

# Integrated Science Platform at CERN

**Enrico Bocchi**

CERN – IT, Storage Group

**Dec 16th, 2019**

ESCAPE Tech Meeting, WP5



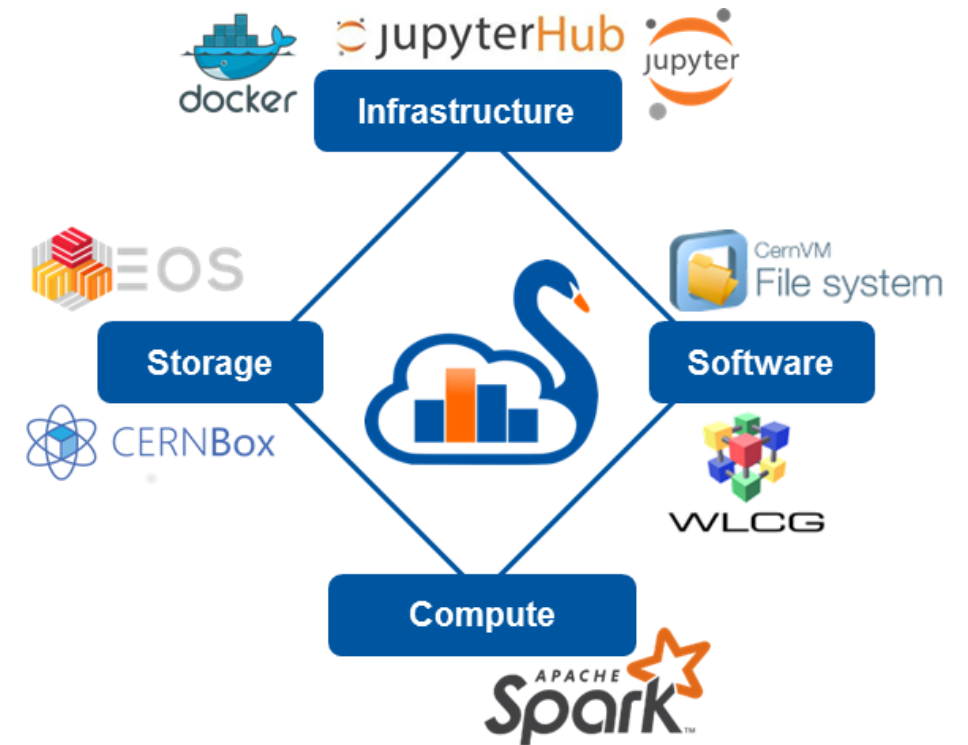
# SWAN

The Jupyter Notebook Service at CERN



# SWAN in a Nutshell

- > CERN's Jupyter Notebook service
  - Based on upstream Jupyter Notebooks // JupyterHub
- > Analysis only with a web browser
  - No local installation needed
  - Calculations, input data, and results “in the Cloud”
- > Support for multiple analysis ecosystems and languages
  - Python, ROOT C++, R and Octave
- > Easy sharing of scientific results: plots, data, code
- > Integration with external resources
  - Storage, Software, mass processing power





# SWAN in a Nutshell





# SWAN: User Interface



SWAN © Copyright CERN 2016-2019. All rights reserved.  
Home | Community | Support | Report a bug

## 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...  
96

**Platform** more...  
CentOS 7 (gcc8)

**Environment script** more...  
e.g. \$CERNBOX\_HOME/MySWAN/myscript.sh


**Number of cores** more...  
2

**Memory** more...  
8 GB

**Spark cluster** more...  
BE NXCALS (NXCals)

Always start with this configuration

[Start my Session](#)



[Projects](#) [Share](#) [CERNBox](#) >\_ ... ↗

SWAN > My Projects

## My Projects +

<input type="checkbox"/> NAME <span>▲</span>	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





# SWAN: User Interface

Text

Code

Graphics

FILE EDIT VIEW INSERT CELL KERNEL NAVIGATE WIDGETS HELP Not Trusted Python 2

## 2 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()
```

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

```
In [6]: h.SetFillColor(ROOT.kBlue-10)  
c.SetGrid()  
h.Draw()  
c.Draw()
```

myHisto	
Entries	1000
Mean	0.02680
Std Dev	1.038



# SWAN: User Interface

FILE EDIT VIEW INSERT CELL KERNEL HELP Trusted Python 2

Do the heavylifting in spark and collect aggregated view to panda DF

```
In [11]: df_loadAvg_pandas = spark.sql("SELECT submitter_host, \
    avg(body.LoadAvg) as avg, \
    hour(from_unixtime(timestamp / 1000, 'yyyy-MM-dd HH:mm:ss')) as hr \
    FROM loadAvg \
    WHERE submitter_hostgroup = 'hadoop/itdb/datanode' \
    AND dayofmonth(from_unixtime(timestamp / 1000, 'yyyy-MM-dd HH:mm:ss')) = 15 \
    GROUP BY hour(from_unixtime(timestamp / 1000, 'yyyy-MM-dd HH:mm:ss'), submitter_host")\
    .toPandas()
```

Apache Spark: 90 EXECUTORS 180 CORES Jobs: 1 COMPLETED

Job ID	Job Name	Status	Stages	Tasks	Submission Time	Duration
3	toPandas	COMPLETED	2/2	388 / 388	4 minutes ago	36s

Visualize with seaborn

```
In [19]: # heatmap of service availability
plt.figure(figsize=(10, 6))
ax = sns.heatmap(df_loadAvg_pandas.pivot(index='submitter_host', columns='hr', values='avg'), cmap="Blues")
ax.set_title("Heatmap of loadAvg")
```

Out[19]: Text(0.5,1,u'Heatmap of loadAvg')

Text

Code

Monitoring

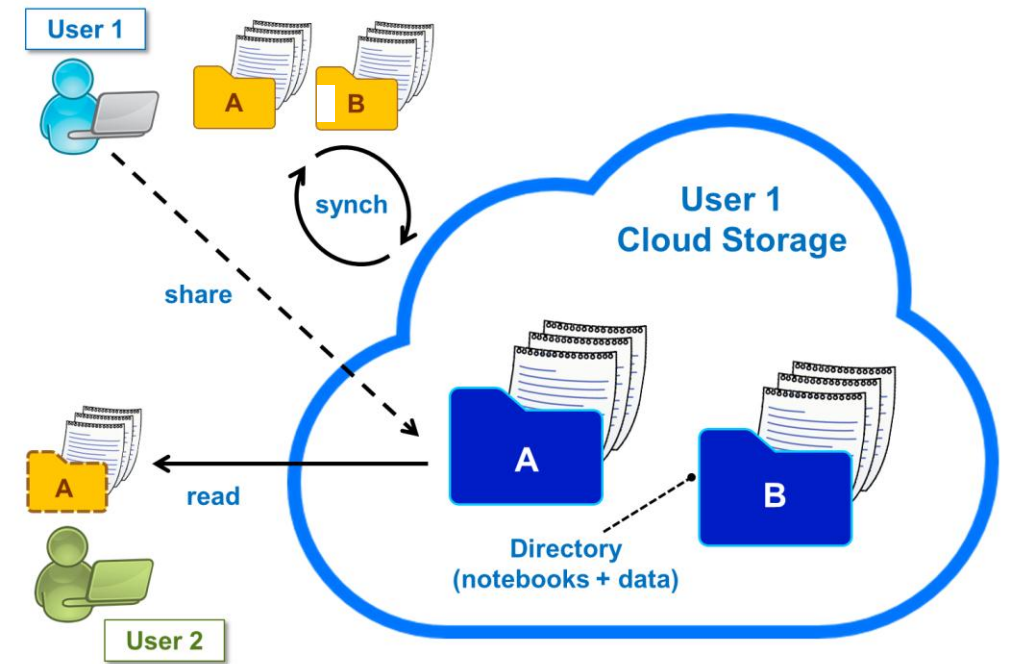
Visualizations





# Storage: The Cloud as your Home

- > CERNBox is SWAN's home directory
  - Based on EOS disk storage system
- > Sync & Share
  - Files synced across devices and the Cloud
  - Collaborative analysis
- > Sharing integration within SWAN UI
  - Users can share “Projects” (special kind of folder containing notebooks and other files)
  - Self contained

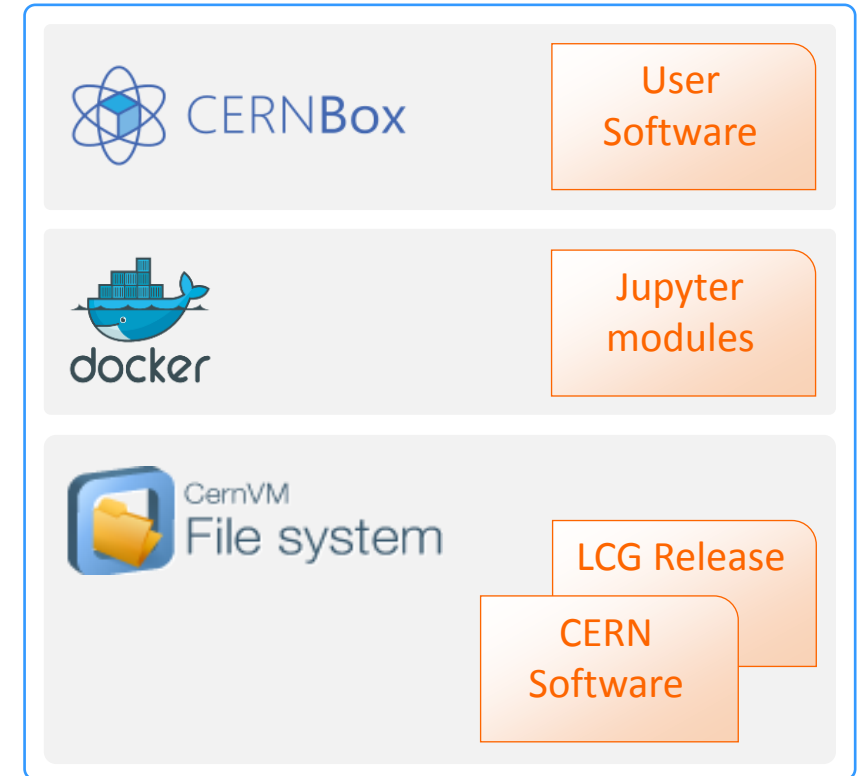






# Software: Leverage on WLCG repositories

- > Software distributed through CVMFS
  - Distributed read-only filesystem
  - "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





# Mass Processing: Integration with Spark

## > Connection to Apache Spark Clusters

- Spark: general purpose distributed computing framework

## > Same environment across platforms (local/remote)

- Software - CVMFS

## > Graphical Jupyter extensions developed

- Spark Connector
- Spark Monitor

The screenshot shows a Jupyter notebook titled "Spark > Spark\_Simple (autosaved)". The interface includes a menu bar (FILE, EDIT, VIEW, INSERT, CELL, KERNEL, HELP) and a toolbar with various icons. The main content area displays a "Simple example with Spark" section with introductory text. On the right side, a "Spark clusters connection" panel is open, showing configuration options for connecting to a "hadalytic" cluster. The panel includes a text input field for "Add a new option", a checkbox for "Bundled configurations" (Include NXCALS options), and a "Selected configuration" section with three settings: "spark.shuffle.service.enabled" (false), "spark.driver.memory" (2g), and "spark.executor.instances" (4). A green "Connect" button is at the bottom of the panel.

The screenshot shows a Jupyter notebook cell with the following text:

Do the heavylifting in spark and collect aggregated view to panda DF

```
In [11]: df_loadAvg_pandas = spark.sql("SELECT submitter_host, \
    avg(body.LoadAvg) as avg, \
    hour(from_unixtime(timestamp / 1000, 'yyyy-MM-dd HH:mm:ss')) as hr \
    FROM loadAvg \
    WHERE submitter_hostgroup = 'hadoop/itdb/datanode' \
    AND dayofmonth(from_unixtime(timestamp / 1000, 'yyyy-MM-dd HH:mm:ss')) = 15 \
    GROUP BY hour(from_unixtime(timestamp / 1000, 'yyyy-MM-dd HH:mm:ss'), submitter_host")\
    .toPandas()")
```

Below the code cell, a status bar shows "Apache Spark: 90 EXECUTORS 180 CORES Jobs: 1 COMPLETED". A table below the status bar provides details for the job:

Job ID	Job Name	Status	Stages	Tasks	Submission Time	Duration
3	toPandas	COMPLETED	2/2	388 / 388	4 minutes ago	36s





# SWAN Galleries

SWAN > Gallery > Basic Examples

**Gallery**

- > Basic Examples
- > ROOT Primer
- > Accelerator Complex
- > FCC
- > LHC Signal Monitoring
- > Beam Dynamics
- > Machine Learning
- > Apache Spark
- > Outreach
- > Awake

**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](http://root.cern.ch).

Simple ROOTbook (Python)

Simple ROOTbook (C++)

Simple Fitting

Simple I/O

C++ from Python w/o bindings

3D Visualisation

**70+ galleries** created with users  
**10 categories**

<https://swan.web.cern.ch/content/basic-examples>





# SWAN Galleries

SWAN > Gallery > Basic Examples

## Gallery

- > Basic Examples
- > ROOT Primer
- > Accelerator Complex
- > FCC
- > LHC Signal Monitoring
- > Beam Dynamics
- > Machine Learning
- > Apache Spark
- > Outreach
- > Awake

```
In [6]: from PyHEADTAIL.particles.rfbucket_matching import ThermalDistribution
plot(ThermalDistribution);
```

The figure displays four plots related to the thermal distribution of particles in a bucket:

- phase space distribution:** A 2D heatmap showing the distribution of particles in the  $(z, \delta)$  plane. The horizontal axis is  $z$  (ranging from -30 to 30) and the vertical axis is  $\delta$  (ranging from -0.002 to 0.002). The distribution is centered at  $(0, 0)$  and is bounded by a purple separatrix. A color bar on the right indicates the density  $\psi(z, \delta)$  from 0.00 to 0.90.
- Hamiltonian contours:** A 2D heatmap showing the Hamiltonian  $\mathcal{H}(z, \delta)$  contours. The axes are the same as the phase space plot. The contours are centered at  $(0, 0)$  and are bounded by a purple separatrix. A color bar on the right indicates the Hamiltonian value from -135 to 135.
- line density:** A 1D plot showing the line density  $\lambda(z) = \int d\delta \psi(z, \delta)$  as a function of  $z$ . The horizontal axis is  $z$  (ranging from -40 to 40) and the vertical axis is  $\lambda(z)$  (ranging from 0 to 250). The distribution is a smooth, bell-shaped curve centered at  $z = 0$ .
- Hamiltonian distribution:** A 1D plot showing the Hamiltonian distribution  $\psi(\mathcal{H})$  as a function of  $\mathcal{H}$ . The horizontal axis is  $\mathcal{H}$  (ranging from -200 to 150) and the vertical axis is  $\psi(\mathcal{H})$  (ranging from 0.0 to 1.0). The distribution is a smooth curve that is zero for  $\mathcal{H} < 0$  and increases for  $\mathcal{H} > 0$ . A vertical purple line at  $\mathcal{H} = 0$  is labeled "separatrix".

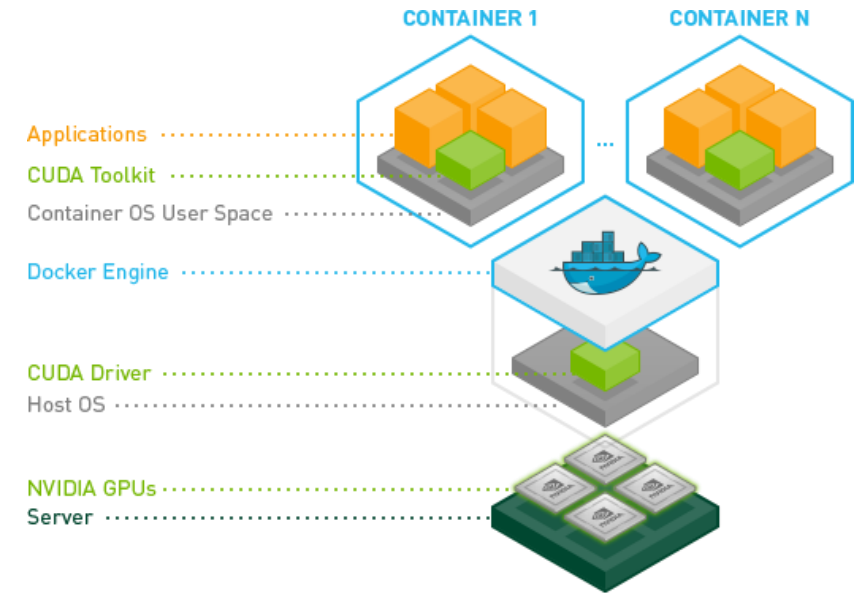
**70+ galleries** created with users  
**10 categories**

<https://swan.web.cern.ch/content/basic-examples>



# Upcoming: NVidia GPU Support

- > Exploitation of container technologies to provide support for NVidia GPUs
- > Prototype server for testing purposes
  - NVidia Tesla V100 PCIe 32GB
- > All the packages are provided by CVMFS
  - Including CUDA enabled machine learning software stack
  - TensorBoard for interactive monitoring





# Upcoming: Configurable SW Environment

- > Adding support for Conda environments
  - Linked to Projects
  - Sharable
- > Easy installation of extra packages
  - Clone/import Projects and install the software automatically
- > Custom images with SWAN
  - For higher customization of software environment
  - BYO Docker image
- > Flexibility of Binder +  
Integrated Environment of SWAN

The screenshot shows a 'Configure Project' dialog box with the following elements:

- Google Summer of Code** logo at the top.
- CERN** and **HSF** logos below the title.
- Install Packages** section with a search bar containing 'Search for available packages'.
- Packages To Install** section with a table:

✓ numpy	1.17.0
---------	--------
- Installed Packages** section with a table:

🗑️ _libgcc_mutex	0.1
🗑️ backcall	0.1.0
🗑️ bzip2	1.0.8
🗑️ ca-certificates	2019.6.16
🗑️ certifi	2019.6.16
🗑️ decorator	4.4.0
🗑️ ipykernel	5.1.2
🗑️ ipython	7.7.0
🗑️ ipython_genutils	0.2.0
🗑️ jedi	0.15.1
🗑️ jupyter_client	5.3.1
🗑️ jupyter_core	4.4.0
🗑️ libffi	3.2.1





# Upcoming: Jupyterlab

## > Next-generation interface for Project Jupyter

- “IDE-like” environment
- Concurrent editing

## > Next steps: integration of current extensions

- SWAN Projects
- CERNBox sharing integration
- Spark Connector and Monitor
- ...

The screenshot displays the JupyterLab environment. On the left, a sidebar shows a file browser with a list of notebooks: Data.ipynb (an hour ago), Fasta.ipynb (a day ago), Julia.ipynb (a day ago), Lorenz.ipynb (seconds ago), R.ipynb (a day ago), iris.csv (a day ago), lightning.json (9 days ago), and lorenz.py (3 minutes ago). The main workspace is divided into several panes. The top pane shows the code editor for 'Lorenz.ipynb', containing text and a code cell. The code cell contains the following code:

```
In [4]: from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```

The bottom-left pane shows the 'Output View' with three sliders for parameters: sigma (10.00), beta (2.67), and rho (28.00). Below the sliders is a 3D plot of the Lorenz attractor, showing a complex, swirling trajectory in a 3D space. The bottom-right pane shows the 'lorenz.py' code file with the following code:

```
9 def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
10     """Plot a solution to the Lorenz differential equations."""
11     fig = plt.figure()
12     ax = fig.add_axes([0, 0, 1, 1], projection='3d')
13     ax.axis('off')
14
15     # prepare the axes limits
16     ax.set_xlim((-25, 25))
17     ax.set_ylim((-35, 35))
18     ax.set_zlim((5, 55))
19
20     def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
21         """Compute the time-derivative of a Lorenz system."""
22         x, y, z = x_y_z
23         return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]
24
25     # Choose random starting points, uniformly distributed from -15 to 15
26     np.random.seed(1)
27     x0 = -15 + 30 * np.random.random((N, 3))
28
```



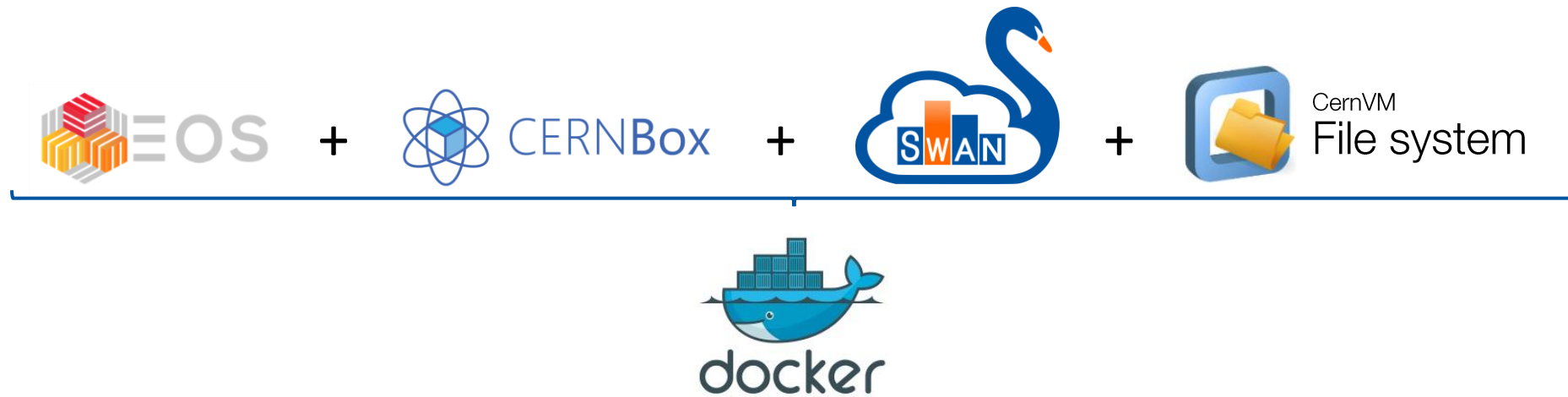
# ScienceBox

SWAN, CERNBox, EOS in Docker containers



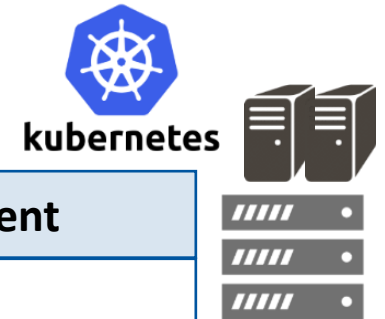
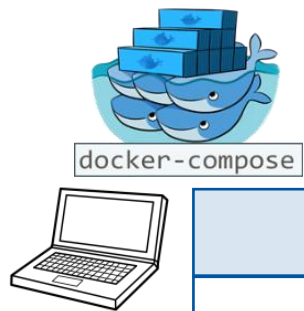


- > Self-contained Docker-based software package





## > Self-contained Docker-based software package



### One-Click Demo Deployment

- Single-box installation
- Download and run in 5 minutes

<https://github.com/cernbox/uboxed>

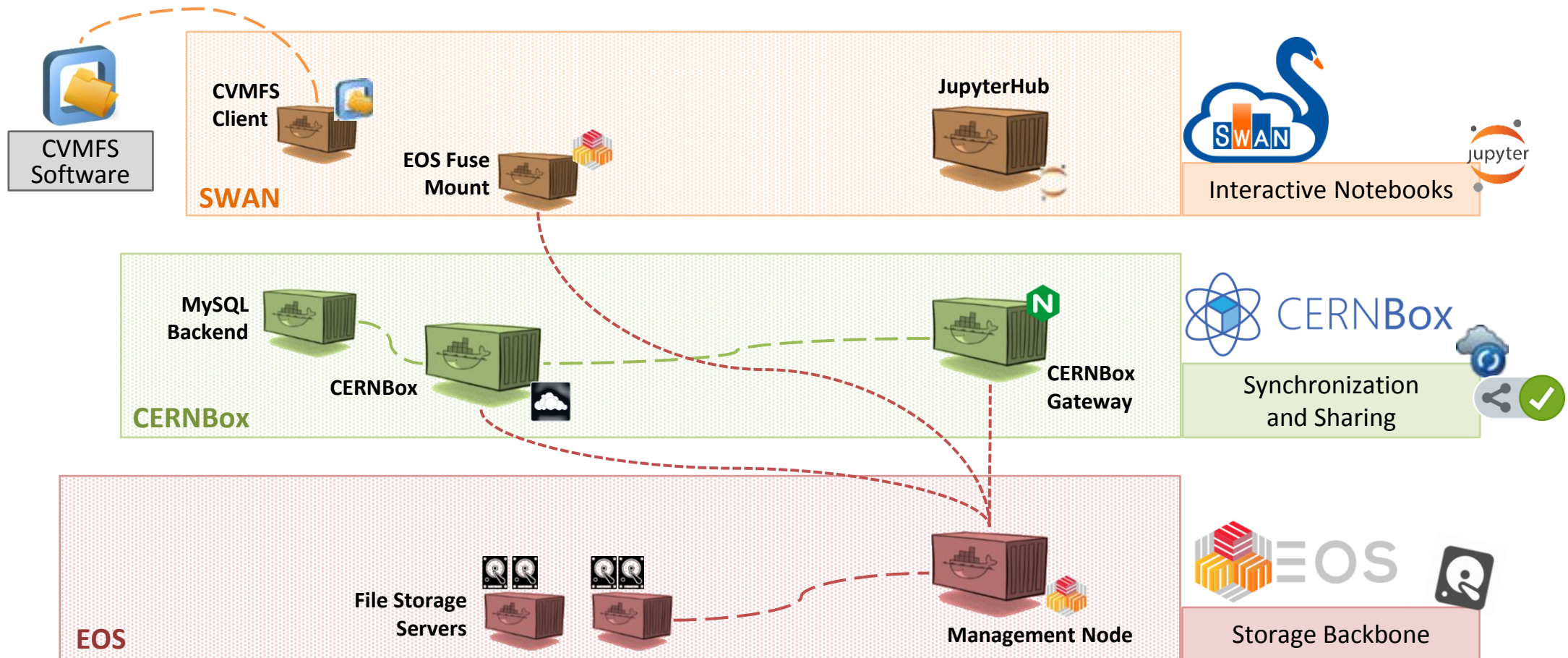
### Production-ready Deployment

- Scale out service capacity
- Tolerant to node failures

<https://github.com/cernbox/kuboxed>



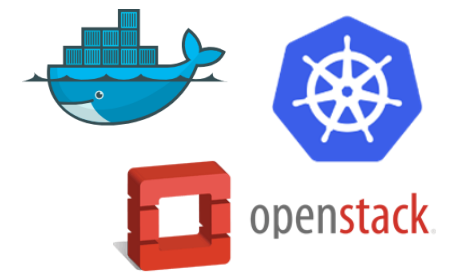
# ScienceBox Architecture





# Use Cases: Up to University

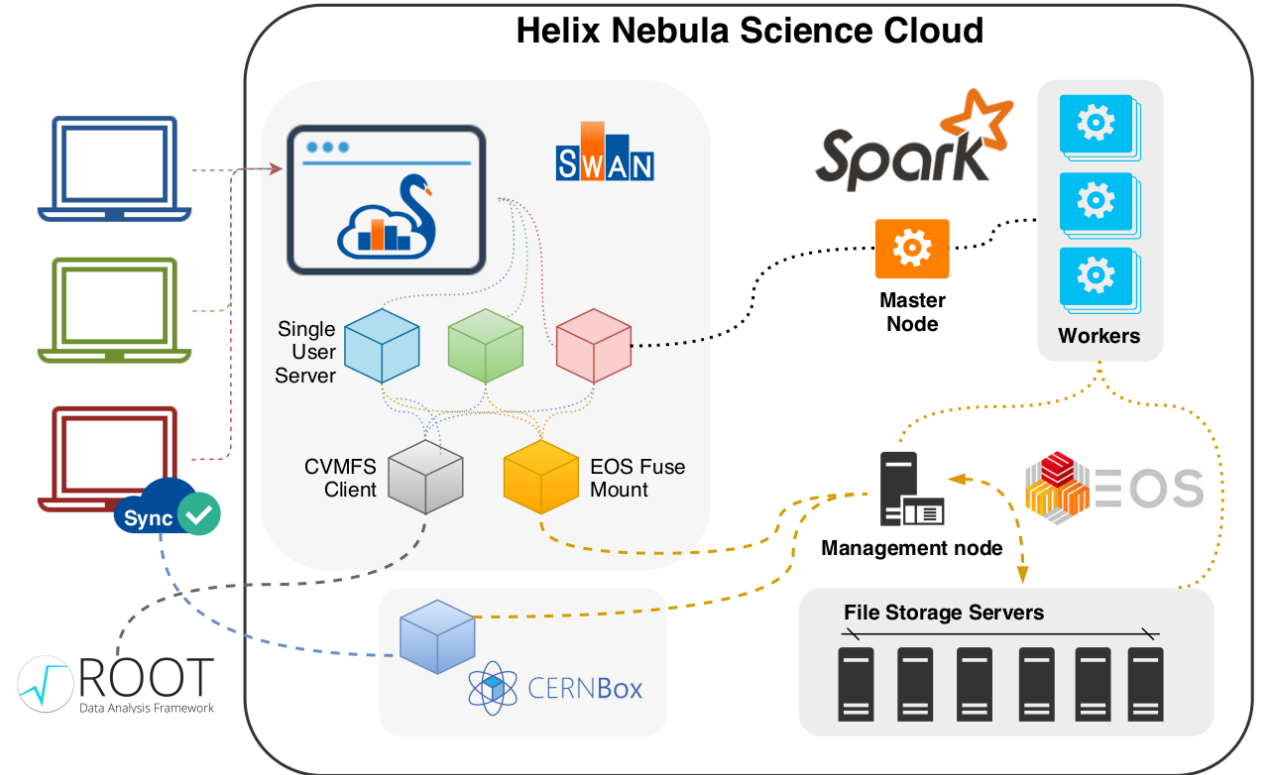
- > Allow students in high-schools to adopt tools used in science
  - SWAN – Full data analysis ecosystem in a web browser
  - CERNBox – Cloud storage for easy sharing and access form any device
  
- > ScienceBox in production for Up2U users for 1.5 years
  - Deployed at Poznan Supercomputing and Networking Center, Poland
  - Kubernetes on VMs, Ceph volumes for persistent storage
  
- > Pilot service at CERN – <http://up2u.cern.ch>
  - CERNBox and SWAN on Kubernetes VMs
  - EOS on VMs and bare metal disks





# Use Cases: Totem Analysis on Helix Nebula Cloud

- > Deployment on commercial cloud
  - 2000+ CPUs
  - 10+ TB memory
  - Virtually unlimited block storage
- > ScienceBox with Apache Spark for massive computations
- > Full TOTEM Analysis
  - Dataset: 4.7 TB, 1153 files
  - Data imported via xrootd
  - Results synchronized back via CERNBox client



*Big Data Tools and Cloud Services for High Energy Physics Analysis in TOTEM Experiment - V. Avati et al.*

<https://ieeexplore.ieee.org/document/8605741>





# More External Sites and Collaborators

## > External SWAN deployments inspired by ScienceBox

- Australia's Academic and Research Network (AARNET)
- SURFSARA, The Netherlands
- Joint Institute for Nuclear Research (JINR), Russia
- Academia Sinica Grid Computing Centre (ASGC), Taiwan



# Opportunities for Collaboration



- > EU-funded project (coordinated by CERN)
  - 6M EUR, 12 partners, 2020-2022
- > Goal: Global collaborative environment for research
  - Share documents, files, projects, data, ...
  - Connected Application Hubs
  - Data Science Environments → **SWAN**
- > Federation of existing CS3 sites
  - 30+ sites (e.g. CERNBox, DesyBox, Universities, ...)
  - 300K+ users
  - [cs3community.org](https://cs3community.org)







# CS3MESH Objectives

## > Collaborative Workflows



Share, access, synchronize



Metadata & Tagging, Open Data  
(OpenAIRE, Zenodo, ...)



Data Science: Jupyter Notebooks  
(SWAN, ...)



Concurrent editing



On-demand data transfers (Direct,  
FTS, DTN, Rucio, ...)

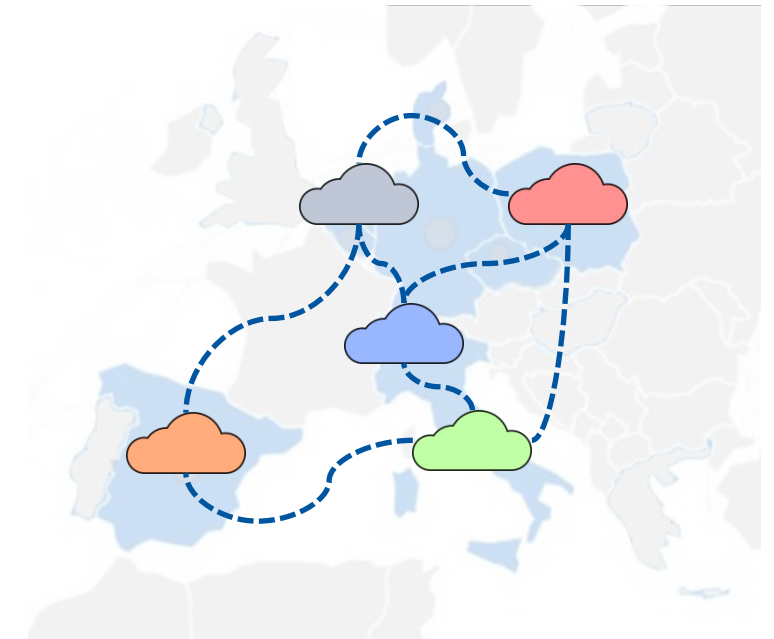
## > Interoperability

- Add thin layer on top of existing services
- Use existing fabric
- Use existing standards
  - Introduce new APIs only if needed
- Close collaboration with industry
- Integrate into upstream products



# ScienceBox for CS3MESH

- > ScienceBox is the reference platform for CS3MESH for distribution and deployment of cloud software
  - SWAN will become part of the core service for future European Open Science Cloud
- > Developer community & upstream
  - Working together on SWAN
- > Benefits for SWAN users at a scale
  - Share SWAN projects beyond the CERN borders
  - Work easily with your experiment collaborators inside/outside CERN





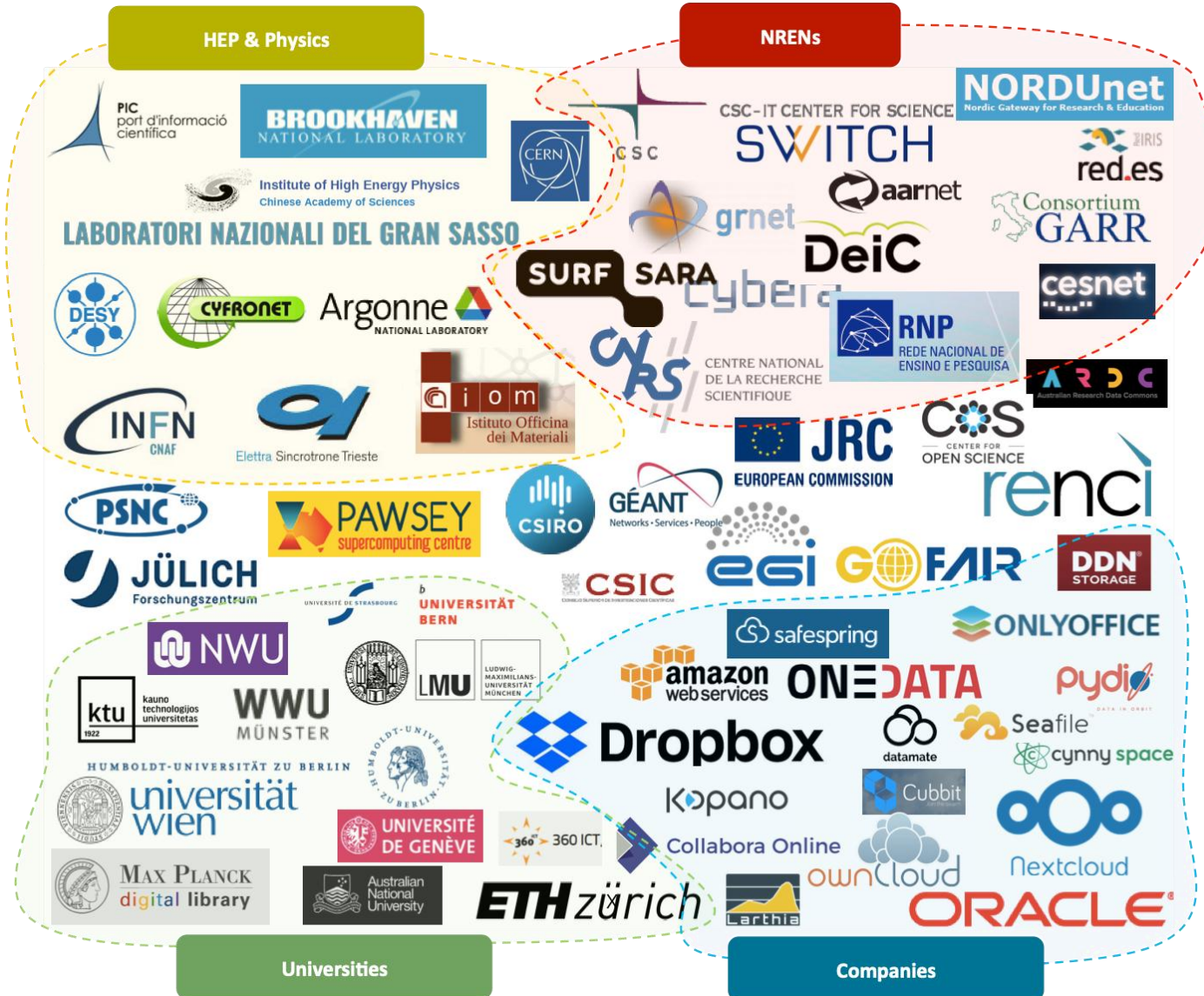
# CS3 Workshop

<https://cs3.deic.dk/>





# CS3 Community



# Where to find us



# Where to find us

## > Swan and Galleries

- <https://swan.web.cern.ch/>
- <https://swan.web.cern.ch/content/basic-examples>

## > Science Box

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

## > CERNBox and EOS

- <https://cernbox.web.cern.ch/>
- <https://eos.web.cern.ch/>

# Demo

## Analysis of Cinemas in Geneva