Data Mining as a Service
DMaaS

P. Mato, D. Piparo, E. Tejedor – EP-SFT
M. Lamanna, L. Mascetti, J. Moscicki – IT-ST

Cloud Services for Synchronisation and Services (CS3)

18/01/2016

Objectives of this Talk

- Describe a distributed service offering a web interface for data analysis based on *Jupyter Notebooks*
- Demonstrate how provision of CPU and storage resources as well as software are its building blocks
  - Focus on sync'd and mass storage
- Illustrate with a demo analysis how it can boost productivity and give access to innovative workflows
- Give you the possibility to try it out!

Prelude:
The “Notebook”

The Notebook

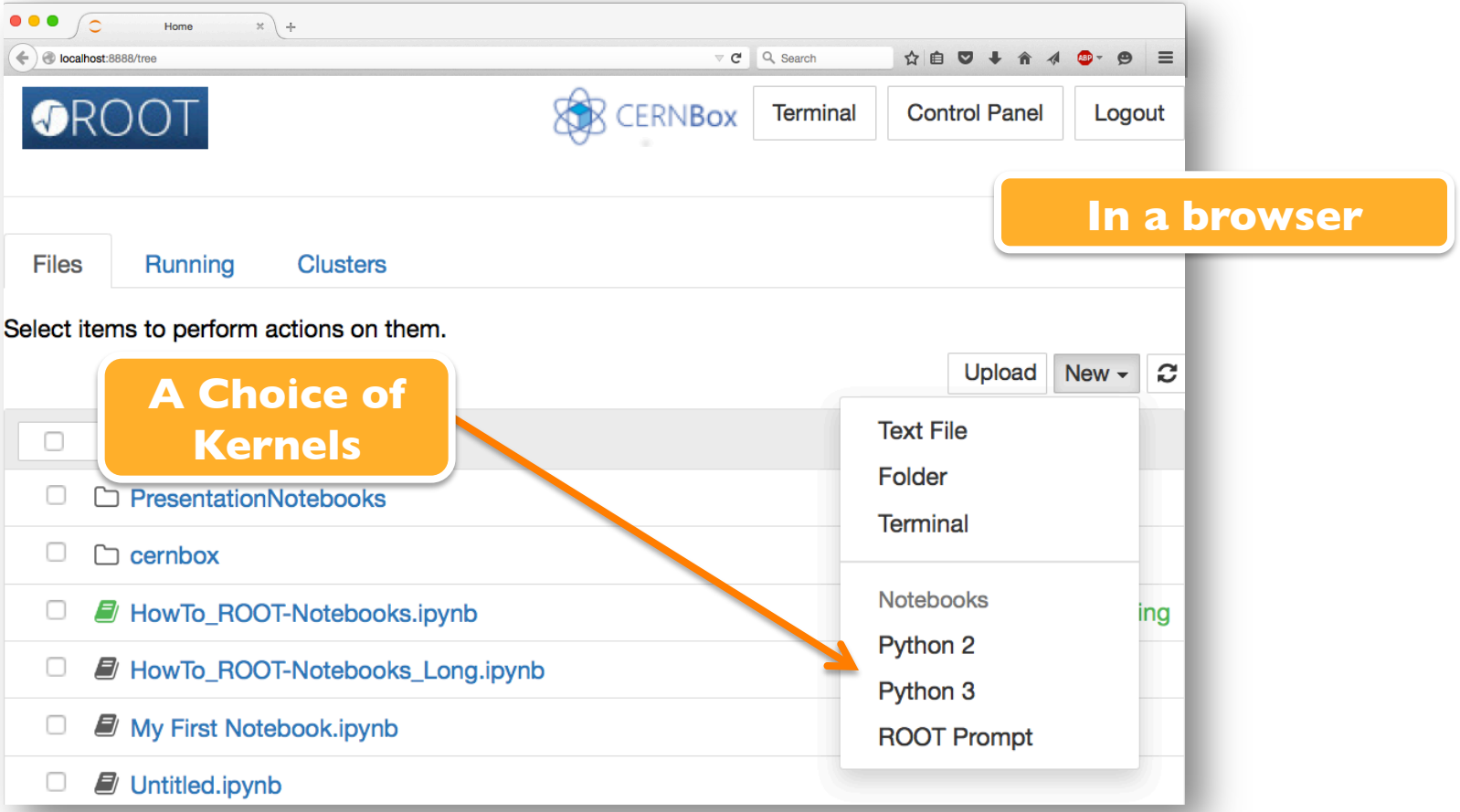
A web-based interactive computing interface and platform that combines code, equations, text and visualisations.

Many supported languages: Python, Haskell, Julia, R … One generally speaks about a "kernel" for a specific language

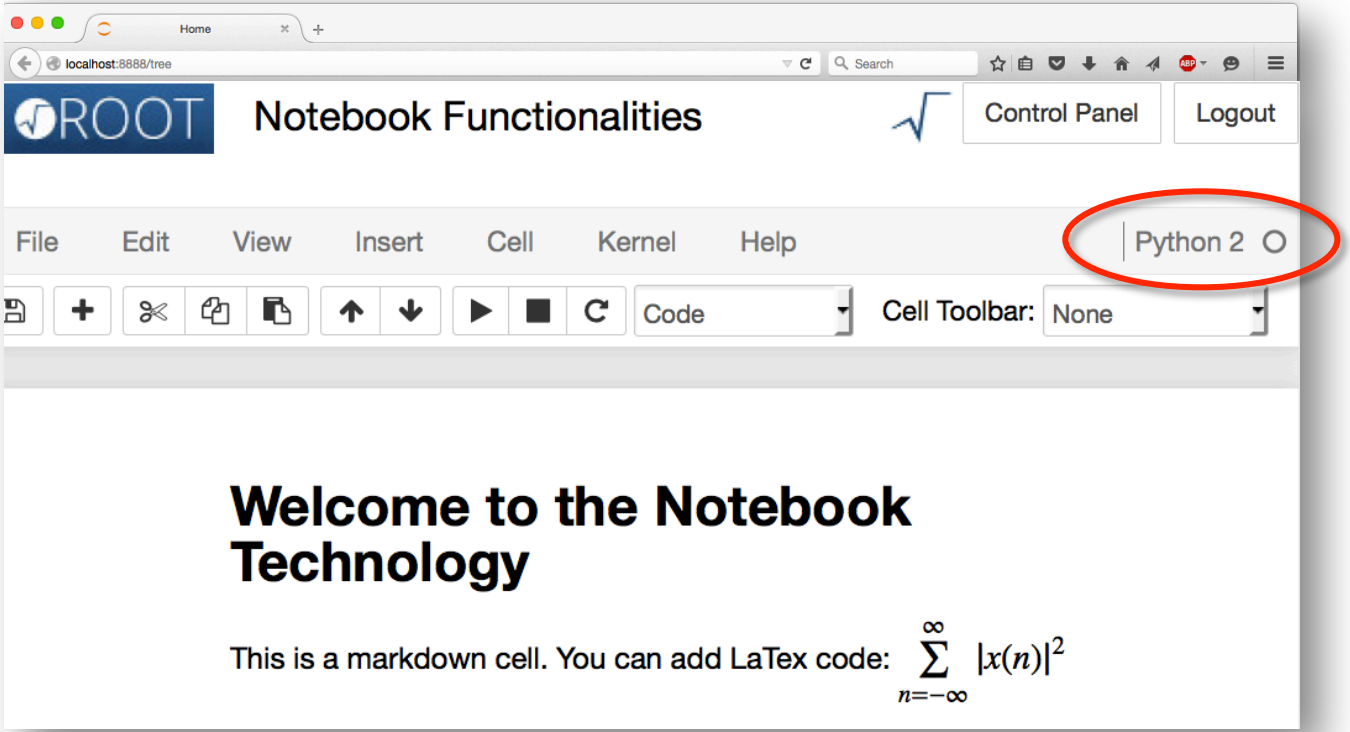
http://www.jupyter.org

In a nutshell: an "interactive shell opened within the browser"

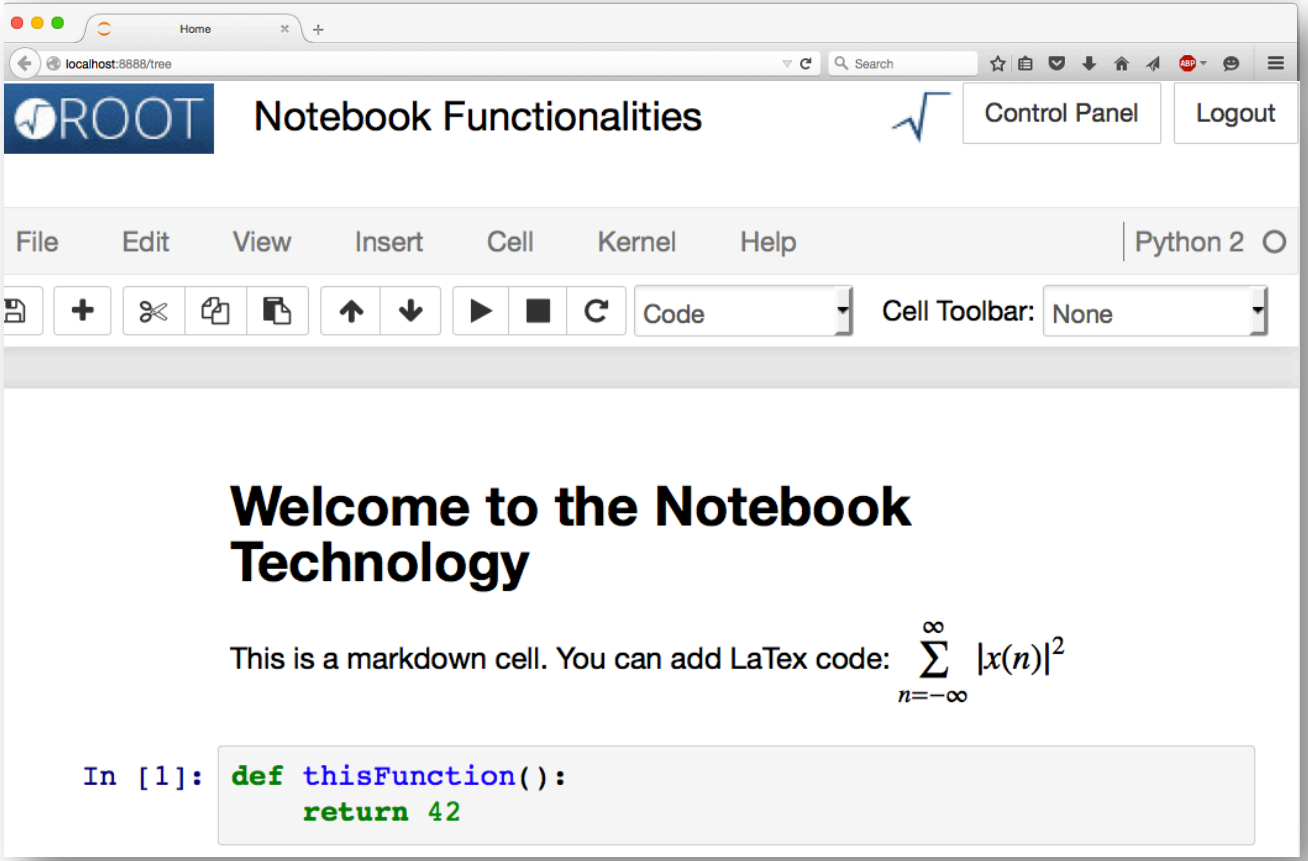
Also called:
"Jupyter Notebook" or "IPython Notebook"



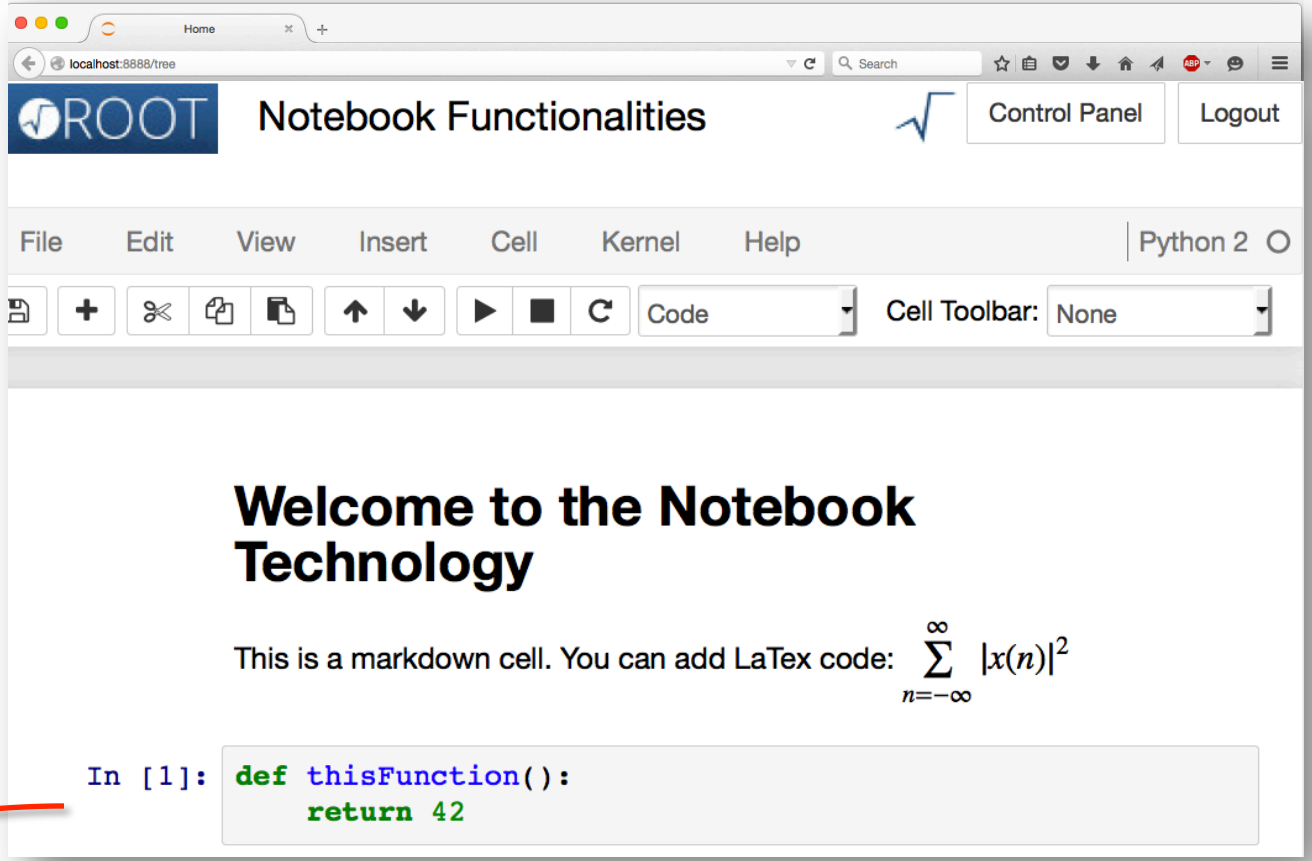
Kernels are processes that run interactive code in a particular programming language and return output to the user. Kernels also respond to tab completion and introspection requests.



Text and Formulas

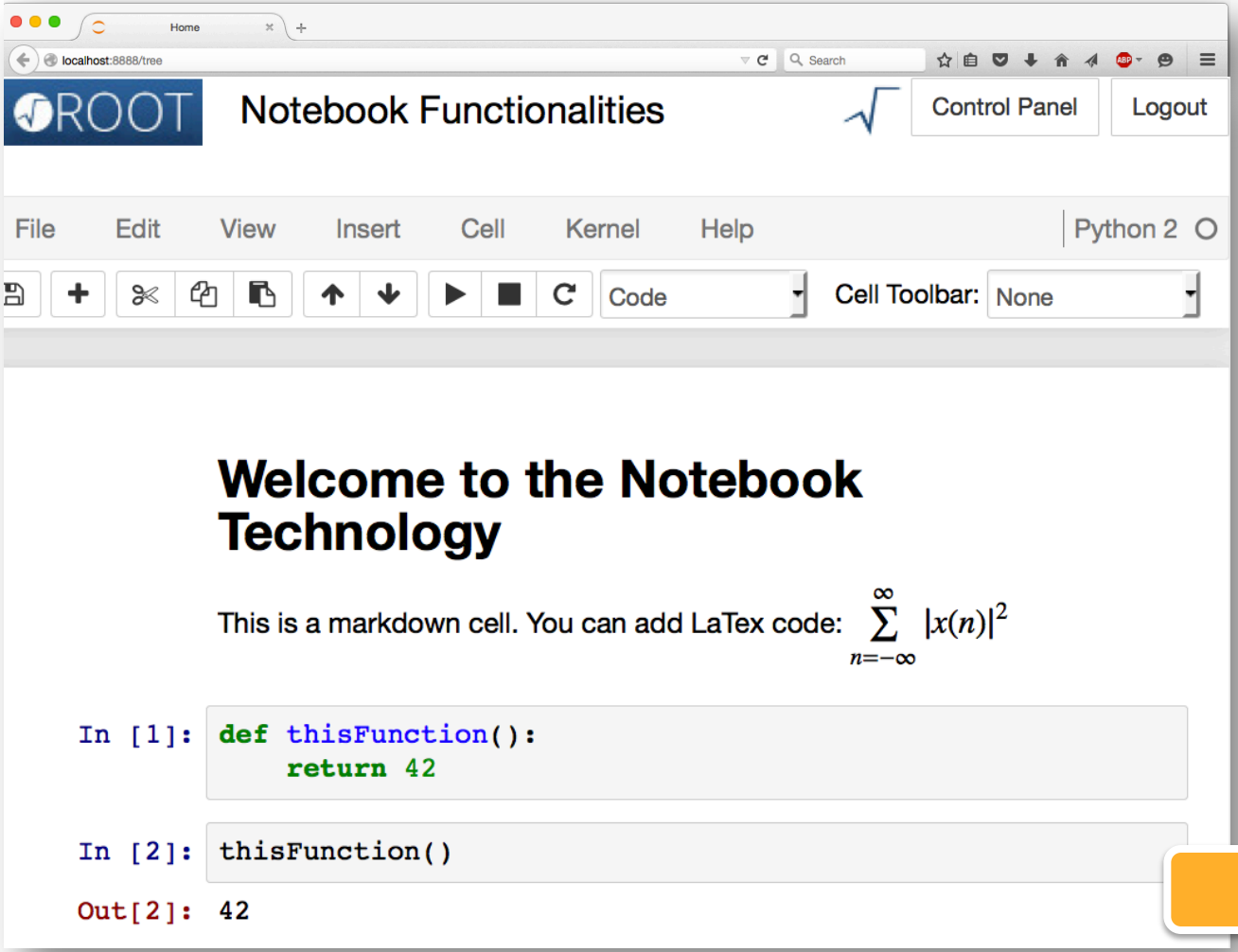

Home
localhost:8888/tree
Search
ROOT
Notebook Functionalities
Control Panel
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Python 2
Code
Cell Toolbar:
None
Welcome to the Notebook Technology
This is a markdown cell. You can add LaTex code: $\sum_{n=-\infty}^{\infty} |x(n)|^2$
In [1]:
def thisFunction():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

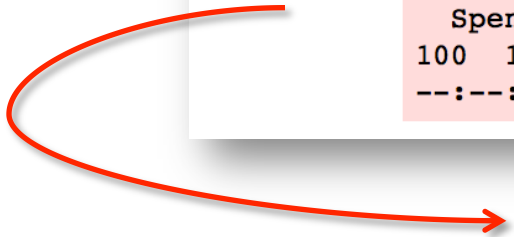
```
In [1]: def thisFunction():
            return 42

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

```
aasdemo.web.cern.ch/rootaasdemo/SaasFee.jpg \
> 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

A Distributed Service Building on top of CERN Services Portfolio

Data Mining as a Service

- Platform independent: only with a web browser
  - Analyse data via the Notebook web interface
- Calculations, input and results "in the cloud"
- Allow easy sharing of scientific results: plots, data, code
  - Storage is crucial
- Simplify teaching of data processing and programming
  - Not HEP specific, not only for cutting edge fundamental research
- C++, Python and other languages or analysis "ecosystems"
  - Also interface to widely adopted scientific libraries (e.g. ROOT*)

* www.root.cern.ch

Novel Application, Existing Components

The DMaaS project relies on technologies provided by CERN

- Scientific libraries Notebook integration (EP-SFT)
- Software distribution (*EP-SFT, IT-ST*): CVMFS
  - All software potentially available
- Virtualised CPU resources in OpenStack Cloud (*IT-CM*)
  - Interactive and batch usage
- Synergy with document sharing and publication (*IT-CDA*)
- Security, e.g. CERN credentials (*IT-DI-CSO*)
- Storage access (*IT-ST*): CERNBox, EOS
  - All data potentially available

A coherent view at CERN

Glossary

EOS

Disk-based low latency storage infrastructure for physics users. Main target: physics data analysis. Storage backend for CERNBox.

Indico

Manage complex conferences, workshops and meetings.

CVMFS

HTTP based network FS, optimized to deliver experiment software Files aggressively cached and downloaded on demand.

ROOT

Software framework for data mining, visualisation and storage. Hundreds of PB of HEP data saved in ROOT format. Try it in your browser (notebooks!):

mybinder.org/repo/cernphsft/rootbinder

Other Technologies

Jupyterhub: Server application - manages login of users and redirection to notebook

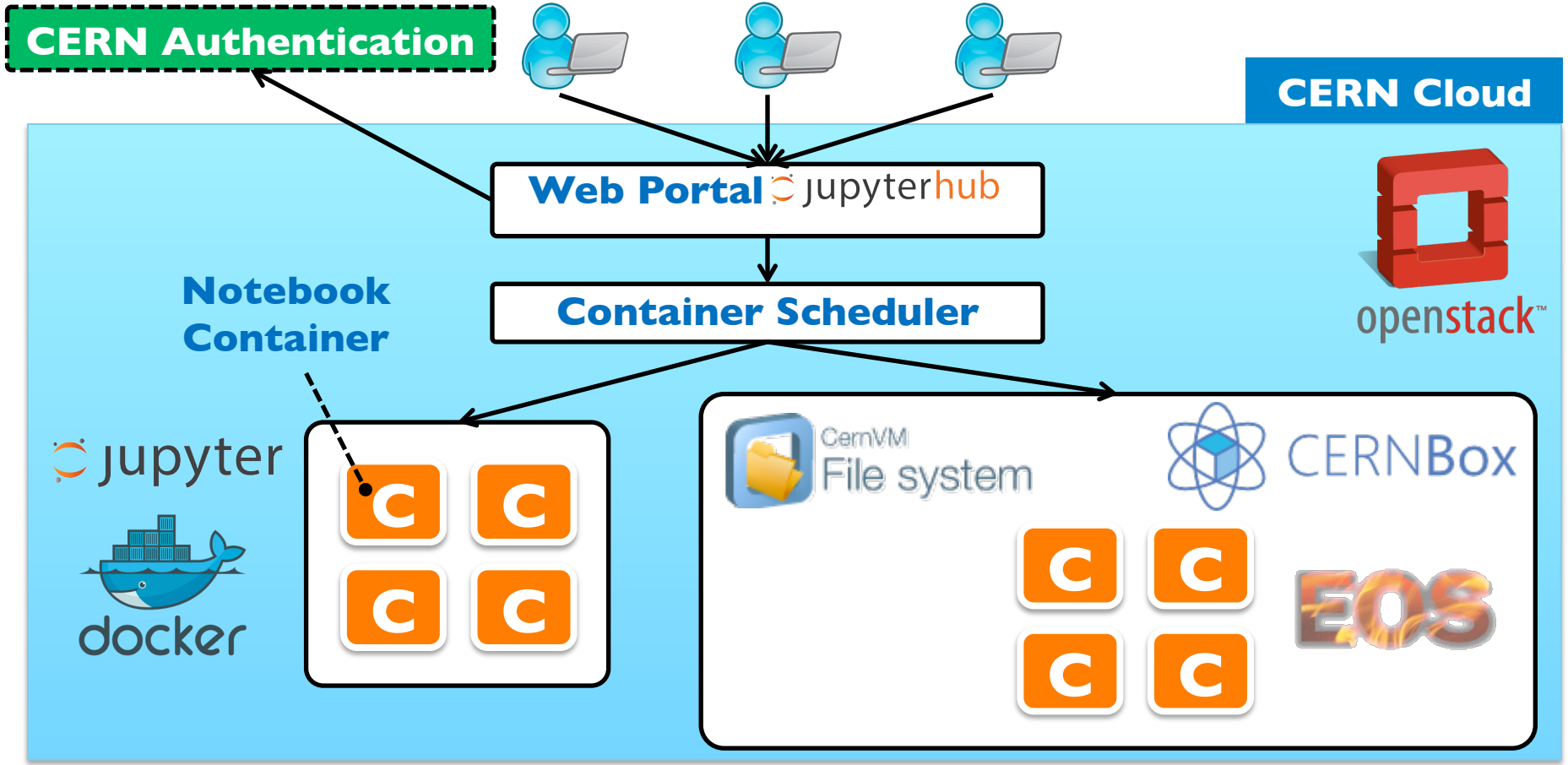
- Existing solution
- Allows encapsulation: spawn Docker container at logon

Docker

- Isolation of users
- Boot faster than Virtual Machines
- Openstack support

Both have large user bases and an active community behind

The Big Picture



Potential Daily Usage

- Launch jobs on the batch farm
- Access notebook running on a container in the OpenStack instance
- Inspect produced data via CERNBox/EOS from the notebook
- Create plots and output data
- Share, access plots (and output data!) on the web with CERNBox web interface
- Security guaranteed by the usual CERN standards

e.g.

Added value: remote users cannot open graphical connections to CERN (latency): Problem automatically solved in the above workflow

A Taste of the User Experience

Time for a demo:

- Download data
- Produce a plot after a simple analysis
- Share it via CERNBox

Test Node at CERN being used

DEMO

Number of **cinemas** and their screens in canton **Zurich** in the last 45 years

Timeline

Intermediate steps accomplished:

1) Single node, CERNBox, no CERN credentials
2) Single node on Openstack, CVMFS, CERNBox, CERN authentication (just demoed)

See backup for more details about these setups

TODO:

- Distributed setup on Openstack, CVMFS, CERNBox, CERN authentication

DMaaS accessible to CERN users: 2nd quarter 2016

2015: Success Stories

CERN Summer Student Program, ROOT lectures: Interactive notebooks offered

- 50 participants, perfect scaling, a success!
  - https://indico.cern.ch/event/407519

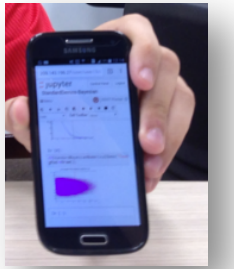
Example of notebooks on Indico

Data Science @ LHC Workshop, Multivariate analysis tutorial:
http://indico.cern.ch/event/395374/

E-Planet exchange @ UERJ, Brazil

- 30 participants, every day for a week, 3h a day

https://indico.cern.ch/event/402660/



In addition, clear signs of appreciation of Notebook technology: see backup

Conclusions

- We will provide a service for data analysis in the cloud via a web interface
  - Platform independent: no need to install software
  - Rely on the robust services already provided by CERN
- Sync'd storage and access to the mass storage are crucial
  - Share data, code, documentation, results
  - CERNBox + EOS – An optimal solution
- New ways of approaching data mining made accessible: boost productivity thanks to sync and file sharing services
- Give you the possibility to try it out!

Try it now!

- Access from the conference site until tomorrow

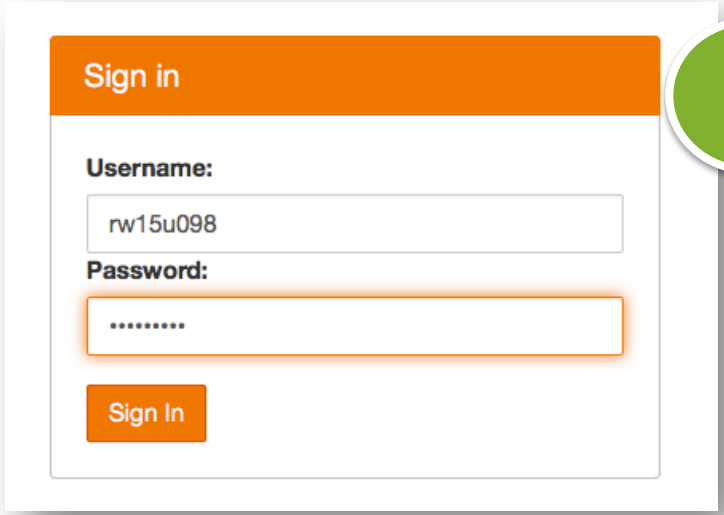
  dmaasdemo.web.cern.ch

From the ETH "public" network only!

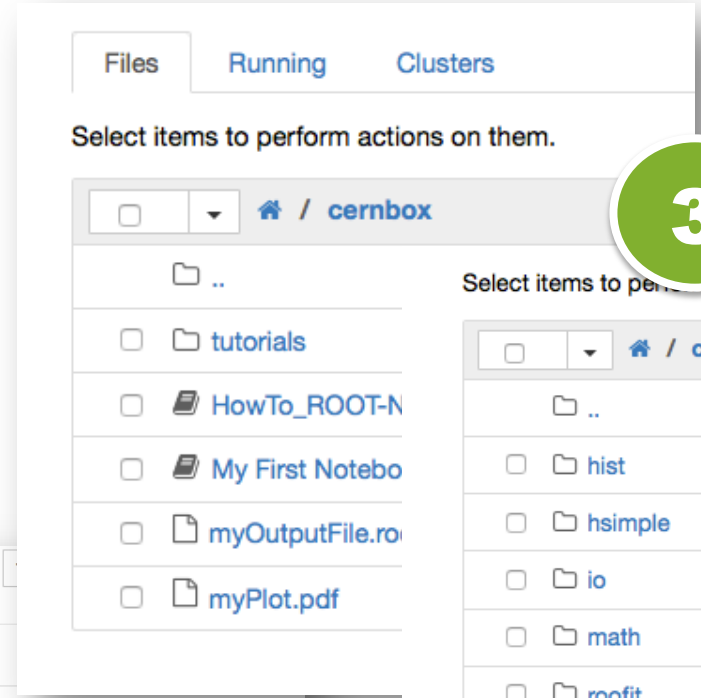
- Take a look to the provided notebooks, modify them, run them
  - Produce results!
  - Access them via CERNBox (https://cernbox.cern.ch)

Ask me for your user name and password!

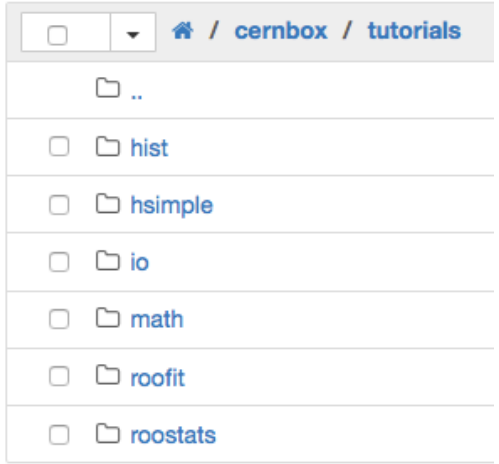
Several Tutorials Available



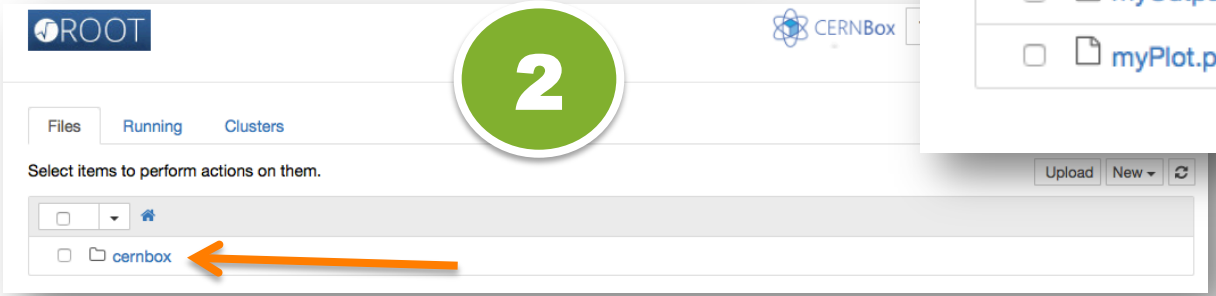
1



3

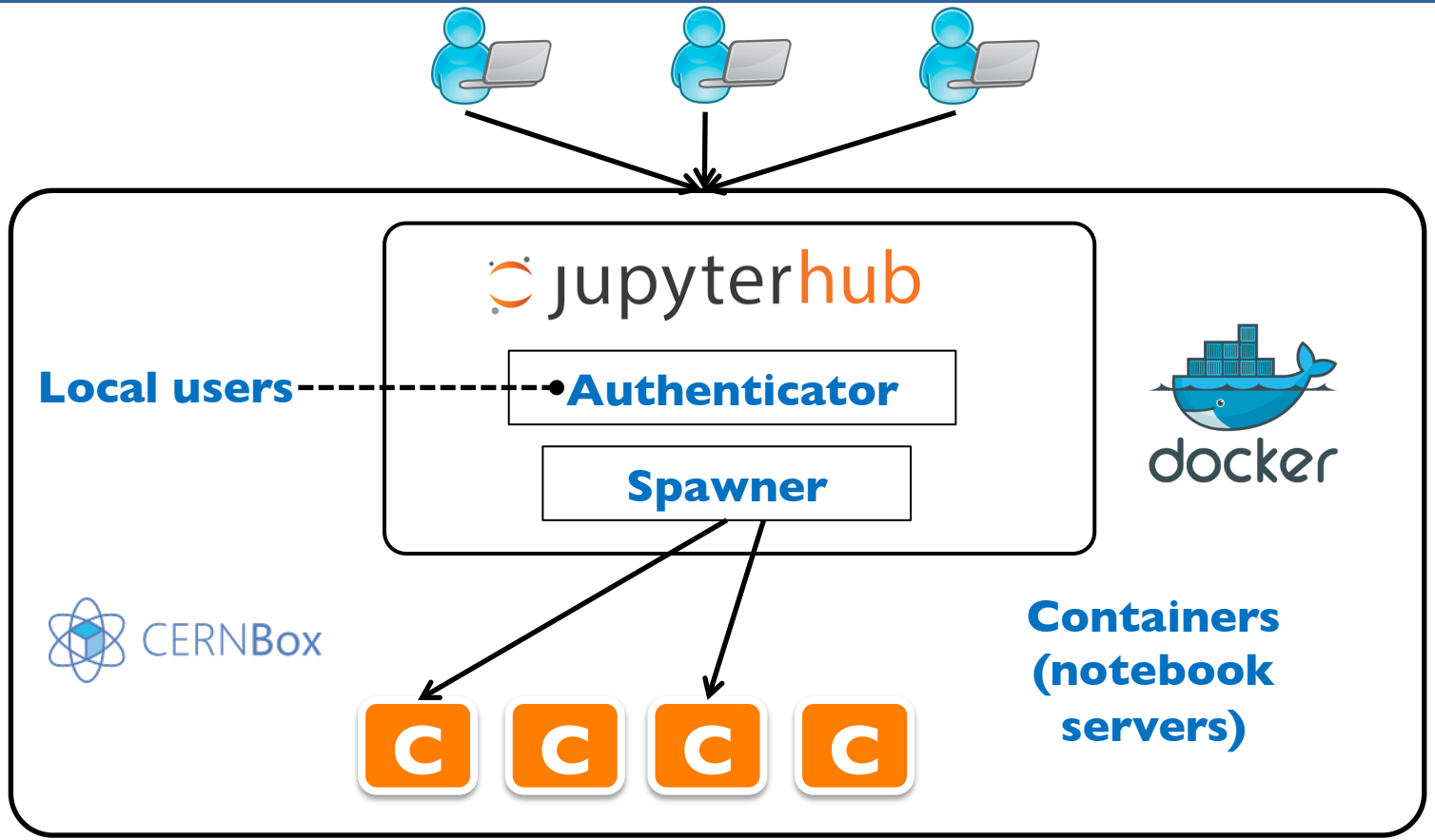


2

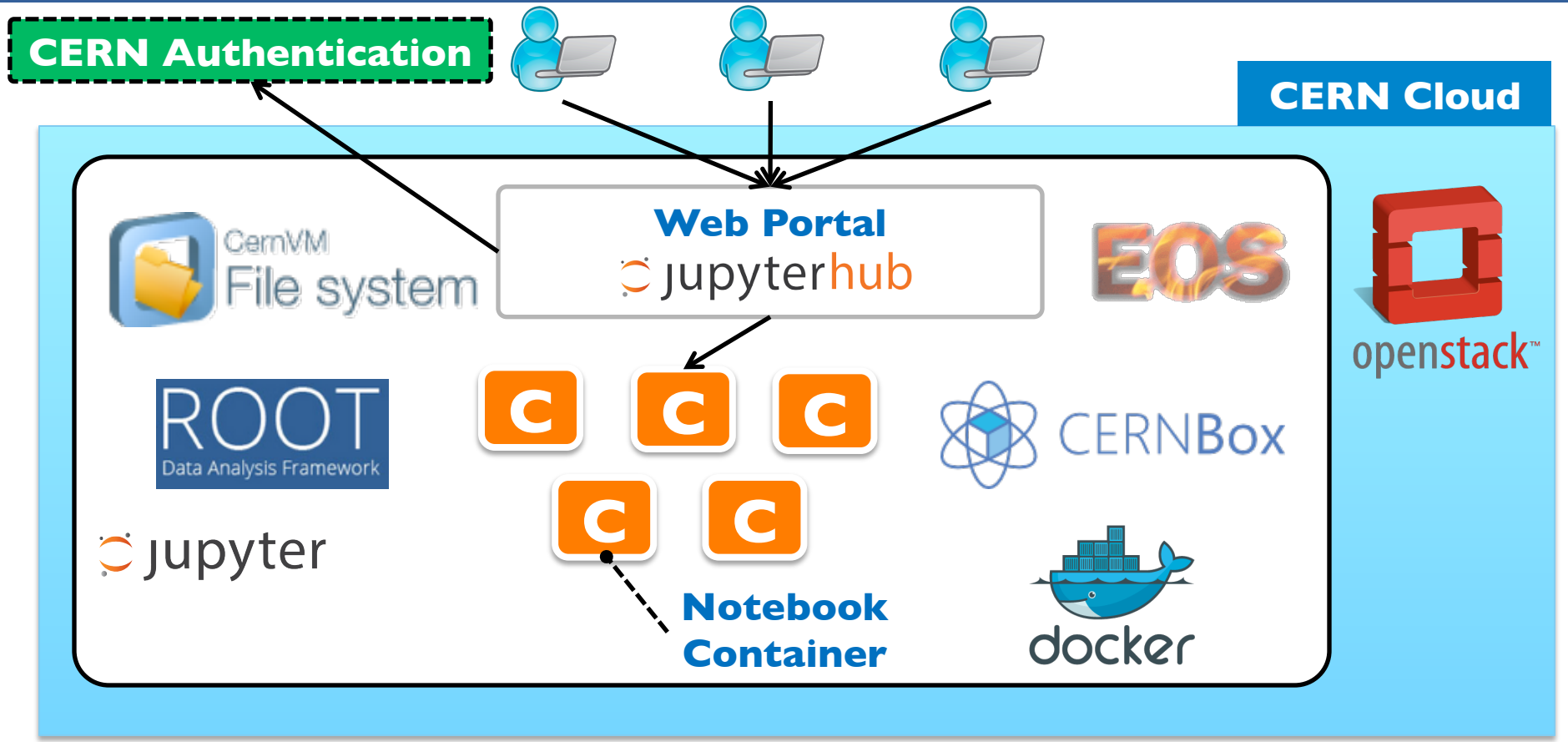


Backup Slides

Demo Setup


jupyterhub
Local users
Authenticator
Spawner
docker
CERNBox
Containers (notebook servers)
A single powerful machine

Pilot Service Single Node



Interface to Job Submission Tools

Large volume of data – complex analysis: need to use many cores

1) Single node: TProcPool, IPython Parallel, etherogeneous/ multithreaded code
2) Many nodes: Batch/Grid jobs

CERN Batch Service being considered in the full picture!

Several production grade, Python based job submission tools available:

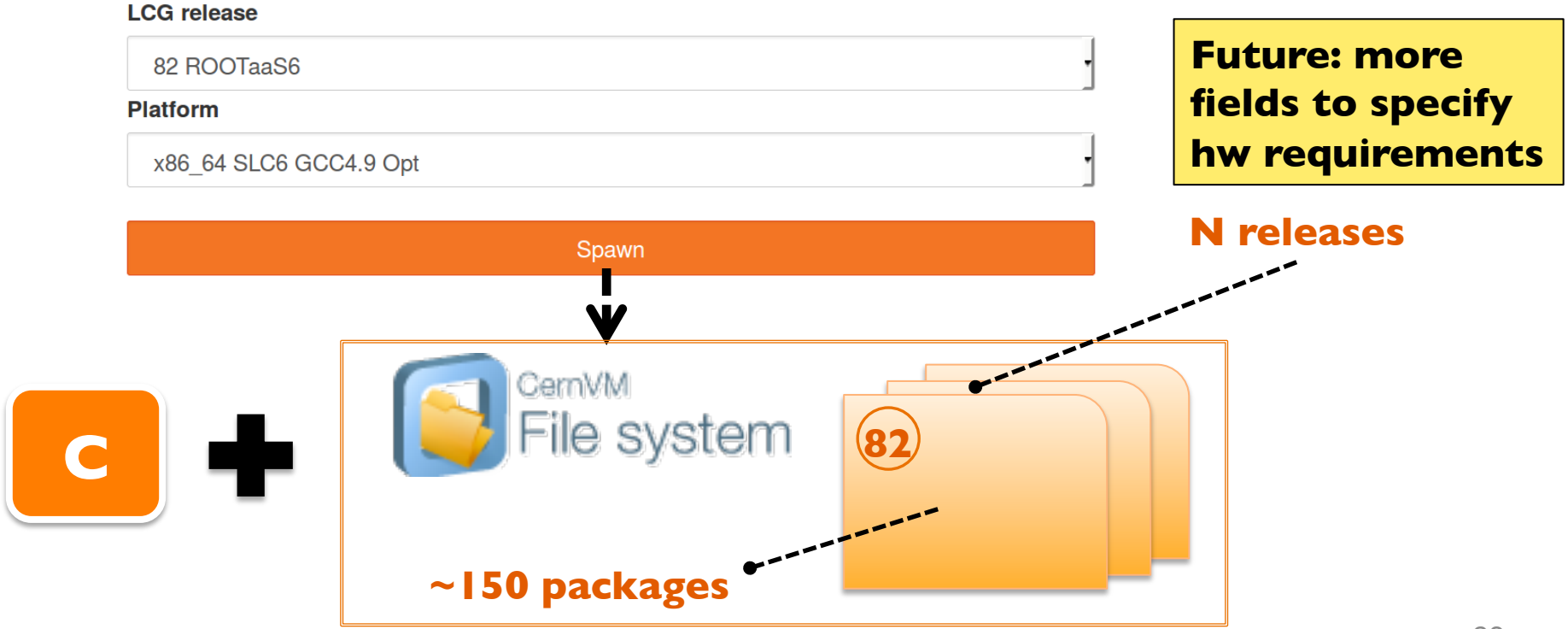
- Ganga, GridControl, Panda, …
- See A. Richards Presentation

Opportunity: Steer job submission to WLCG or local batch resources from the notebook.

CVMFS: Software Environment

Define a custom software environment via a web form

- Same mechanism for selecting hardware (e.g. GPU, N CPU cores, SSD disk)



And Notebooks

A C++ Interactive Shell

Integration of ROOT & Jupyter Notebooks **delivered**

- Python flavour
  - import ROOT: all goodies activated
  - %%cpp magic
- ROOT C++ flavour
  - Kernel distributed with ROOT itself
- Goodies
  - Tab completion
  - Display of graphics
  - Syntax highlighting
  - Asynchronous output capturing

ROOT comes with a C++ 11/14 compatible interpreter based on LLVM Technology

Try it Out: Local Server

Follow some simple instructions at:
https://root.cern.ch/how/how-create-rootbook
(basically build ROOT) and…

```
$ 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 ROOTbook goodies

Documentation and Links

ROOTbooks How-Tos

https://root.cern.ch/howtos#Jupyter%20Notebooks

ROOT bindings for Jupyter

https://github.com/root-mirror/root/tree/master/bindings/pyroot/JupyROOT

ROOT C++ Kernel

https://github.com/ipython/ipython/wiki/IPython-kernels-for-other-languages

Examples (15 already) from the *new* ROOT Tutorials can be found at:

https://root.cern.ch/code-examples#notebooks

both in Python and C++ (and mixed!)

ROOT: Binder Usage

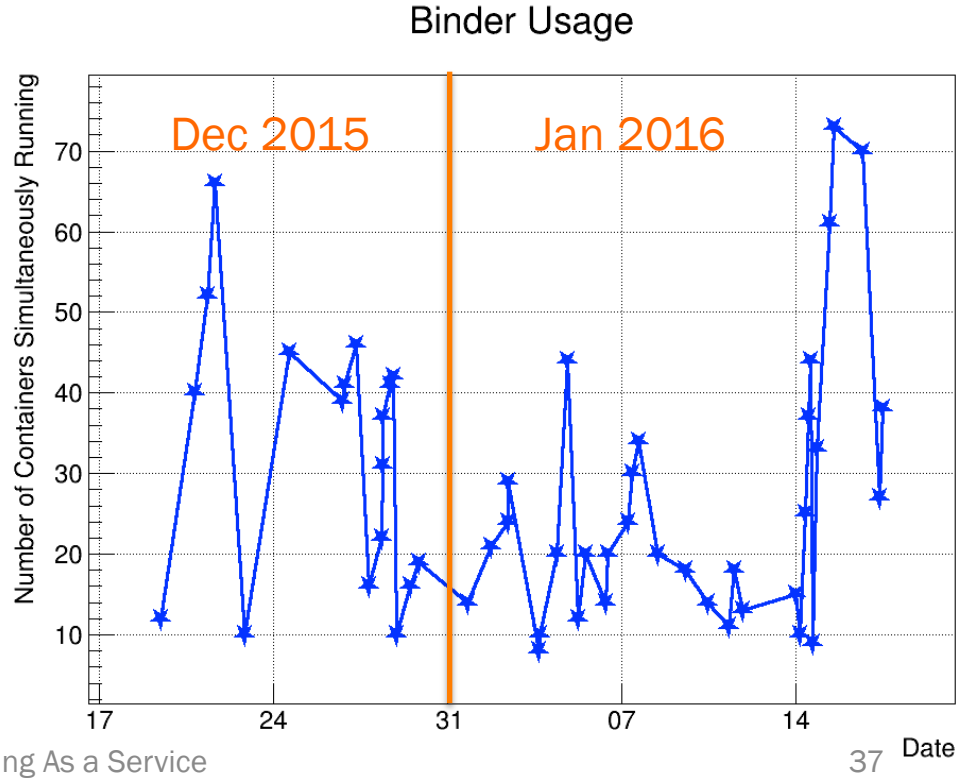
Binder is a software package and a web-service (100% free and open source) to turn a GitHub repo into a collection of interactive notebooks powered by Jupyter and Kubernetes. ROOT is on Binder: you can try it at ROOTBinder.

On ROOTBinder you can find a collection of Notebooks aiming to illustrate the potential of the ROOT Framework.

View, Create and Run ROOTbooks!

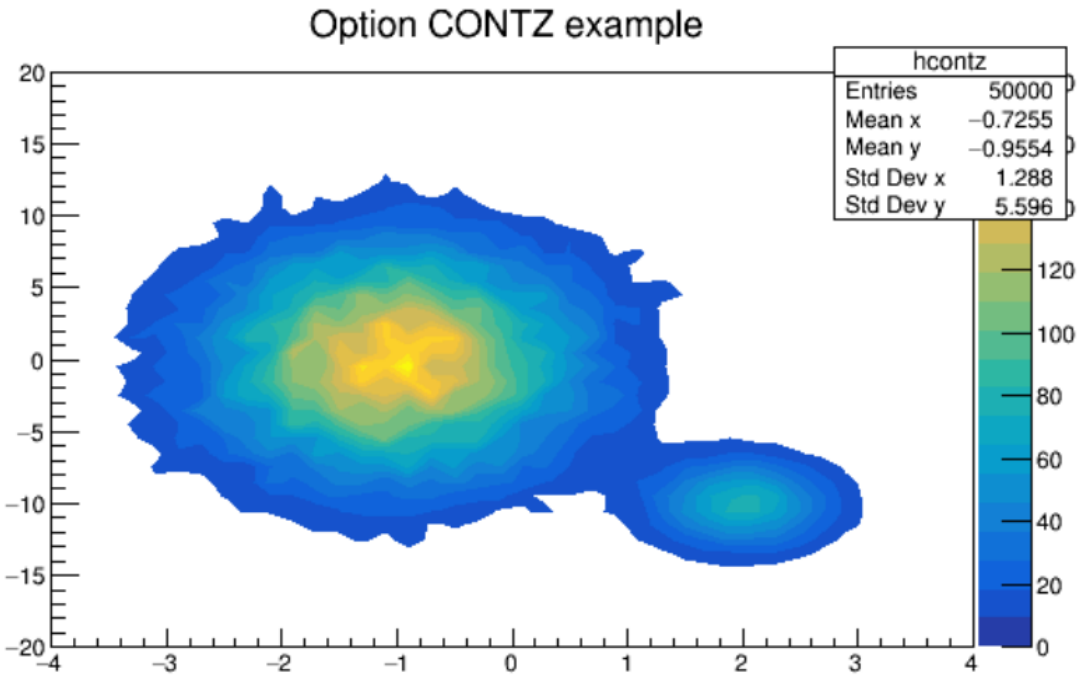
binder

Anonymous access, no persistent storage



A C++ ROOTbook

```
hcontz.Draw("CONTZ");
c1.Draw();
```

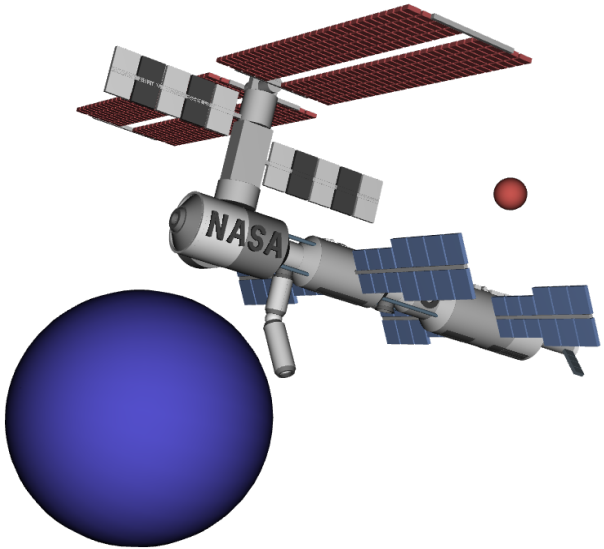


mybinder.org/repo/cernphsft/rootbinder

A C++ ROOTbook

In [2]:
```
TGeoManager::Import("../data/spaceStation.root");
auto topVolume = gGeoManager->GetTopVolume();
topVolume->Draw();
```

```
Info in <TGeoManager::Import>: Reading geometry from file:
../data/spaceStation.root
```



mybinder.org/repo/cernphsft/rootbinder