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

This Talk

Explore the new Jupyter interface of ROOT and the SWAN service

Part 1:

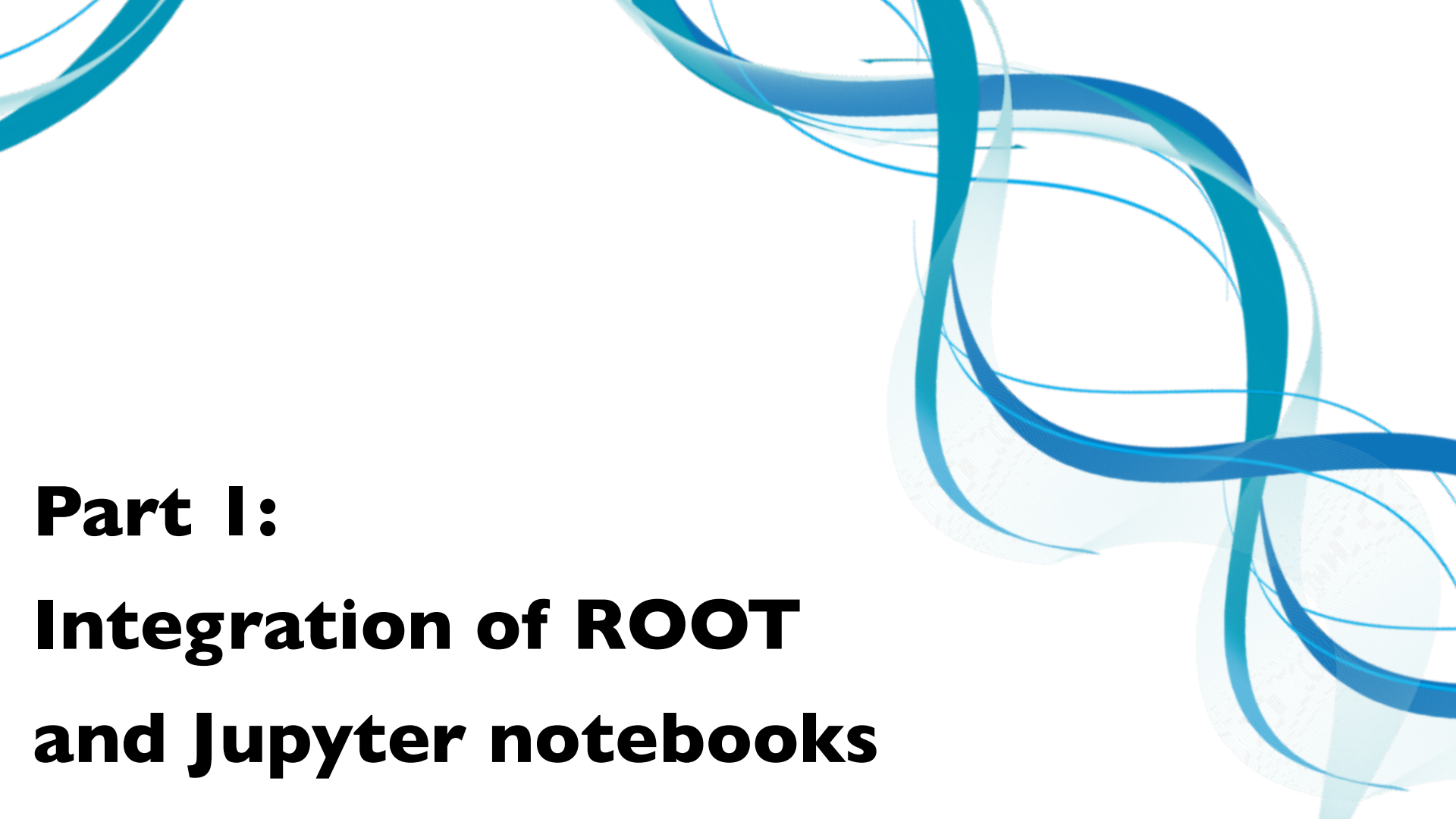
- Introduction to the Jupyter Notebooks
- ROOT integration: C++ and Python kernels
- Added value of this synergy

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 1:
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

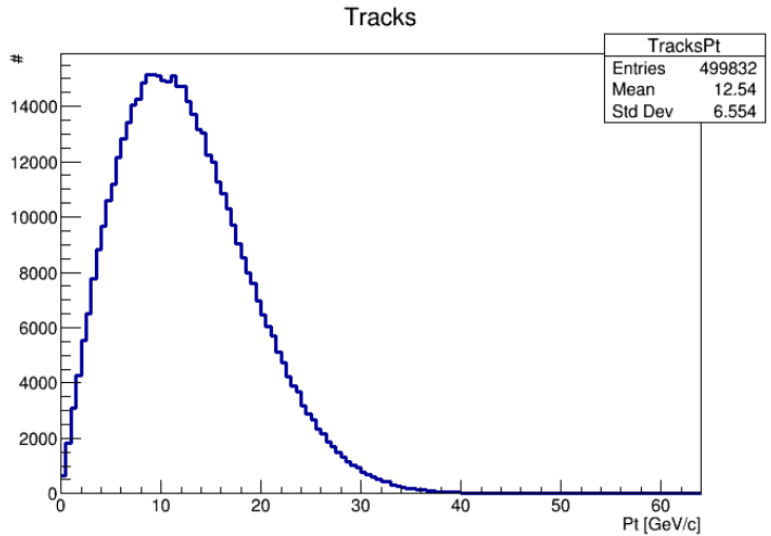
In a Browser

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()
```



Motivation

- 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

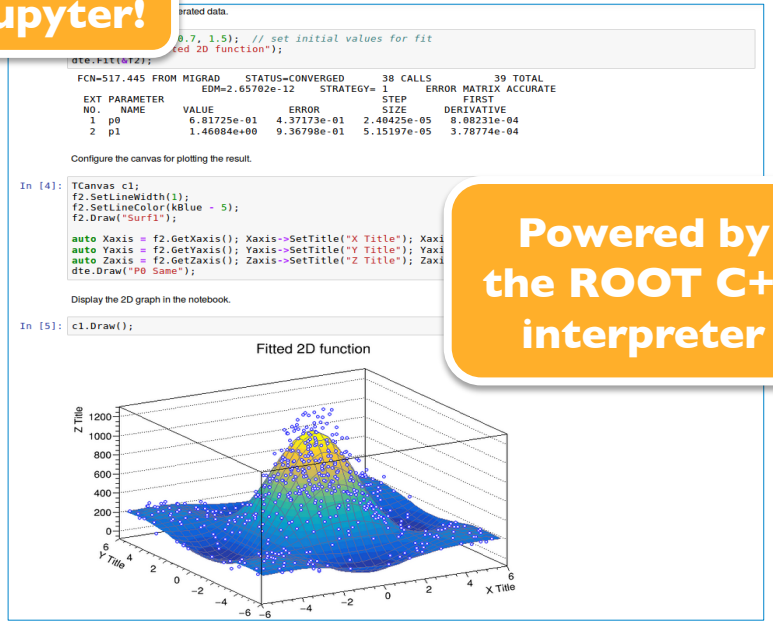
The ROOTBooks

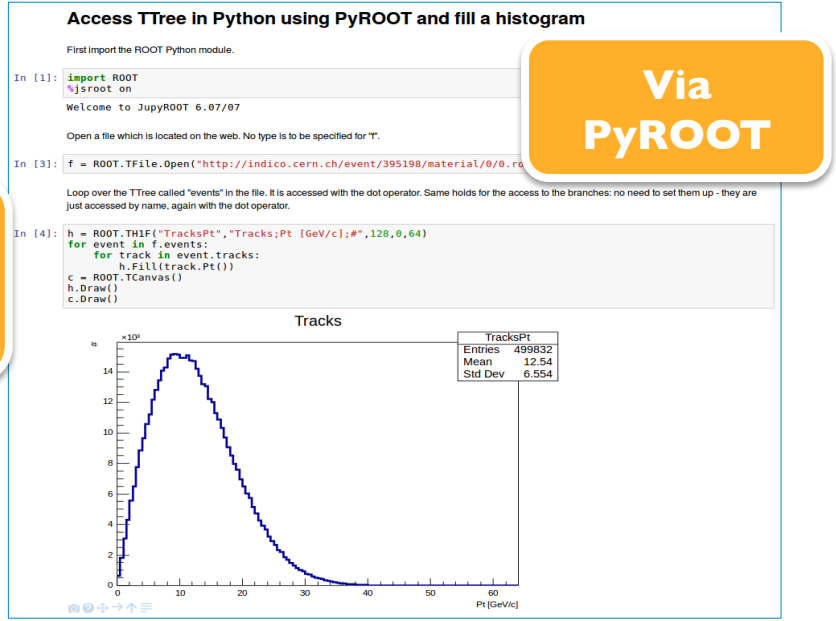
- Two language flavours (a.k.a. kernels) are available:

New in Jupyter!

Powered by the ROOT C++ interpreter

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; }
};
```

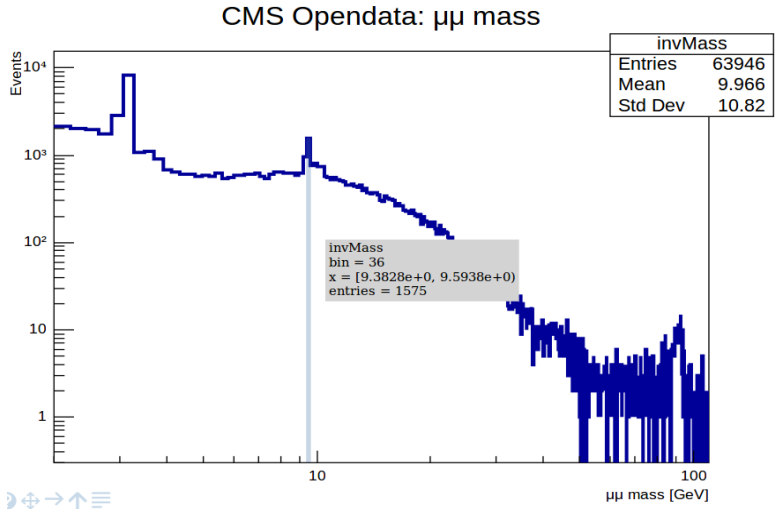
%%python also available in C++ notebooks

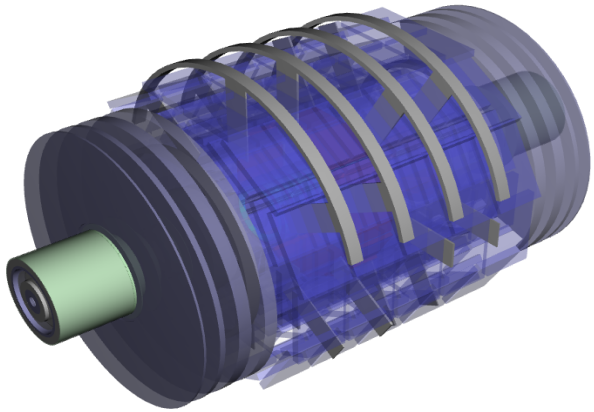
In [3]:
```
a = ROOT.A()
```

```
Constructor of A!
```

Interactive Graphics: JSROOT

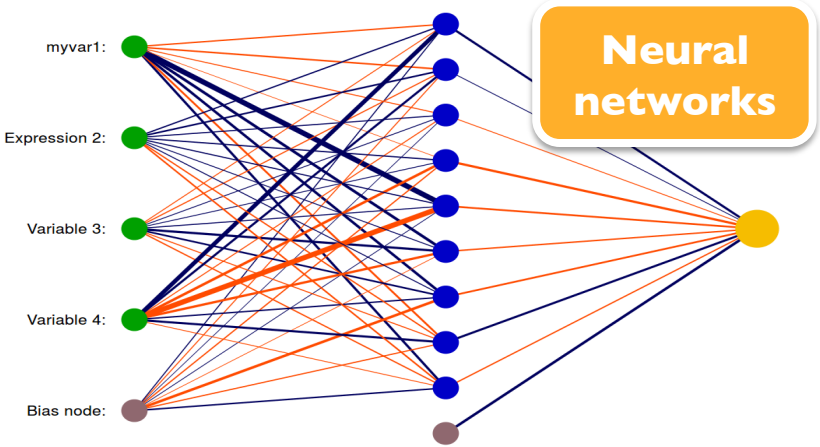
- Both flavours (C++ and Python) allow to **inline ROOT graphics** in a notebook
- Two modes: static image and JavaScript visualisation
  - Activate JSROOT mode with **%jsroot on**
  - Interact with your plot: zoom, modify axis, inspect bins, …

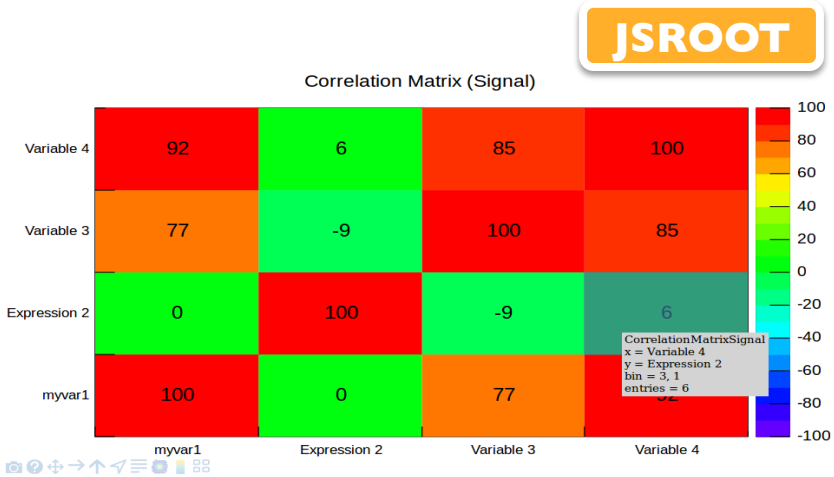




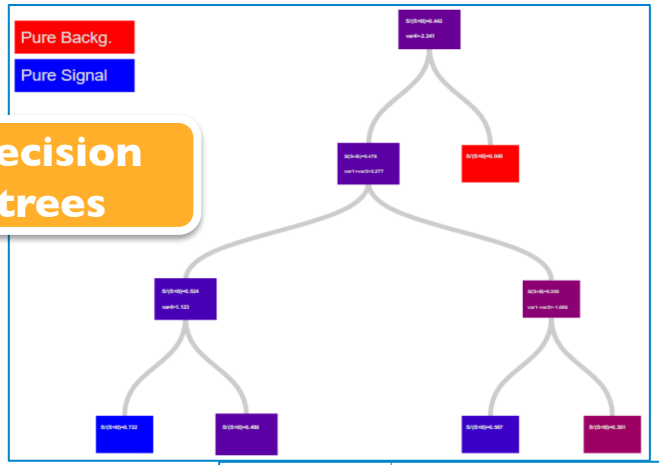
Interactive Machine Learning

- 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

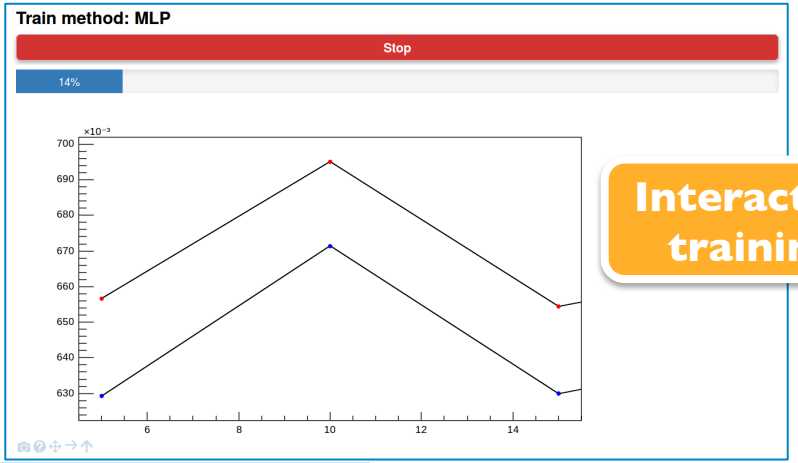




More Added Value



Decision trees



Interactive training

DataSetInfo	Correlation matrix (Signal)
DataSetInfo	Correlation matrix (Background)
DataSetFactory	**Dataset: tmva_class_example**
TFHandler_MLP	

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

	Training Network
	Elapsed time for training with 6000 events : 4.45 sec
MLP	**Dataset: tmva_class_example** Evaluation of MLP on training sample (6000 events)
	Elapsed time for evaluation of 6000 events : 0.0187 sec
	Creating xml weight file: tmva_class_example/weights/TMVAClassification_MLP.weights.xml
	Creating standalone class: tmva_class_example/weights/TMVAClassification_MLP.class.C
	Write special histos to file: TMVA.root:/tmva_class_example/Method_MLP/MLP

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 **candleplot.C**
View Notebook Open in SWAN Example of candle plot with 2-D histograms.

file **candleplotoption.C**
View Notebook Open in SWAN Example showing how to combine the various candle plot options.

file **candleplotstack.C**
View Notebook Open in SWAN Example showing how a **THStack** with candle plot option.

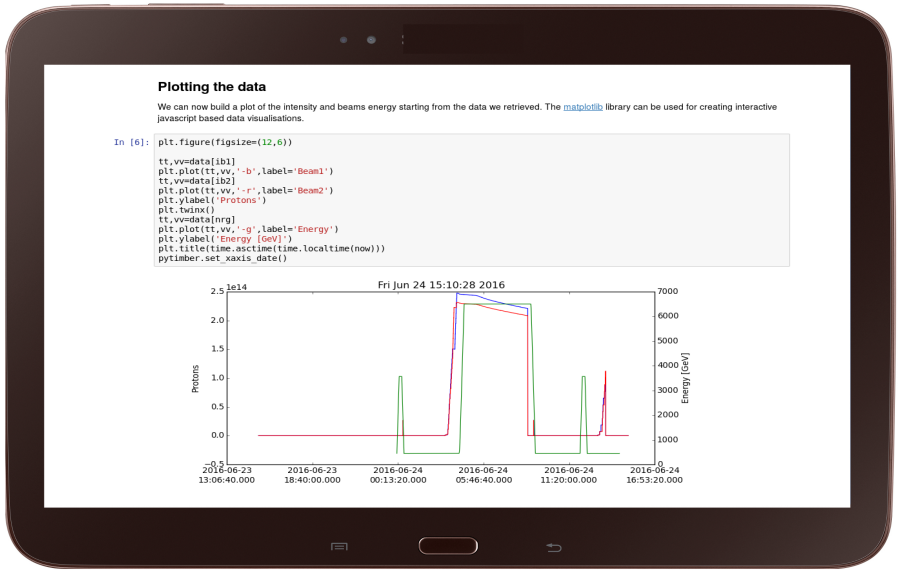
file **candleplotwhiskers.C**
View Notebook Open in SWAN Example of candle plot showing the whiskers definition.

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

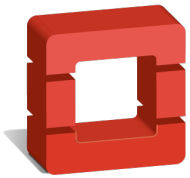
Data Analysis as a Service

- 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

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

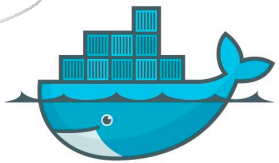


A coherent view at CERN

The “home directory” in SWAN

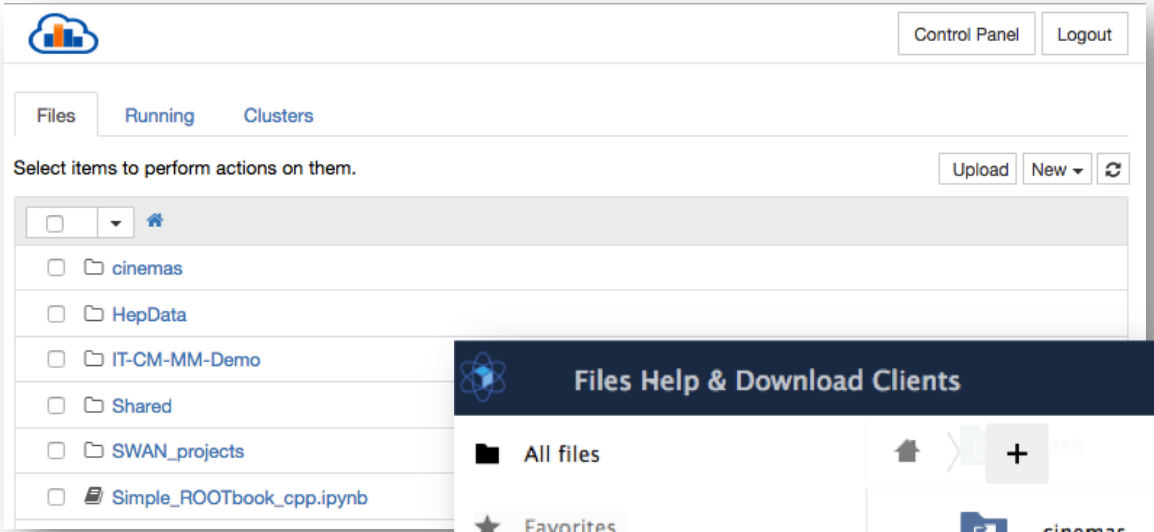
External but mainstream technologies

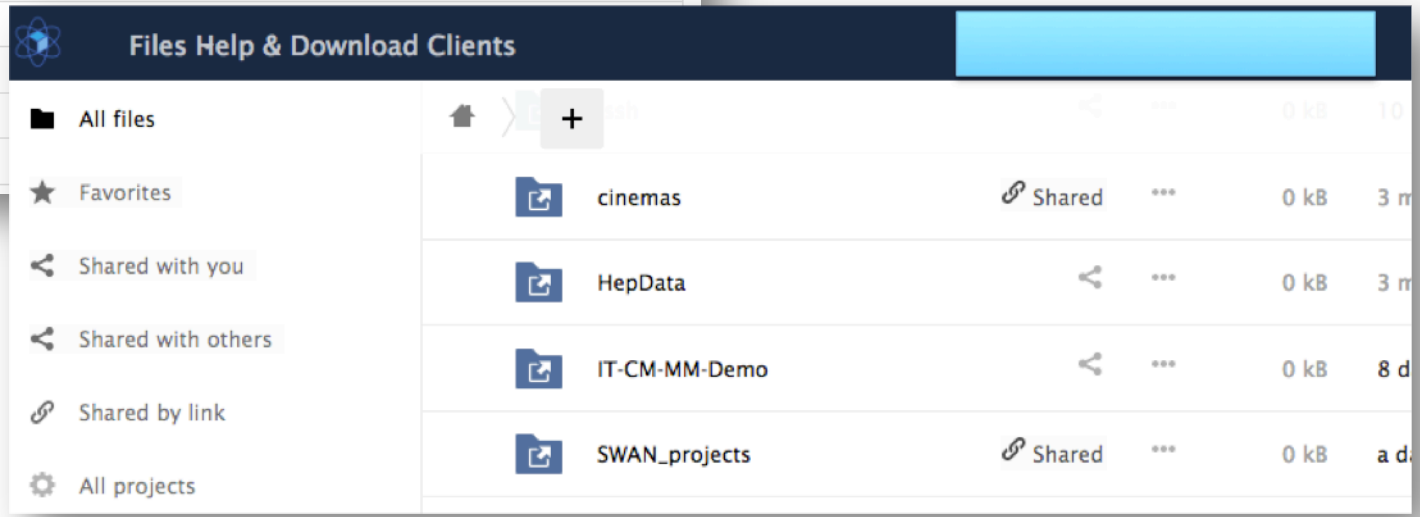
- Jupyterhub
- Docker



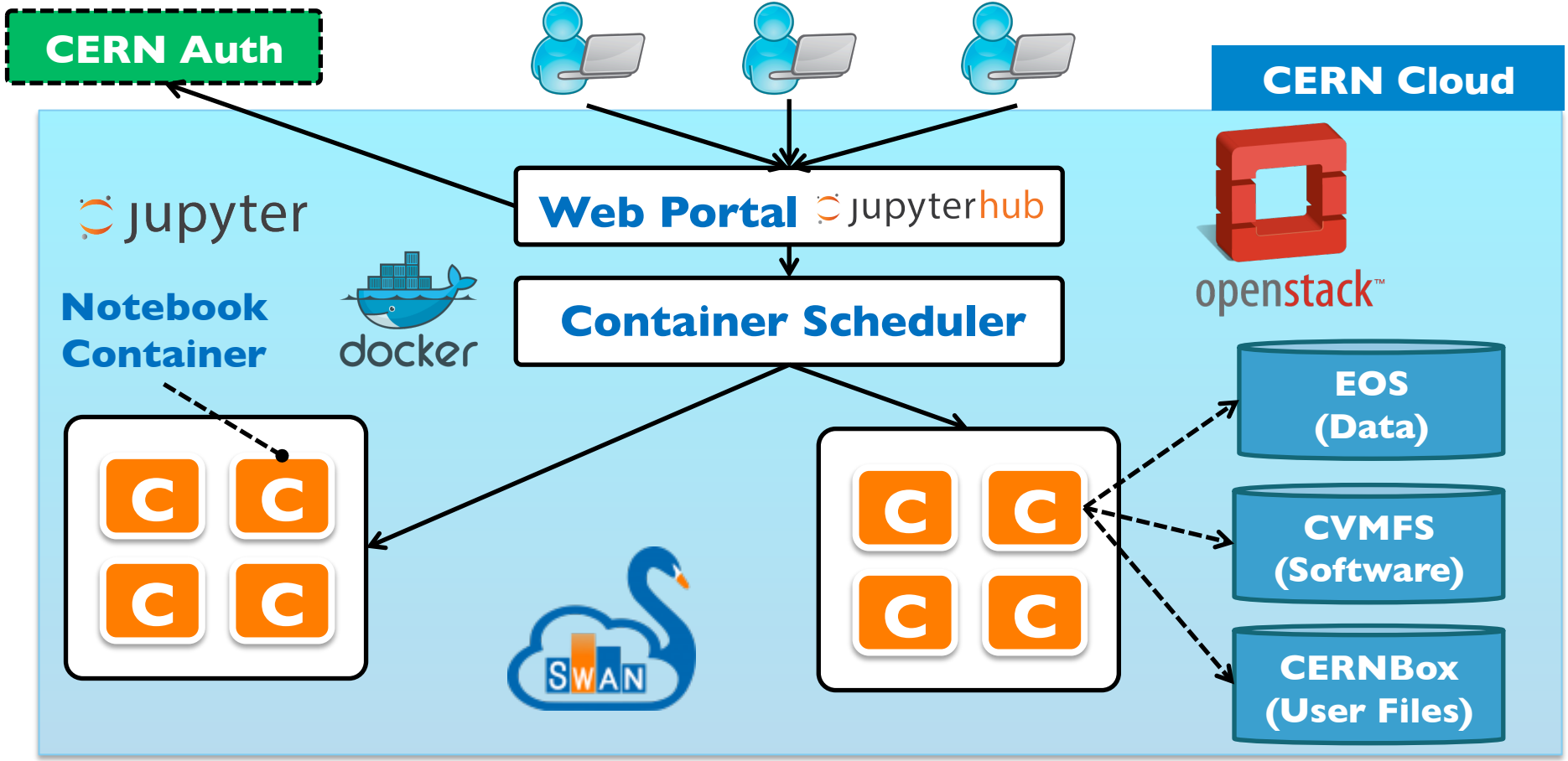
Both have large user bases and an active community behind

CERNBox is Your Home



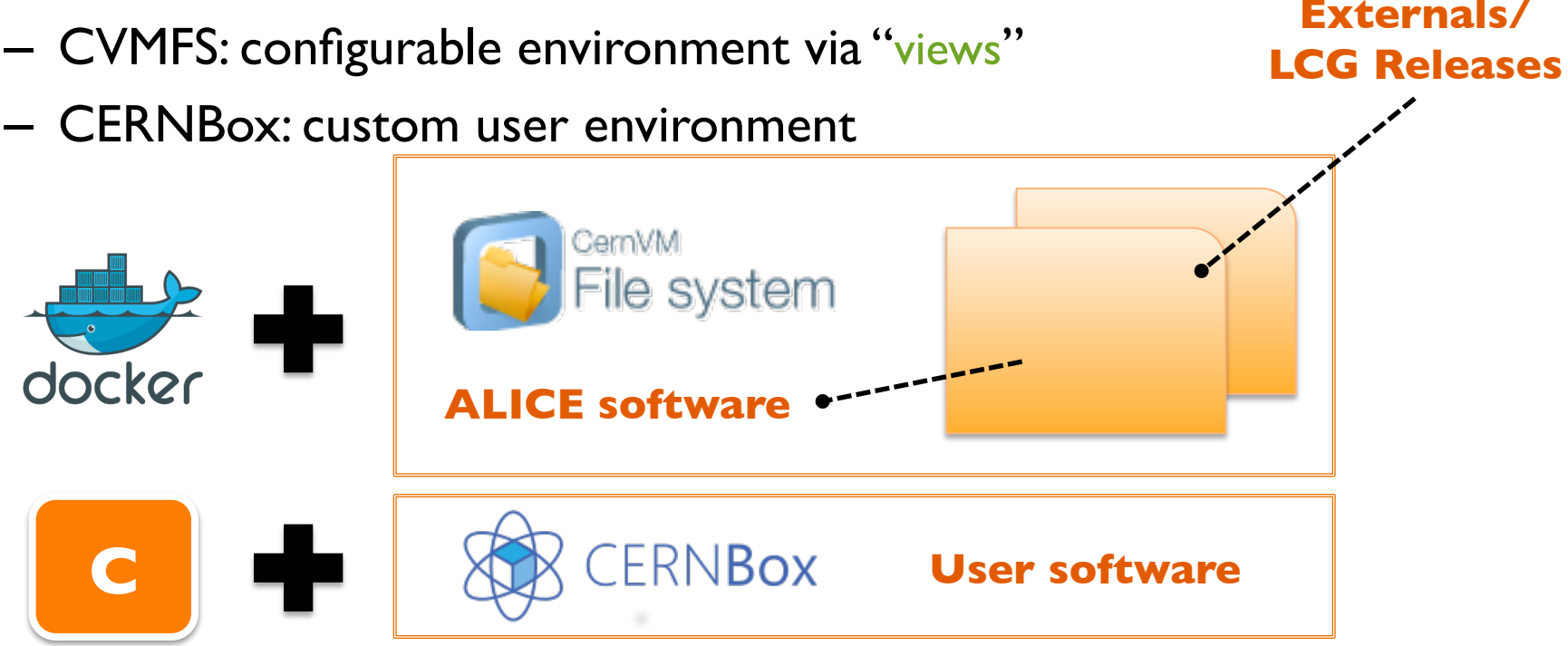


The Architecture

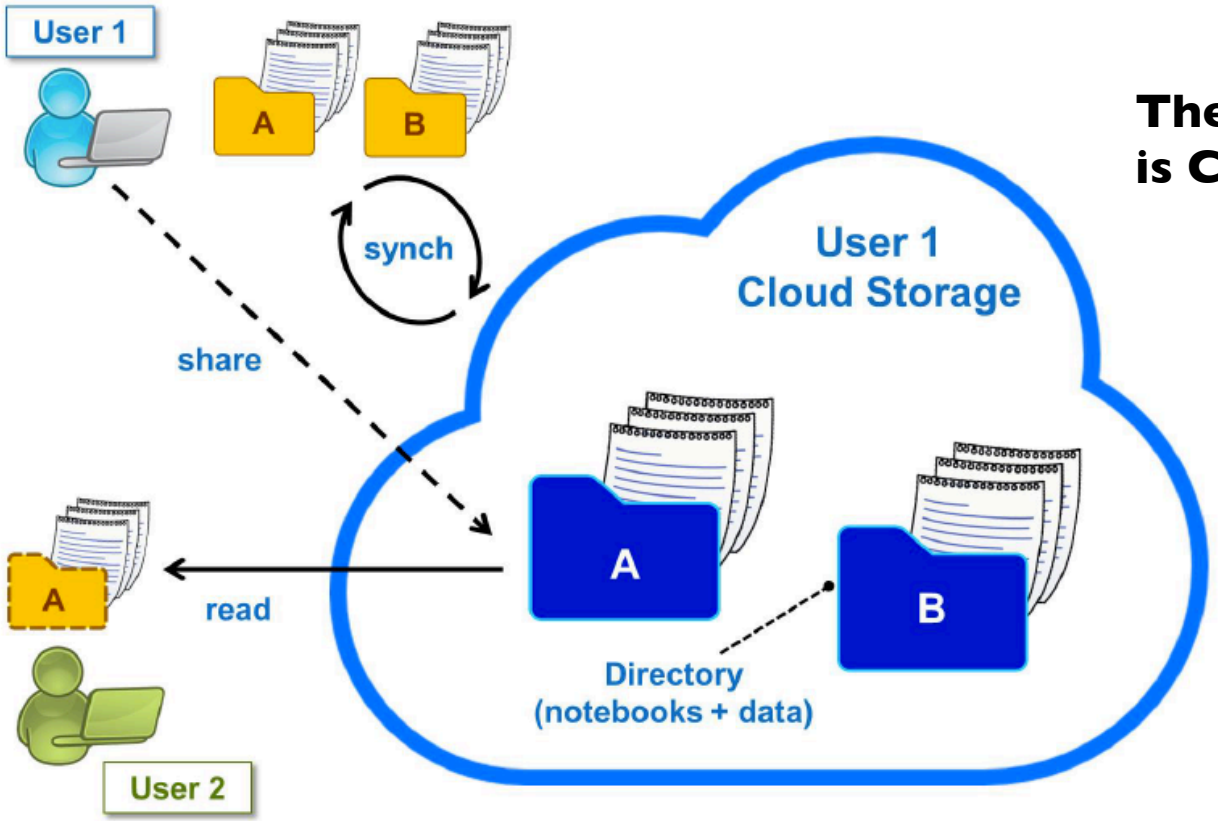


Software Environment

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

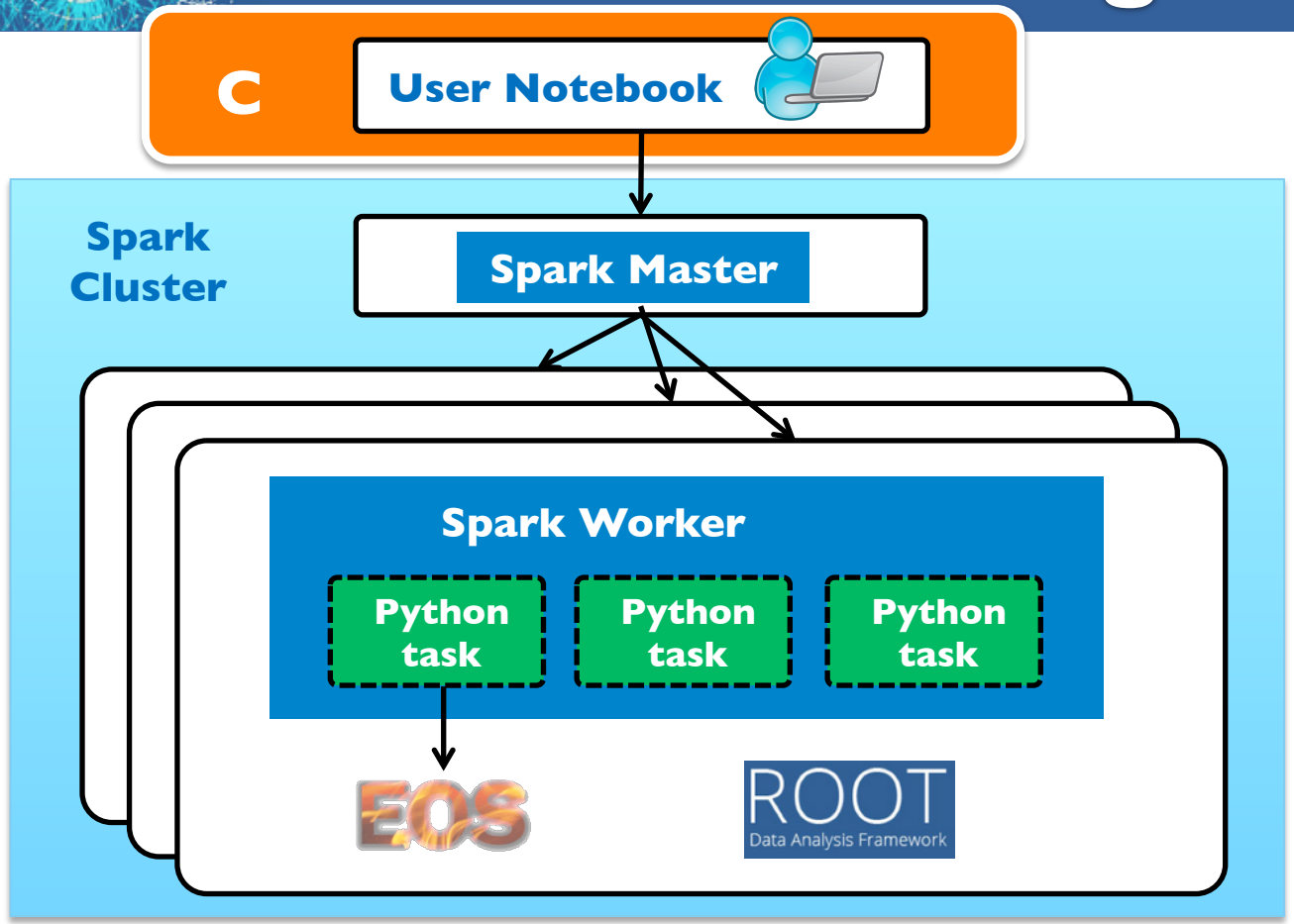


Cloud Based Analysis Model



The home directory is CERNBox

Offloading Calculations



In collaboration with IT-DB, IT-ST

Spark

R&D:

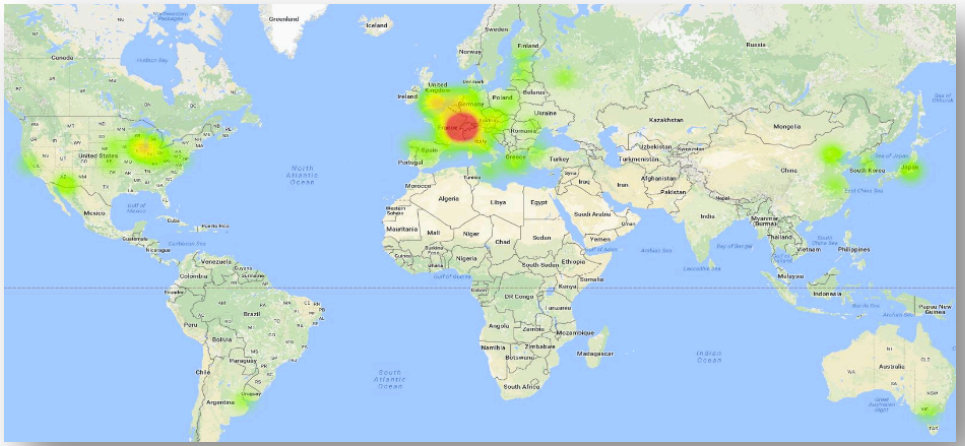
Not in production yet

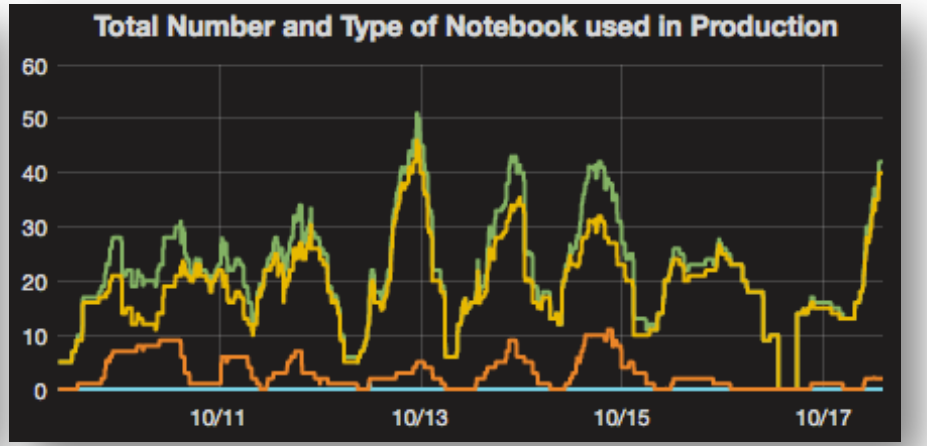
Looking into Condor submission too.

Indications of a Successful Model

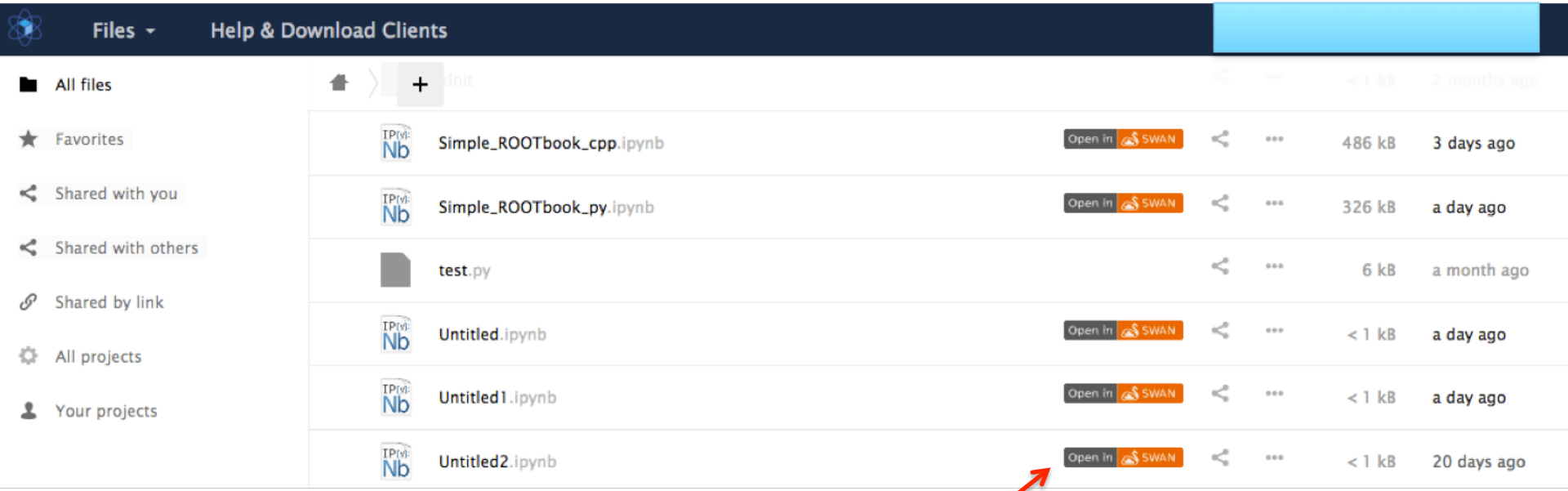
The first100 days of SWAN:

- 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



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

M. Diefenthaler, Oral 610

- 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,

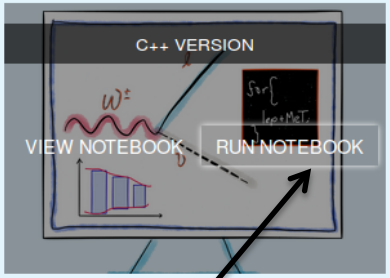
Note: in the end it was not a SWAN issue but rather matplotlib trying to interact with a non-existing X server.

Atlas Opendata and SWAN

ATLAS ROOTbooks Gallery!

How deep can you go?

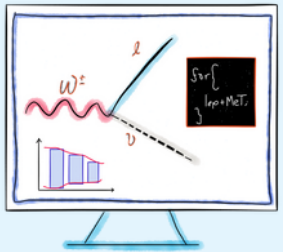
Analysis notebooks at
http://opendata.atlas.cern/webanalysis/ROOTbooks.php

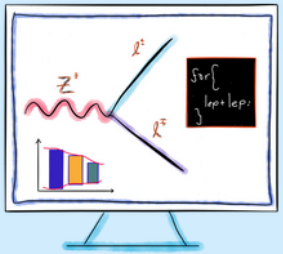


Runnable in SWAN!

The W Analysis ROOTbook

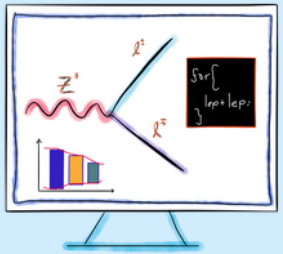
The W boson analysis is intended to provide an example for a high statistics analysis using the ATLAS open data dataset. Furthermore it tests the description of the real data by the simulated W boson data which represents the most extensive dataset in terms of luminosity.





The Z Analysis ROOTbook

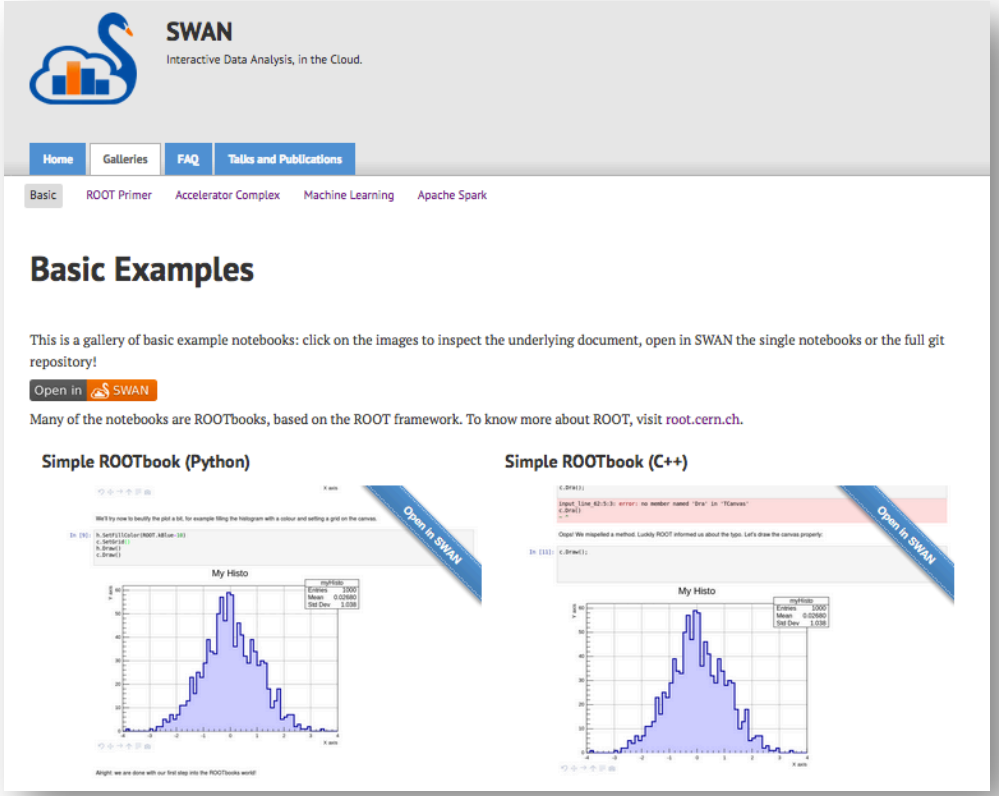
Many analyses selecting leptons suffer from Z + jets as a contributing background due to its large production cross section. It is therefore vital to check the correct modelling of this process by the Monte-Carlo simulated data. It is important to measure well known Standard Model particles, to confirm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



open data ATLAS

Quick Demo

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



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

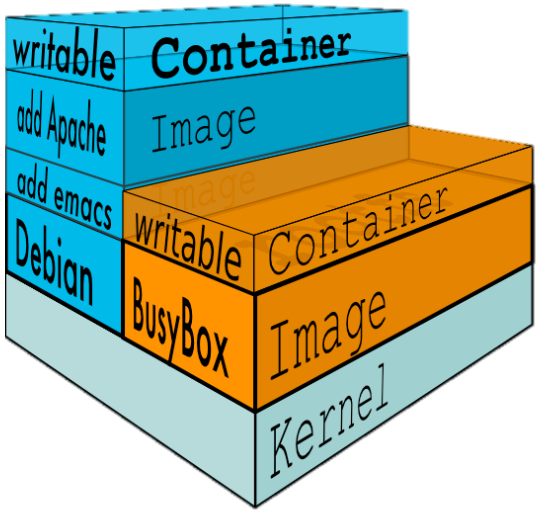
How can ROOTBooks and SWAN be an useful complement to the already existing ALICE (analysis) workflows?

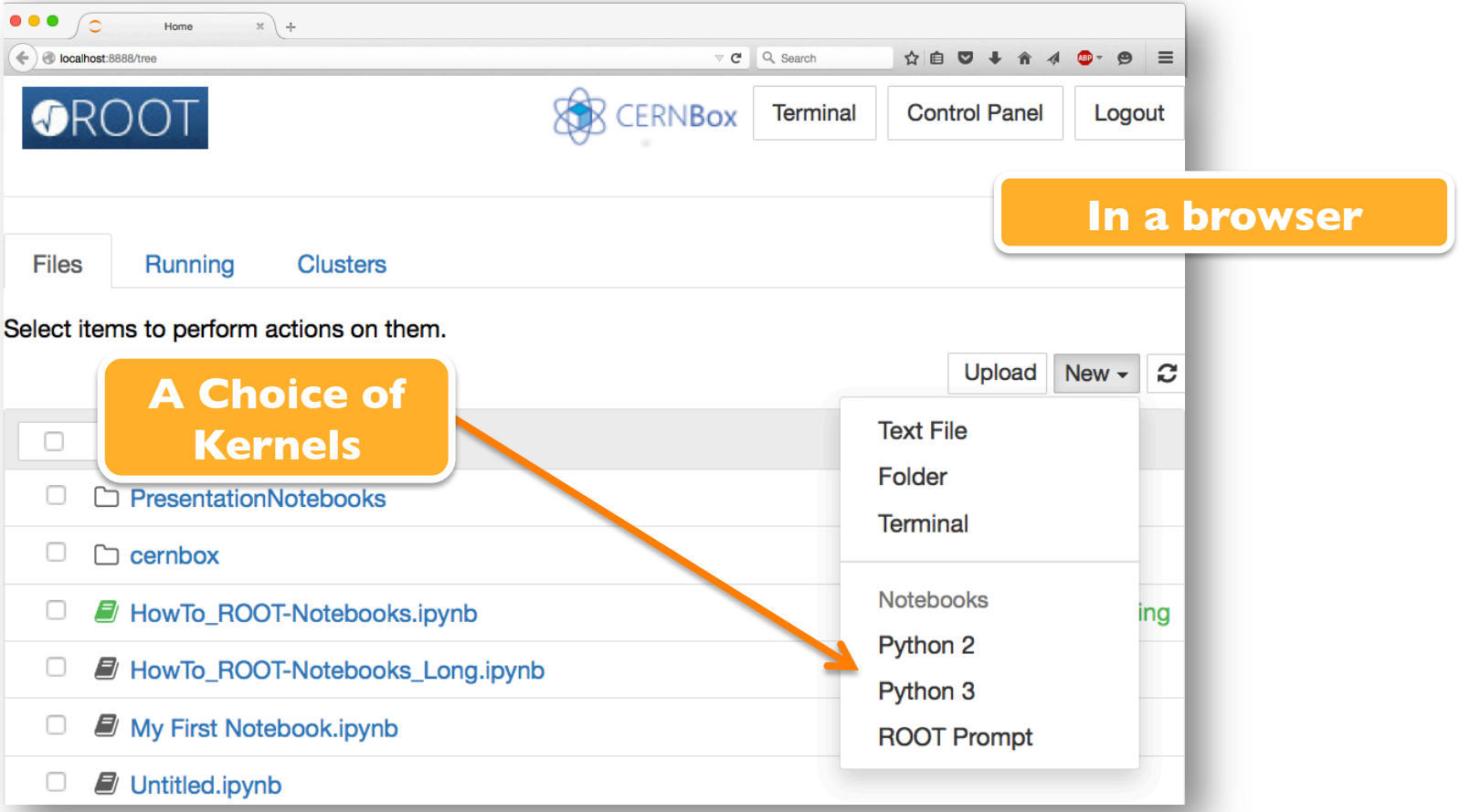
Backup

Docker Container

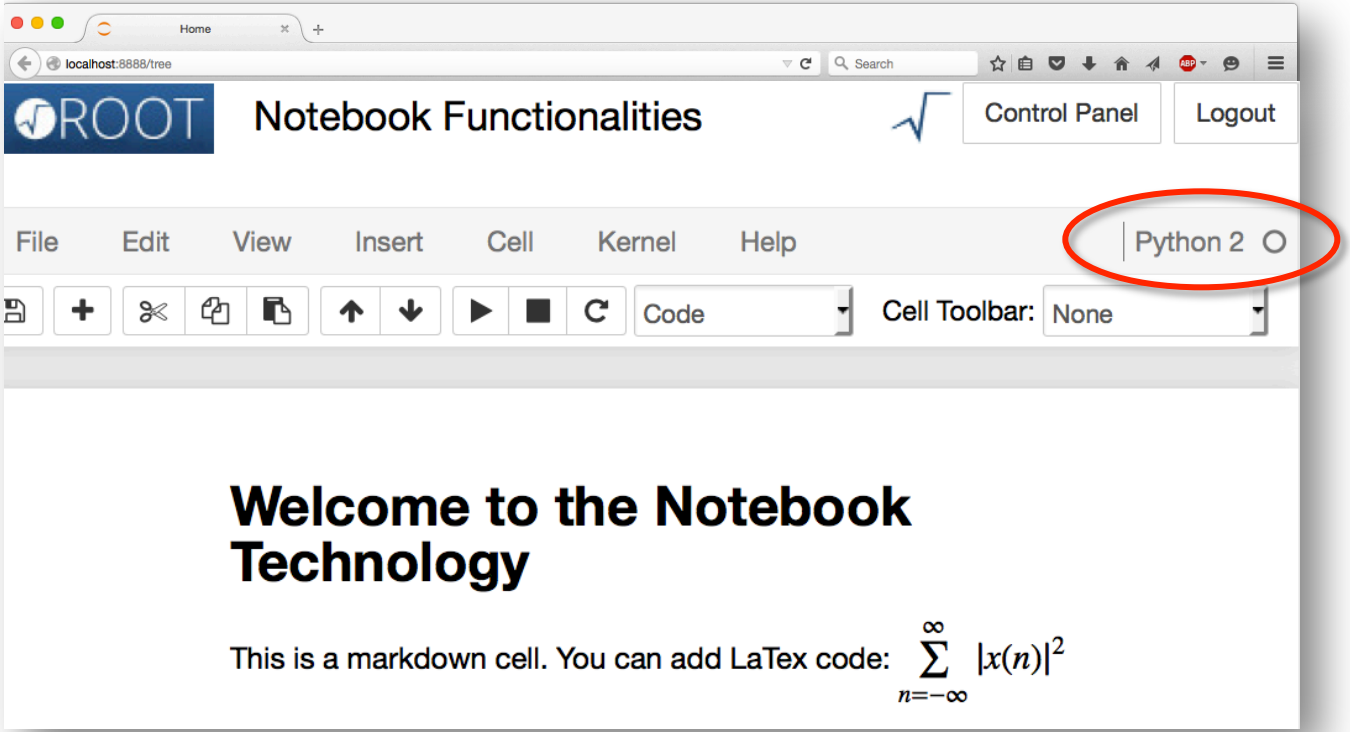
- A "Light-weight virtual machine"
- Complete isolation of users: many linux systems sharing the same kernel
- Works on OSx and Windows too
  - Need VM in the background to run the Kernel!
- Openstack support

Transparent to the SWAN users!

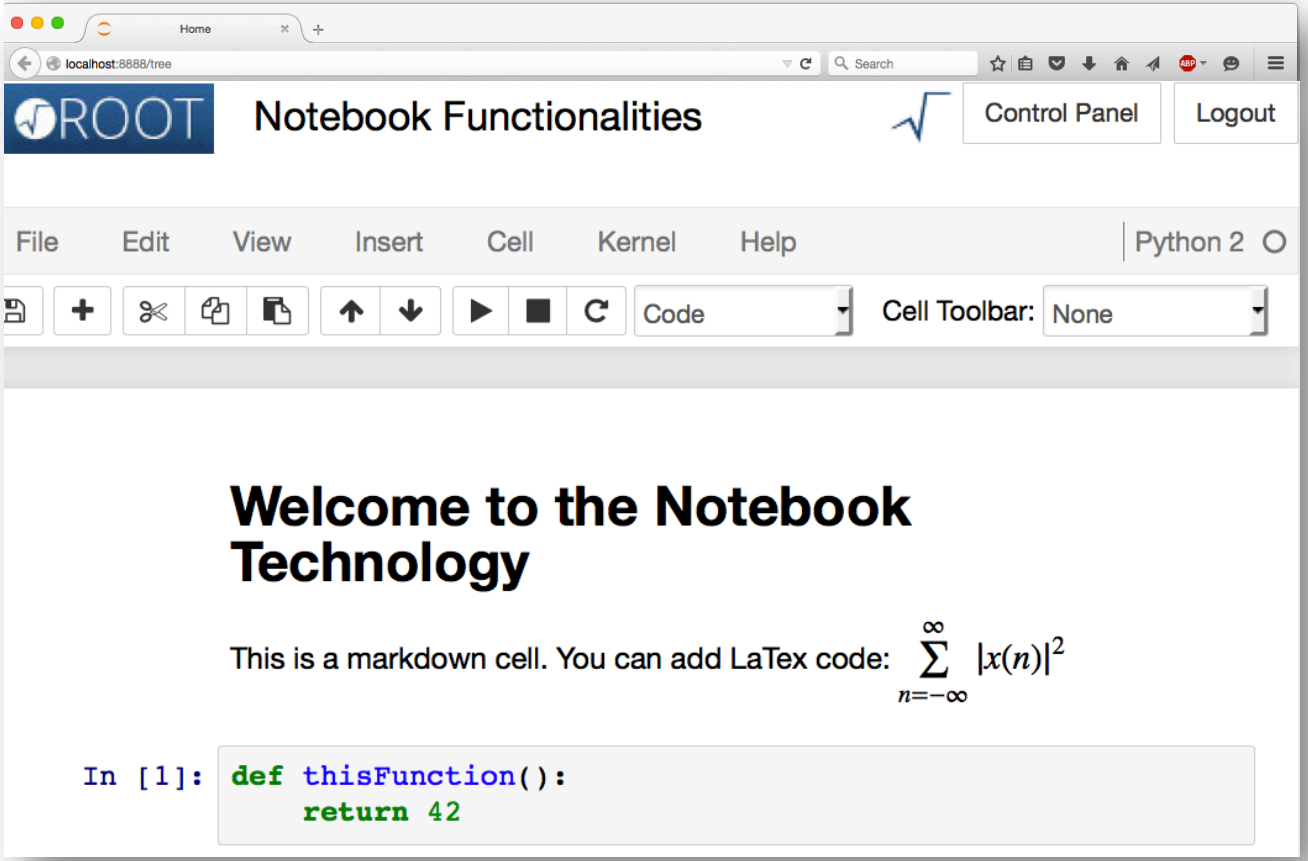




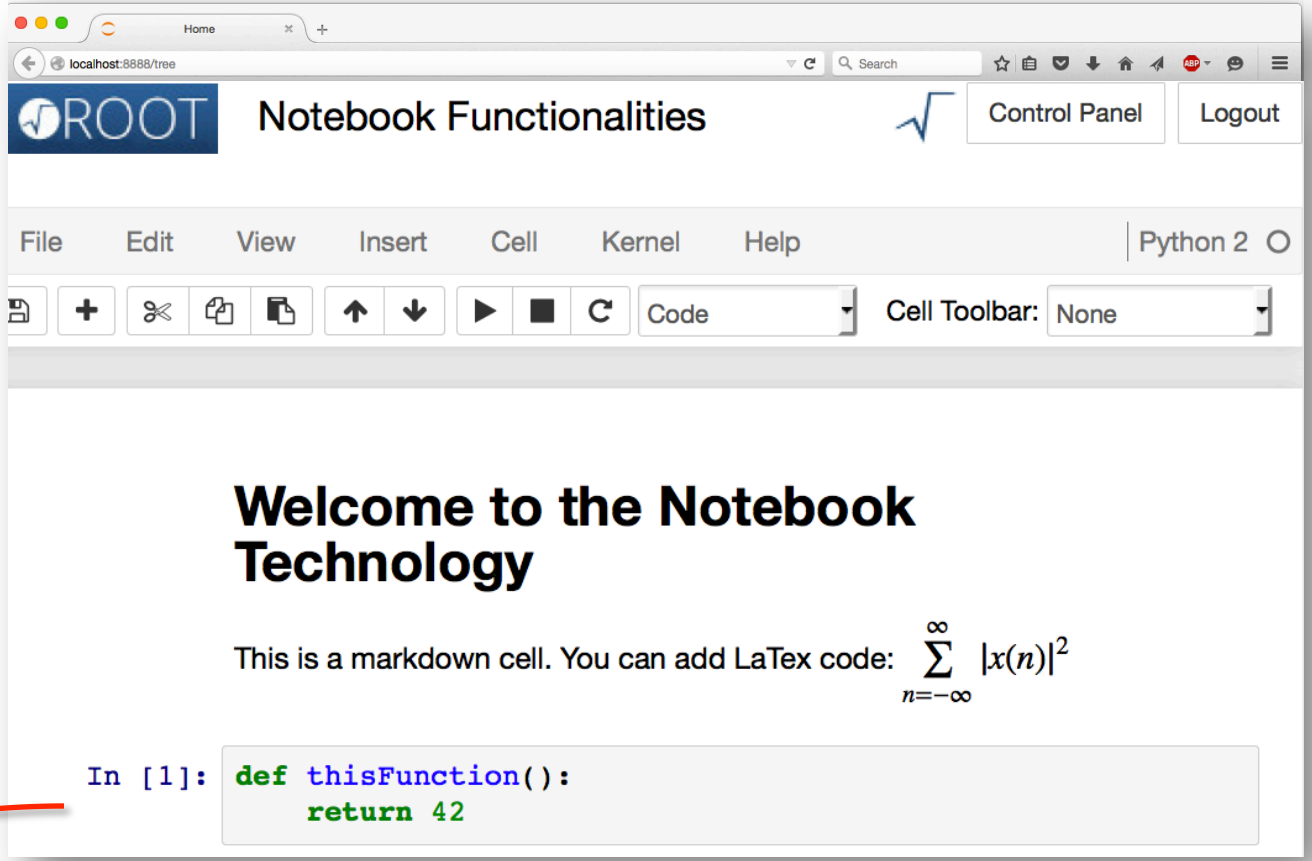
Kernels are processes that run interactive code in a particular programming language and return output to the user. Kernels also respond to tab completion and introspection requests.



Text and Formulas

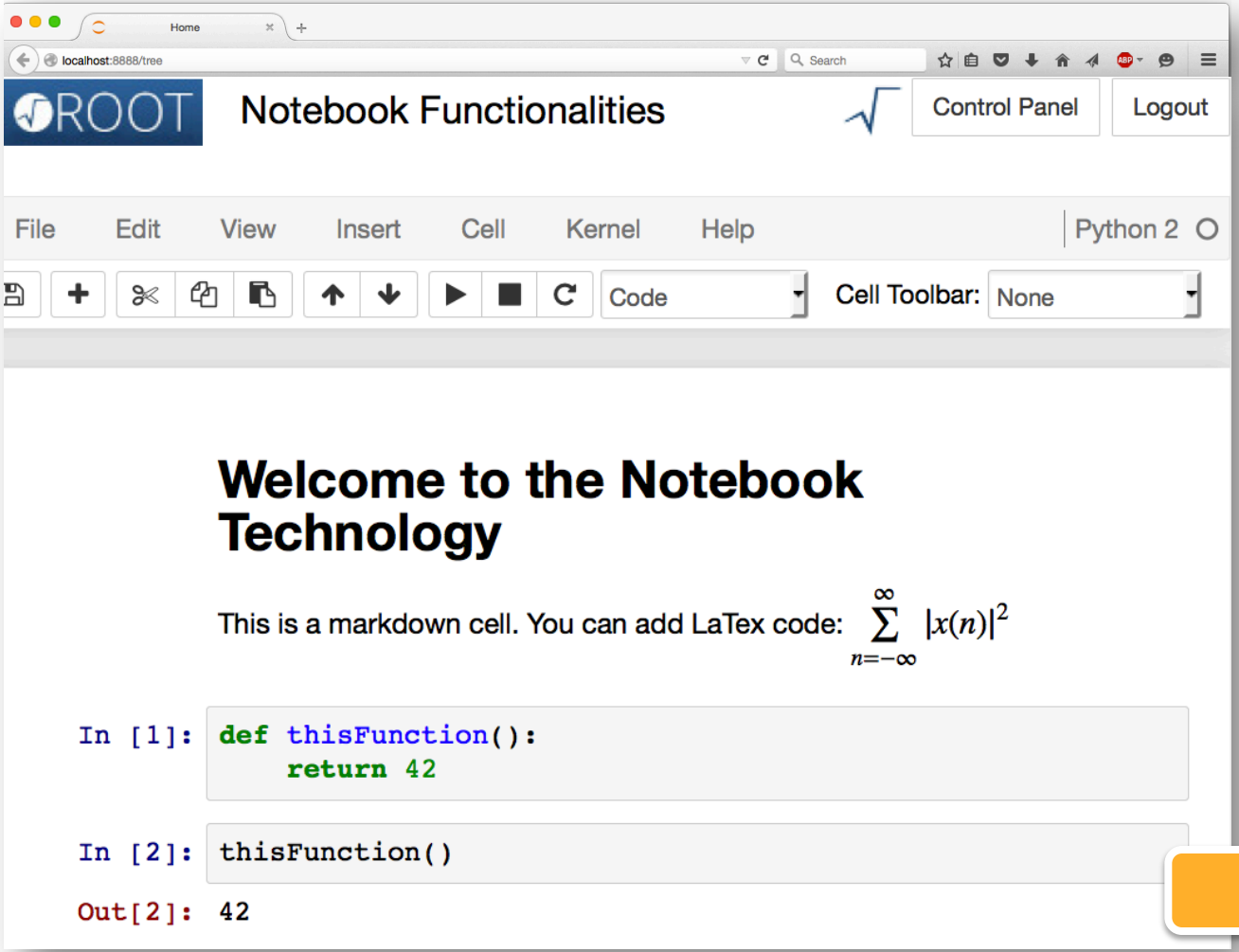

Home
localhost:8888/tree
Search
ROOT
Notebook Functionalities
Control Panel
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Python 2
Code
Cell Toolbar:
None
Welcome to the Notebook Technology
This is a markdown cell. You can add LaTex code: $\sum_{n=-\infty}^{\infty} |x(n)|^2$
In [1]:
def thisFunction():
return 42

Code



This is a notebook in Python

Code


Home
localhost:8888/tree
Search
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

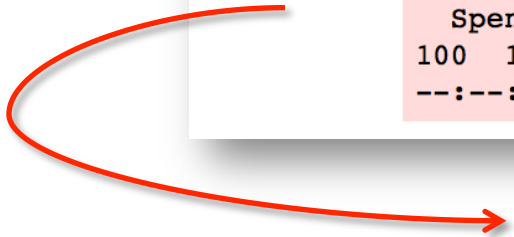
```
In [1]: def thisFunction():
            return 42

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

Text and Formulas

```
In [2]: thisFunction()

Out[2]: 42
```

Code

```
> SF.jpg
```

```
  % Total    % Received % Xferd  Average Speed   Time
  Time     Time  Current

  Spent    Left  Speed
100  128k  100  128k    0
--:--:-- --:--:-- 2787k
```

Shell Commands

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

Out[4]:

Images