Analysis Facility @ CERN

Enric Tejedor for the SWAN team

https://cern.ch/swan

Analysis Ecosystems Workshop II

May 23rd, 2022

SWAN: the interface of an AF @ CERN

- SWAN: Service for Web-based Analysis

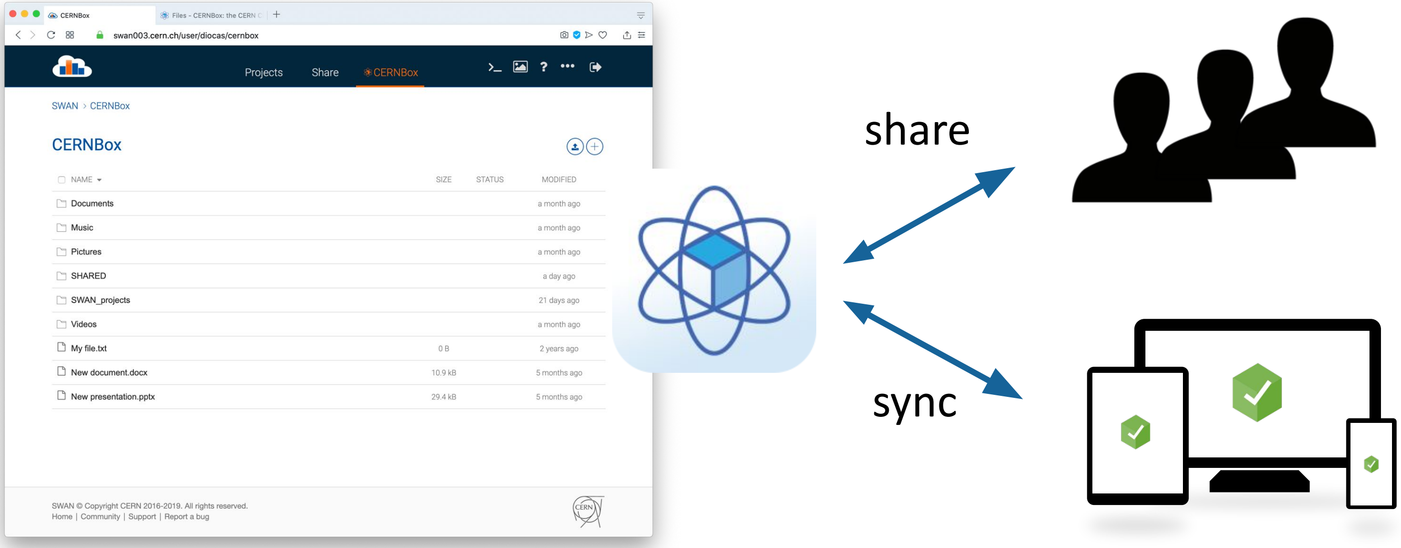
- CERN's Jupyter notebook service
  - Created in 2016
  - Used by 200-250 people daily

- Jupyter interface + federation of CERN services → added value!
  - Software (CVMFS)
  - Storage (EOS, CERNBox)
  - Computing resources (GPU, Spark, HTCondor)

- Platform for physics analysis: supports both *single-node* and *distributed* analysis

Storage: EOS, CERNBox

- Find the data you need for your analysis
  - EOS: experiment repositories (/eos/atlas, /eos/cms, …), projects, open data
  - CERNBox as home directory, sync & share



CERNBox | Files - CERNBox: the CERN C

swan003.cern.ch/user/diocas/cernbox

SWAN › CERNBox

CERNBox

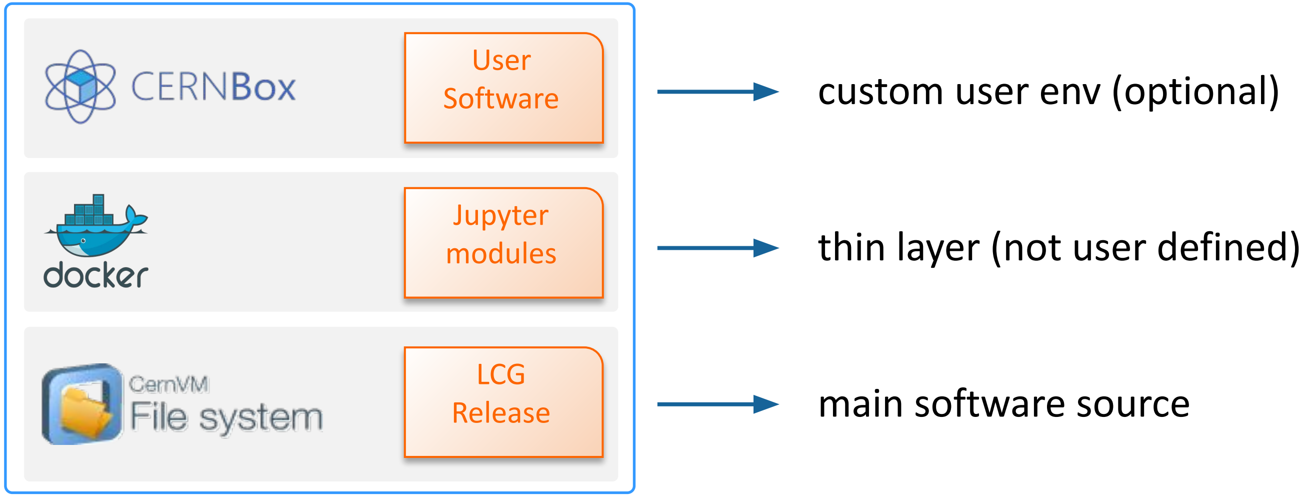
NAME	SIZE	STATUS	MODIFIED
Documents			a month ago
Music			a month ago
Pictures			a month ago
SHARED			a day ago
SWAN_projects			21 days ago
Videos			a month ago
My file.txt	0 B		2 years ago
New document.docx	10.9 kB		5 months ago
New presentation.pptx	29.4 kB		5 months ago

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

Software: CVMFS

> Find the software you need for your analysis

- CVMFS: LCG releases (and soon CMSSW, FCC)
- EOS: custom software environment



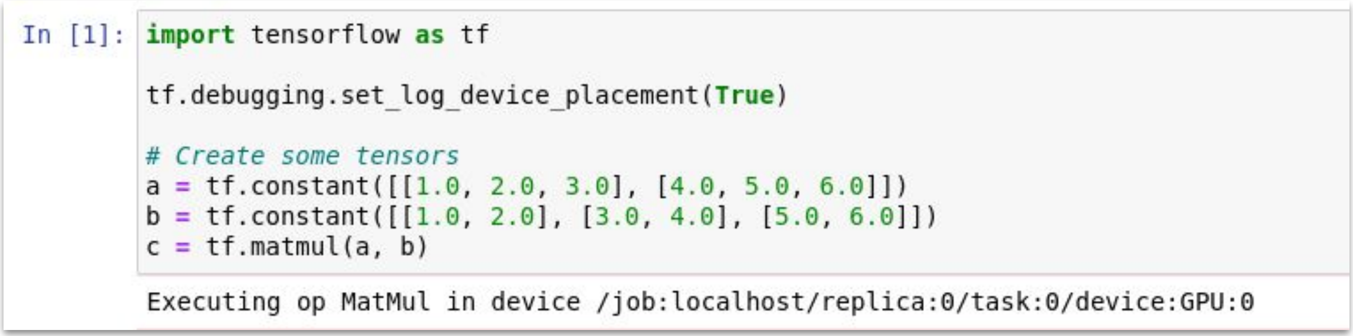
custom user env (optional)

thin layer (not user defined)

main software source

GPUs

- SWAN allows to attach a GPU to a user session
  - Feature of the new SWAN k8s deployment (https://swan-k8s.cern.ch)
  - ~10 GPUS (Tesla T4 and V100)
- The GPUs are used interactively
  - When starting their session, the user selects a CUDA software stack and gets a GPU
  - GPU-enabled packages (e.g. tensorflow, PyTorch) can then be used in a notebook and offload to the GPU by default

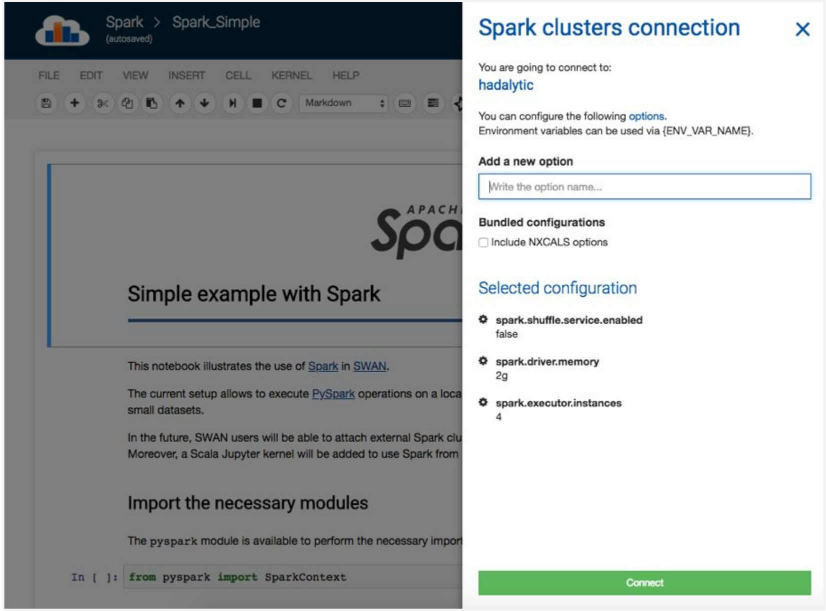


```
In [1]: import tensorflow as tf

        tf.debugging.set_log_device_placement(True)

        # Create some tensors
        a = tf.constant([[1.0, 2.0, 3.0], [4.0, 5.0, 6.0]])
        b = tf.constant([[1.0, 2.0], [3.0, 4.0], [5.0, 6.0]])
        c = tf.matmul(a, b)

        Executing op MatMul in device /job:localhost/replica:0/task:0/device:GPU:0
```

Spark

- SWAN is connected to the Spark clusters at CERN
  - Physical: ~3800 cores, some dedicated
  - Virtual: ~250 cores, on demand (kubernetes)

- Jupyter extensions available to:
  - Connect to a certain cluster
  - Monitor the execution



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

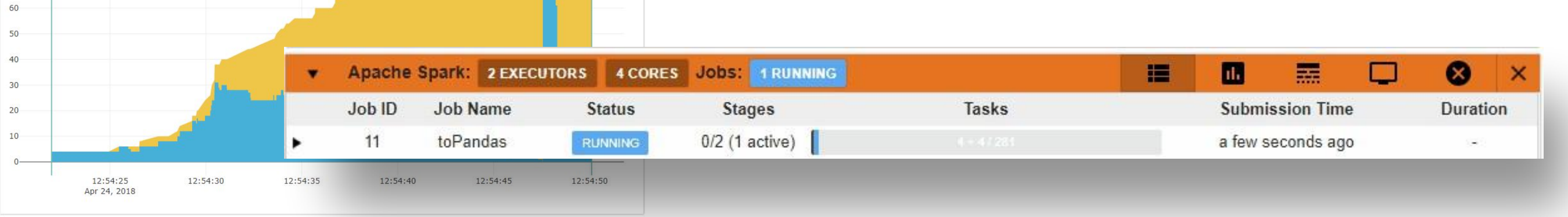
Selected configuration

spark.shuffle.service.enabled
false

spark.driver.memory
2g

spark.executor.instances
4

Connect

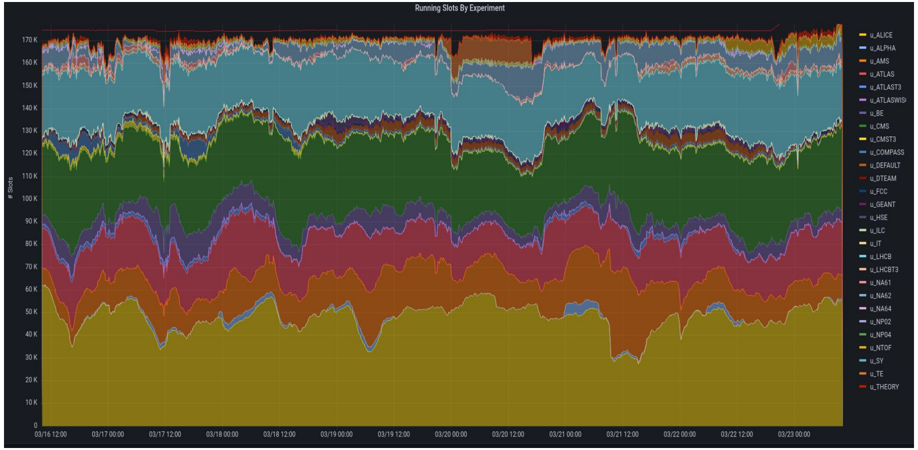


Apache Spark: 2 EXECUTORS 4 CORES Jobs: 1 RUNNING

Job ID	Job Name	Status	Stages	Tasks	Submission Time	Duration
11	toPandas	RUNNING	0/2 (1 active)		a few seconds ago	-

HTCondor

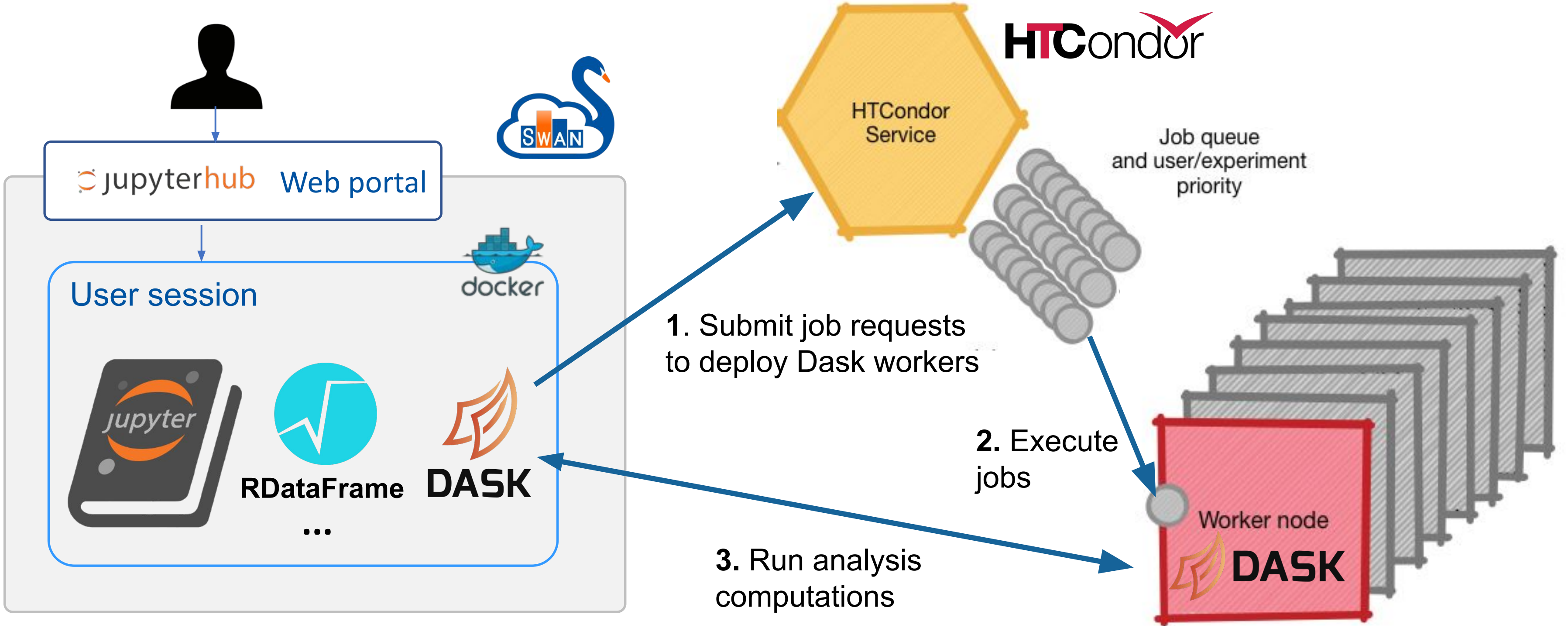
- Goal: leverage HTCondor pools at CERN from SWAN
  - Up to ~175k cores in shared pools at CERN – limited by the quotas assigned depending on experiment affiliation
  - Already used for analysis



- Batch submission: already supported
  - Condor packages available on CVMFS

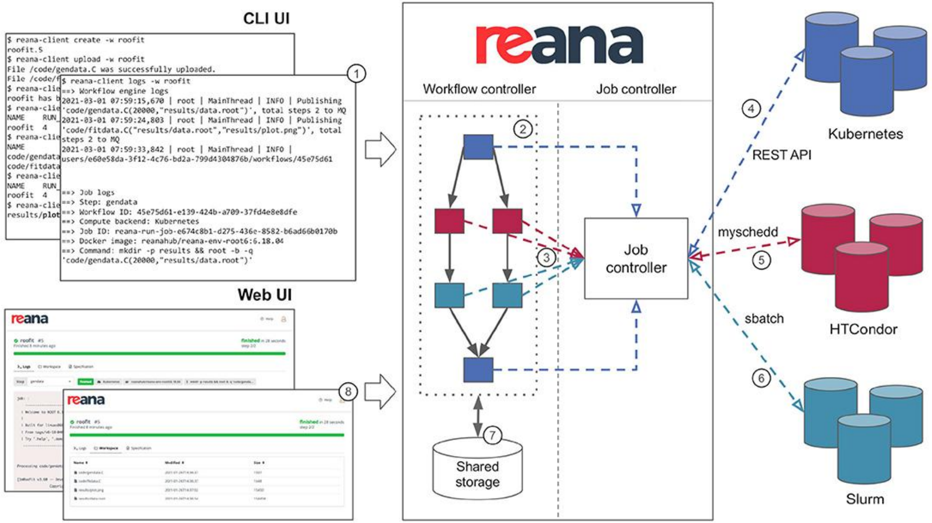
- Interactive usage: in pilot phase
  - Collaboration with Batch Service@CERN
  - Dask packages available on CVMFS
  - Will be exposed to users when migration to JupyterLab is finished (Q2-Q3 2022)

SWAN + HTCondor for interactive analysis



REANA Reusable Analysis Platform

- Born within analysis preservation and reuse context
- Run computational workflows on containerised clouds
  - multiple workflow systems (CWL, Serial, Snakemake, Yadage)
  - multiple container technologies (Docker, Singularity)
  - multiple compute backends (Kubernetes, HTCondor, Slurm)

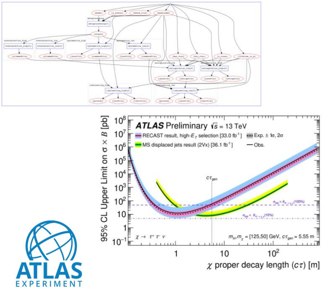


CLI UI

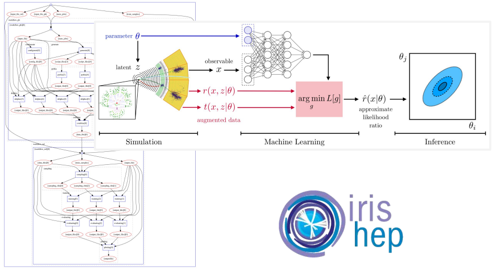
```
$ reana-client create -w roofit
roofit.5
$ reana-client upload -w roofit
File /code/gendata.C was successfully uploaded.
File /code/f
$ reana-clie
roofit has b
$ reana-clie
NAME     RUN_
roofit  4
$ reana-clie
NAME
code/gendata
code/fitdata
$ reana-clie
NAME     RUN_
roofit  4
$ reana-clie
results/plot
```

```
$ reana-client logs -w roofit
==> Workflow engine logs
2021-03-01 07:59:15,670 | root | MainThread | INFO | Publishing
'code/gendata.C(20000,"results/data.root")', total steps 2 to MQ
2021-03-01 07:59:24,803 | root | MainThread | INFO | Publishing
'code/fitdata.C("results/data.root","results/plot.png")', total
steps 2 to MQ
2021-03-01 07:59:33,842 | root | MainThread | INFO |
users/e60e58da-3f12-4c76-bd2a-799d4304876b/workflows/45e75d61



==> Job logs
==> Step: gendata
==> Workflow ID: 45e75d61-e139-424b-a709-37fd4e8e8dfe
==> Compute backend: Kubernetes
==> Job ID: reana-run-job-e674c8b1-d275-436e-8582-b6ad66b0170b
==> Docker image: reanahub/reana-env-root6:6.18.04
==> Command: mkdir -p results && root -b -q
'code/gendata.C(20000,"results/data.root")'
```

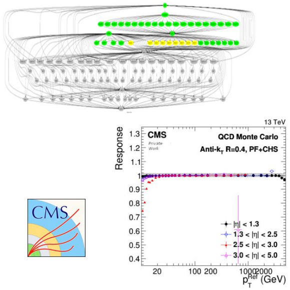
Web UI



Example: ATLAS search reinterpretations



Example: MadMiner ML workflows
reana @BNL @NYU @ND



Example: CMS jet energy resolution and corrections

More info here

Backup slides

Context and motivation

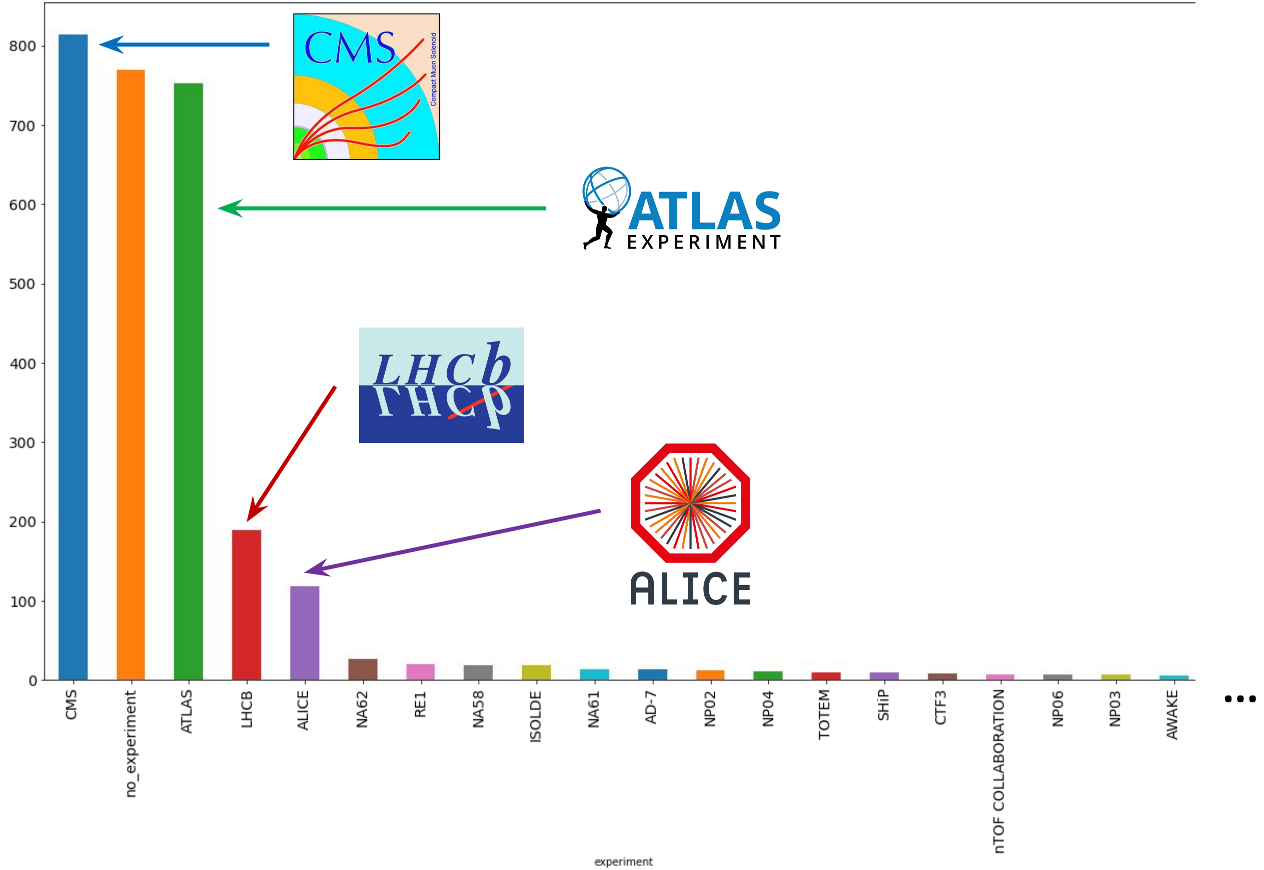
- HL-LHC needs are pushing us to build **modern Analysis Facilities**
  - Traditional batch processing
  - Interactive computing on big datasets, with new interfaces (Jupyter)

- An AF should facilitate access to:
  - Software
  - Storage (+ sharing)
  - Computing resources (elastic)

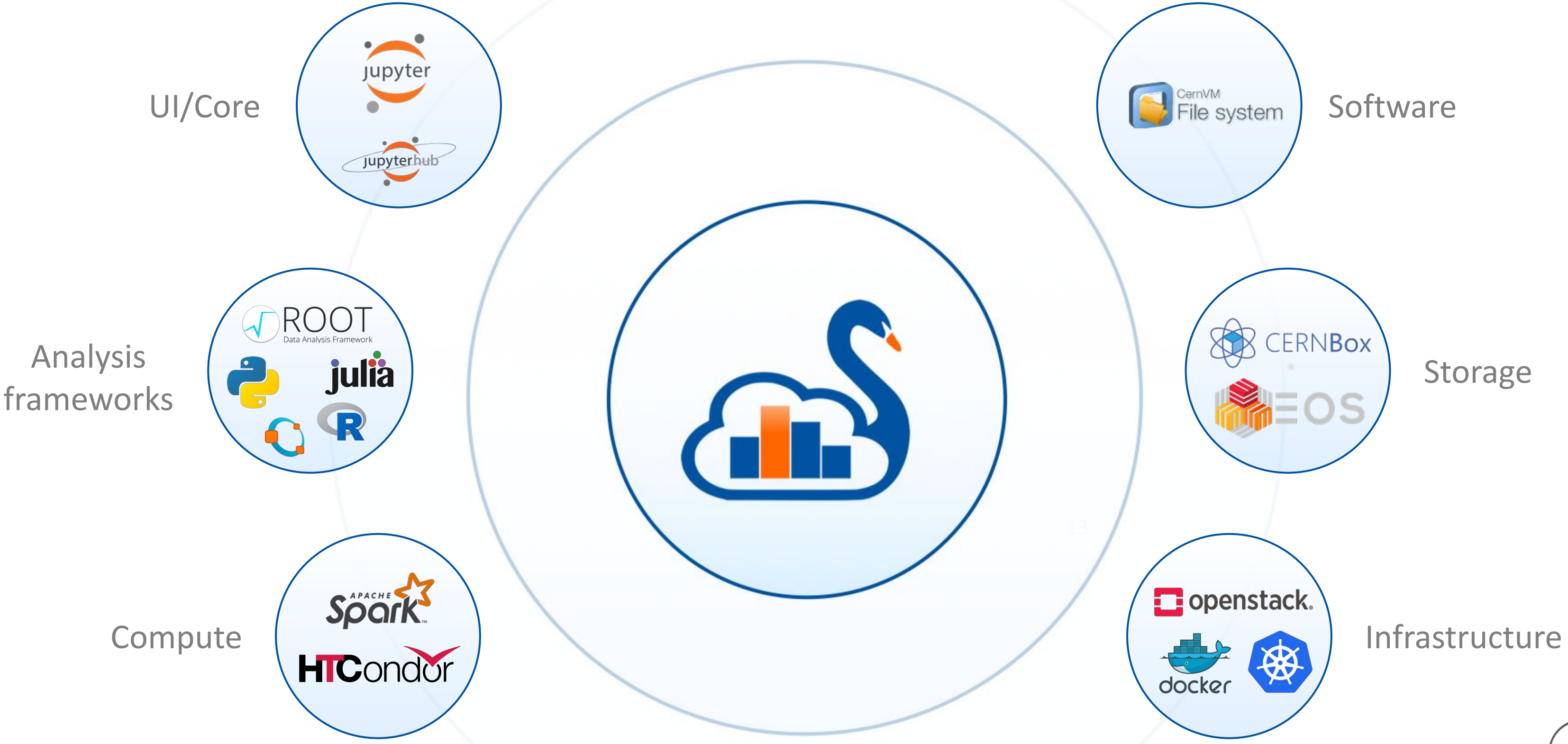
- Ongoing effort to provide an AF @ CERN
  - Interdepartmental collaboration (EP, IT)
  - In contact with Analysis Facility WG

- Build on what already exists whenever possible!

SWAN unique users in 2021 (by experiment)



SWAN’s building blocks

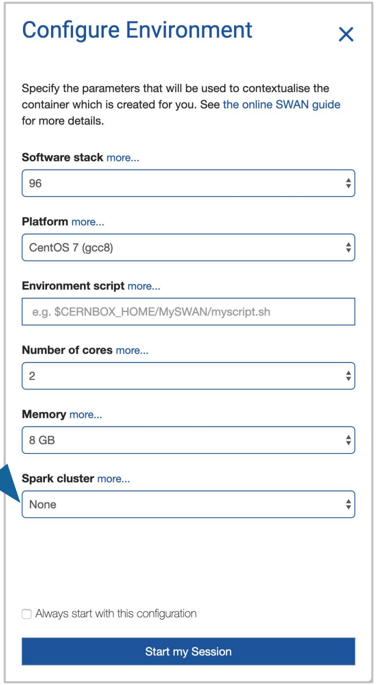


ScienceBox: installable SWAN

- SWAN can be installed on premises thanks to ScienceBox
  - Packaged SWAN, CVMFS, EOS (and soon CERNBox)
  - https://sciencebox.web.cern.ch
- Two alternatives for installation
  - Single-node: for testing, minikube
  - Multi-node: for production, kubernetes Helm charts
- Successfully deployed outside CERN
  - Aarnet, JRC, education and outreach projects
  - In progress/discussion: WUR, Purdue university (CMS Tier 2)
- In sync with CERN’s production SWAN
  - Will benefit too from the integration with Dask and resource managers (HTCondor, kubernetes)

Spark clusters

Cluster Name	Configuration	Primary Usage
analytix	46 nodes (Cores – 1956, Mem – 24.4 TB, Storage – 17.5 PB)	General Purpose
nxcals	38 nodes (Cores – 1820, Mem – 17 TB, Storage – 13 PB)	Accelerator logging (NXCALS) project dedicated cluster
Cloud containers	OpenStack project, Spark-as-a-Service, CPU-optimized (Cores 256, Mem – 2 TB, Storage – EOS) + possibly more	General Purpose Compute ONLY



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

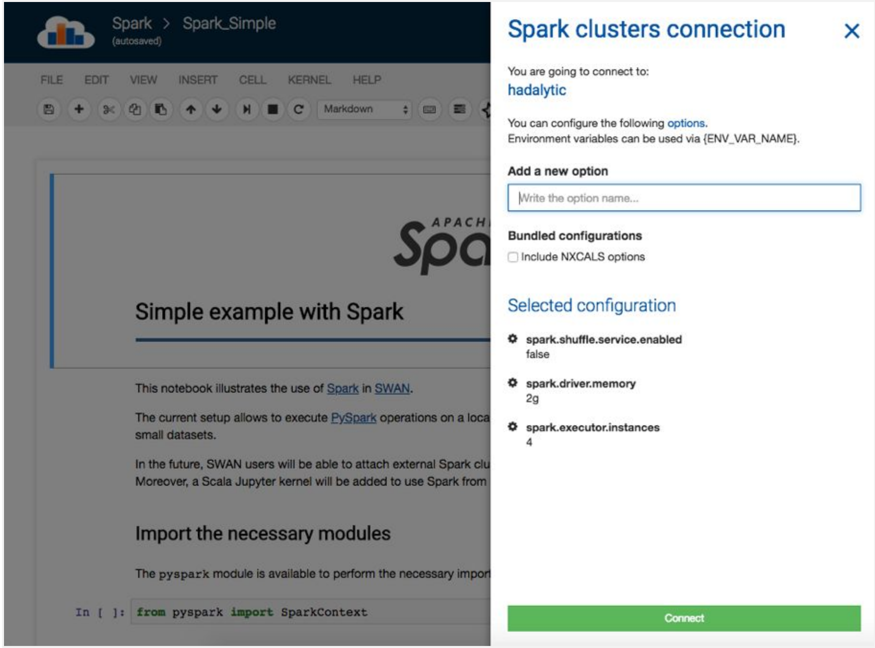
Number of cores more...: 2

Memory more...: 8 GB

Spark cluster more...: None

□ Always start with this configuration

Spark Connector



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

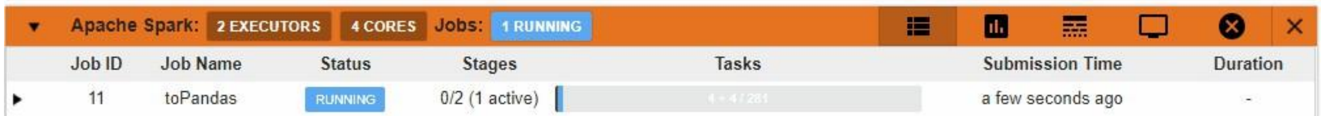
spark.driver.memory
2g

spark.executor.instances
4

- **Spark Connector** – handling the spark configuration complexity
  - User is presented with Spark Session (Spark) and Spark Context (sc)
  - Ability to bundle configurations specific to user communities
  - Ability to specify additional configuration

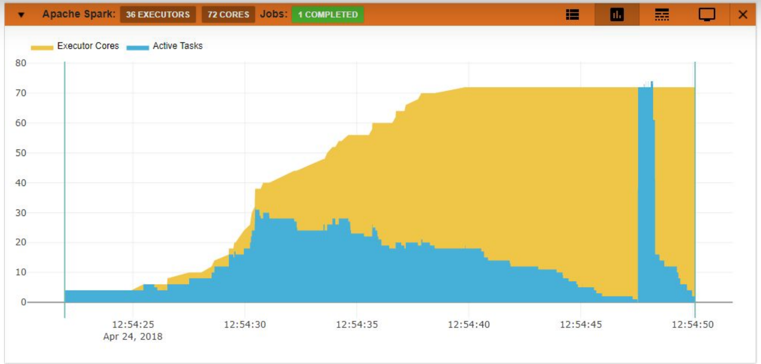
Spark Monitor

- **Spark Monitor** – Jupyter notebook extension
  - For live monitoring of spark jobs spawned from the notebook
  - A graph showing number of active tasks & executor cores vs time
  - A timeline which shows jobs, stages, and tasks

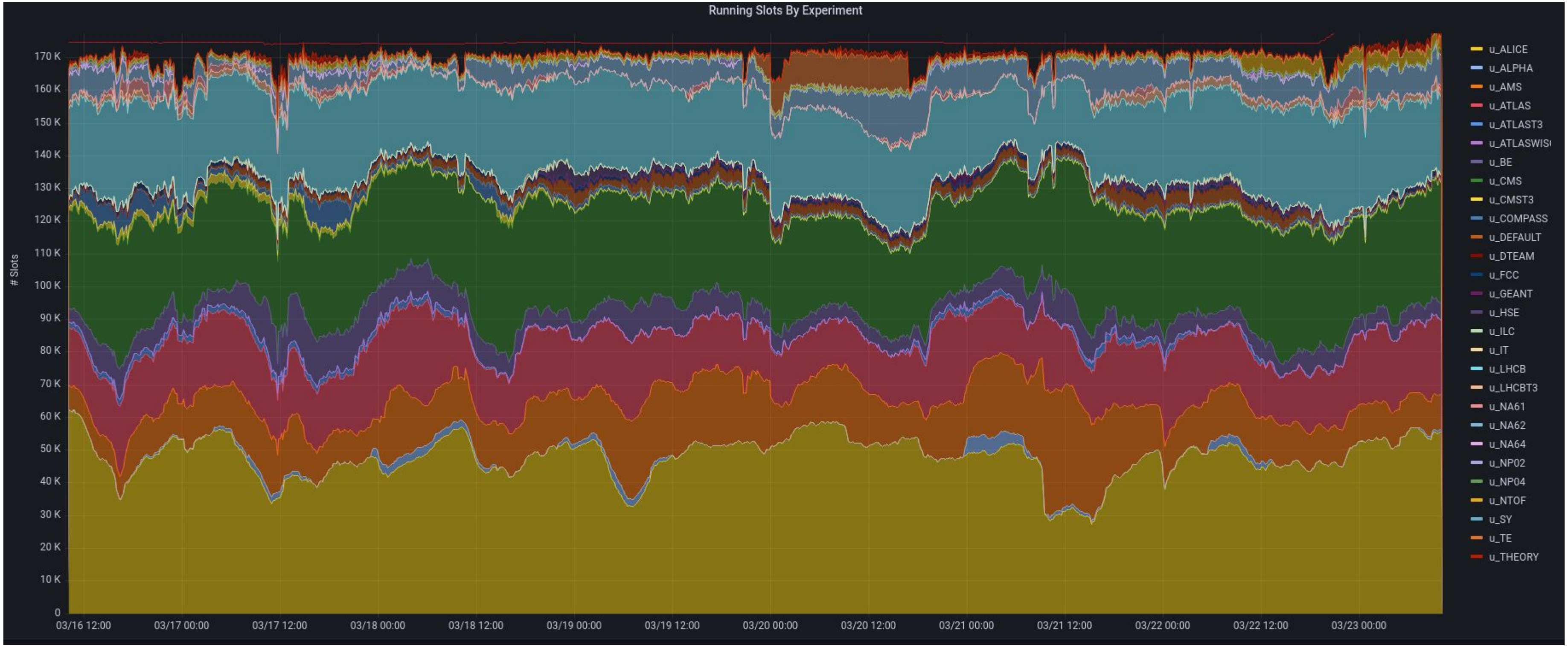


▼ Apache Spark: 2 EXECUTORS 4 CORES Jobs: 1 RUNNING

	Job ID	Job Name	Status	Stages	Tasks	Submission Time	Duration
▶	11	toPandas	RUNNING	0/2 (1 active)	0 + 4 / 281	a few seconds ago	-



HTCondor @ CERN


Running Slots By Experiment
# Slots
170 K
160 K
150 K
140 K
130 K
120 K
110 K
100 K
90 K
80 K
70 K
60 K
50 K
40 K
30 K
20 K
10 K
03/16 12:00
03/17 00:00
03/17 12:00
03/18 00:00
03/18 12:00
03/19 00:00
03/19 12:00
03/20 00:00
03/20 12:00
03/21 00:00
03/21 12:00
03/22 00:00
03/22 12:00
03/23 00:00
u_ALICE
u_ALPHA
u_AMS
u_ATLAS
u_ATLAST3
u_ATLASWIS
u_BE
u_CMS
u_CMST3
u_COMPASS
u_DEFAULT
u_DTEAM
u_FCC
u_GEANT
u_HSE
u_ILC
u_IT
u_LHCB
u_LHCBT3
u_NA61
u_NA62
u_NA64
u_NP02
u_NP04
u_NTOF
u_SY
u_TE
u_THEORY