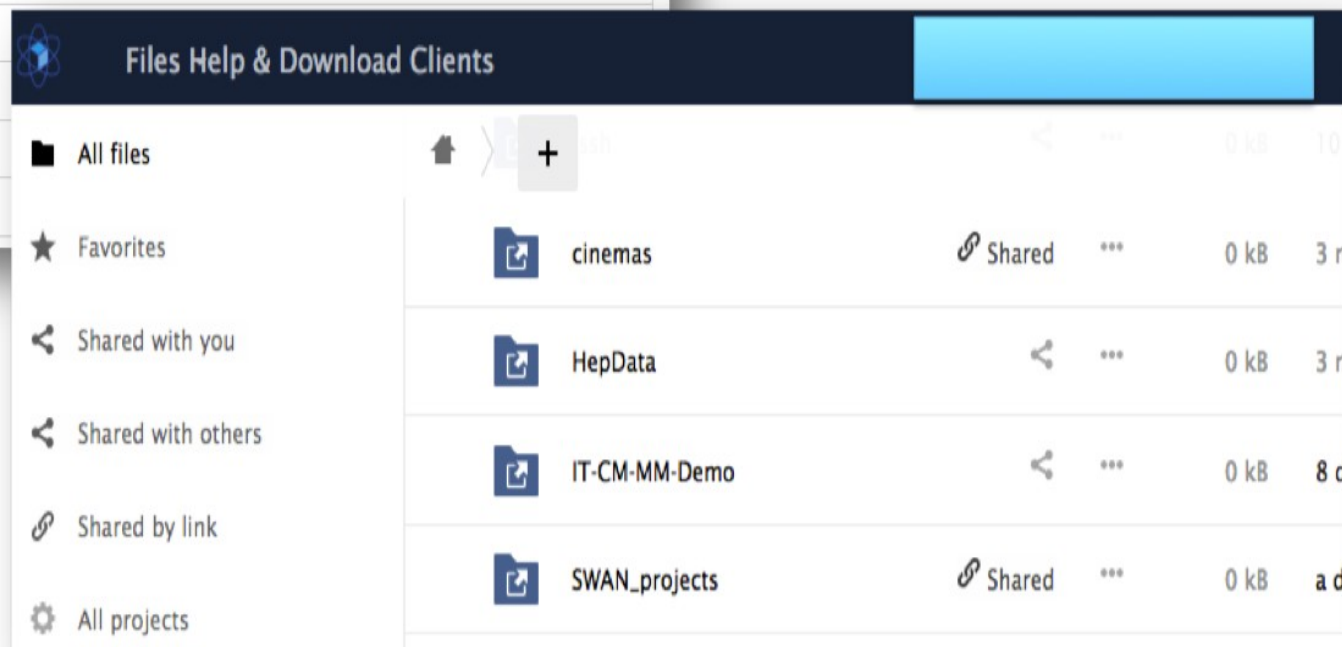


# CERNBox is Your HOME



The screenshot shows the CERNBox Control Panel interface. At the top right, there are buttons for "Control Panel" and "Logout". Below this, there are tabs for "Files", "Running", and "Clusters". A message says "Select items to perform actions on them." followed by "Upload", "New", and a refresh icon. A list of items is shown with checkboxes and a home icon:

- Home
- cinemas
- HepData
- IT-CM-MM-Demo
- Shared
- SWAN\_projects
- Simple\_ROOTbook\_cpp.ipynb



The screenshot shows the "Files Help & Download Clients" interface. It features a sidebar with navigation options and a main content area with a file list.

**Files Help & Download Clients**

- All files
- Favorites
- Shared with you
- Shared with others
- Shared by link
- All projects

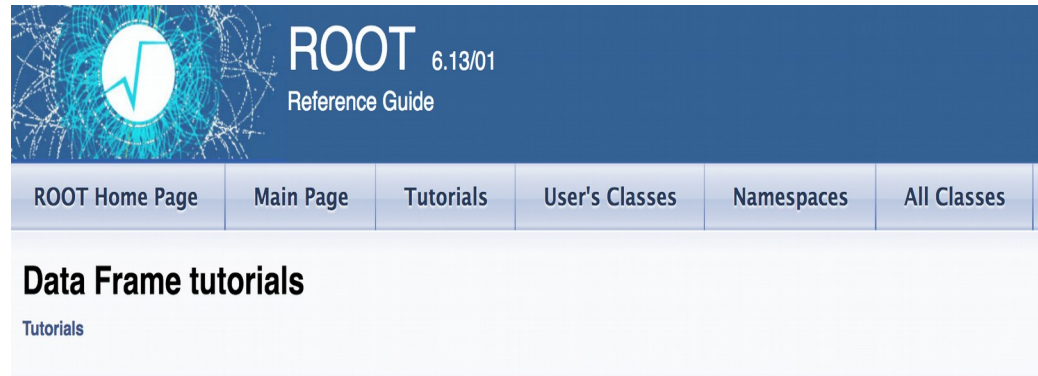
Item	Shared	Size	Modified
cinemas	Shared	0 kB	3 m
HepData		0 kB	3 m
IT-CM-MM-Demo		0 kB	8 d
SWAN_projects	Shared	0 kB	a d



# Interactive Tour of ROOTBooks



# More Examples



ROOT 6.13/01  
Reference Guide

ROOT Home Page | Main Page | Tutorials | User's Classes | Namespaces | All Classes

## Data Frame tutorials

Tutorials

[https://root.cern/doc/master/group\\_\\_Tutorials.html](https://root.cern/doc/master/group__Tutorials.html)

These examples show the functionalities of the TDataFrame class.

## Files

file [tdf001\\_introduction.C](#)

[View](#) [Notebook](#) [Open in SWAN](#) This tutorial illustrates the basic features of the TDataFrame class, a utility with a chain like approach.

file [tdf001\\_introduction.py](#)

[View](#) [Notebook](#) [Open in SWAN](#) This tutorial illustrates the basic features of the TDataFrame class, a utility with a chain like approach.

file [tdf002\\_dataModel.C](#)

[View](#) [Notebook](#) [Open in SWAN](#) This tutorial shows the possibility to use data models which are more complex.

file [tdf002\\_dataModel.py](#)


[View](#) [Notebook](#) [Open in SWAN](#) This tutorial shows the possibility to use data models which are more complex.

file [tdf003\\_profiles.C](#)

[View](#) [Notebook](#) [Open in SWAN](#) This tutorial illustrates how to use TProfiles in combination with the TDataFrame.

file [tdf003\\_profiles.py](#)

# More Examples



**SWAN**  
Interactive Data Analysis, in the Cloud.

Home Galleries **FAQ** Talks and Publications

Basic ROOT Primer Accelerator Complex Beam Dynamics Machine Learning Apache Spark Outreach

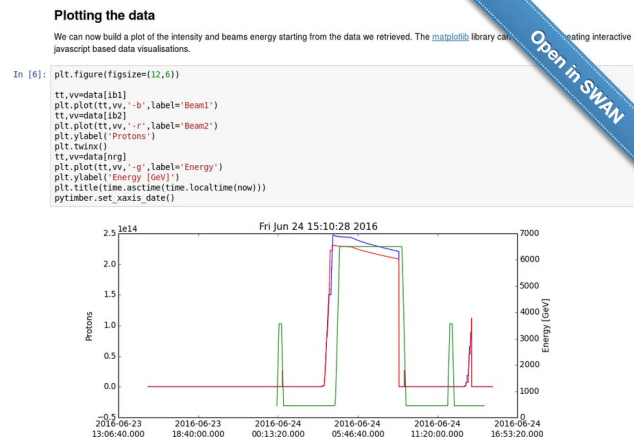
<https://swan.web.cern.ch>

## Accelerator Complex

This gallery shows examples of machine studies relative to the CERN accelerators' complex.

Open in  SWAN

### LHC Page1



### Experiments' Luminosities

