SWAN: interactive data analysis on the web

Diogo Castro
On behalf of the SWAN team

https://cern.ch/swan

Oct 17th, 2019
PyHEP 2019
Introduction
SWAN is the Jupyter Notebook service at CERN
SWAN community

> Built for and with the community
  - New features (libs, kernels, ...) are requested by users from their real usage needs

> ~200 users’ sessions per day on average
  - Peaks of 300+ users per day with trainings / courses

> 2,000+ unique users over the last 6 months
Users by Department

Users by Experiment
SWAN Users’ Workshop

- SWAN usage at COMPASS
- SWAN for HGCAL Beam Tests Analysis
- Integrating CMSSW in SWAN
- ALICE Data Analysis without wired PC feat. SWAN
- SWAN as a tool in Atlas TDAQ operations
- My SWAN experience as an ALICE analyzer
- AWAKE Data Analysis on SWAN
- Experiments
SWAN Users’ Workshop

- SWAN for NXCALS (CERN Accelerator Logging Service)
- SWAN for machine studies
- TE/ABT experience with SWAN
- Application of SWAN to the LHC Signal Monitoring Project
Usage of SWAN with the CERN Open Data portal for education and outreach

SWAN for operational radiation protection and possible improvements

Automatic generation of superconducting magnet input files for the LEDET software

Using SWAN to Analyse Tape Server Logs

Rucio & SWAN Integration idea

Statistical Methods for the LHC

Towards EduSWAN
> SWAN is useful for many different use cases
  - SWAN is successfully used for data analysis in the represented experiments
    - But also for other daily tasks (important ones!)
  - SWAN is also used for use cases outside of physics
    - Analysis of the LHC logging, tapes, etc.
  - SWAN is a powerful ally in education and outreach

> SWAN provides many functionalities to our community
  - Easy to plug some experiments software stack
  - Still is easy to use

> The community is demanding
  - But it gives us suggestions and help us develop the service
SWAN: service for web-based analysis
SWAN in a Nutshell

> CERN’s Jupyter Notebook service

> Analysis only with a web browser
  - No local installation and configuration needed
  - Calculations, input data and results “in the Cloud”
  - Support for: Python (2 and 3), ROOT C++, R and Octave

> What makes it different? The integration with CERN resources
  - Software, storage, mass processing power
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...
- **Platform**: more...
- **Environment script**: more...
  - Example: `SCERNBOX_HOME/My/SWAN/my_script.sh`
- **Number of cores**: more...
  - 2
- **Memory**: more...
  - 8 GB
- **Spark cluster**: more...
  - Hadoop

☐ Always start with this configuration

Start my Session
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 *JROOT*, 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()
```

![Histogram](image)

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.SetFillColot(ROOT.kBlue-16)
   : c.SetGrid()
   : h.Draw()
   : c.Draw()
```

![Histogram](image)
Integrating services

Compute  Software  Storage
Cloud storage as your Home

> CERNBox is SWAN's home directory
  - Storage for your notebooks and data
  - 16k users and 6PB of user data

> Uses EOS disk storage system
  - All experiment data potentially available
  - 250PB of experimental data at CERN (LHC and others)

> Sync&Share
  - Files synced across devices and the Cloud
  - Collaborative analysis
Sharing made easy

> Sharing from inside SWAN interface
  - Integration with CERNBox
  - List shares from other users

> Users can share “Projects”
  - Special kind of folder that contains notebooks and other files, like input data
  - Self contained
  - Fosters collaboration

> Concurrent editing not supported yet by Jupyter
  - Safer to clone
  - Will be available with Jupyterlab
> Software distributed through CVMFS
- Distributed read-only filesystem
- "LCG Releases" - pack a series of compatible packages
- Reduced Docker Images size
- Lazy fetching of software
- Step towards reproducibility (across time and people)

> Possibility to install libraries in user cloud storage
- Good way to use custom/not mainstream packages
- Configurable environment
Access to Computing Resources
Integration with Spark - Production

> Connection to CERN Spark Clusters
  - Spark: general purpose distributed computing framework

> Same environment across platforms (local/remote)
  - Software - CVMFS

> Graphical Jupyter extensions developed
  - Spark Connector
  - Spark Monitor
> Exploitation of container technologies to provide support for NVidia GPUs
  - Already integrated with ScienceBox (more details soon)

> Prototype server for testing purposes
  - NVidia Tesla V100 PCIe 32GB
  - If interested, ask us to join the beta program

> All the packages are provided by CVMFS
  - Including CUDA enabled machine learning software stack
  - TensorBoard for interactive monitoring
Ongoing effort: Batch jobs submission

- Using Ganga
- Monitoring display
- Jobs tab

Possibility to connect to user managed Kubernetes clusters

- Offload Spark computations
- Control and use your own resources
- Quickly create, use and dispose
- Share access with other users
Collaboration and configuration
Next-generation interface for Project Jupyter
- Concurrent editing
- "IDE in the browser"
Configurable software environment - Prototype

> Adding support for Conda environments
  - Linked to Projects
  - Sharable

> Easy installation of extra packages
  - Clone/import Projects and install the software automatically

> Still a proof of concept
  - Integration with EOS is starting
SWAN Users’ Workshop
> First get together with the users of the service
  - Share use cases and knowledge among the community
  - Allow SWAN admins to understand how the service is used
  - Collect wishes for future improvements

> [https://indico.cern.ch/event/834069](https://indico.cern.ch/event/834069)
Use cases for SWAN

Two types of debugging:

1. live! There’s an issue NOW. Probably just need to load in a small amount of recent “bad” data, maybe some previous “good” data and investigate as many relevant variables as possible
   
   Here, the interactive visualization is the best part – can see ALL the possibilities in one place, with the exact code that made them – discuss with others, etc.

2. post-run: Something looked a little off last run. Did it look off the run before? When did it start? What else changed? Could need to load a large amount of data, overlay many runs…
   
   Only need to load data once – and can then play around with many plots, without having to wait for the data to load again

And then another use, for post-mortem of new changes: want to document effects of updates on the system and verify that changes had the expected outcome

   e.g. new trigger had the expected rate and the CPU changed accordingly
SWAN usage at COMPASS

Study of the DVCS process at COMPASS

For all the events candidates, extract the DVCS exact topology

\[ \Delta \phi = \phi_{\text{cam}} - \phi_{\text{spec}} \]

\[ \Delta p_T = p_{T,\text{cam}} - p_{T,\text{spec}} \]

+ plot many other physics observables

\[ \Delta Z_A = Z_{A,\text{cam}} - Z_{A,\text{spec}} \text{ and vertex} \]

\[ M^2_{\text{Undet}} = (k+p-k'-q'\cdot p)^2 \]

**Pros & cons of SWAN**

- Well organised analysis in notebooks using sections (use of markdown, latex, etc)
- Flexible enough to make systematic studies: use SWAN to adjust some cuts
- Drawback of interactivity: can be slow with too many events

Spark helps!
NXCALS Data After Few Clicks!

- After requesting authorisation to NXCALS service it is sufficient to:
  - Provide CERN credentials
  - Select Environment (NXCALS Python3 software stack and BE NXCALS Spark cluster)
  - Establish Spark clusters connection (with bundled NXCALS configuration)
  - Import NXCALS builders and execute a code as in the example below:

```python
from cern.nxcal.pyquery.builders import *

df1 = DevicePropertyQuery.builder(spark)
    .system('CMN')
    .startTime('2017-08-29 00:00:00.000')
    .duration(10000000000)
    .entity().parameter('RADMON.PS-10/ExpertMonitoringAcquisition')
    .buildDataset()
```

11-10-2019

BE-CO-DS 9
How SWAN works for me

- Usage with cernbox and EOS is great
  » Develop locally, run your code both on-line and offline
  » Output is always at hand

My SWAN experience as an ALICE analyzer – Nicolò Jacazio
Possible solution: combined use of SWAN and GitLab

- **Step 1:**
  - Creating a SWAN project

- **Step 2:**
  - Making the created SWAN project locally available (CERNBox client)

- **Step 3:**
  - Creating a Gitlab project
  - Setting the folder as local repository for the GitLab project

- **Step 4:**
  - Sharing the GitLab repository

**Possible improvement:**
- Integration of Git inside the SWAN service
Our Experience with SWAN

- Seamless report generation
- No local dependencies
- No installation required
- Integrated access to NXCALS
- Easy to start and collaborate

- Connection/kernel issues
- Limited execution time
- Conversion to a script
- IDE capabilities
- Versioning with GitLab
Experiment namespace integration

Rucio & SWAN Integration idea – Mario Lassnig
Where to find us
Where to find us

> Contacts
  - swan-admins@cern.ch
  - http://cern.ch/swan
  - https://cern.ch/swan-community

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

> ScienceBox
  - https://cern.ch/sciencebox
ScienceBox: SWAN on Premises

> Packaged deployment of SWAN

- Includes all SWAN components: CERNBox/EOS, CVMFS, JupyterHub
- Deployable through Kubernetes or docker-compose

One-Click Demo Deployment
- Single-box installation
- Download and run in 5 minutes
  [https://github.com/cernbox/uboxed](https://github.com/cernbox/uboxed)

Production-ready Deployment
- Scale out service capacity
- Tolerant to node failures
  [https://github.com/cernbox/kuboxed](https://github.com/cernbox/kuboxed)
Conclusion
Conclusion

> SWAN is a CERN service that provides Jupyter Notebooks on demand
  - Promotes a cloud-based analysis model

> Valued by the community
  - Used for many use cases: Data analysis (Physics or others), Exploration, Teaching, …

> SWAN became a fundamental Interface for Mass Processing Resources
  - Currently it gives access to Spark
  - In development the access to GPUs and others
SWAN: interactive data analysis on the web

Thank you

Diogo Castro
diogo.castro@cern.ch