Facilitating Collaborative Analysis in SWAN

E. Tejedor, D. Castro, D. Piparo, P. Mato
E. Bocchi, J. Moscicki, M. Lamanna

https://swan.cern.ch

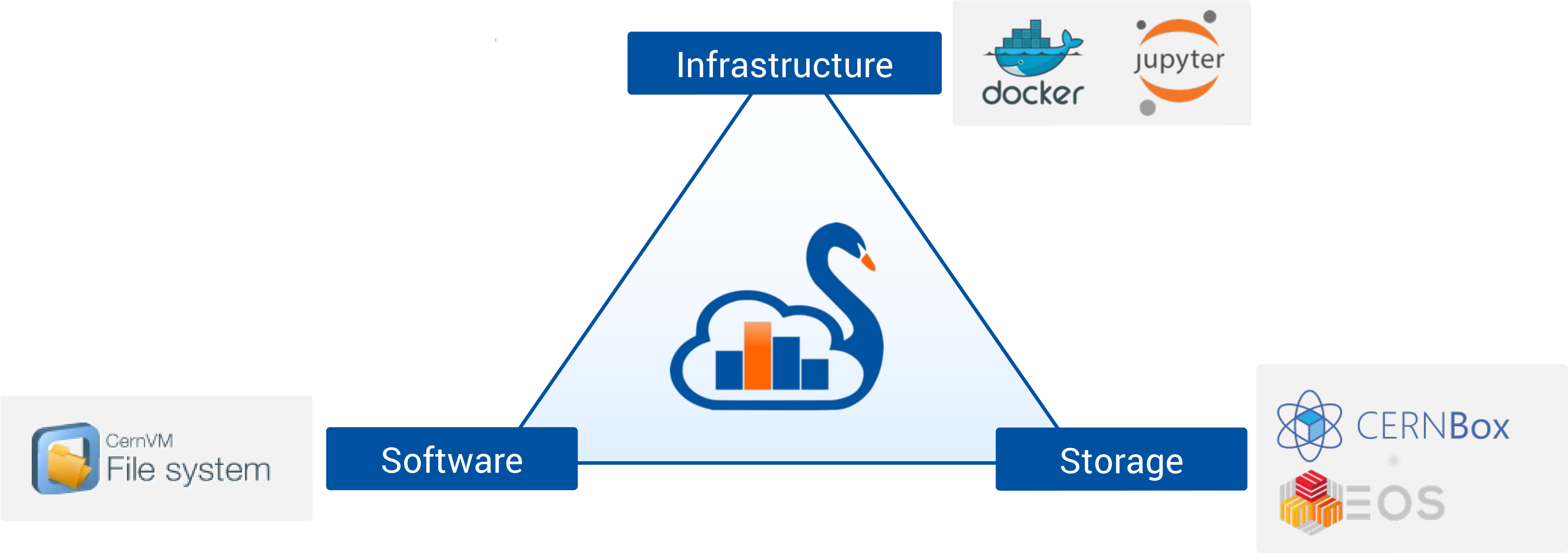
July 4th, 2018
CHEP 2018, Sofia (Bulgaria)

Introduction

SWAN in a Nutshell

- Analysis only with a web browser
  - No local installation needed
  - Based on Jupyter Notebooks – interactive computing
  - Calculations, input data and results “in the Cloud”
- Support for multiple analysis ecosystems
  - ROOT, Python, R, …
- Easy sharing of scientific results: plots, data, code
- Integration with CERN resources
  - Sofware, storage, mass processing power

The Pillars of SWAN



Infrastructure

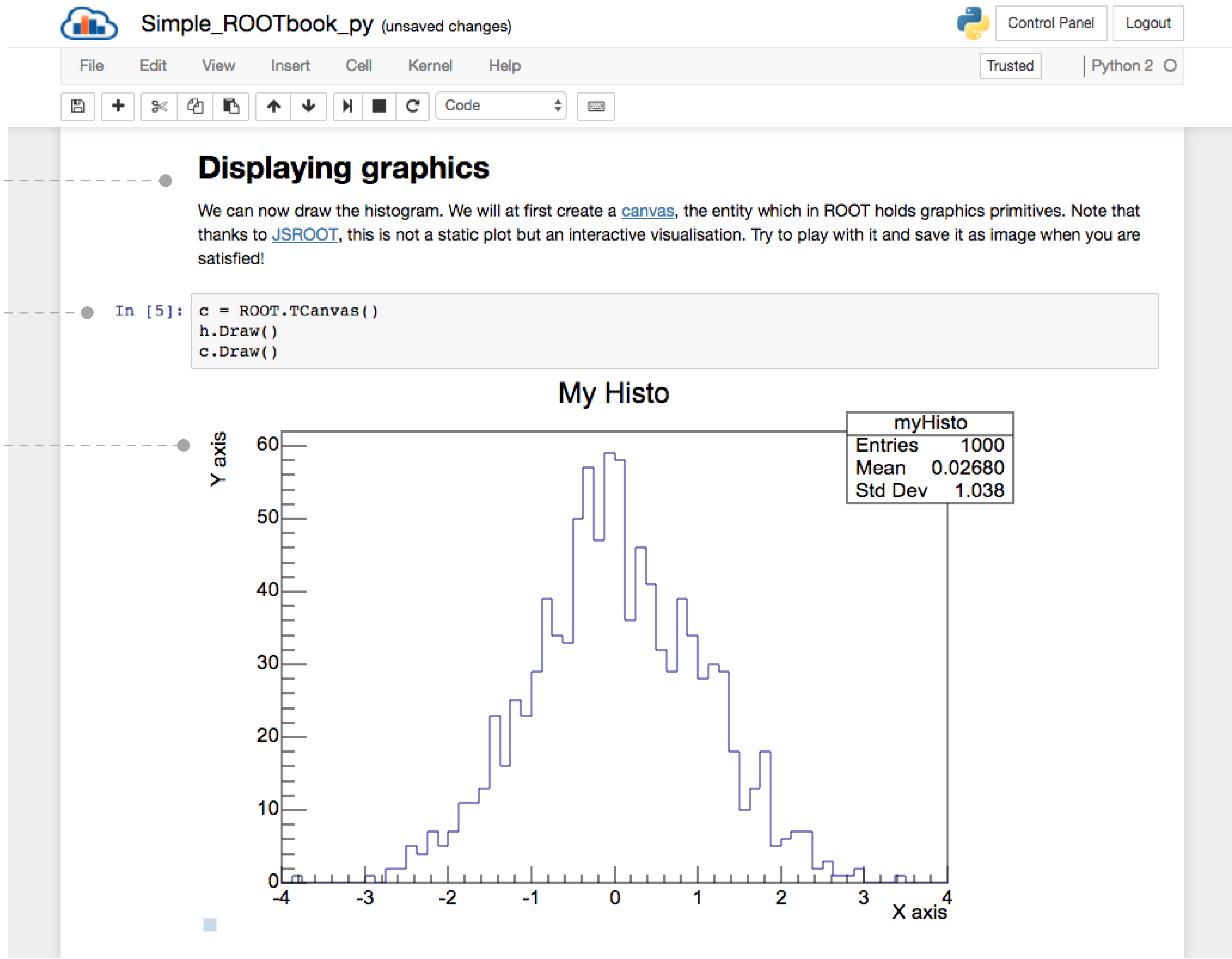
Software

Storage

Text

Code

Graphics



Displaying graphics

We can now draw the histogram. We will at first create a canvas, the entity which in ROOT holds graphics primitives. Note that thanks to JSROOT, this is not a static plot but an interactive visualisation. Try to play with it and save it as image when you are satisfied!

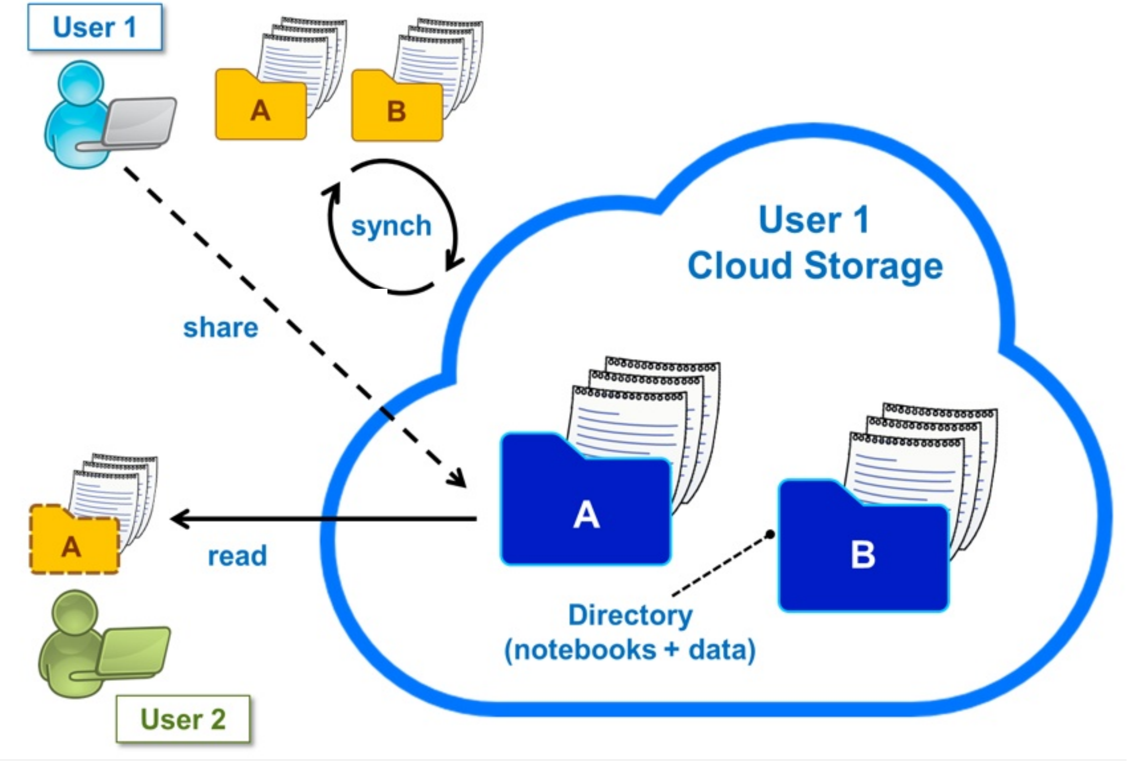
In [5]:
```
c = ROOT.TCanvas()
h.Draw()
c.Draw()
```

Collaborative Analysis

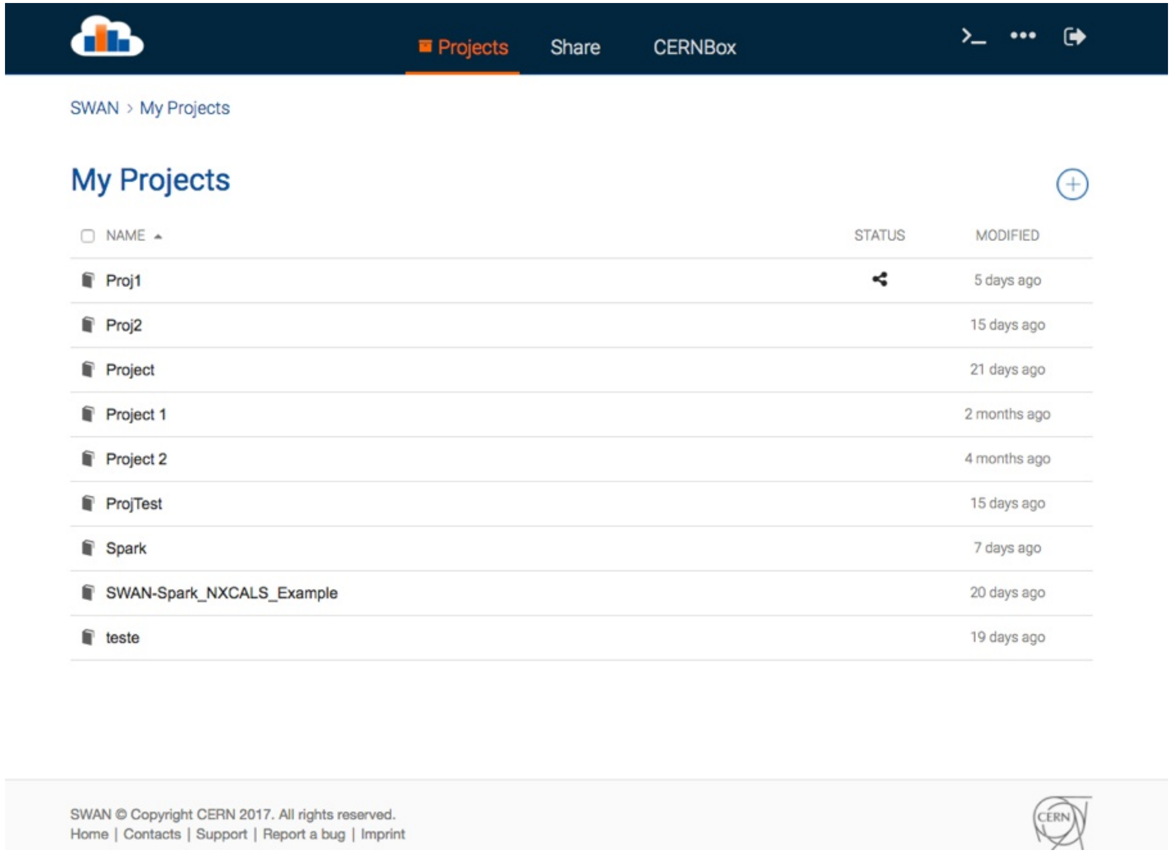
Cloud Storage as your Home

- CERNBox is SWAN's home directory
  - Storage for your notebooks and data
- Automatic synchronization
  - Files synced across devices and the Cloud
- Provides foundations for sharing
  - Collaborative analysis

EOS CERNBox



New User Interface

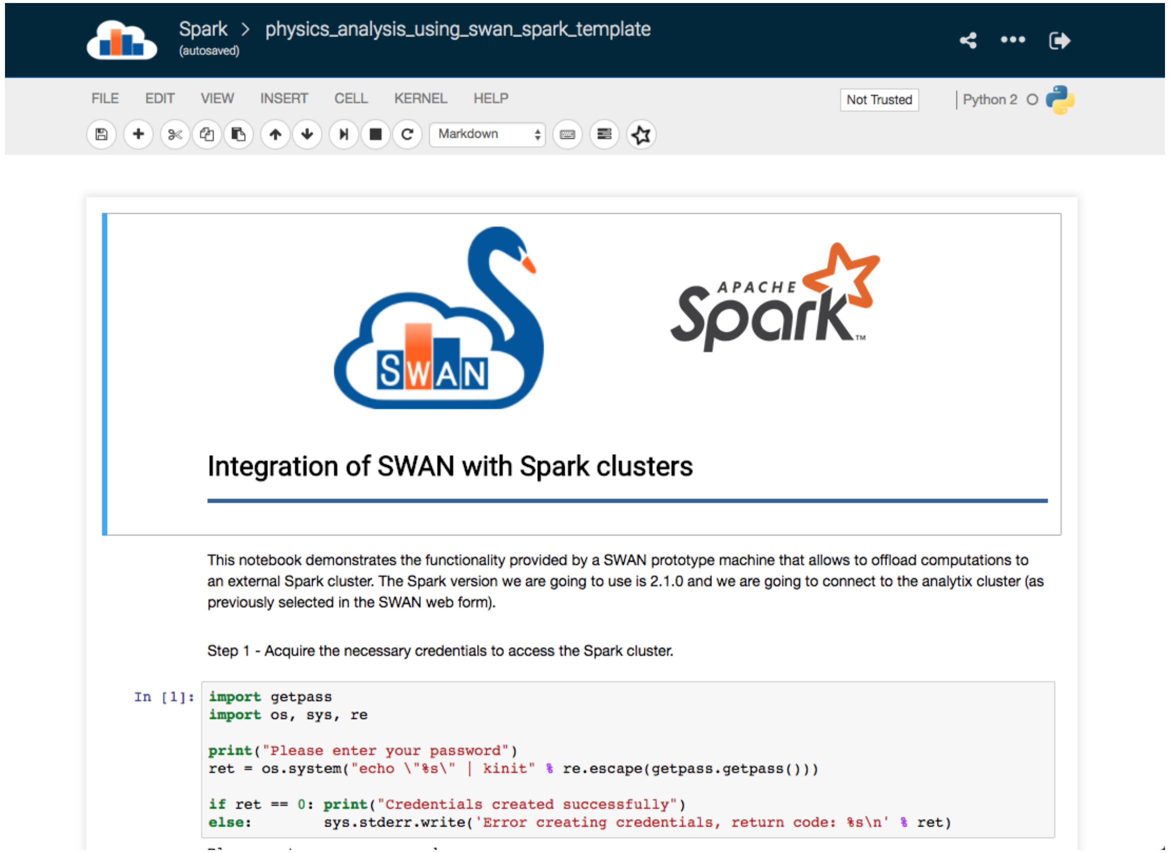


SWAN > My Projects

My Projects

□ NAME ▲	STATUS	MODIFIED
Proj1	<	5 days ago
Proj2		15 days ago
Project		21 days ago
Project 1		2 months ago
Project 2		4 months ago
ProjTest		15 days ago
Spark		7 days ago
SWAN-Spark_NXCALS_Example		20 days ago
teste		19 days ago

SWAN © Copyright CERN 2017. All rights reserved.
Home | Contacts | Support | Report a bug | Imprint



Integration of SWAN with Spark clusters

This notebook demonstrates the functionality provided by a SWAN prototype machine that allows to offload computations to an external Spark cluster. The Spark version we are going to use is 2.1.0 and we are going to connect to the analytix cluster (as previously selected in the SWAN web form).

Step 1 - Acquire the necessary credentials to access the Spark cluster.

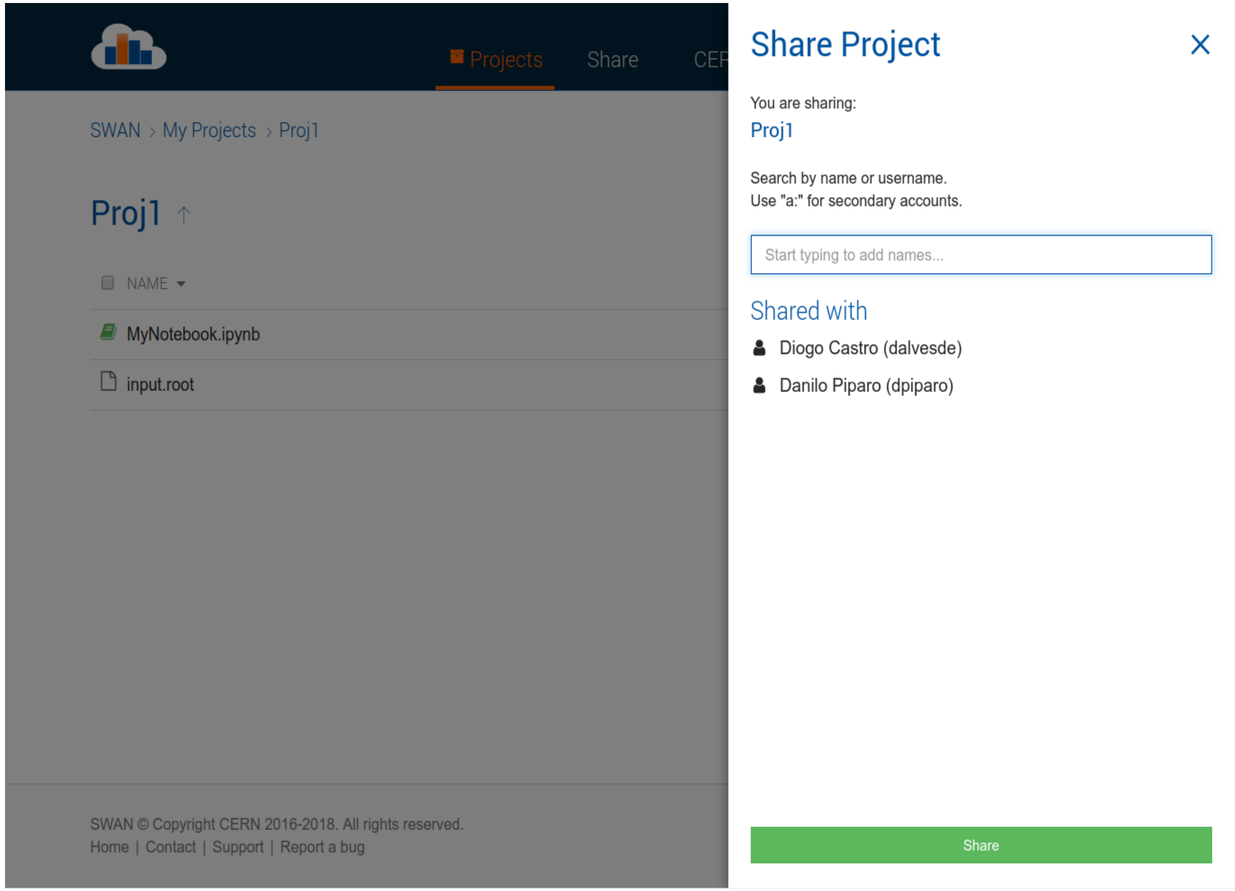
```
In [1]: import getpass
        import os, sys, re

        print("Please enter your password")
        ret = os.system("echo \"%s\" | kinit" % re.escape(getpass.getpass()))

        if ret == 0: print("Credentials created successfully")
        else:        sys.stderr.write('Error creating credentials, return code: %s\n' % ret)
```

Sharing Made Easy

- Sharing from within the SWAN interface
  - Integration with CERNBox
- Users can share “Projects”
  - Special kind of folder that contains notebooks and other files, like input data
  - Self contained



Share Project

You are sharing:
Proj1

Search by name or username.
Use "a:" for secondary accounts.

Start typing to add names...

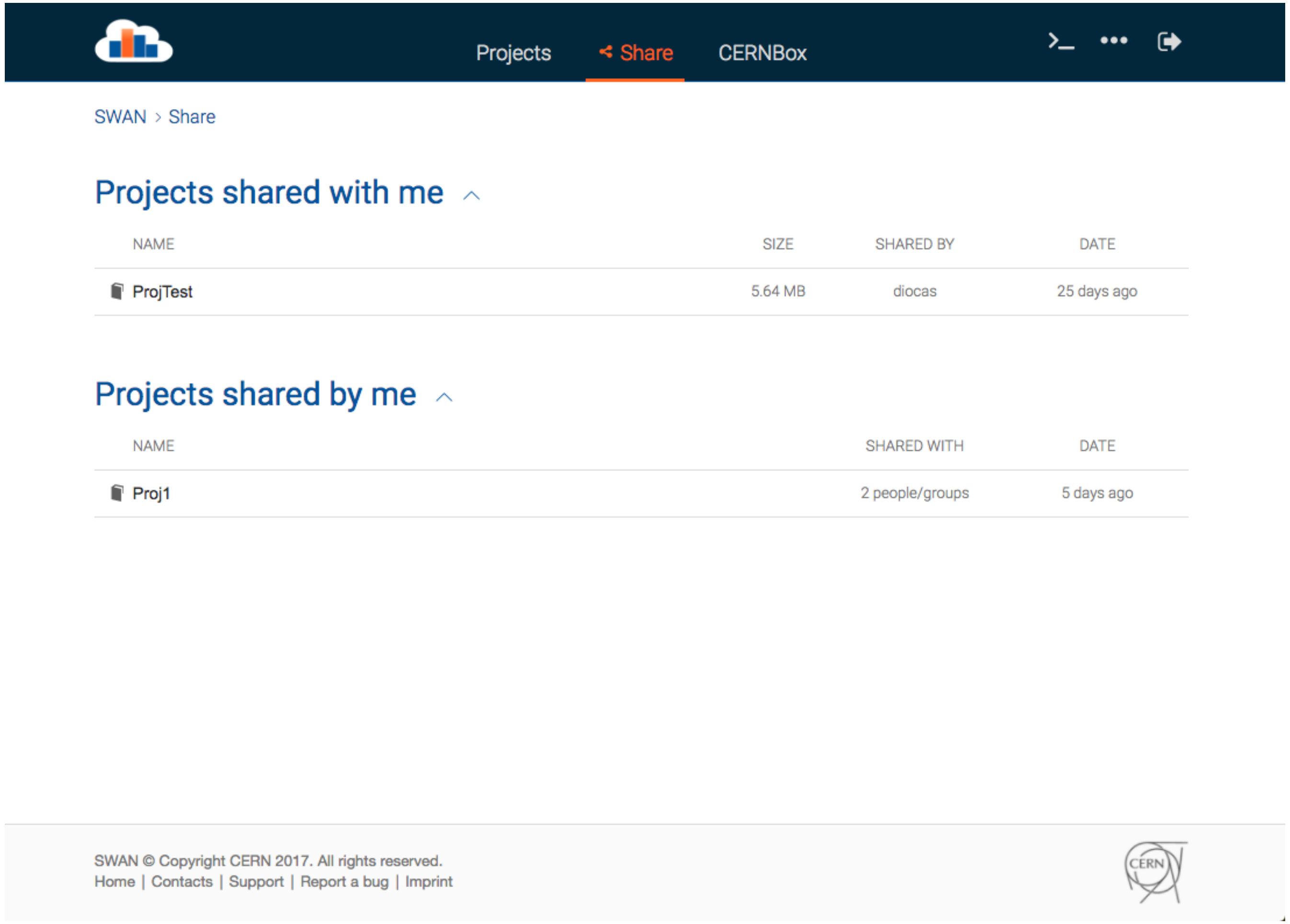
Shared with

- Diogo Castro (dalvesde)
- Danilo Piparo (dpiparo)

SWAN © Copyright CERN 2016-2018. All rights reserved.
Home | Contact | Support | Report a bug

The Share Tab

- Users can list which projects...
  - they have shared
  - others have shared with them
- Project information
  - Sharer
  - Size
  - Date



SWAN › Share

Projects shared with me

NAME	SIZE	SHARED BY	DATE
ProjTest	5.64 MB	diocas	25 days ago

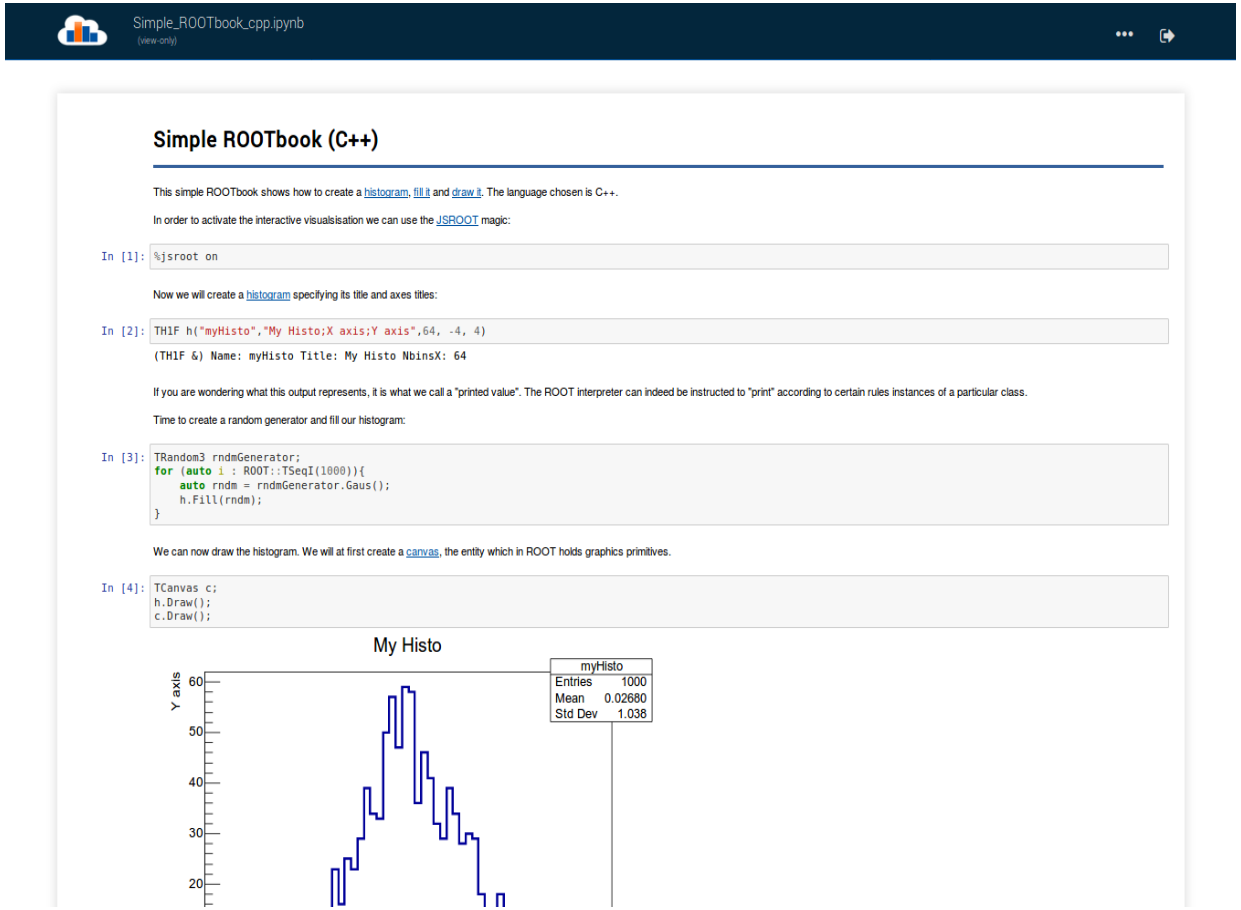
Projects shared by me

NAME	SHARED WITH	DATE
Proj1	2 people/groups	5 days ago

SWAN © Copyright CERN 2017. All rights reserved.
Home | Contacts | Support | Report a bug | Imprint

Inspecting a Project

- By clicking on a shared project, a user can inspect its contents
  - Browsing of the project files
  - Static rendering of notebooks
- Useful to decide whether to accept or not the shared project



Simple_ROOTbook_cpp.ipynb
(view-only)

Simple ROOTbook (C++)

This simple ROOTbook shows how to create a histogram, fill it and draw it. The language chosen is C++.

In order to activate the interactive visualisation we can use the JSROOT magic:

```
In [1]: %jsroot on
```

Now we will create a histogram specifying its title and axes titles:

```
In [2]: TH1F h("myHisto","My Histo;X axis;Y axis",64, -4, 4)
        (TH1F &) Name: myHisto Title: My Histo NbinsX: 64
```

If you are wondering what this output represents, it is what we call a "printed value". The ROOT interpreter can indeed be instructed to "print" according to certain rules instances of a particular class.

Time to create a random generator and fill our histogram:

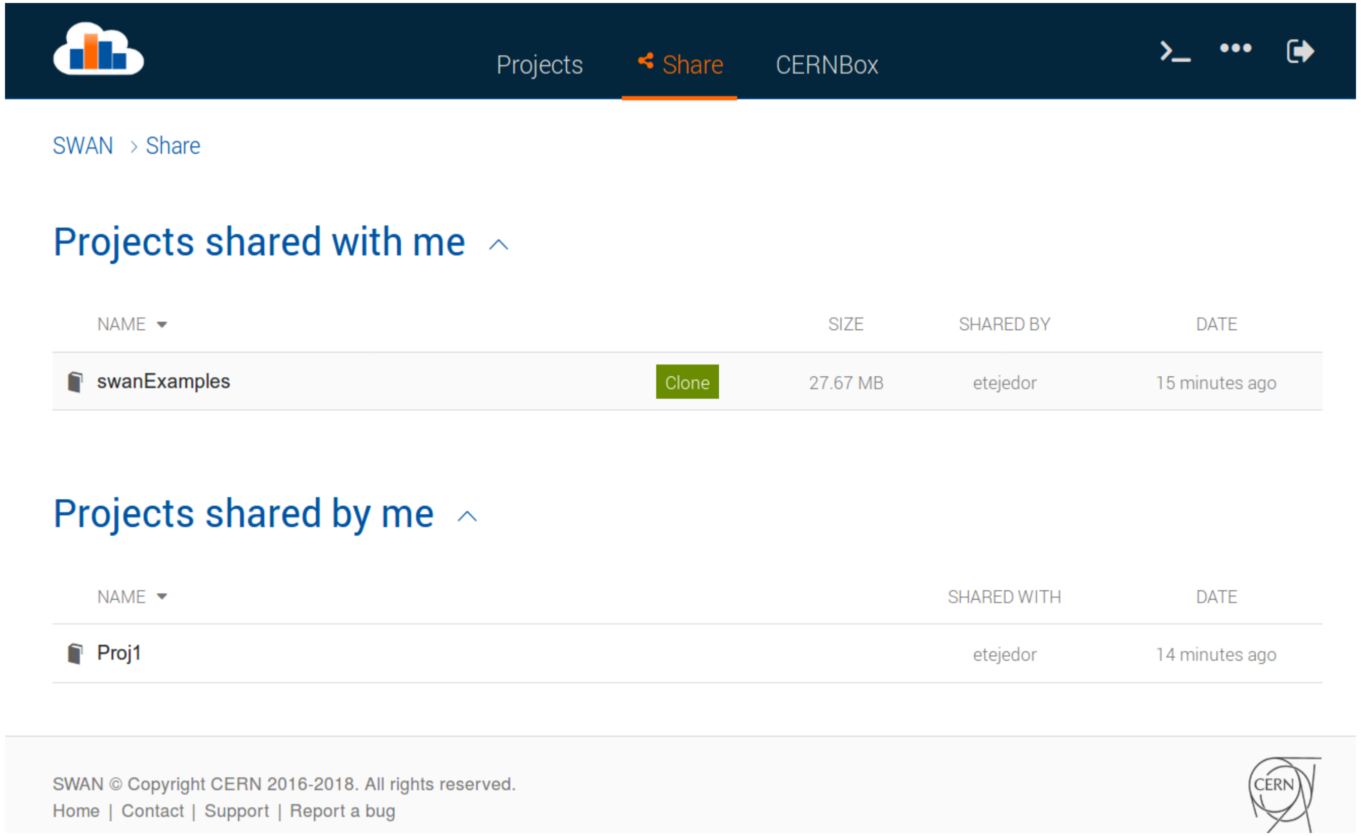
```
In [3]: TRandom3 rndmGenerator;
        for (auto i : ROOT::TSeqI(1000)){
            auto rndm = rndmGenerator.Gaus();
            h.Fill(rndm);
        }
```

We can now draw the histogram. We will at first create a canvas, the entity which in ROOT holds graphics primitives.

```
In [4]: TCanvas c;
        h.Draw();
        c.Draw();
```

Accepting a Shared Project

- When accepting a shared project, its contents are cloned to the receiver's CERNBox
  - The receiver will work on their own copy
- Concurrent editing not supported by Jupyter
  - Safer to clone



SWAN > Share

Projects shared with me

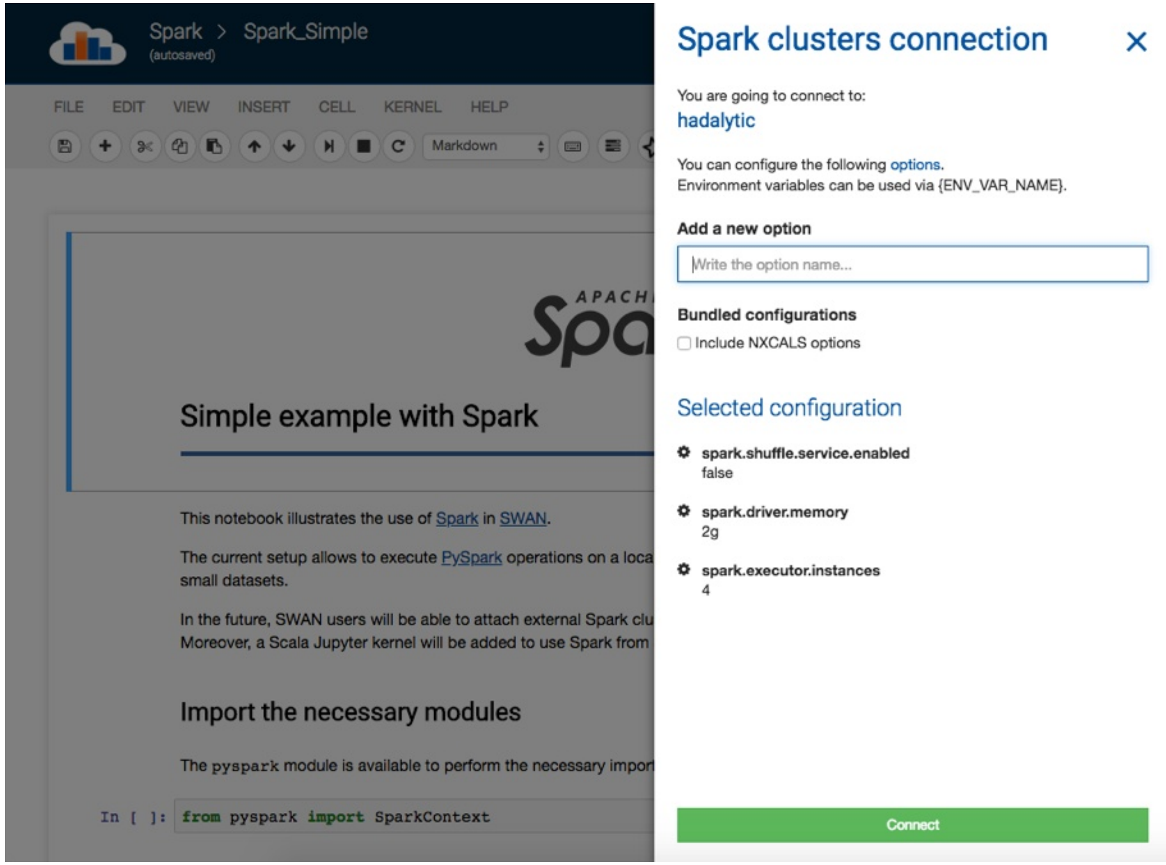
NAME		SIZE	SHARED BY	DATE
swanExamples	Clone	27.67 MB	etejedor	15 minutes ago

Projects shared by me

NAME	SHARED WITH	DATE
Proj1	etejedor	14 minutes ago

Home | Contact | Support | Report a bug

Sharing Spark Projects



Simple example with Spark

This notebook illustrates the use of Spark in SWAN.

The current setup allows to execute PySpark operations on a loca
small datasets.

In the future, SWAN users will be able to attach external Spark clu
Moreover, a Scala Jupyter kernel will be added to use Spark from

Import the necessary modules

The pyspark module is available to perform the necessary impor

```
In [ ]: from pyspark import SparkContext
```

Spark clusters connection

You are going to connect to:
hadalytic

You can configure the following options.
Environment variables can be used via {ENV_VAR_NAME}.

Add a new option

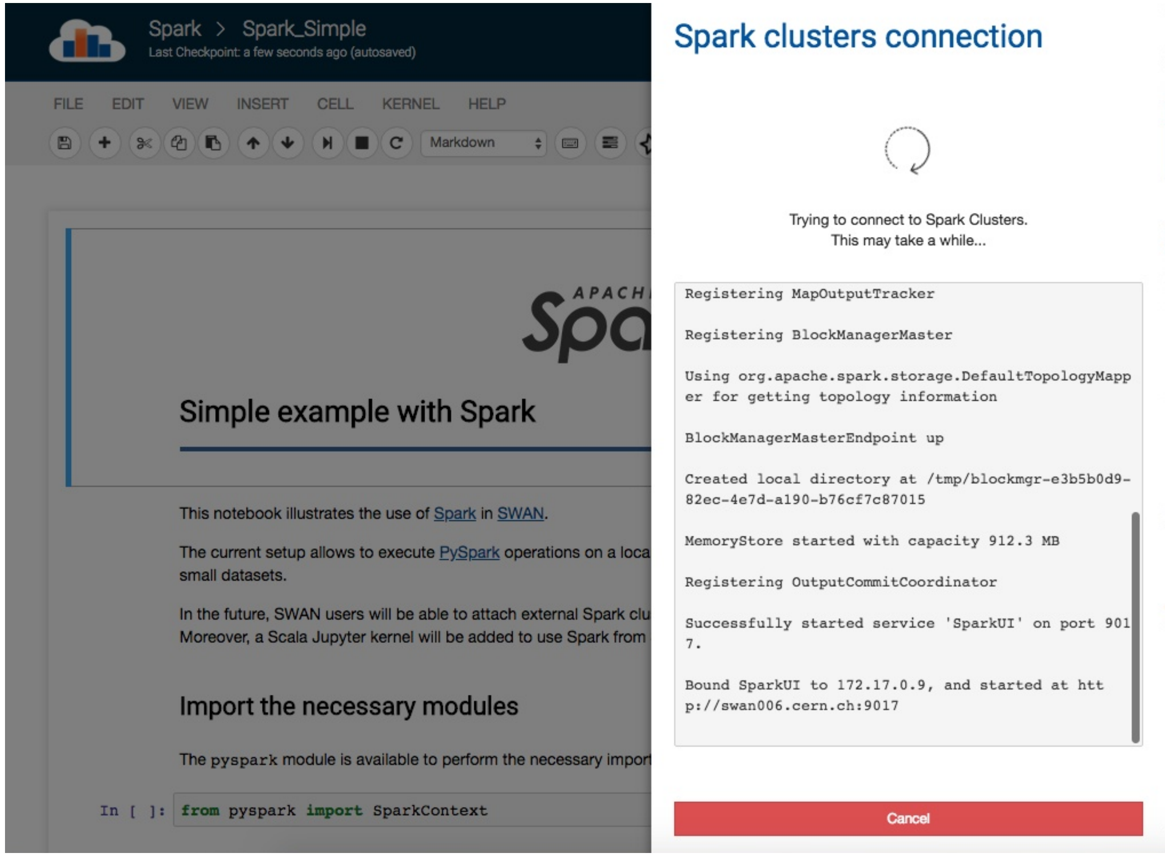
Write the option name...

Bundled configurations

□ Include NXCALS options

Selected configuration

- spark.shuffle.service.enabled: false
- spark.driver.memory: 2g
- spark.executor.instances: 4


Spark > Spark_Simple
Last Checkpoint: a few seconds ago (autosaved)
FILE EDIT VIEW INSERT CELL KERNEL HELP
Markdown
APACHE Spa

Simple example with Spark

This notebook illustrates the use of Spark in SWAN.

The current setup allows to execute PySpark operations on a loca
small datasets.

In the future, SWAN users will be able to attach external Spark clu
Moreover, a Scala Jupyter kernel will be added to use Spark from

Import the necessary modules

The pyspark module is available to perform the necessary impor

```
In [ ]: from pyspark import SparkContext
```

Spark clusters connection

Trying to connect to Spark Clusters.
This may take a while...

```
Registering MapOutputTracker

Registering BlockManagerMaster

Using org.apache.spark.storage.DefaultTopologyMapp
er for getting topology information

BlockManagerMasterEndpoint up

Created local directory at /tmp/blockmgr-e3b5b0d9-
82ec-4e7d-a190-b76cf7c87015

MemoryStore started with capacity 912.3 MB

Registering OutputCommitCoordinator

Successfully started service 'SparkUI' on port 901
7.

Bound SparkUI to 172.17.0.9, and started at htt
p://swan006.cern.ch:9017
```

Talk: Apache Spark usage and deployment. P. Kothuri, E. Tejedor et al

SWAN for Education

- UP2University European Project
  - Bridge the gap between secondary schools, higher education and the research domain
- SWAN used by students to learn physics and other sciences
  - Let them use the very same tools & services used by scientists at CERN
- SWAN Boxed: distribution easily deployable on premises
  - https://github.com/cernbox/uboxed

Poster: The EU Up to University Project, E.Bocchi, J. Moscicki

Conclusion

Summary

- The interface of SWAN has been completely redesigned to foster collaboration and sharing of results among scientists

- Sharing is now fully integrated in SWAN, where users can share their work in the form of Projects (notebooks + data)

- New functionality to share a project, list shared projects, inspect and clone shared projects into users CERNBox

- Concurrent edition of notebooks in Jupyter would give another dimension to sharing

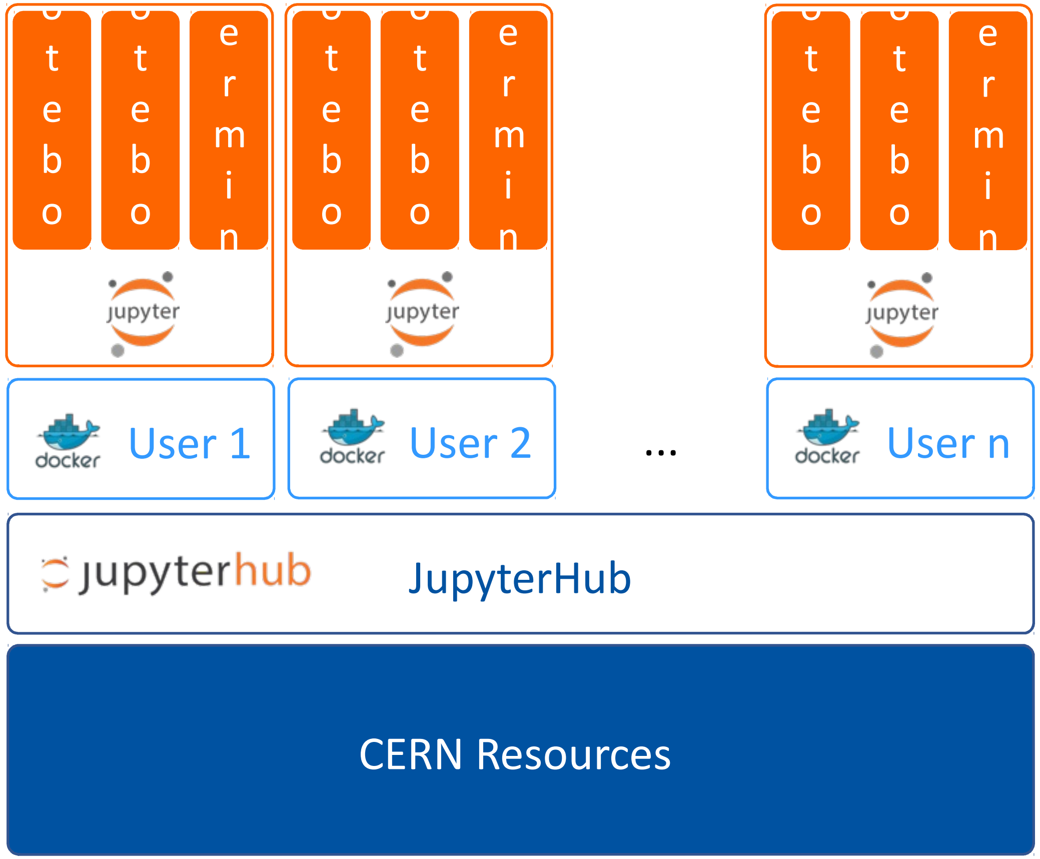
Facilitating Collaborative Analysis in SWAN

Thank you

Backup Slides

Integrating Jupyter

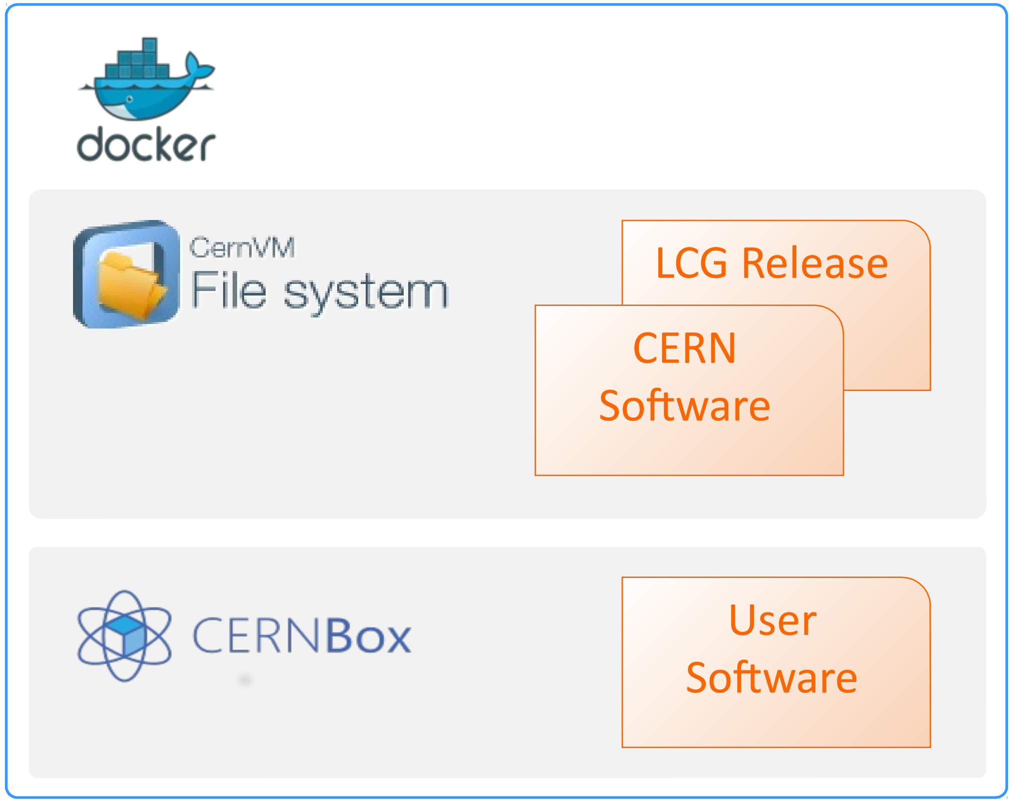
- Configurable environments through user defined scripts
- Jupyterhub to allow multiple Jupyter instances
  - Single instance of Jupyter per user
- User sessions spawned as Docker containers
  - Enforces resource limits per user
  - To isolate users work



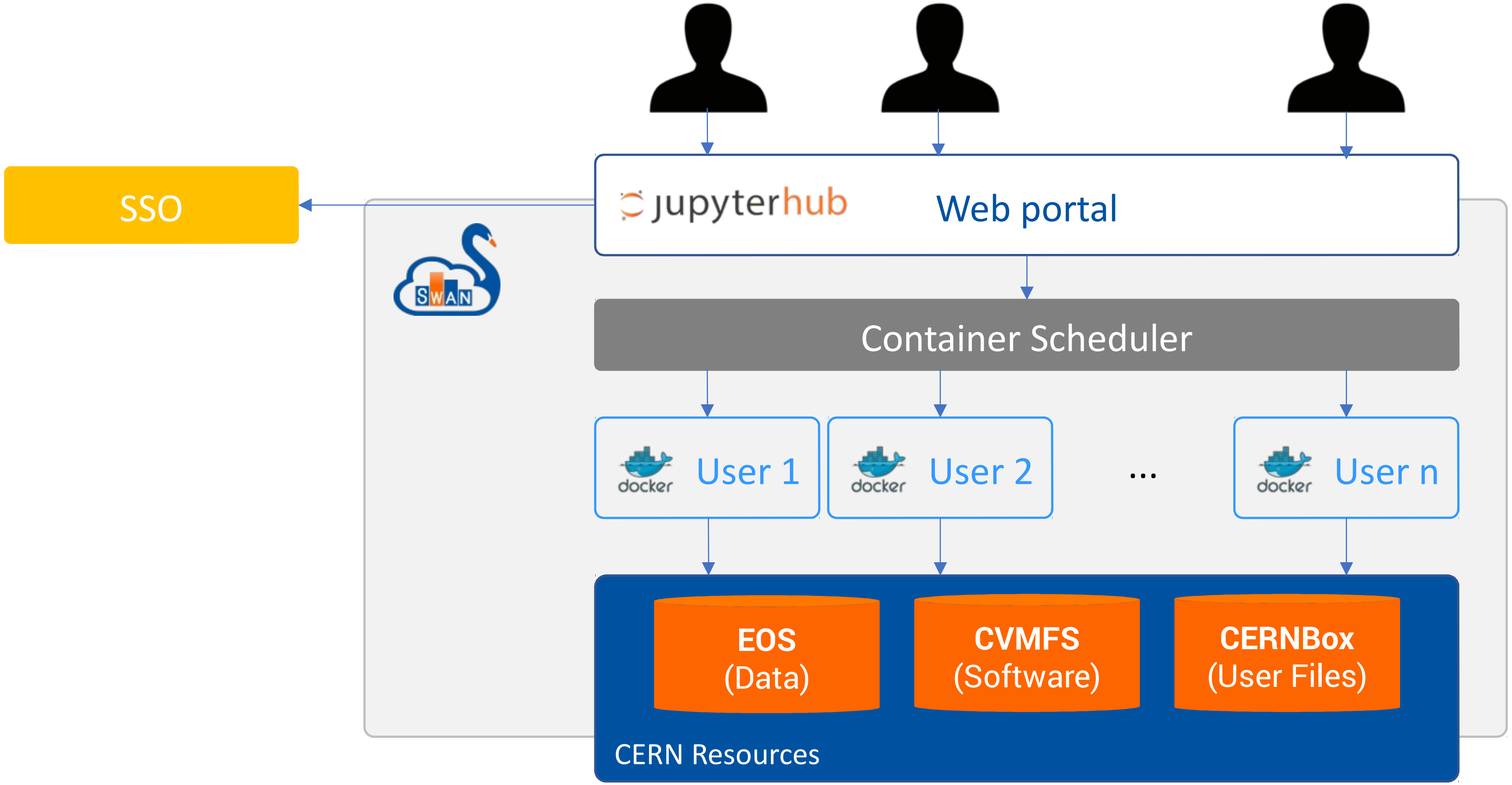
Software

- Software distributed through CVMFS
  - LCG Releases: distribute a series of compatible packages
  - Software used by researchers is available

- Possibility to install other libraries in user storage (CERNBox)



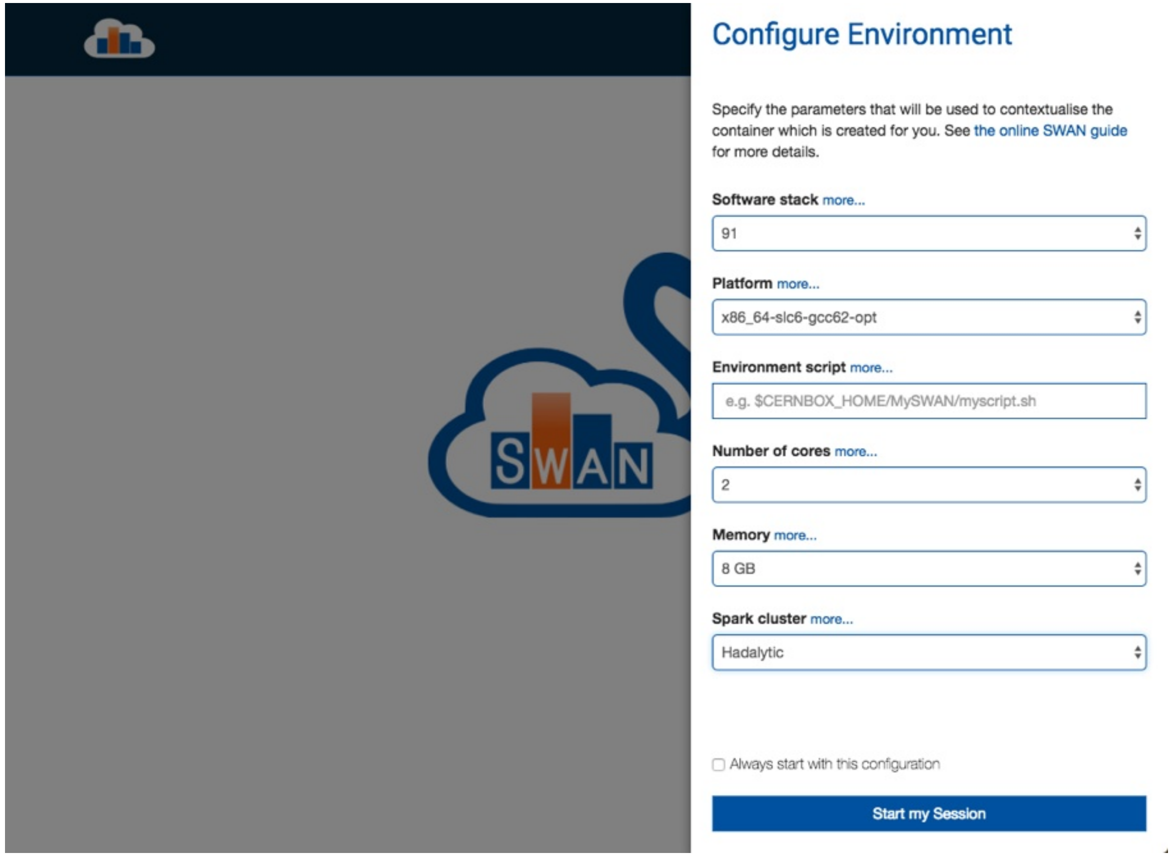
Architecture



The Notebook as Interface

- A web-based interactive interface and platform that combines code, equations, text and visualisations
  - Ideal for sharing/collaboration
  - A “shell opened within the browser”
- Many supported languages (kernels)
  - In SWAN: Python, ROOT C++, R and Octave
- Interactive, usually lightweight computations
- Very useful for some use cases
  - Final steps of an analysis, exploration, teaching, documentation and reproducibility

New User Interface



Configure Environment

Specify the parameters that will be used to contextualise the container which is created for you. See the online SWAN guide for more details.

Software stack more...
91

Platform more...
x86_64-slc6-gcc62-opt

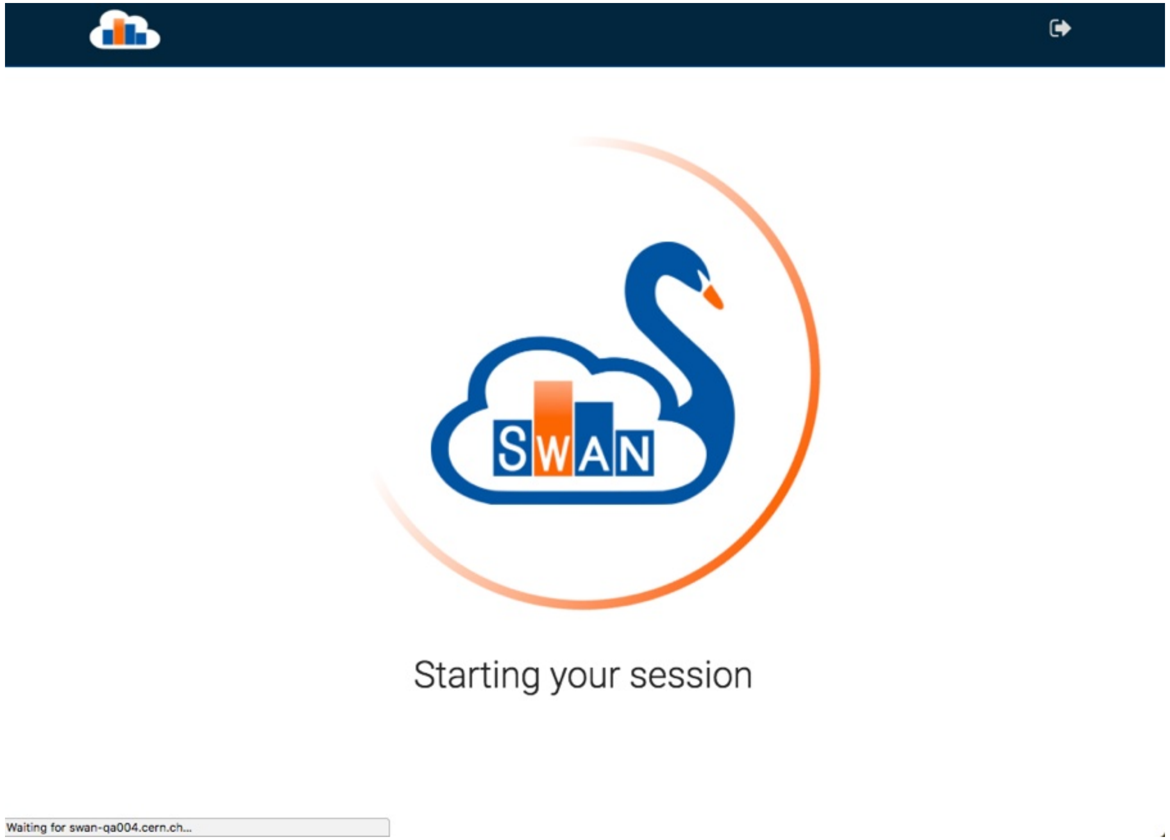
Environment script more...
e.g. $CERNBOX_HOME/MySWAN/myscript.sh

Number of cores more...
2

Memory more...
8 GB

Spark cluster more...
Hadalytic

□ Always start with this configuration



Starting your session

Waiting for swan-qa004.cern.ch...

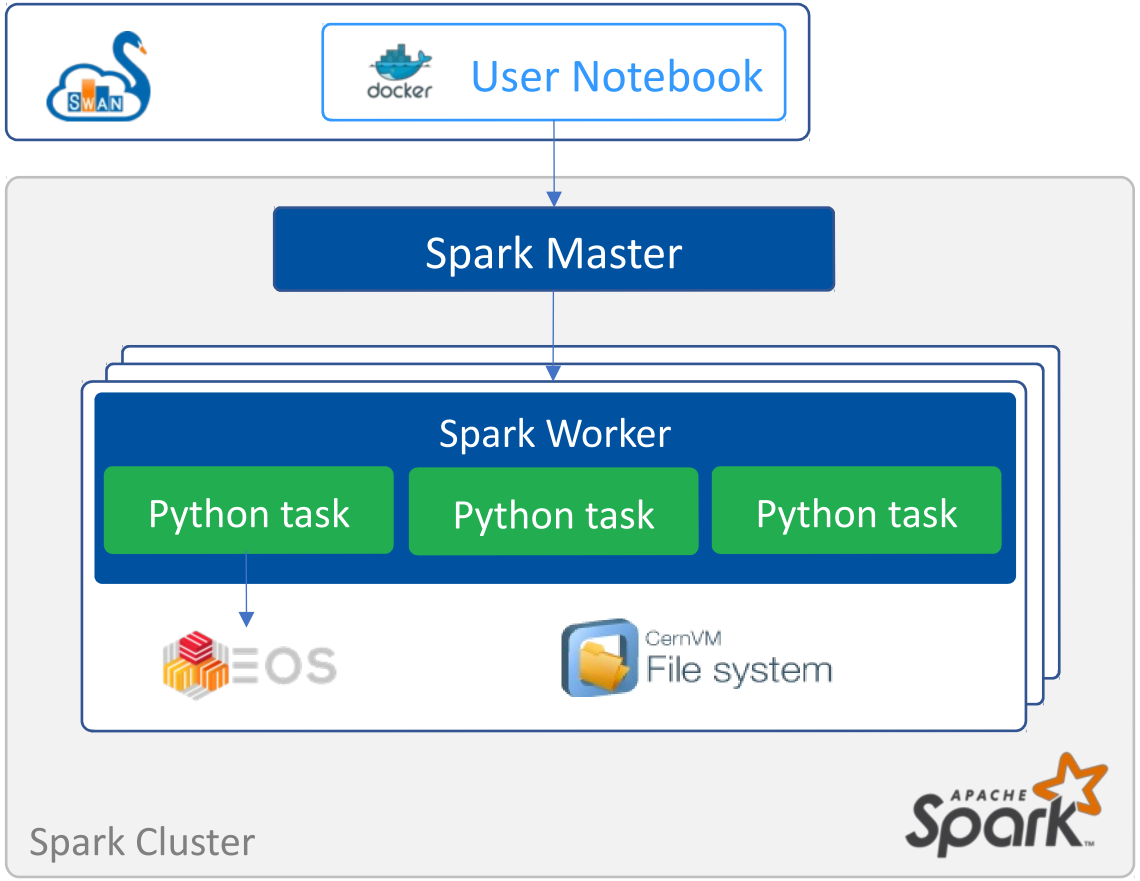
Boxed

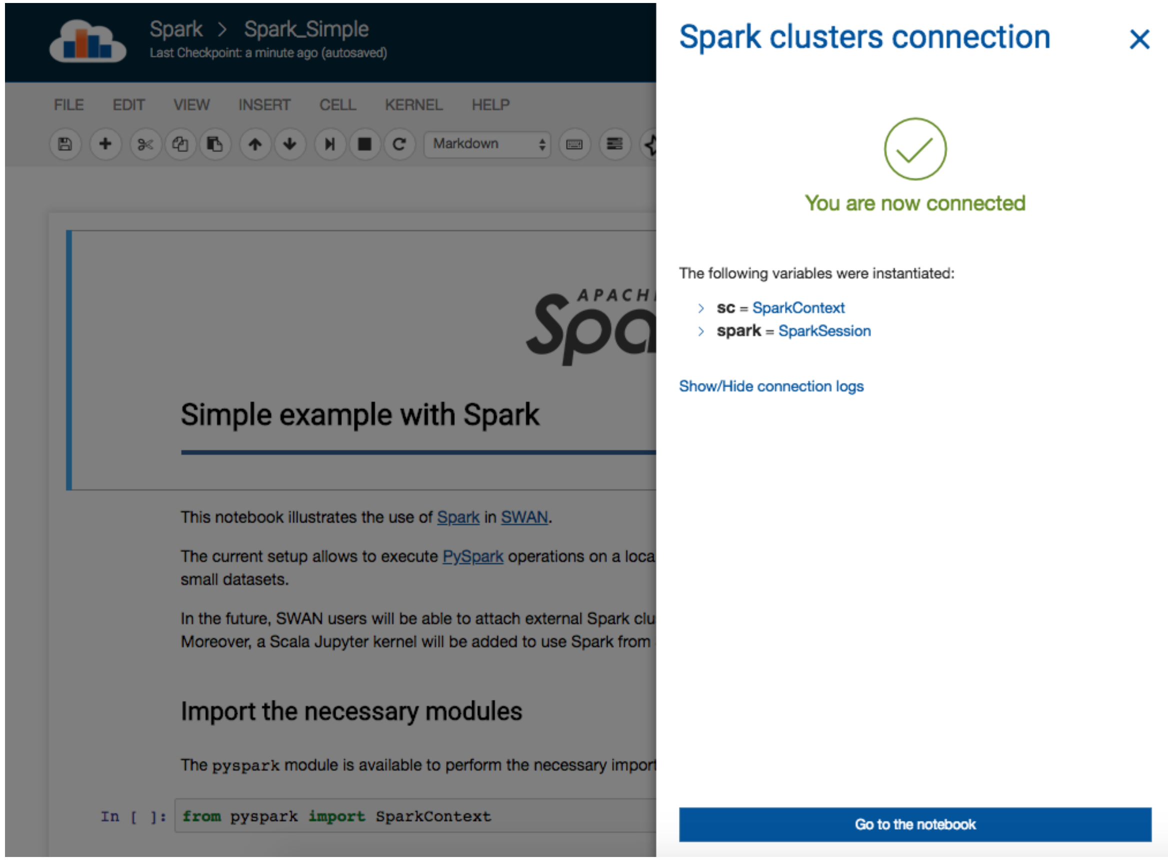
- Containerized version of all the infrastructure
  - Includes EOS, CERNBox, CVMFS and all Swan services (Jupyter Docker image, JupyterHub)
  - Available in https://github.com/cernbox/uboxed

- Easily deployable on premises
  - Installable in Linux systems
  - Based on Docker Compose

Integration with Spark

- Connection to CERN Spark Clusters
- User data accessed through EOS
- Graphical Jupyter extensions developed
  - Spark Connector
  - Spark Monitor





Simple example with Spark

This notebook illustrates the use of Spark in SWAN.

The current setup allows to execute PySpark operations on a loca
small datasets.

In the future, SWAN users will be able to attach external Spark clu
Moreover, a Scala Jupyter kernel will be added to use Spark from

Import the necessary modules

The pyspark module is available to perform the necessary impor

```
In [ ]: from pyspark import SparkContext
```

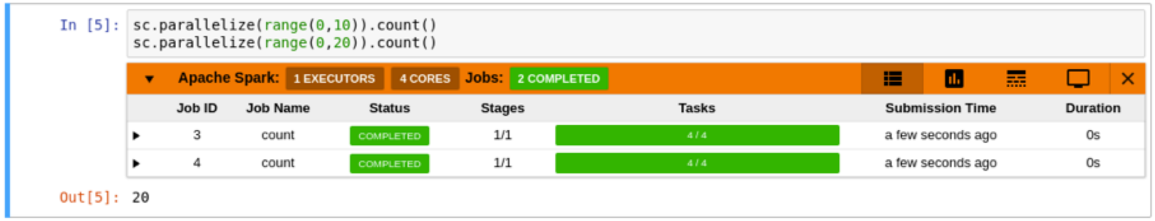
Spark clusters connection

You are now connected

The following variables were instantiated:

- sc = SparkContext
- spark = SparkSession

Show/Hide connection logs



```
In [5]: sc.parallelize(range(0,10)).count()
        sc.parallelize(range(0,20)).count()
```

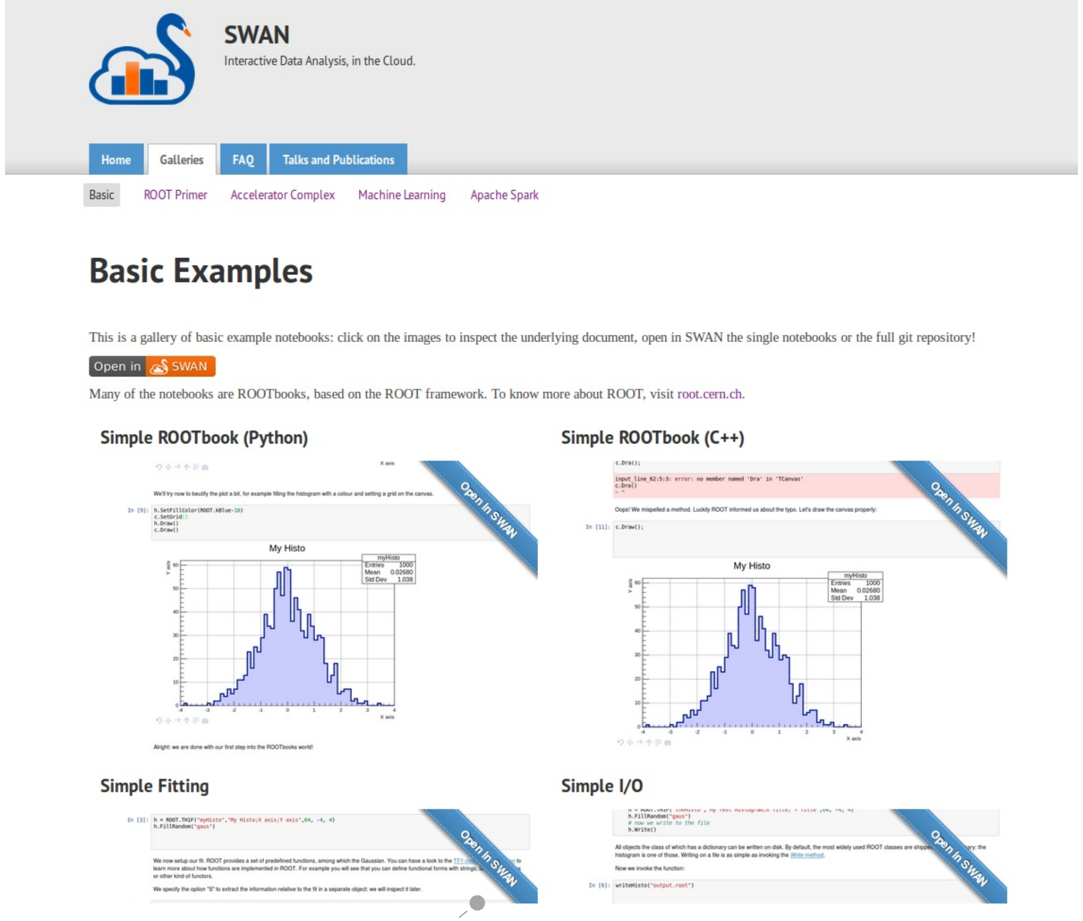
	Job ID	Job Name	Status	Stages	Tasks	Submission Time	Duration
▸	3	count	COMPLETED	1/1	4 / 4	a few seconds ago	0s
▸	4	count	COMPLETED	1/1	4 / 4	a few seconds ago	0s

```
Out[5]: 20
```

HEP User Community

- SWAN development is guided by our user community
  - New features (libs, kernels, ...) are requested by users from their real usage needs
- Gallery of examples
  - Made in collaboration with our users
  - Almost 50 notebooks in 7 categories

Example notebooks at **swan.web.cern.ch**



SWAN

Interactive Data Analysis, in the Cloud.

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!

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

Simple ROOTbook (Python)

We'll try now to beutify the plot a bit, for example filling the histogram with a colour and setting a grid on the canvas.

```
In [9]: h.SetFillColor(ROOT.kBlue-10)
        c.SetGrid()
        h.Draw()
        c.Draw()
```

Alright: we are done with our first step into the ROOTbooks world!

Simple ROOTbook (C++)

```
c.Dra();
input_line_62:5:3: error: no member named 'Dra' in 'TCanvas'
c.Dra()
~ ^
```

Oops! We mispelled a method. Luckily ROOT informed us about the typo. Let's draw the canvas properly:

```
In [11]: c.Draw();
```

Simple Fitting

```
In [3]: h = ROOT.TH1F("myHisto","My Histo;X axis;Y axis",64, -4, 4)
        h.FillRandom("gaus")
```

We now setup our fit. ROOT provides a set of predefined functions, among which the Gaussian. You can have a look to the TF1 cl… to learn more about how functions are implemented in ROOT. For example you will see that you can define functional forms with strings, … or other kind of functors.

We specify the option "S" to extract the information relative to the fit in a separate object: we will inspect it later.

Simple I/O

```
h.FillRandom("gaus")
# now we write to the file
h.Write()
```

All objects the class of which has a dictionary can be written on disk. By default, the most widely used ROOT classes are shipp… :ry: the histogram is one of those. Writing on a file is as simple as invoking the Write method.

Now we invoke the function:

```
In [6]: writeHisto("output.root")
```

Access with only a click