

# SWAN and its analysis ecosystem

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

<https://cern.ch/swan>



Jan 29th, 2019

CS3 2019 - Cloud Storage Synchronization and Sharing Services



# Introduction



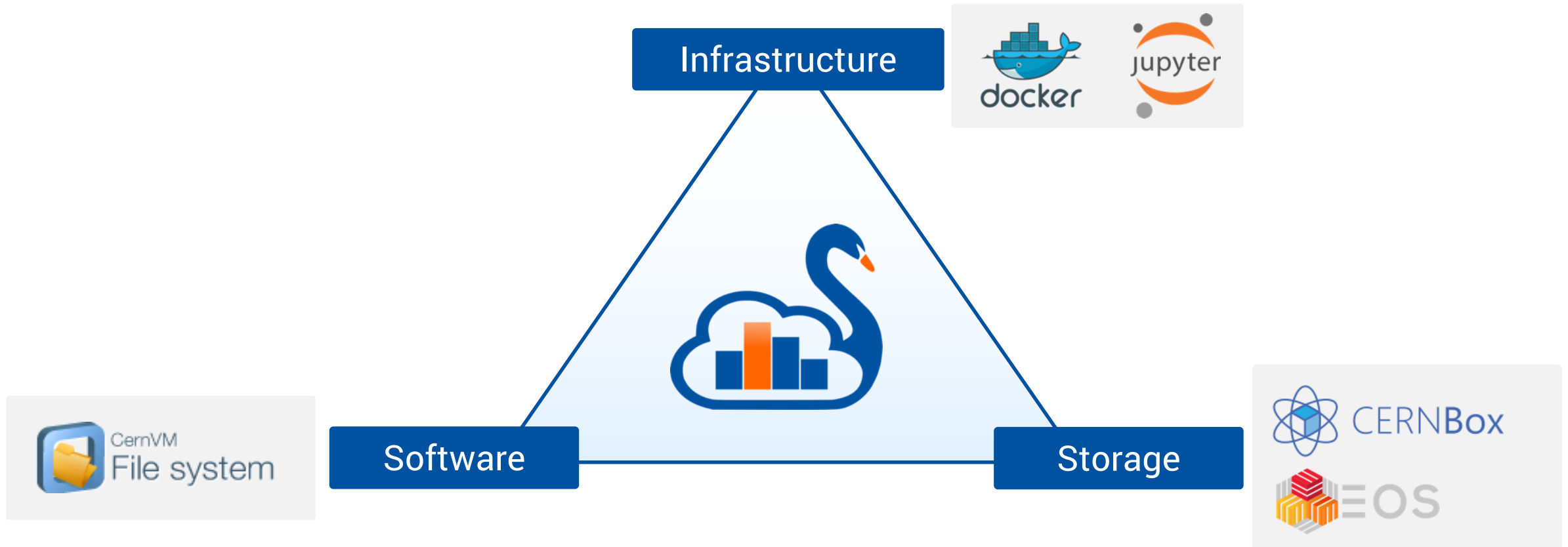
# SWAN in a Nutshell

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





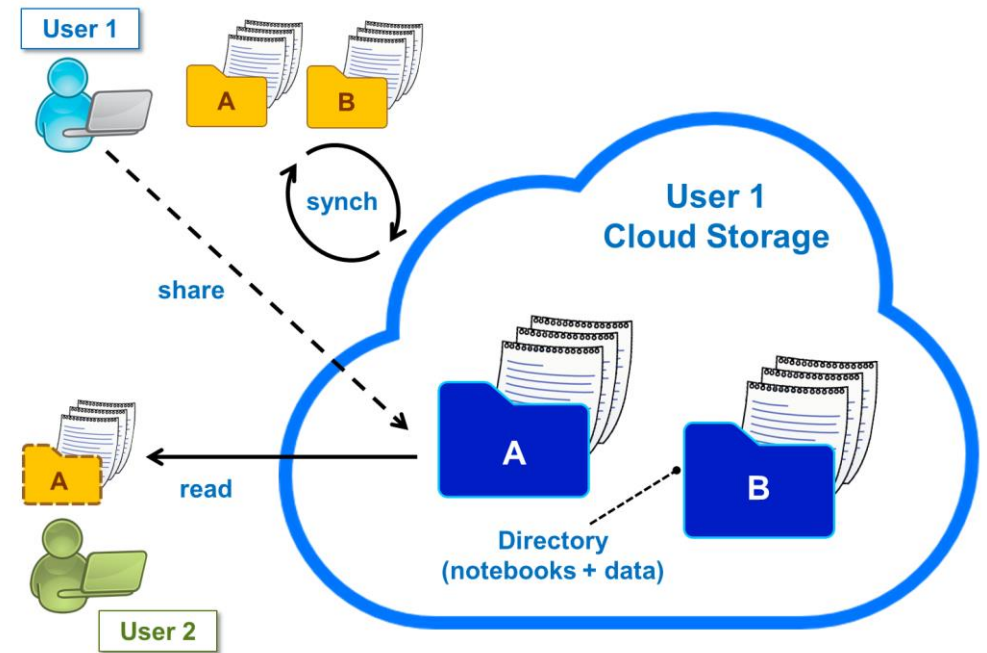
# Integrating services





# Storage

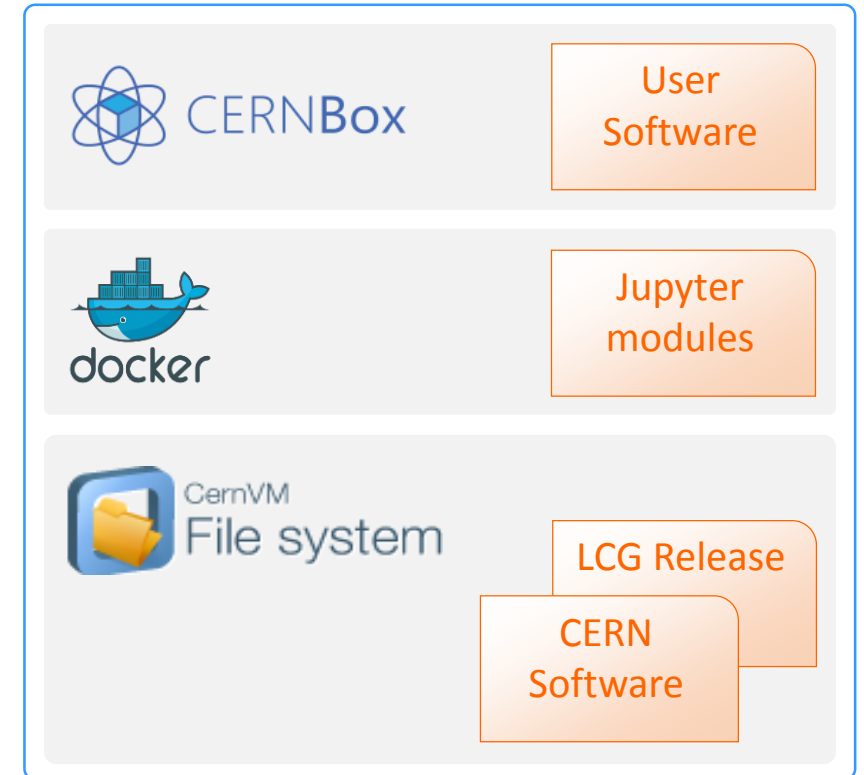
- > Uses EOS disk storage system
  - All experiment data potentially available
- > CERNBox is SWAN's home directory
  - Storage for your notebooks and data
- > Sync&Share
  - Files synced across devices and the Cloud
  - Collaborative analysis





# Software

- > Software distributed through CVMFS
  - "LCG Releases" - pack a series of compatible packages
  - Reduced Docker Images size
  - Lazy fetching of software
- > Possibility to install libraries in user cloud storage
  - Good way to use custom/not mainstream packages
  - Configurable environment



Previously on last CS3 conference...



# 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
  - Self contained

The screenshot displays the SWAN interface for sharing a project. The main view shows a project titled "Super Real Analysis with TOTEM data" containing files like "DistilDistribution.ipynb" and "dataset.root". A "Share Project" dialog is open on the right, showing the project name, a search box for users, and a list of users already shared with: Danilo Piparo (danilo) and Enric Tejedor Saavedra (enric). Buttons for "Stop Sharing" and "Update" are at the bottom of the dialog.



# The Share tab

- > Users can list which projects...
  - they have shared
  - others have shared with them
- > Projects can be cloned to the receiver's CERNBox
  - The receiver will work on his own copy
- > Concurrent editing not supported by Jupyter
  - Safer to clone

SWAN > Share

### Projects shared with me ^

NAME ▾	SIZE	SHARED BY	DATE
UP2University Pilot	Empty	jupytercon2	7 minutes ago

[Clone](#)

### Projects shared by me ^

NAME ▾	SHARED WITH	DATE
Higgs Boson discovery	2 people/groups	18 hours ago
Super Real Analysis with TOTEM data	diogo	19 hours ago

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

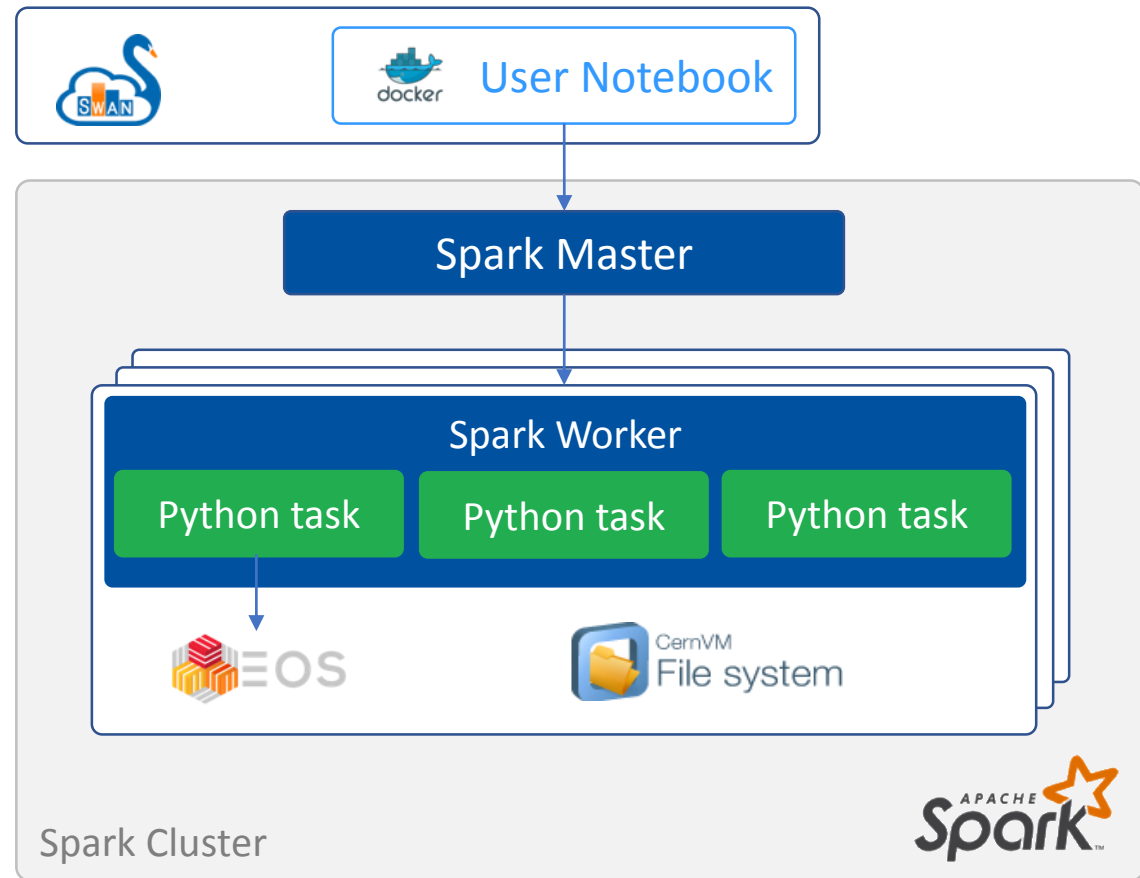
javascript:





# Integration with Spark

- > Connection to CERN Spark Clusters
- > Same environment across platforms
  - User data - EOS
  - Software - CVMFS
- > Graphical Jupyter extensions developed
  - Spark Connector
  - Spark Monitor





# Spark Connector/Monitor



The screenshot shows a Jupyter notebook titled "Spark > Spark\_Simple (autosaved)". The notebook content includes a section "Simple example with Spark" and a code cell with the following text:

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

Overlaid on the notebook is a "Spark clusters connection" dialog box. It contains the following information:

- Header: Spark clusters connection
- Text: You are going to connect to: **hadalytic**
- Text: You can configure the following options. Environment variables can be used via {ENV\_VAR\_NAME}.
- Section: Add a new option. A text input field with the placeholder "Write the option name...".
- Section: Bundled configurations. A checkbox labeled "Include NXCALs options" which is currently unchecked.
- Section: Selected configuration. A list of configurations:
  - spark.shuffle.service.enabled: false
  - spark.driver.memory: 2g
  - spark.executor.instances: 4
- Bottom button: Connect

```
In [ ]: def f(x):  
        global a  
        a+=x  
        RDD9.foreach(f)  
        RDD9.foreach(f)  
        print(a.value)  
        #Display should appear automatically
```

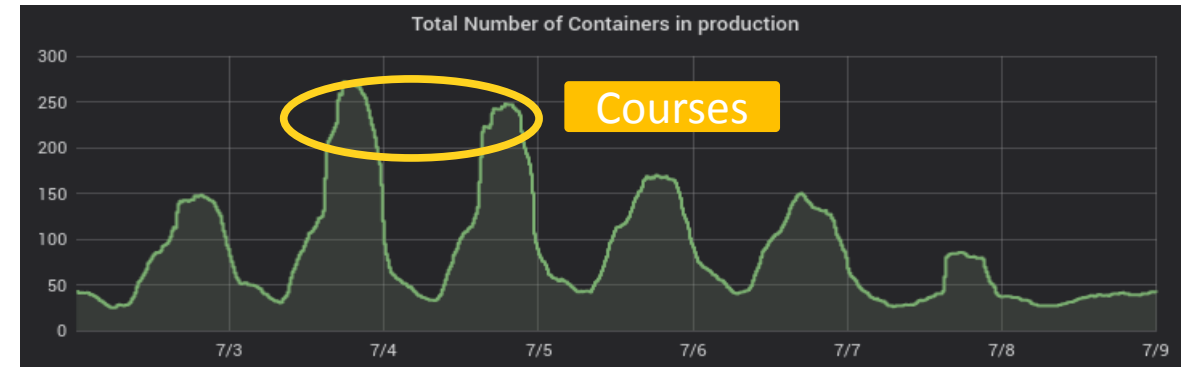


# The result



# Stats

- > **~200 user sessions** a day on average
  - Users doubled last year with new SWAN interface
- > **~1300 unique users** in 6 months
- > Spark cluster connection: 15 – 20 % of users
  - SWAN as entry point for accessing computational resources
  - Used for monitoring LHC accelerator hardware devices (NXCals)



# New developments



# Inspecting a Project

- > Users can inspect shared project contents
  - Browsing of the 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;
```

```
In [5]: h.Draw();
c.Draw();
```

my-Histo	
Entries	1000
Mean	0.02680
Std Dev	1.038



# Spark improvements

Spark > Spark\_Simple (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 going to connect to: **hadalytic**

You can configure the following options. Environment variables can be used via {ENV\_VAR\_NAME}.

**Add a new option**

**Bundled configurations**

Include NXCALs options

**Selected configuration**

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

**Connect**

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**

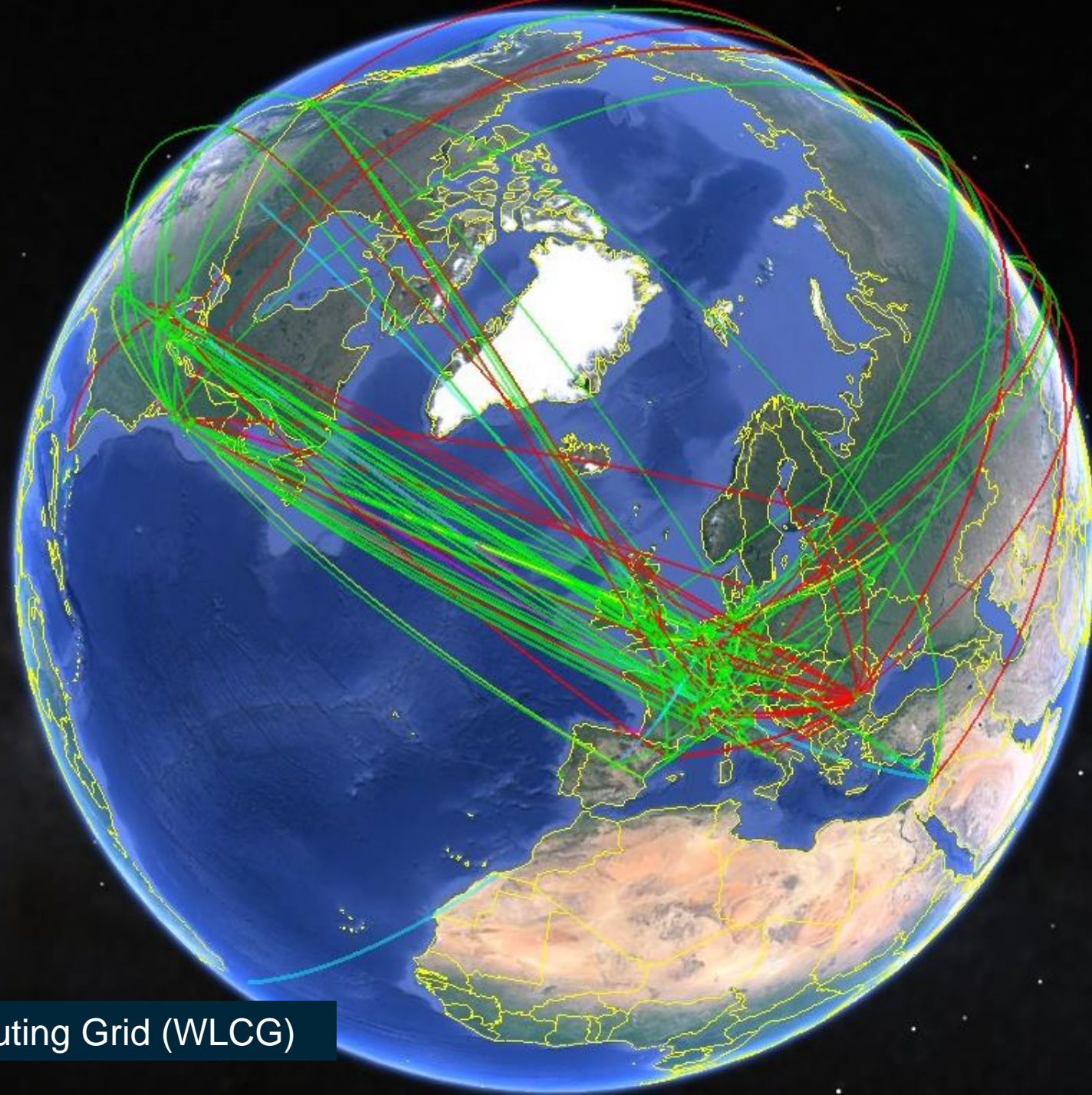
The following variables were instantiated:

- sc = SparkContext
- spark = SparkSession

[Show/Hide connection logs](#)

**Go to the notebook**





Worldwide LHC Computing Grid (WLCG)



# Connecting More Resources

> Ongoing effort: submit **batch jobs** from the notebook

- Monitoring display
- Jobs tab



Projects Share **Jobs** CERNBox

Select jobs to perform action on them. Job ID Jobs per page

<input type="checkbox"/> 0	Job ID	Job Name	Backend	Application	File Name	Status	Submission Time	Runtime
<input type="checkbox"/>	3		Condor	Executable		SUBMITTING	Jul 24th, 3:26 pm	-
<input type="checkbox"/>	2		Condor	Executable		NEW	-	-
<input type="checkbox"/>	1		Localhost	Executable		NEW	-	-
<input type="checkbox"/>	0		Localhost	Executable		COMPLETED	Jul 24th, 3:21 pm	00 seconds

```
In [6]: %%ganga
j=ganga.Job()
j.submit()
```

Backend: LOCALHOST Application: EXECUTABLE Splitter: None 0 SUBJOBS

Job ID	Job Name	Status	Subjobs	Submission Time	Runtime
0		COMPLETED	No Subjobs	Jul 24th, 3:21 pm	00 seconds



# Outreach, Education



# Science Box: SWAN on Premises

- › Packaged deployment of SWAN
  - Includes all SWAN components: CERNBox/EOS, CVMFS, JupyterHub
- › Deployable through Kubernetes or docker-compose
- › Some successful community installations
  - AARNet
  - PSNC
  - Open Telekom Cloud (Helix Nebula)



# Science Box: SWAN on Premises

- › UP2University European Project
  - Bridge the gap between secondary schools, higher education and the research domain
  - Partner universities (OU, UROMA, NTUA, ...), pilot schools
  - <http://up2university.eu>
- › SWAN used by students to learn physics and other sciences
  - Let them use the very same tools & services used by scientists at CERN
  - Pilot with University of Geneva (Physiscope)
- › Establishing collaboration with Callysto project



# Looking ahead



# Future work/challenges

- › Move to Jupyterlab
  - Porting the current extensions
  - Concurrent editing
- › New architecture
  - Based on Kubernetes
- › Exploitation of GPUs
  - HEP is looking to ML
  - Speed up computation of GPU-ready libraries (e.g. TensorFlow)

# Where to find us



# Where to find us

## > Contacts

- [swan-talk@cern.ch](mailto:swan-talk@cern.ch)
- <http://cern.ch/swan>

## > Repository

- <https://github.com/swan-cern/>

## > Science Box

- <https://cern.ch/sciencebox>

# Conclusion



# Conclusion

- › Changes introduced since last year improved user experience
  - Which translated on more users using the service
- › SWAN became a fundamental Interface for Mass Processing Resources (Spark)
  - Not only for Physics analysis but also for monitoring the LHC hardware
- › The new Jupyterlab interface will bring new possibilities for collaborative analysis
  - With the introduction of concurrent editing of notebooks
  - Which can help reach more users
- › Successfully deployed outside CERN premises
  - Including on education related projects

# SWAN and its analysis ecosystem

Thank you

Diogo Castro  
diogo.castro@cern.ch