



The ROOT's Notebook interface and the SWAN Service

<https://root.cern>

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



D. Piparo (EP-SFT) for the ROOT team
Alice Offline Week





Explore the new Jupyter interface of ROOT and the SWAN service

Part 1:

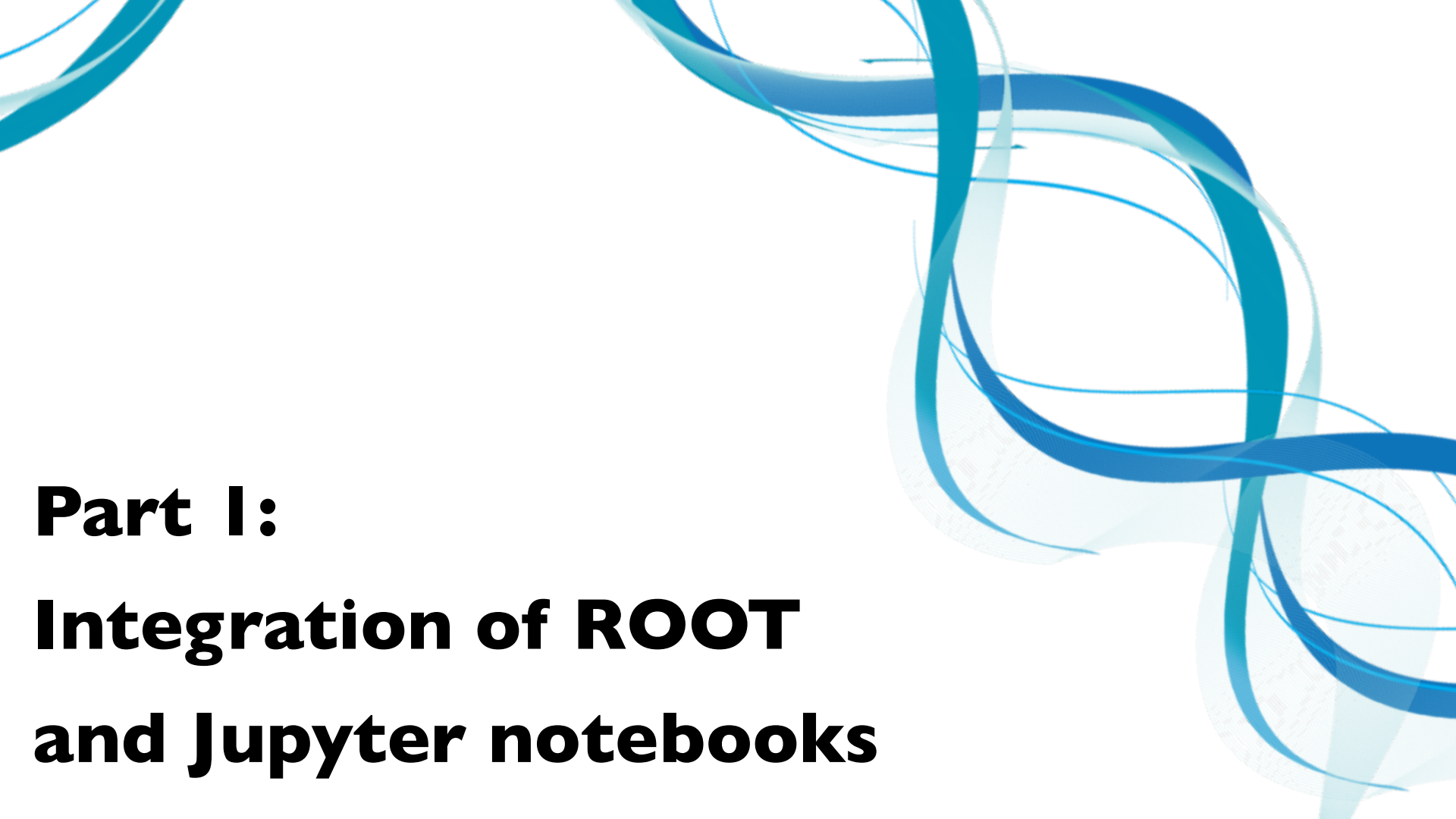
- Introduction to the Jupyter Notebooks
- ROOT integration: C++ and Python kernels
- Added value of this synergy

**All available in
ROOT 6.08 !**

Part 2:

- CERN's notebook service: SWAN
- Provision of ROOT (and more) “as a service”
- Added value of SWAN with respect to other cloud based models

**Available now to
every CERN user !**

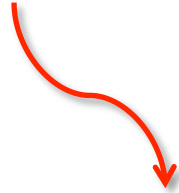


Part I: Integration of ROOT and Jupyter notebooks



Prelude: the Notebook

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



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

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



An Example Notebook

Text

Code

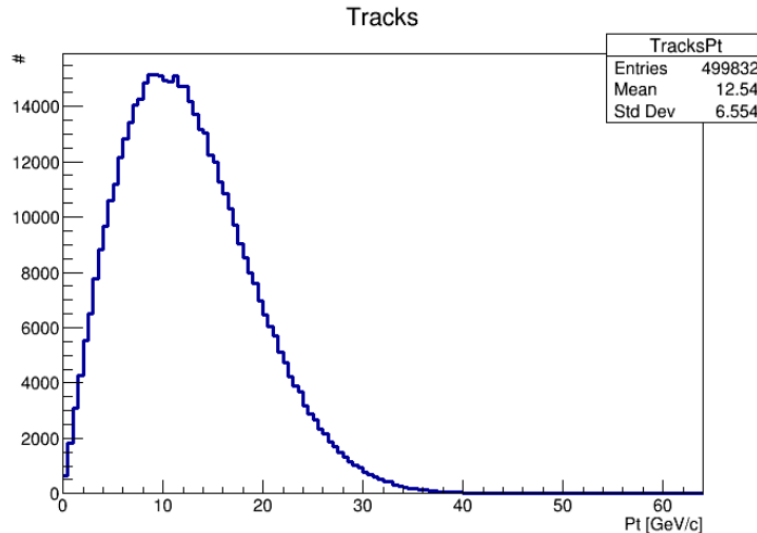
(Interactive)
Graphics

Access TTree in Python using PyROOT and fill a histogram

Loop over the TTree called "events" in a file located on the web. The tree is accessed with the dot operator. Same holds for the access to the branches: no need to set them up - they are just accessed by name, again with the dot operator.

```
In [1]: import ROOT

f = ROOT.TFile.Open("http://indico.cern.ch/event/395198/material/0/0.root");
h = ROOT.TH1F("TracksPt", "Tracks;Pt [GeV/c];#", 128, 0, 64)
for event in f.events:
    for track in event.tracks:
        h.Fill(track.Pt())
c = ROOT.TCanvas()
h.Draw()
c.Draw()
```



In a Browser



- Notebook features appealing to ROOT:
 - **Sharing**: scientists can share their results (code, plots, text) in the form of notebooks
 - **Teaching**: runnable tutorials and exercises, combining code and explanations
 - **Reproducibility**: a notebook contains results and the code that led to them
- Widely adopted outside HEP

The ROOTBooks

- Two language flavours (a.k.a. kernels) are available:

New in
Jupyter!



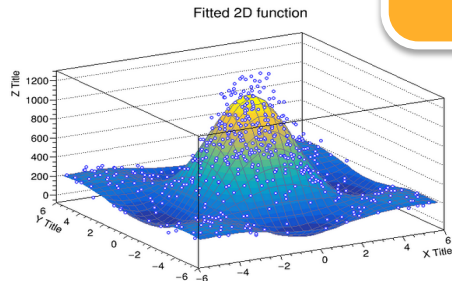
```
prated data.  
0.7, 1.5); // set initial values for fit  
ed 2D function*);  
ote:fit(f2);  
FCN=517.445 FROM MIGRAD STATUS=CONVERGED 38 CALLS 39 TOTAL  
EDM=2.65702e-12 STRATEGY= 1 ERROR MATRIX ACCURATE  
EXT PARAMETER STEP FIRST  
NO. NAME VALUE ERROR SIZE DERIVATIVE  
1 p0 6.81725e-01 4.37173e-01 2.40425e-05 8.08231e-04  
2 p1 1.46084e+00 9.36798e-01 5.15197e-05 3.78774e-04
```

Configure the canvas for plotting the result.

```
In [4]: TCanvas c1;  
f2.SetLineWidth(1);  
f2.SetLineColor(kBlue - 5);  
f2.Draw("Surf1");  
  
auto Xaxis = f2.GetAxis(); Xaxis->SetTitle("X Title"); Xaxis->SetRange(0, 6);  
auto Yaxis = f2.GetAxis(); Yaxis->SetTitle("Y Title"); Yaxis->SetRange(-6, 6);  
auto Zaxis = f2.GetAxis(); Zaxis->SetTitle("Z Title"); Zaxis->SetRange(0, 1200);  
dte.Draw("P0 Same");
```

Display the 2D graph in the notebook.

```
In [5]: c1.Draw();
```



Powered by
the ROOT C++
interpreter



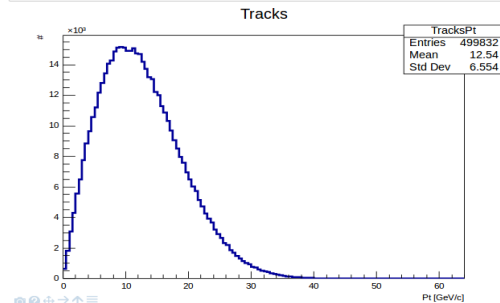
Access TTree in Python using PyROOT and fill a histogram

First import the ROOT Python module.

```
In [1]: import ROOT  
%jsroot on  
Welcome to JupyROOT 6.07/07  
  
Open a file which is located on the web. No type is to be specified for "f".  
  
In [3]: f = ROOT.TFile.Open("http://indico.cern.ch/event/395198/material/0/0.root")
```

Loop over the TTree called "events" in the file. It is accessed with the dot operator. Same holds for the access to the branches: no need to set them up - they are just accessed by name, again with the dot operator.

```
In [4]: h = ROOT.TH1F("TracksPt", "Tracks;Pt [GeV/c];#", 128, 0, 64)  
for event in f.events:  
    for track in event.tracks:  
        h.Fill(track.Pt())  
c = ROOT.TCanvas()  
h.Draw()  
c.Draw()
```



Via
PyROOT



Mixing Languages

- C++ and Python can be mixed in the same ROOTBook
 - Thanks to the ROOT type system

Interleave Python with C++: the %%cpp magic



```
In [1]: import ROOT
```

Welcome to JupyROOT 6.07/03

Thanks to its [interpreter](#) and [type system](#), entities such as functions, classes and variables, created in a C++ cell, can be accessed from within Python.

```
In [2]: %%cpp
class A {
public:
    A() { cout << "Constructor of A!" << endl; }
};
```

**%%python also
available in C+
+ notebooks**

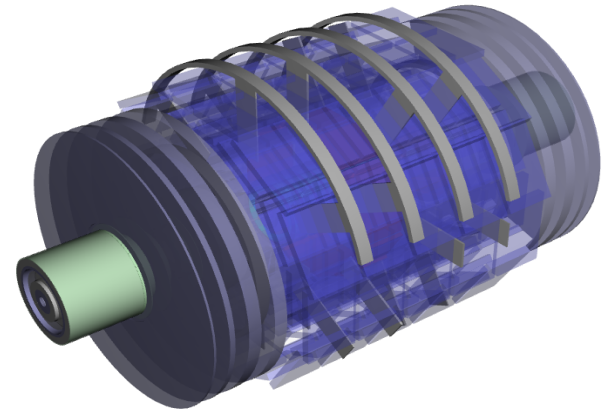
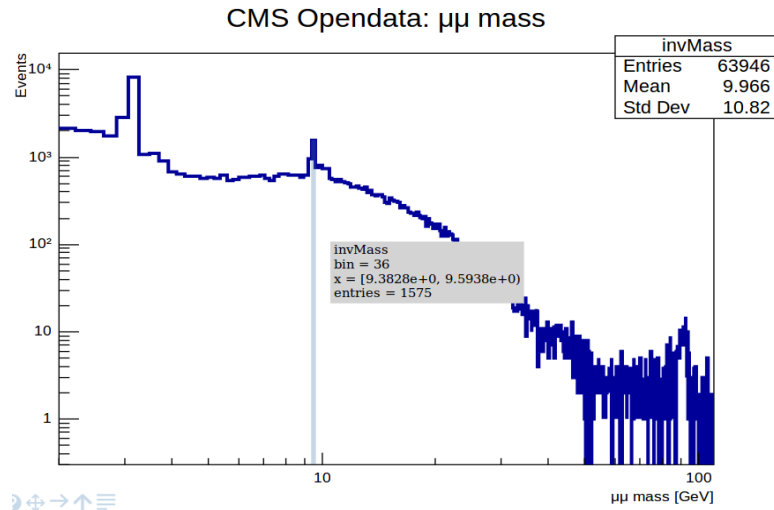
```
In [3]: a = ROOT.A()
```

Constructor of A!



Interactive Graphics: JSROOT

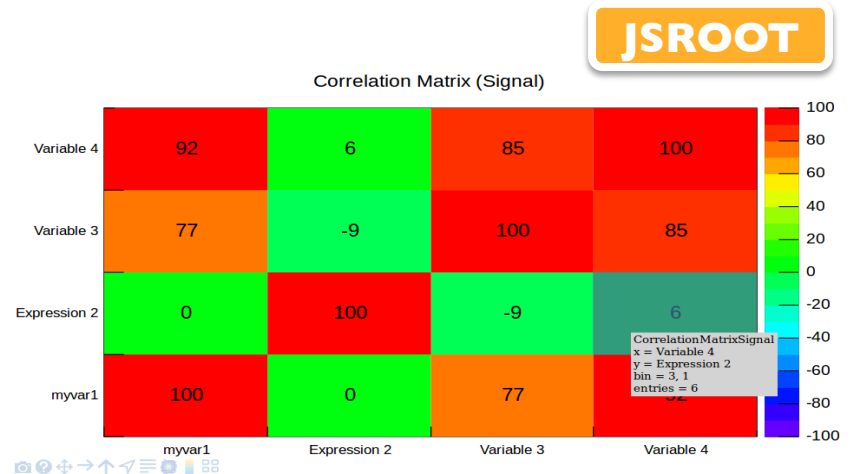
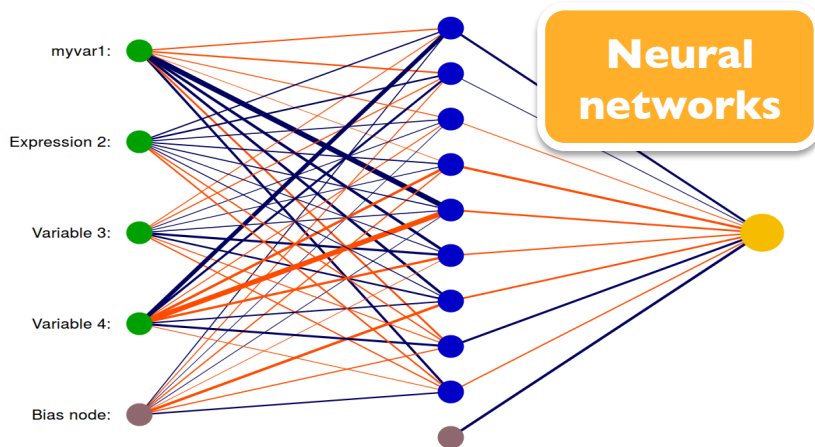
- Both flavours (C++ and Python) allow to **inline ROOT graphics** in a notebook
- Two modes: static image and **JavaScript visualisation**
 - Activate **JSROOT** mode with **%jsroot on**
 - Interact with your plot: zoom, modify axis, inspect bins, ...





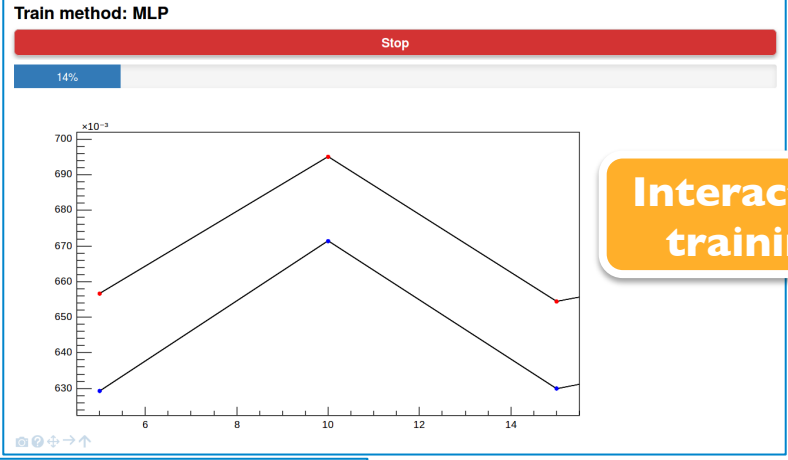
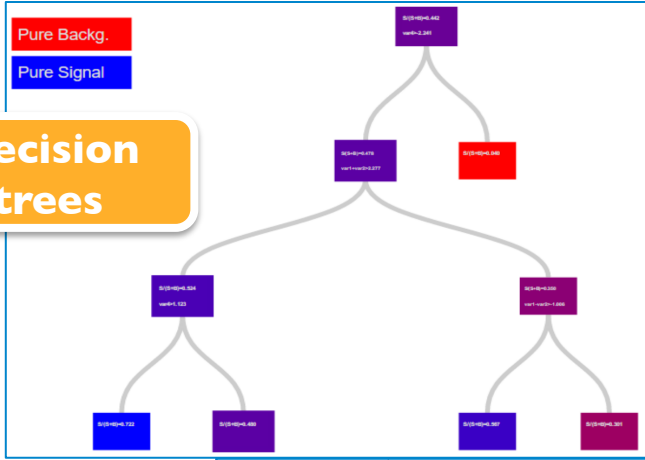
Interactive Machine Learning

- TMVA: **machine learning** toolkit in ROOT
 - Recently integrated with Jupyter as well: **%jsmva on**
 - JSROOT plots for input variables
 - Visualisation of neural networks and decision trees, DNN designer
 - Interactive training: stop a server computation
 - HTML output formatting



More Added Value

Decision trees



Interactive training

DataSetInfo	Correlation matrix (Signal)																									
DataSetInfo	Correlation matrix (Background)																									
DataSetFactory	Dataset: <code>tmva_class_example</code>																									
TFHandler_MLP	<table border="1"> <thead> <tr> <th>Variable</th> <th>Mean</th> <th>RMS</th> <th>Min</th> <th>Max</th> </tr> </thead> <tbody> <tr> <td>myvar1</td> <td>0.083989</td> <td>0.36407</td> <td>-1.0000</td> <td>1.0000</td> </tr> <tr> <td>myvar2</td> <td>0.0094778</td> <td>0.27696</td> <td>-1.0000</td> <td>1.0000</td> </tr> <tr> <td>var3</td> <td>0.080279</td> <td>0.36720</td> <td>-1.0000</td> <td>1.0000</td> </tr> <tr> <td>var4</td> <td>0.12986</td> <td>0.39603</td> <td>-1.0000</td> <td>1.0000</td> </tr> </tbody> </table> <p>Training Network Elapsed time for training with 6000 events : 4.45 sec</p>	Variable	Mean	RMS	Min	Max	myvar1	0.083989	0.36407	-1.0000	1.0000	myvar2	0.0094778	0.27696	-1.0000	1.0000	var3	0.080279	0.36720	-1.0000	1.0000	var4	0.12986	0.39603	-1.0000	1.0000
Variable	Mean	RMS	Min	Max																						
myvar1	0.083989	0.36407	-1.0000	1.0000																						
myvar2	0.0094778	0.27696	-1.0000	1.0000																						
var3	0.080279	0.36720	-1.0000	1.0000																						
var4	0.12986	0.39603	-1.0000	1.0000																						
MLP	Dataset: <code>tmva_class_example</code> Evaluation of MLP on training sample (6000 events) Elapsed time for evaluation of 6000 events : 0.0187 sec Creating xml weight file: <code>tmva_class_example/weights/TMVAClassification_MLP.weights.xml</code> Creating standalone class: <code>tmva_class_example/weights/TMVAClassification_MLP.class.C</code> Write special histos to file: <code>TMVA.root/tmva_class_example/Method_MLP/MLP</code>																									

HTML output



Try it on Your Machine!

Follow simple instructions at

<https://root.cern.ch/how/how-create-rootbook>

and type...

```
$ root --notebook
```

This command:

1. Starts a local notebook server
2. Connects to it via the browser

Provides a ROOT C++ kernel and the rest of ROOTbooks' added value



Do you Need Examples?

[ROOT Home Page](#)[Main Page](#)[Tutorials](#)[User's Classes](#)[Namespaces](#)[All Classes](#)[Files](#)[Release Notes](#)

Histograms tutorials

Tutorials

**ROOT tutorials available as notebooks,
also to be opened in SWAN**

Examples showing the "histograms' classes" usage..

Files

file **andleplot.C**

[View](#)[Notebook](#)[Open in](#)[SWAN](#)

Example of candle plot with 2-D histograms.

file **andleplotoption.C**

[View](#)[Notebook](#)[Open in](#)[SWAN](#)

Example showing how to combine the various candle plot options.

file **andleplotstack.C**

[View](#)[Notebook](#)[Open in](#)[SWAN](#)

Example showing how a **THStack** with candle plot option.

file **andleplotwhiskers.C**

[View](#)[Notebook](#)[Open in](#)[SWAN](#)

Example of candle plot showing the whiskers definition.

Part 2:

Service for Web Based

ANalysis: SWAN

Reference paper: <https://cds.cern.ch/record/2158559>

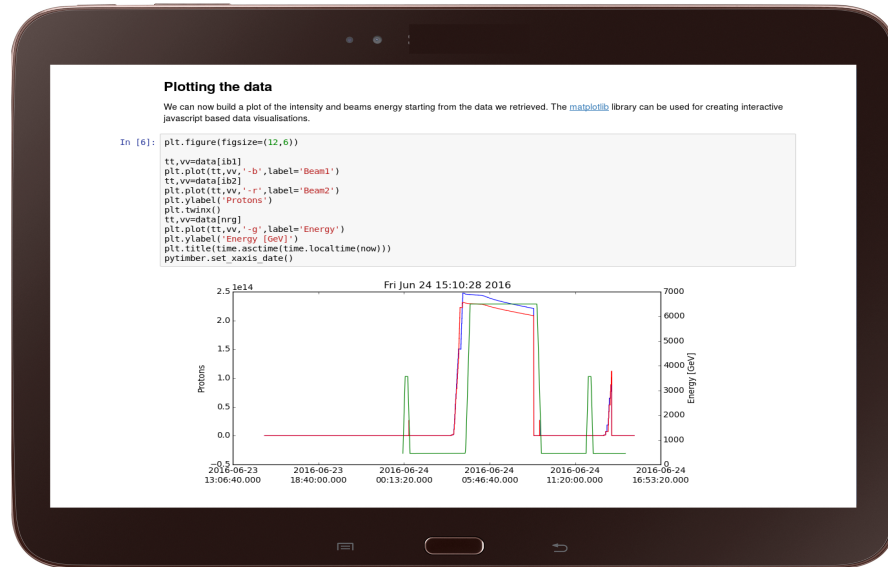




A Turn-Key System

The SWAN Service: A ready-to-use system for performing data analysis with **all the software on all the data we need.**

Only requirement: a web browser.



Available now: <https://swan.cern.ch>



Data Analysis as a Service

- Platform independent: **only with a web browser**
 - Analyse data **via the Notebook web interface**
 - **No need to install and configure software**
- Calculations, input and results **“in the cloud”**
- Allow **easy sharing of scientific results**: plots, data, code
 - Storage is crucial, both mass and synchronised
- **Simplify teaching** of data processing and programming
 - ROOT Summer Student course, ML trainings
- **Ease reproducibility of results and documentation**
- **C++, Python** and other languages or analysis **“ecosystems”**
 - Also interface to widely adopted scientific libraries

Spark



C++





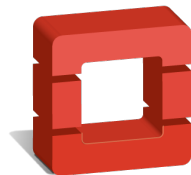
Novel Application, Existing Components

SWAN relies on the CERN ecosystem:

- Authentication with CERN credentials
- Machines in the Openstack cloud
- Software distribution: CVMFS
- Storage access: EOS, CERNBox
 - User and experiments data available

External but mainstream technologies

- Jupyterhub
- Docker



A coherent view at CERN



The “home directory” in SWAN



Both have large user bases and an active community behind

CERNBox is Your Home



Control Panel Logout

Files Running Clusters

Select items to perform actions on them. Upload New ↕

- Home
- cinemas
- HepData
- IT-CM-MM-Demo
- Shared
- SWAN_projects
- Simple_ROOTbook_cpp.ipynb

Files Help & Download Clients

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

Folder	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



The Architecture

CERN Auth



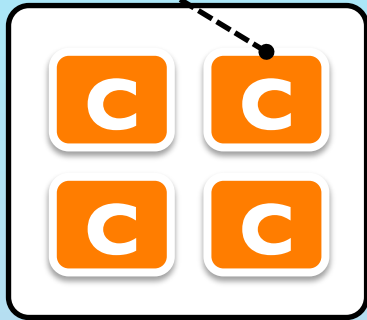
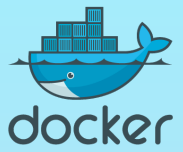
CERN Cloud

Web Portal jupyterhub

Container Scheduler

jupyter

Notebook Container



EOS (Data)

CVMFS (Software)

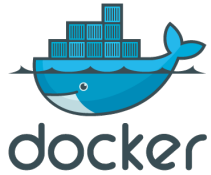
CERNBox (User Files)



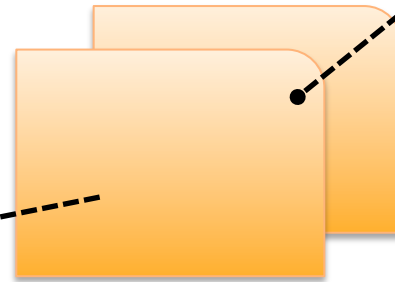
Software Environment

- Strategy to configure the software environment:
 - Docker: **single** thin image, not managed by the user!
 - CVMFS: configurable environment via “**views**”
 - CERNBox: custom user environment

**Externals/
LCG Releases**



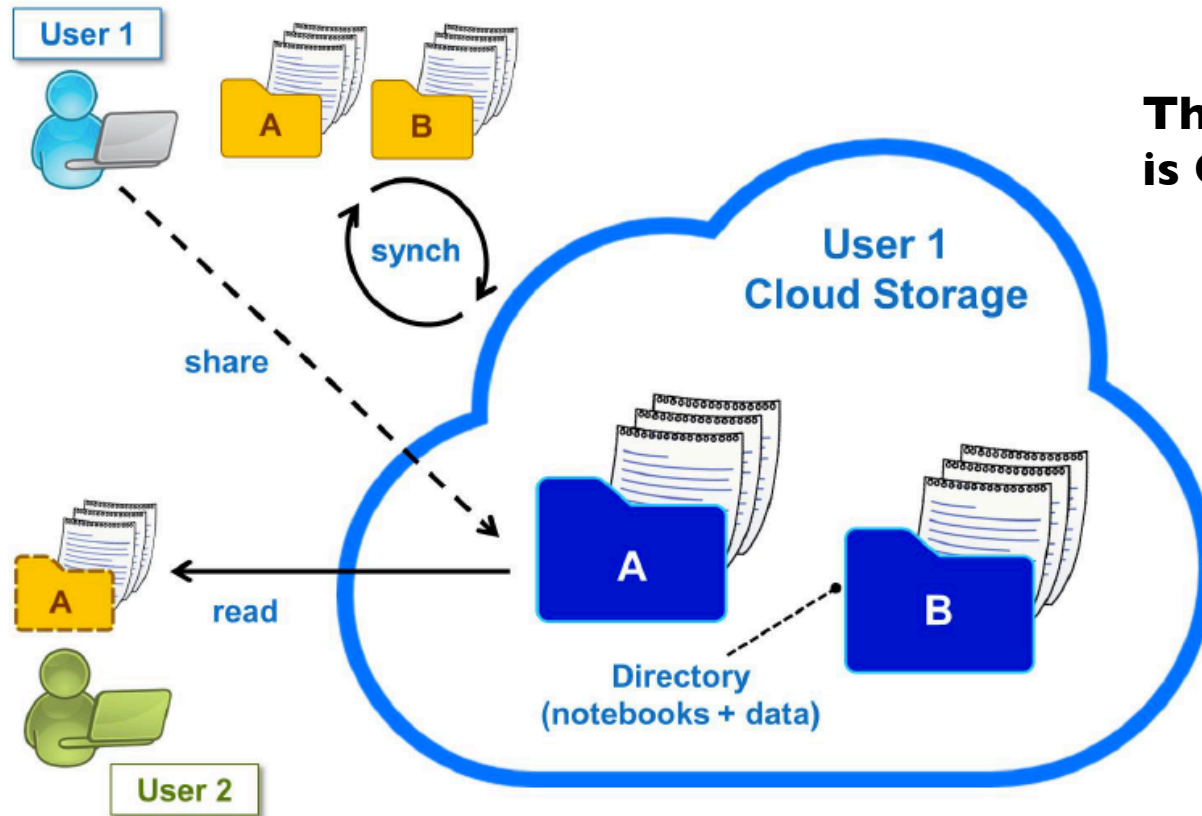
ALICE software



User software

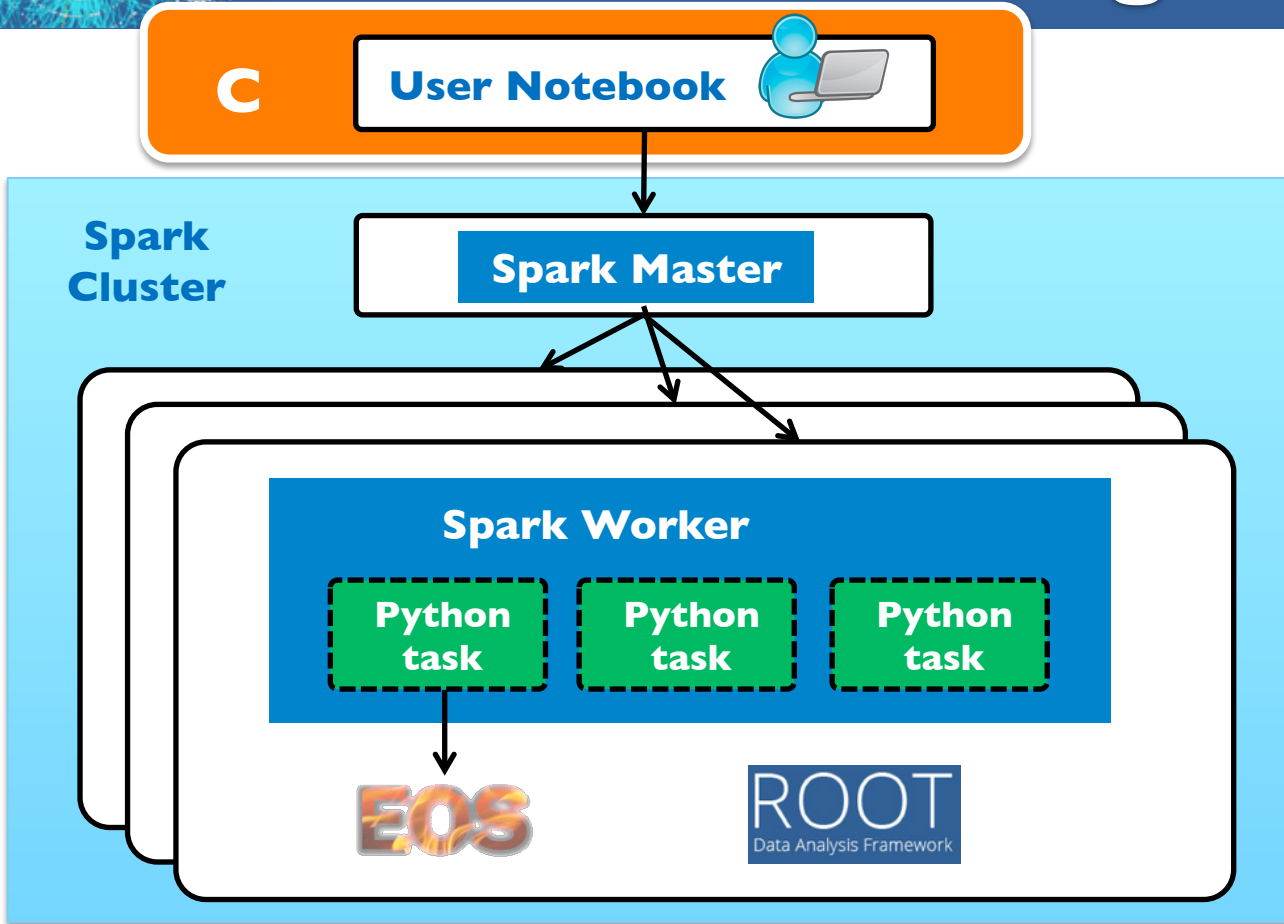


Cloud Based Analysis Model



The home directory is CERNBox

Offloading Calculations



In collaboration
with IT-DB, IT-ST



R&D:
Not in
production yet

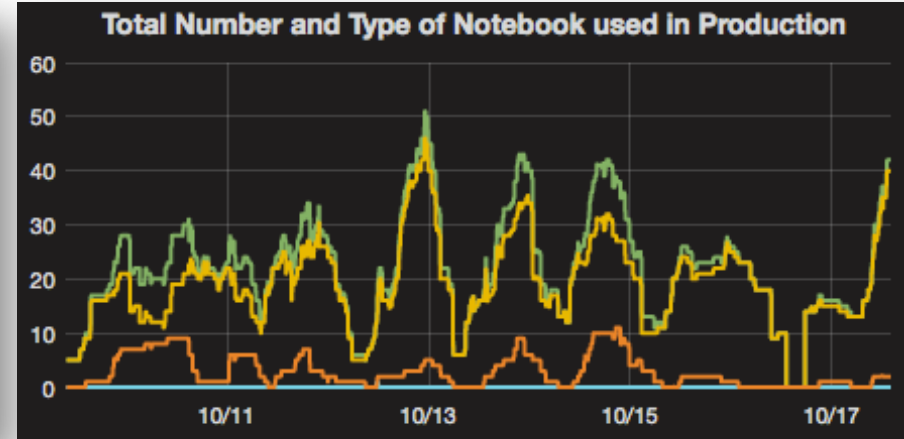
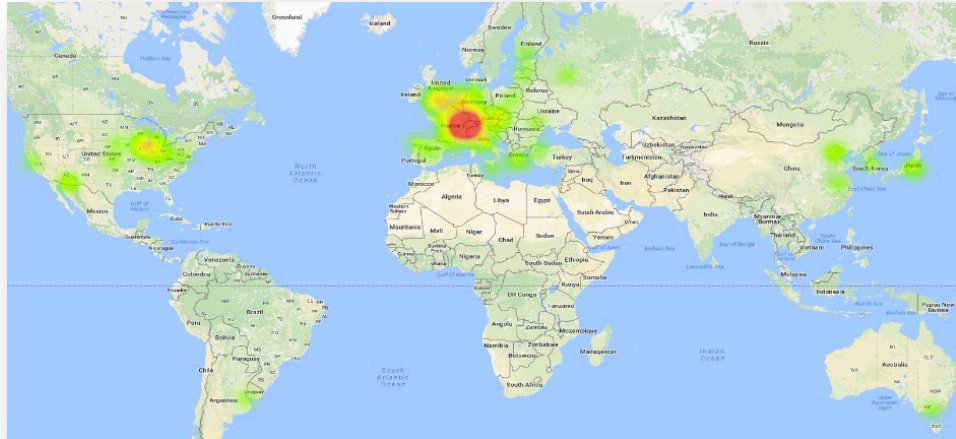
Looking into
Condor submission
too.



Indications of a Successful Model

The first 100 days of SWAN:

- 1800 logins, 3700 notebooks opened, peak of ~100 simultaneous active users
- Used by IT, EN, BE, EP – connections from all over the World
- Lots of feedback received from the users community





CernBOX Integration

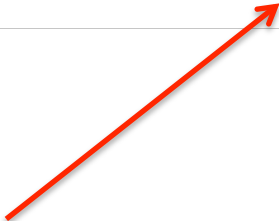
Files ▾ Help & Download Clients

All files

- ★ Favorites
- Shared with you
- Shared with others
- Shared by link
- ⚙ All projects
- 👤 Your projects

🏠 > + `init`

	Simple_ROOTbook_cpp.ipynb	Open in	🔗	⋮	486 kB	3 days ago
	Simple_ROOTbook_py.ipynb	Open in	🔗	⋮	326 kB	a day ago
	test.py		🔗	⋮	6 kB	a month ago
	Untitled.ipynb	Open in	🔗	⋮	< 1 kB	a day ago
	Untitled1.ipynb	Open in	🔗	⋮	< 1 kB	a day ago
	Untitled2.ipynb	Open in	🔗	⋮	< 1 kB	20 days ago





Nuclear Physics (From CHEP)

LHC experiments: tremendous success in achieving their analysis goals and producing results in timely manners

Lesson learned at LHC experiments:

- as the complexity and size of the experiments grew
- the complexity of analysis environment grew
- time dealing with the analysis infrastructure grew

- **Big Data** is not about size
- **Big Data** is about the ability to quickly analyze large amounts of data. i.e.

User centered design

- understand the user requirements first and foremost
- engage wider community of physicists in design whose primary interest is not computing
- make design decisions solely based on user requirements
- web-based user interfaces, e.g. interactive analysis in Jupyter Notebook

EIC: electron ion collider



- statistical language of **R**:
 - driven statistical analysis for more than a decade
 - emerging as the leader in statistical languages for **Big Data**
 - an alternative to ROOT?
 - NP should acquire some knowledge here (cooperate with other fields / industry)



People Counting on It!

urgent: figures do not open any more



Hello,

the figures suddenly stopped working. If I do a simple thing:

```
import matplotlib.pyplot as plt  
plt.figure()
```

Did you change anything in the last 10 min? I obtain the error message below. Could you please fix it asap as I have an analysis to finish tonight.

Thanks,

Note: in the end it was not a SWAN issue but rather matplotlib trying to interact with a non-existing X server.



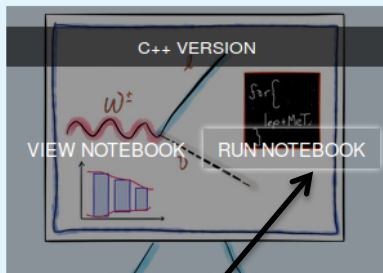
Atlas Opendata and SWAN

ATLAS ROOTbooks Gallery!

How deep can you go?

Analysis notebooks at

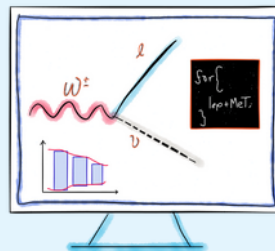
<http://opendata.atlas.cern/webanalysis/ROOTbooks.php>



Runnable in SWAN!

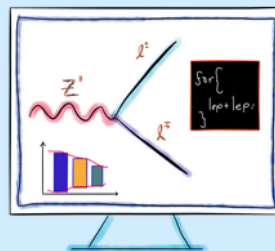
The W Analysis ROOTbook

The W boson analysis is intended to provide an example for a high statistics analysis using the ATLAS open data dataset. Furthermore it tests the description of the real data by the simulated W boson data which represents the most extensive dataset in terms of luminosity.



The Z Analysis ROOTbook

Many analyses selecting leptons suffer from Z + jets as a contributing background due to its large production cross section. It is therefore vital to check the correct modelling of this process by the Monte-Carlo simulated data. It is important to measure well known Standard Model particles, to confirm that we understand properly the detector and software. We are then ready to search for new physics.



We will use examples from: <http://swan.web.cern.ch/content/notebook-gallery>

The screenshot shows the SWAN website interface. At the top left is the SWAN logo, a stylized blue swan with a bar chart and a line graph. To its right, the text reads "SWAN Interactive Data Analysis, in the Cloud." Below the logo is a navigation menu with buttons for "Home", "Galleries", "FAQ", and "Talks and Publications". Underneath, there are sub-menus for "Basic", "ROOT Primer", "Accelerator Complex", "Machine Learning", and "Apache Spark". The main heading is "Basic Examples". A paragraph follows: "This is a gallery of basic example notebooks: click on the images to inspect the underlying document, open in SWAN the single notebooks or the full git repository!" Below this is a button labeled "Open in SWAN" with the SWAN logo. A second paragraph states: "Many of the notebooks are ROOTbooks, based on the ROOT framework. To know more about ROOT, visit root.cern.ch." Two example notebooks are displayed side-by-side. The left one is titled "Simple ROOTbook (Python)" and shows a histogram plot with a blue fill and a grid. A blue diagonal banner on the right of the plot says "Open in SWAN". The right one is titled "Simple ROOTbook (C++)" and shows a similar histogram plot with a blue diagonal banner on the right that says "Open in SWAN". Both plots include a statistics box with values for Entries, Mean, and Std Dev.

SWAN
Interactive Data Analysis, in the Cloud.

Home Galleries FAQ Talks and Publications

Basic ROOT Primer Accelerator Complex Machine Learning Apache Spark

Basic Examples

This is a gallery of basic example notebooks: click on the images to inspect the underlying document, open in SWAN the single notebooks or the full git repository!

Open in SWAN

Many of the notebooks are ROOTbooks, based on the ROOT framework. To know more about ROOT, visit root.cern.ch.

Simple ROOTbook (Python)

My Histo

Entries	1000
Mean	0.02060
Std Dev	1.038

Open in SWAN

Simple ROOTbook (C++)

My Histo

Entries	1000
Mean	0.02060
Std Dev	1.038

Open in SWAN



Conclusions

- ROOT has been integrated with Jupyter Notebooks
 - Both C++ and “PyROOT” notebooks available
 - Added value: inline interactive graphics, tab completion, interactive ML...
- ROOT is available within SWAN
 - CERN’s Notebook Service
 - Data on EOS, software on CVMFS, home in CERNBox

All available in
ROOT 6.08 !

For all CERN
users

How can ROOTBooks and SWAN be an useful complement to the already existing ALICE (analysis) workflows?



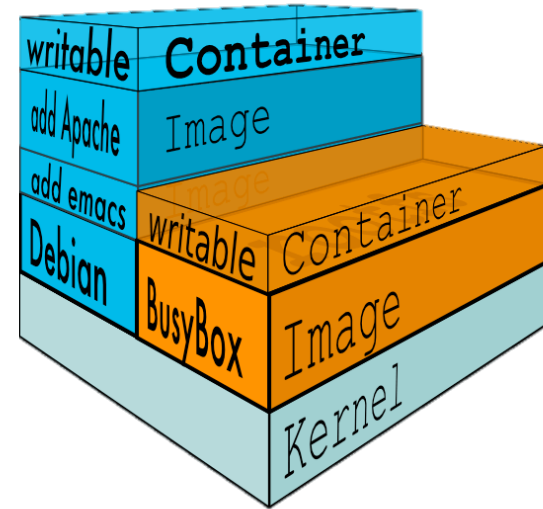
Backup

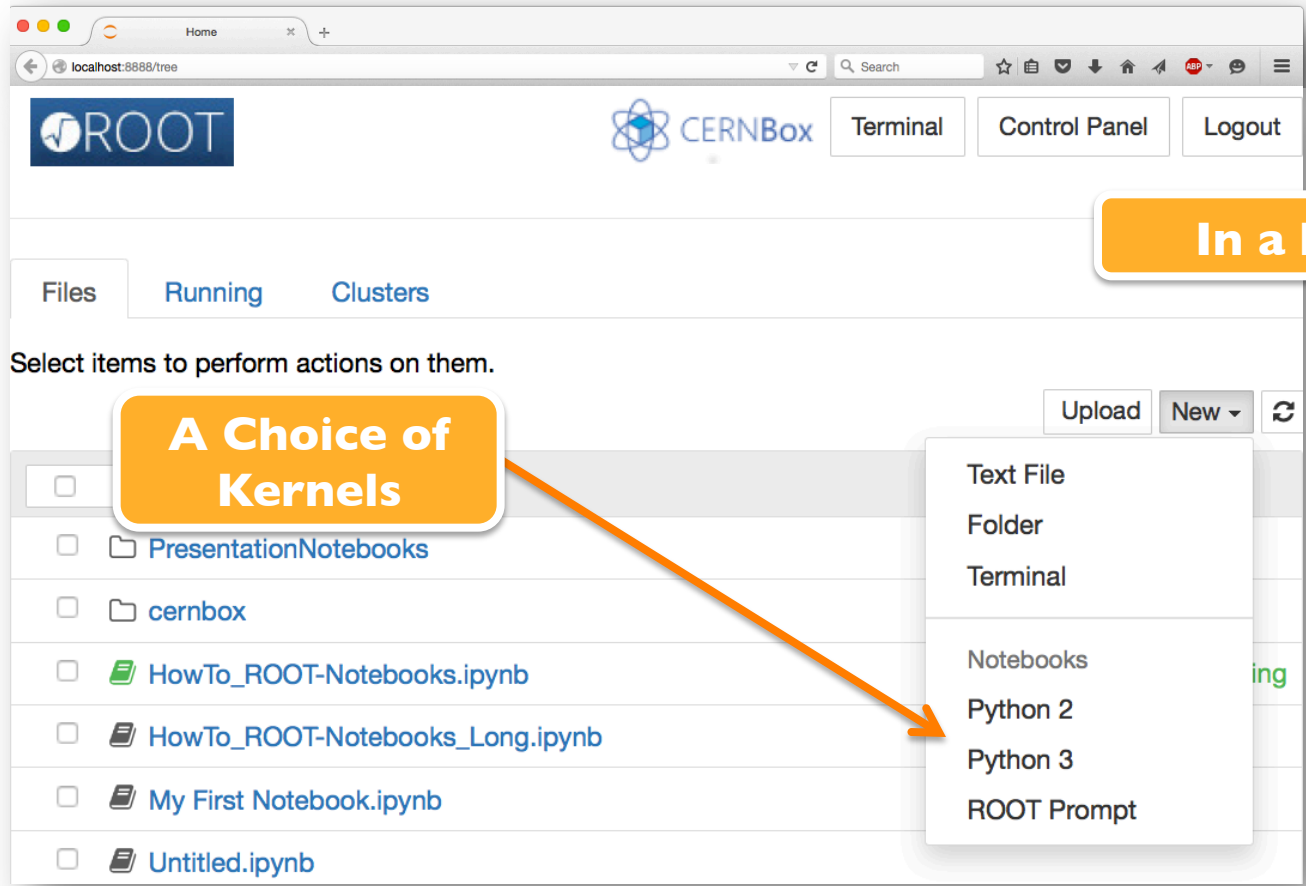


Docker Container

Transparent to the SWAN users!

- A “Light-weight virtual machine”
- Complete isolation of users: many linux systems sharing the same kernel
- Works on **OSx and Windows** too
 - Need VM in the background to run the Kernel!
- Openstack support





In a browser

A Choice of Kernels

- Text File
- Folder
- Terminal
- Notebooks
- Python 2
- Python 3
- ROOT Prompt

Kernels are processes that run interactive code in a particular programming language and return output to the user. Kernels also respond to tab completion and introspection requests.

The image shows a web browser window displaying a Jupyter Notebook interface. The browser's address bar shows 'localhost:8888/tree'. The page title is 'Notebook Functionalities'. The interface includes a 'ROOT' logo, a search bar, and buttons for 'Control Panel' and 'Logout'. A menu bar contains 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. Below the menu bar is a toolbar with icons for file operations and cell execution. The 'Kernel' dropdown menu is open, showing 'Python 2' selected and circled in red. The main content area displays a welcome message and a LaTeX formula.

localhost:8888/tree

ROOT Notebook 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 and
Formulas**

The image shows a Jupyter Notebook interface in a web browser. The browser address bar shows 'localhost:8888/tree'. The notebook title is 'Notebook Functionalities'. The interface includes a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. Below the menu bar is a toolbar with icons for file operations and cell execution. The main content area displays a large heading 'Welcome to the Notebook Technology'. Below the heading, there is a text block: 'This is a markdown cell. You can add LaTeX code: $\sum_{n=-\infty}^{\infty} |x(n)|^2$ '. At the bottom, there is a code cell with the following Python code:

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

Code

ROOT Notebook 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
```

→ This is a notebook in Python

Code

localhost:8888/tree

ROOT Notebook 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
```

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

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

Code

```
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
```



We can invoke commands in the shell...

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 capture their output

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
```

```
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 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
```

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

```
Out[4]:
```



Images

In a browser

```
function():  
42
```

Text and
Formulas

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

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

Code

```
rootaasdemo.web.cern.ch/rootaasdemo/SaaSfee.jpg \  
> SF.jpg
```

% Total Time	% Received Time	% Xferd Current	Average Speed	Time
Spent	Left	Speed		
100 128k	100 128k	0	0 2787k	0 00:00:00
---:---:--	---:---:--	2787k		

Shell Commands

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

```
Out[4]:
```



Images