

SWAN: service for web based analysis



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

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



Mar 27th, 2018
WLCG-HSF workshop

Introduction



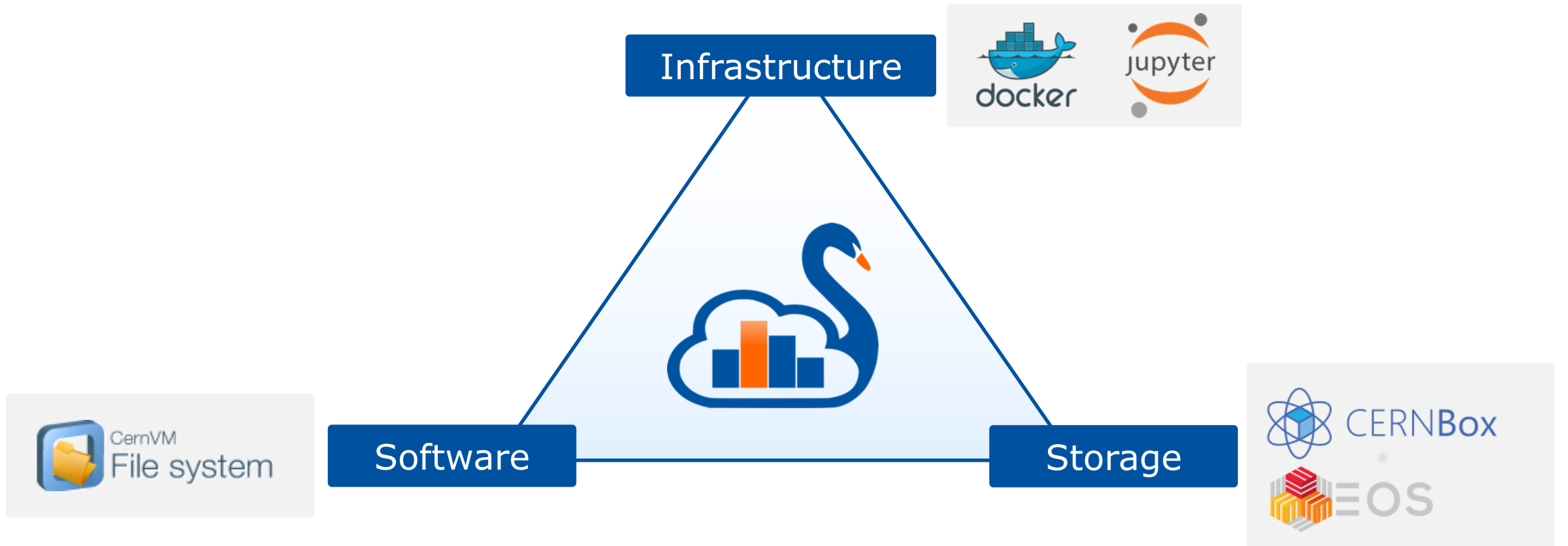
Motivation

- › Data Analysis only with a web browser
 - Available everywhere and at anytime
 - Integrated with other analysis ecosystems : ROOT, R, Python, Octave, ...
- › Easy to use (but powerful)
 - No local installation and configuration needed
- › Sharable scientific results
 - Plots, data, code
- › Integration with CERN resources
 - Access software, user/experiments data, computing resources

SWAN



Integrating services





Jupyter - The Notebook as Interface

- > A web-based interactive interface and platform that combines code, equations, text and visualisations
 - Ideal for sharing/collaboration
 - ... In a nutshell: an “interactive 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





Text

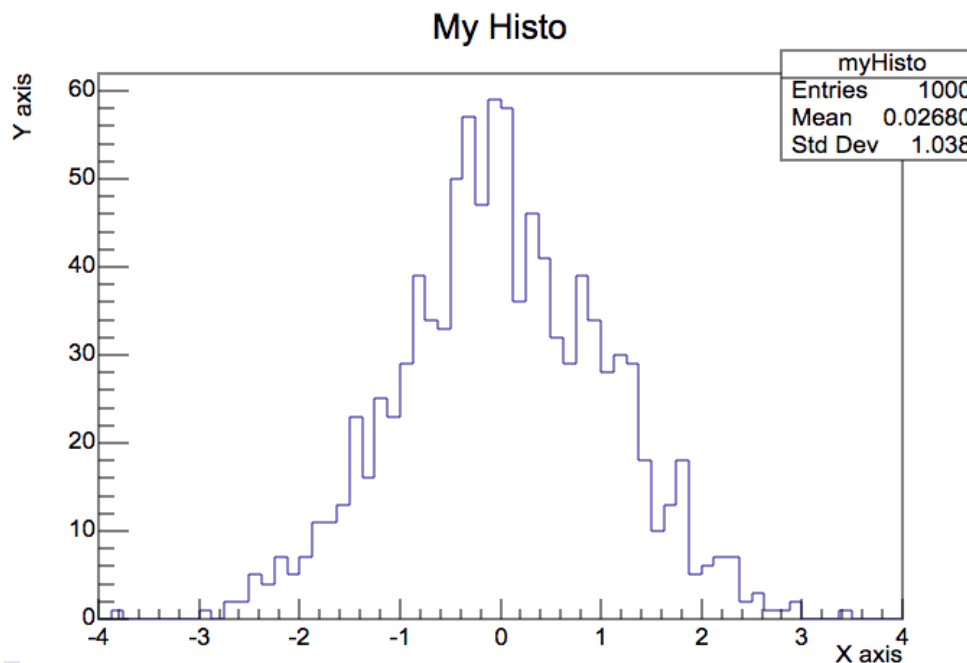
Displaying graphics

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!

Code

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

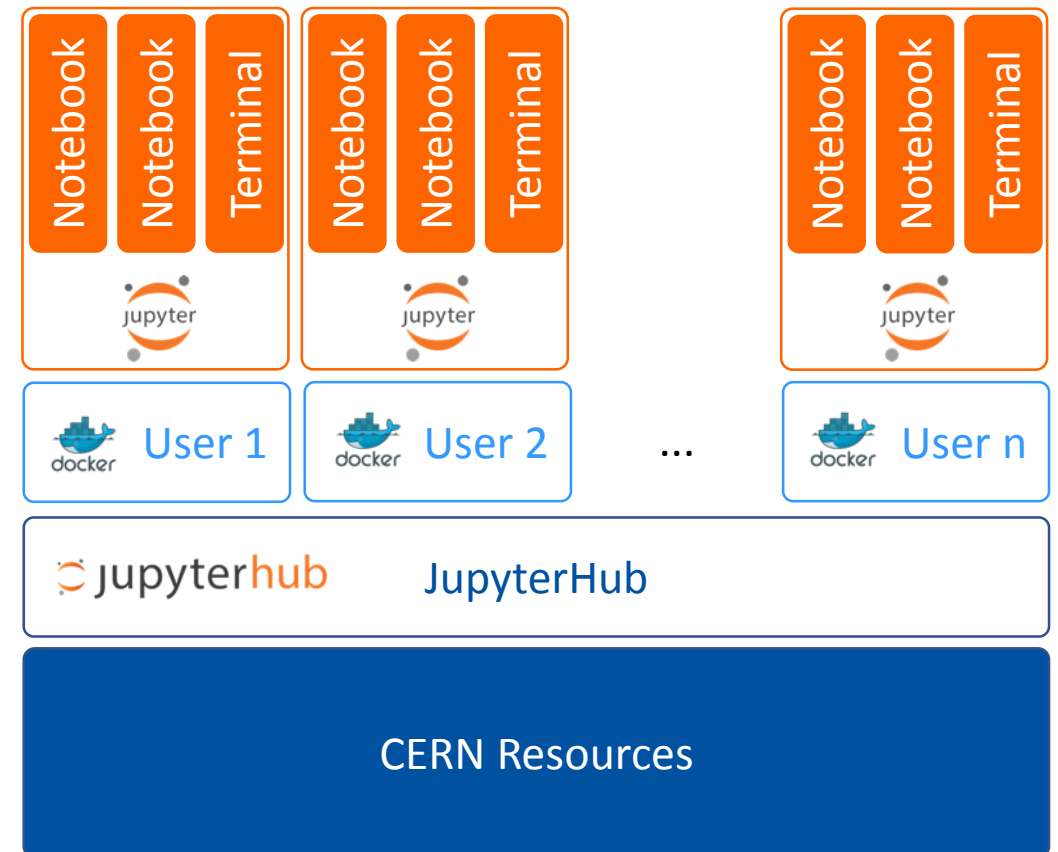
Graphics





Integrating Jupyter

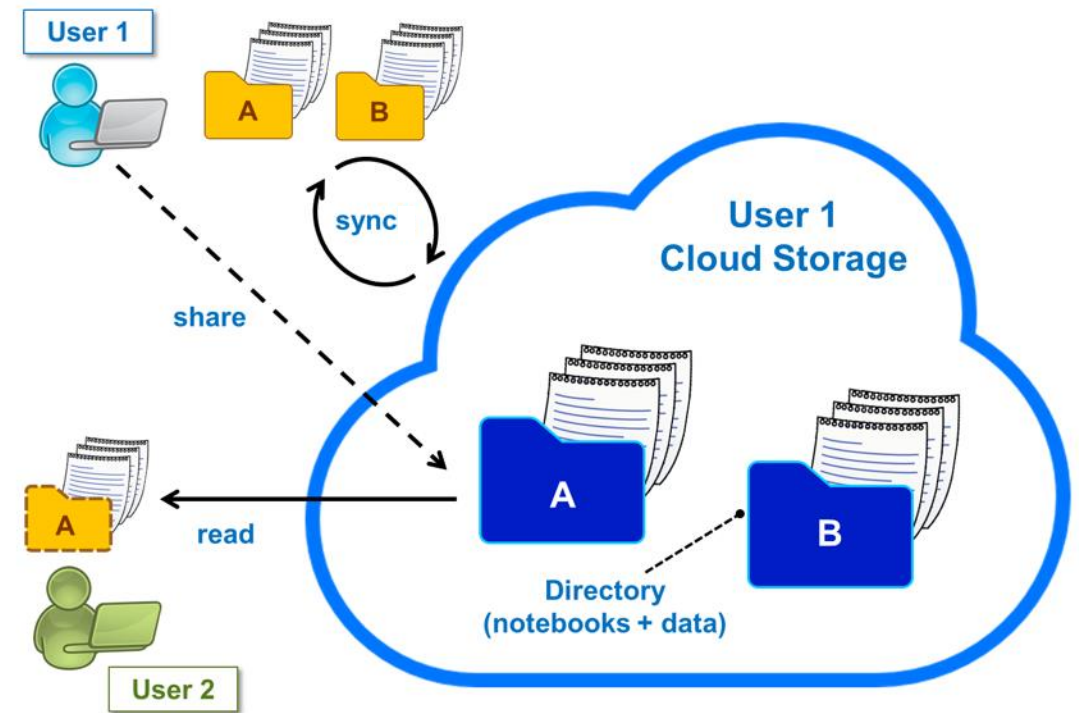
- > Jupyterhub to allow multiple Jupyter instances
 - Single instance of Jupyter per user
- > User sessions spawned as Docker containers
 - Enforces resource limits per user
 - To isolate users work
- > Configurable environments through user defined scripts





Storage

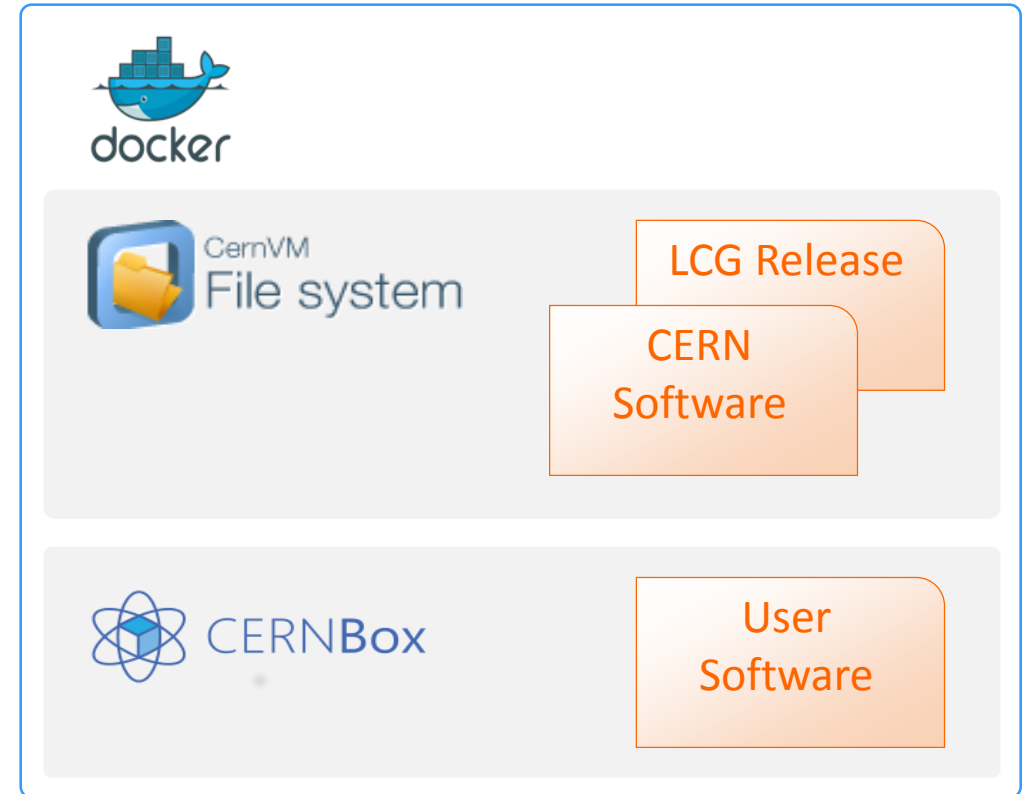
- > User personal user space
 - Synchronized through CERNBox
 - All files synced across devices, the cloud and other users
- > Uses EOS mass storage system
 - All experiment data potentially available





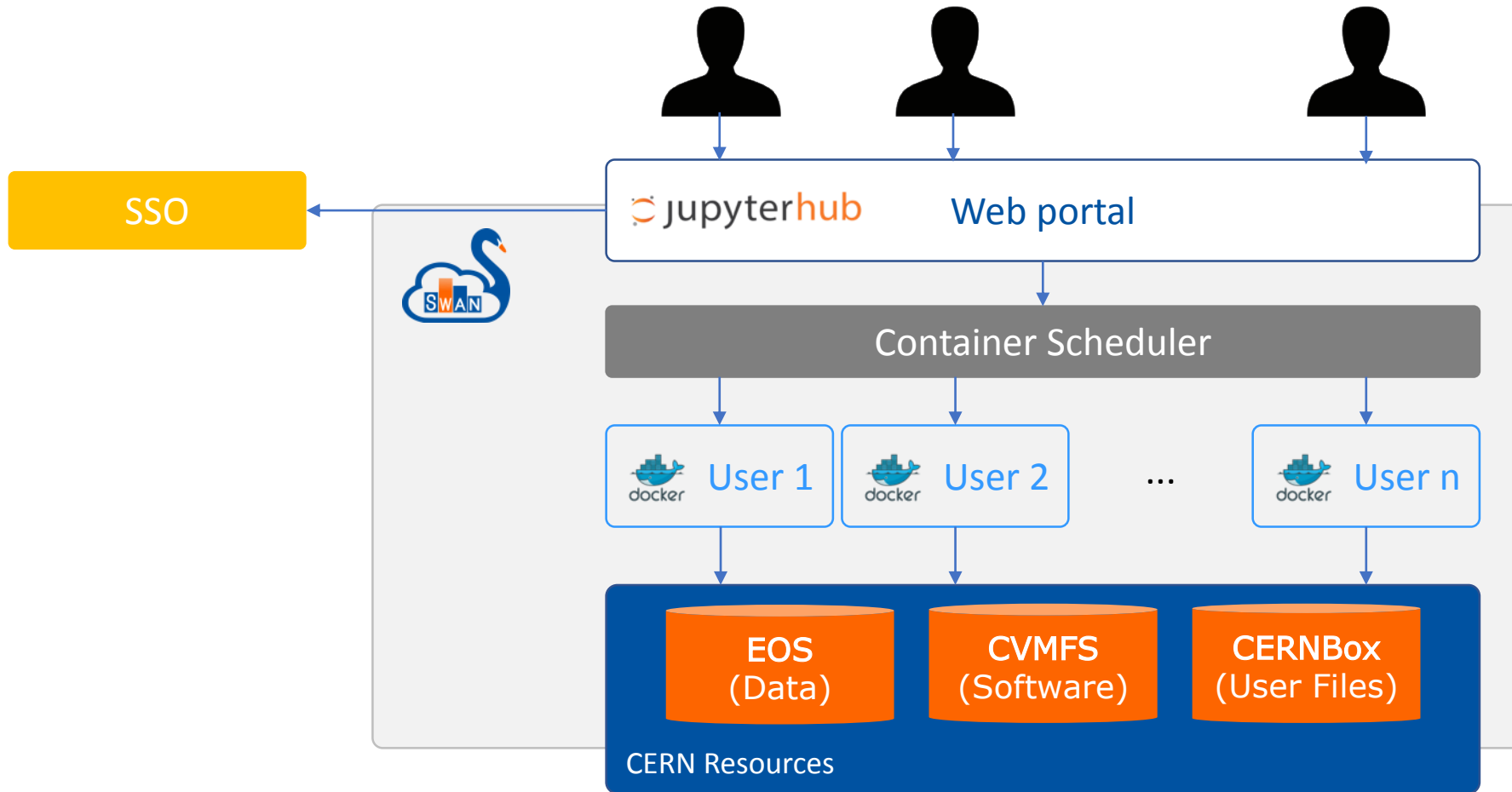
Software

- > Software distributed through CVMFS
 - "LCG Releases" - packing a large software stack with most common software
 - Experiment software used by researchers is available
- > Possibility to install other libraries in user storage (CERNBox)





Architecture





Boxed SWAN

- › Containerized version with all the needed 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

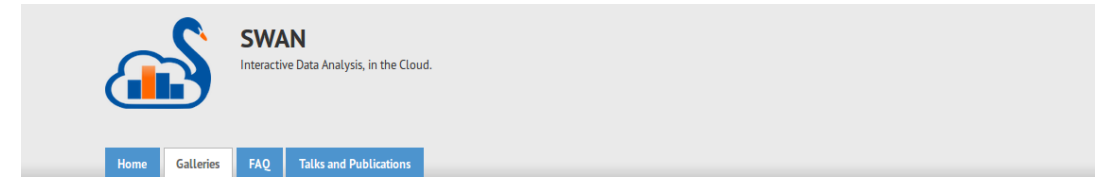
SWAN and the community



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



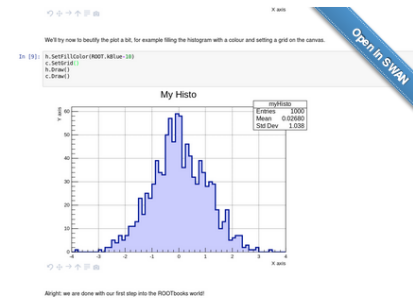
Basic Examples

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.

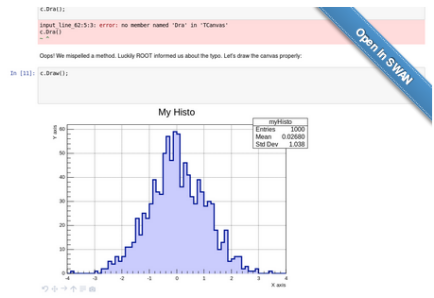
Simple ROOTbook (Python)



Simple Fitting



Simple ROOTbook (C++)



Simple I/O



Access with only a click





Other Communities


- > Building block in UP2University European Project
 - Bridge the gap between secondary schools, higher education, and the research domain
 - SWAN used by students to learn Physics and other Natural Sciences
 - Let the kids use the very same tools & services used by real researchers doing Big Science at CERN
 - Using Boxed distribution



Recent developments



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

Start my Session



Starting your session

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



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

Spark > physics_analysis_using_swan_spark_template (autosaved)

FILE EDIT VIEW INSERT CELL KERNEL HELP Not Trusted Python 2

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 inside SWAN interface
 - Integration with CERNBox
- > Users can share “Projects”
 - Special kind of folder that contains notebooks and other files, like input data

The screenshot displays the SWAN interface with a 'Share Project' dialog box open. The background shows a 'Projects' view with 'Projects shared with me' and 'Projects shared by me' sections. The 'Share Project' dialog is open, showing the project name 'Proj1', a search input field, and a list of users to share with: 'etejedor' and 'dpiparo'. There are 'Stop Sharing' and 'Update' buttons at the bottom of the dialog.



Sharing made easy

- > Users can clone a shared Project directly from the interface
 - Jupyter doesn't allow concurrent editing

The screenshot shows the 'Share' interface in CERNBox. The breadcrumb path is 'SWAN > Share'. There are two sections: 'Projects shared with me' and 'Projects shared by me'. The first section contains a table with one entry: 'ProjTest' (5.64 MB, shared by 'diocas' 25 days ago). The second section contains a table with one entry: 'Proj1' (shared with '2 people/groups' 5 days ago). The footer includes copyright information for CERN 2017 and a CERN logo.

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

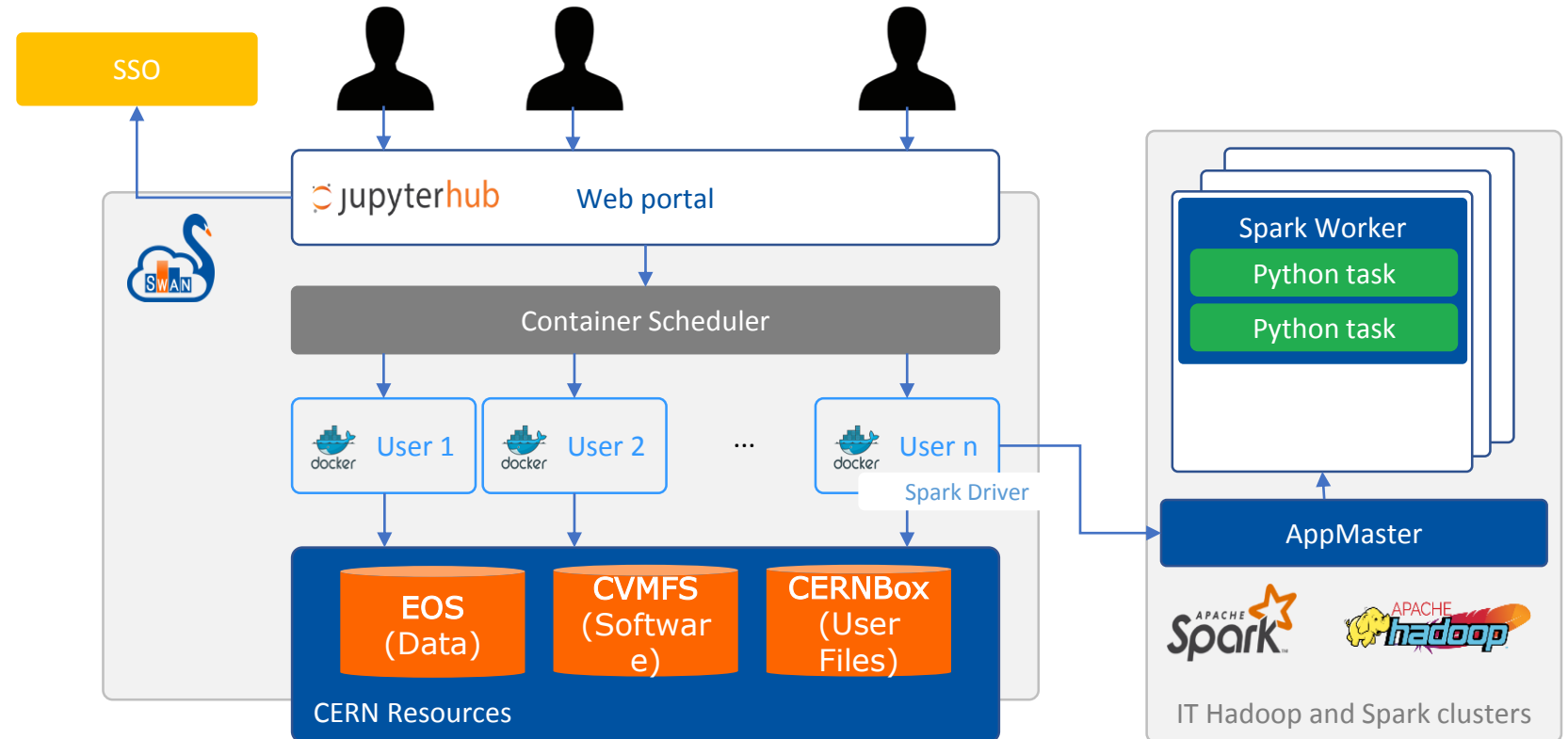
| NAME | SHARED WITH | DATE |
|-------|-----------------|------------|
| Proj1 | 2 people/groups | 5 days ago |





Integration with Spark

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





Integration with Spark

Spark > Spark_Simple
Last Checkpoint: a minute ago (autosaved)

FILE EDIT VIEW INSERT CELL KERNEL HELP

Markdown

Simple example with Spark

This notebook illustrates the use of [Spark](#) in [SWAN](#).

The current setup allows to execute [PySpark](#) operations on a local small datasets.

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

Import the necessary modules

The `pyspark` module is available to perform the necessary imports.

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

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...](#)
93

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...](#)
None
Hadalytic
Analytic

Always start with this configuration

[Start my Session](#)



Spark > Spark_Simple
Last Checkpoint: a minute ago (autosaved)

FILE EDIT VIEW INSERT CELL KERNEL HELP

Markdown

Simple example with Spark

This notebook illustrates the use of [Spark](#) in [SWAN](#).

The current setup allows to execute [PySpark](#) operations on a local small datasets.


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

Import the necessary modules

The `pyspark` module is available to perform the necessary imports.

```
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](#)

[Go to the notebook](#)

- Spark Monitor – jupyter notebook extension
 - For live monitoring of spark jobs spawned from the notebook
 - Access to Spark WEB UI from the notebook
 - Several other features to debug Spark application

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

| ▼ Apache Spark: 1 EXECUTORS 4 CORES Jobs: 2 COMPLETED | | | | | | | |
|---|----------|-----------|--------|-------|-------------------|----------|--|
| 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
```

Further Research and Development



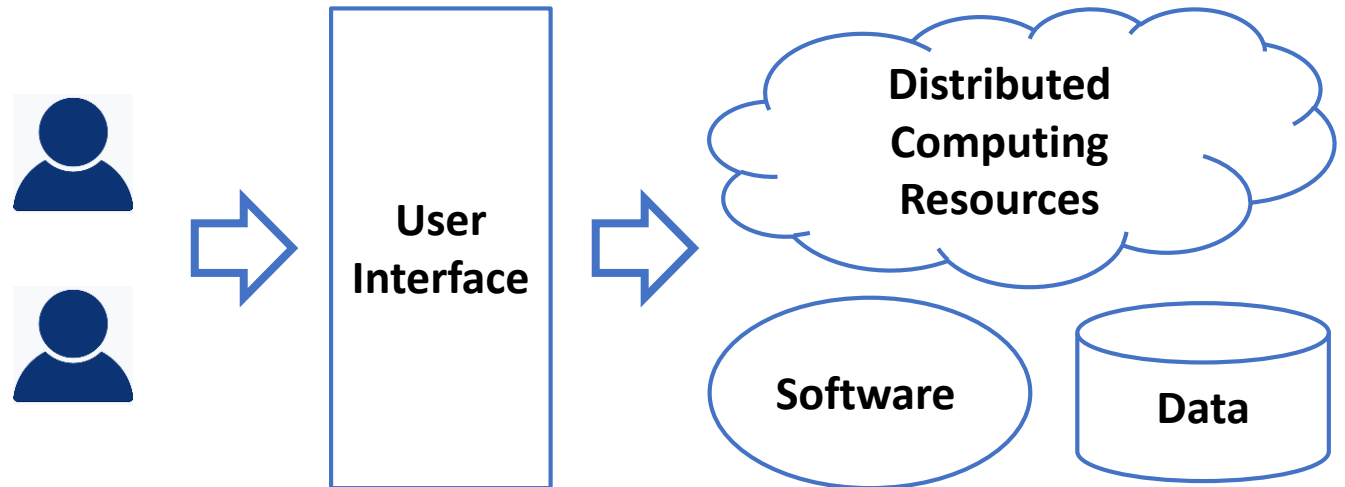
Scaling what you can do with your laptop

- › **Batch/Grid jobs** has been a very successful model for distributed analysis in HEP, but:
 - Wrapper code / scripts necessary to perform submission
 - Not interactive: cannot view intermediate application results
 - Hard to monitor progress / estimate time to completion
 - Hard to debug and analyze performance
- › As soon as they can, physicists work on **ntuples that fit in their computer**
 - Simpler, more user-friendly, interactive, full control of the resources
 - Will this still be possible for HL-LHC?



Interactive Distributed Analysis Environment

- > **Easy entry** point to distributed execution
- > **High-level** programming model
 - Declarative
 - No changes to go distributed
 - Interactive, easily modifiable
- > Pluggable computational **HPC/cloud** resources
- > Leverage on **big data** technologies from industry



Conclusion



Conclusion

- > SWAN is a CERN service that provides Jupyter Notebooks on demand
- > SWAN promotes a cloud based analysis model where users can do analysis only with their browser
- > SWAN federates CERN services for software, storage and infrastructure so that users can find what they need in the service
- > SWAN fosters collaboration and results sharing between scientists
- > SWAN is an interface for large computing resources (Spark)
- > SWAN is the first step towards a **truly scalable and interactive** distributed data analysis environment

SWAN: service for web based analysis

Thank-you