

Hadoop service update

Integration of SWAN with Hadoop and Spark Service

Hadoop User Forum, 24th Apr 2018

Prasanth Kothuri

On behalf of SWAN service and Hadoop and Spark service

SWAN – Jupyter Notebooks On Demand



- SWAN – Service for web based analysis
 - collaboration between EP-SFT, IT-ST and IT-DB
- A web-based interactive interface and platform that combines code, equations, text and visualisations
 - Ideal for exploration, reproducibility, collaboration
- Fully Integrated with IT Spark and Hadoop Clusters
 - Modern, powerful and scalable platform for data analysis
 - Python on Spark (PySpark) at scale





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

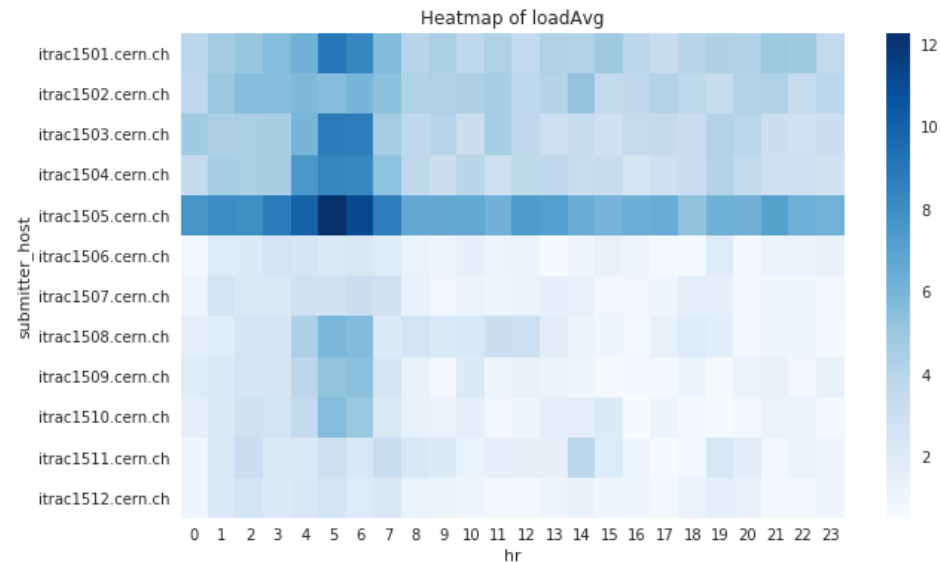
```
In [11]: df_loadAvg_pandas = spark.sql("SELECT submitter_host, \
    avg(body.LoadAvg) as avg, \
    hour(from_unixtime(timestamp / 1000, 'yyyy-MM-dd HH:mm:ss')) as hr \
    FROM loadAvg \
    WHERE submitter_hostgroup = 'hadoop/itdb/datanode' \
    AND dayofmonth(from_unixtime(timestamp / 1000,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

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')



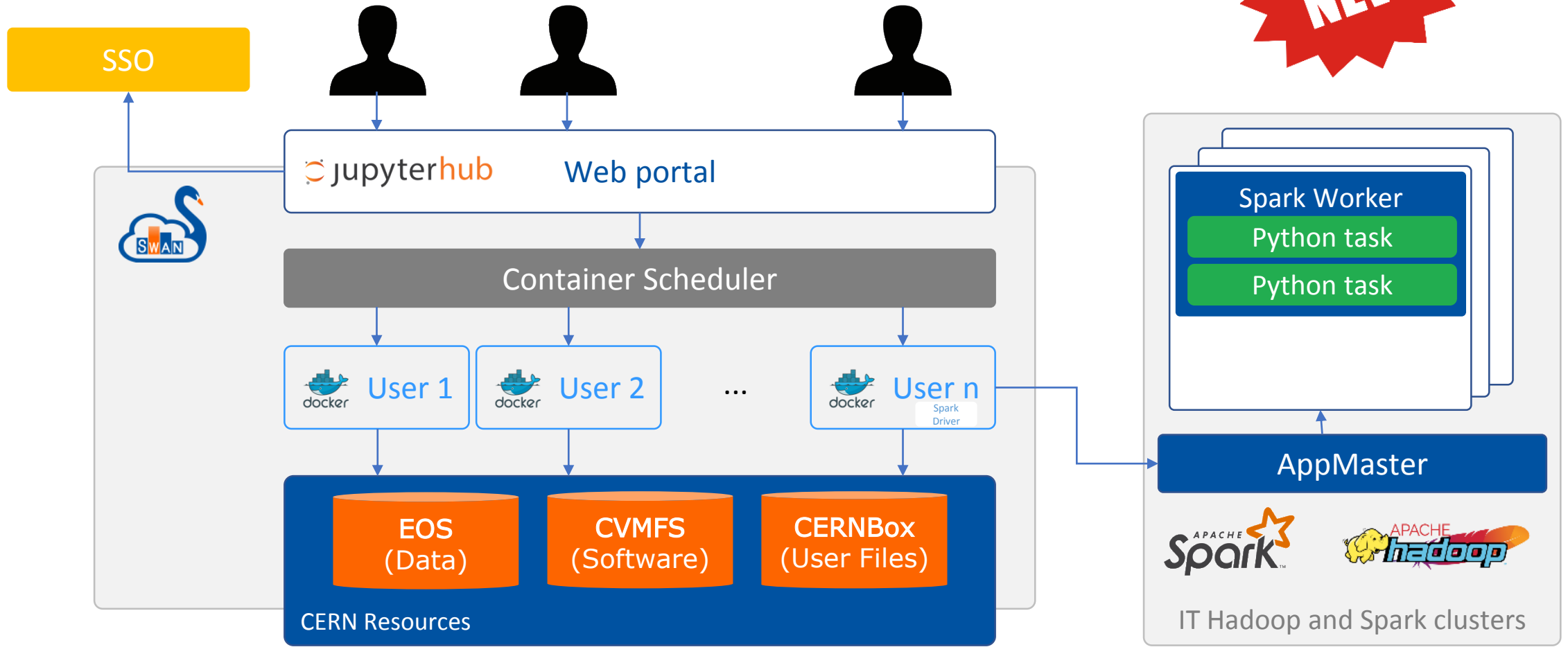
Text

Code

Monitoring

Visualizations

SWAN_Spark – Architecture





Starting your session



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
Analytix
NXCals

Always start with this configuration

Start my Session

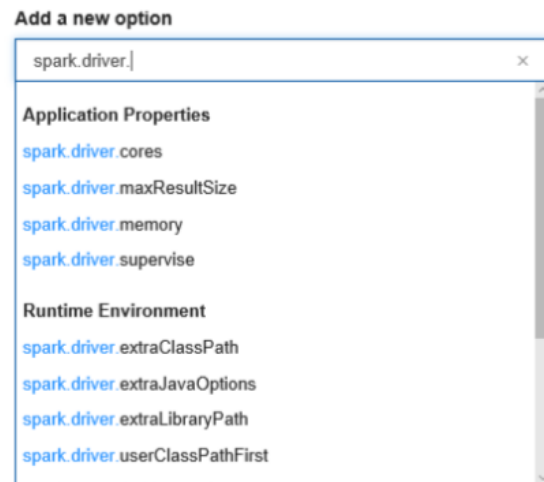
Hadoop-clusters with SWAN integration

Cluster Name	Configuration	Primary Usage	
analytix	48 nodes (Cores – 892,Mem – 7.5TB,Storage – 6 PB)	General Purpose	
hadalytic	14 nodes (Cores – 196,Mem – 768GB,Storage – 2.15 PB)	BE development. Will be decommissioned	
nxcals	20 nodes (Cores 480, Mem - 8 TB, Storage – 5 PB, 96GB in SSD)	Accelerator logging (NXCALS) project dedicated cluster	

- Future *qa* cluster will be integrated with *SWAN QA*

SWAN_Spark features

- 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



Bundled configurations

- Include NXCALS options
- Include CMSSpark options
- Include EOS_ROOT options

Selected configuration

⚙️ NXCALS

⚙️ spark.driver.extraJavaOptions

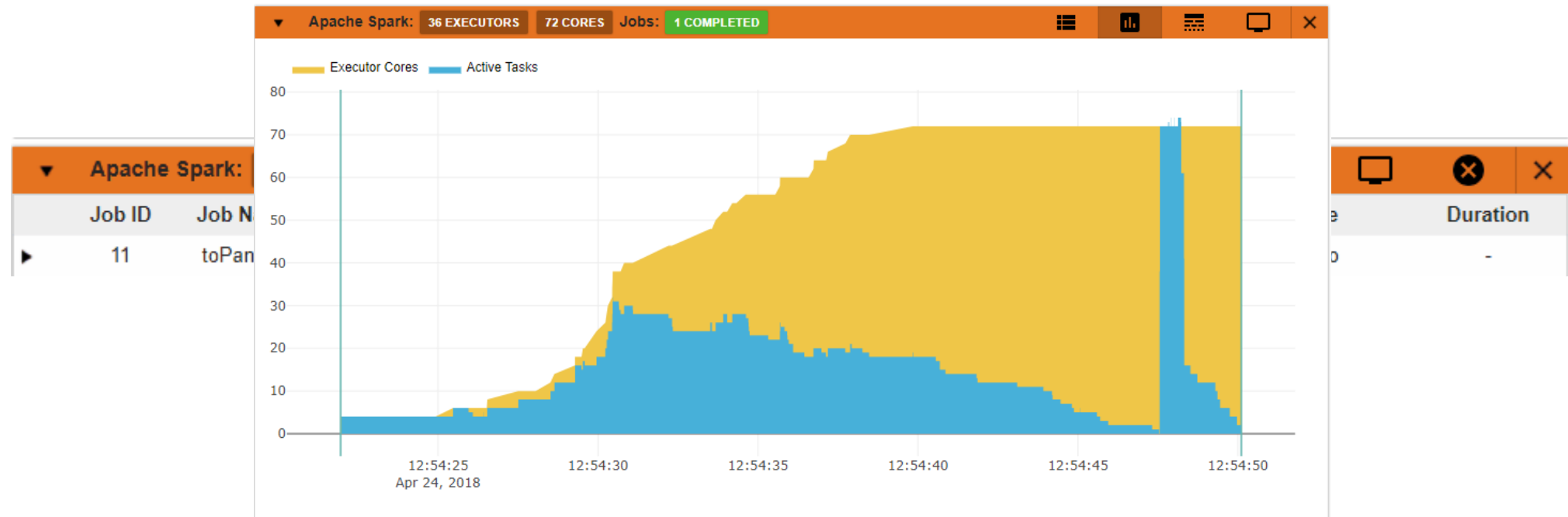
```
-Dservice.url=https://cs-ccr-nxcals6.cern.ch:19093  
-Djavax.net.ssl.trustStore=/etc/pki/tls/certs/truststore.jks  
-Djavax.net.ssl.trustStorePassword=password
```

⚙️ spark.jars

```
{LCG_VIEW}/lib/accsoft/accsoft-nxcals-data-access-0.1.66.jar,  
{LCG_VIEW}/lib/accsoft/dependency/accsoft-nxcals-common-0.1.66.jar,  
{LCG_VIEW}/lib/accsoft/dependency/accsoft-nxcals-service-client-0.1.66.jar,  
{LCG_VIEW}/lib/accsoft/dependency/activation-1.1.1.jar,  
{LCG_VIEW}/lib/accsoft/dependency/animal-sniffer-annotation-1.0.jar,  
{LCG_VIEW}/lib/accsoft/dependency/annotations-2.0.0.jar,  
{LCG_VIEW}/lib/accsoft/dependency/apacheds-i18n-2.0.0-M15.jar,  
{LCG_VIEW}/lib/accsoft/dependency/apacheds-kerberos-codec-2.0.0-M15.jar,  
{LCG_VIEW}/lib/accsoft/dependency/api-asn1-api-1.0.0-M20.jar,  
{LCG_VIEW}/lib/accsoft/dependency/api-util-1.0.0-M20.jar,
```

SWAN_Spark features

- 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 and troubleshoot Spark application

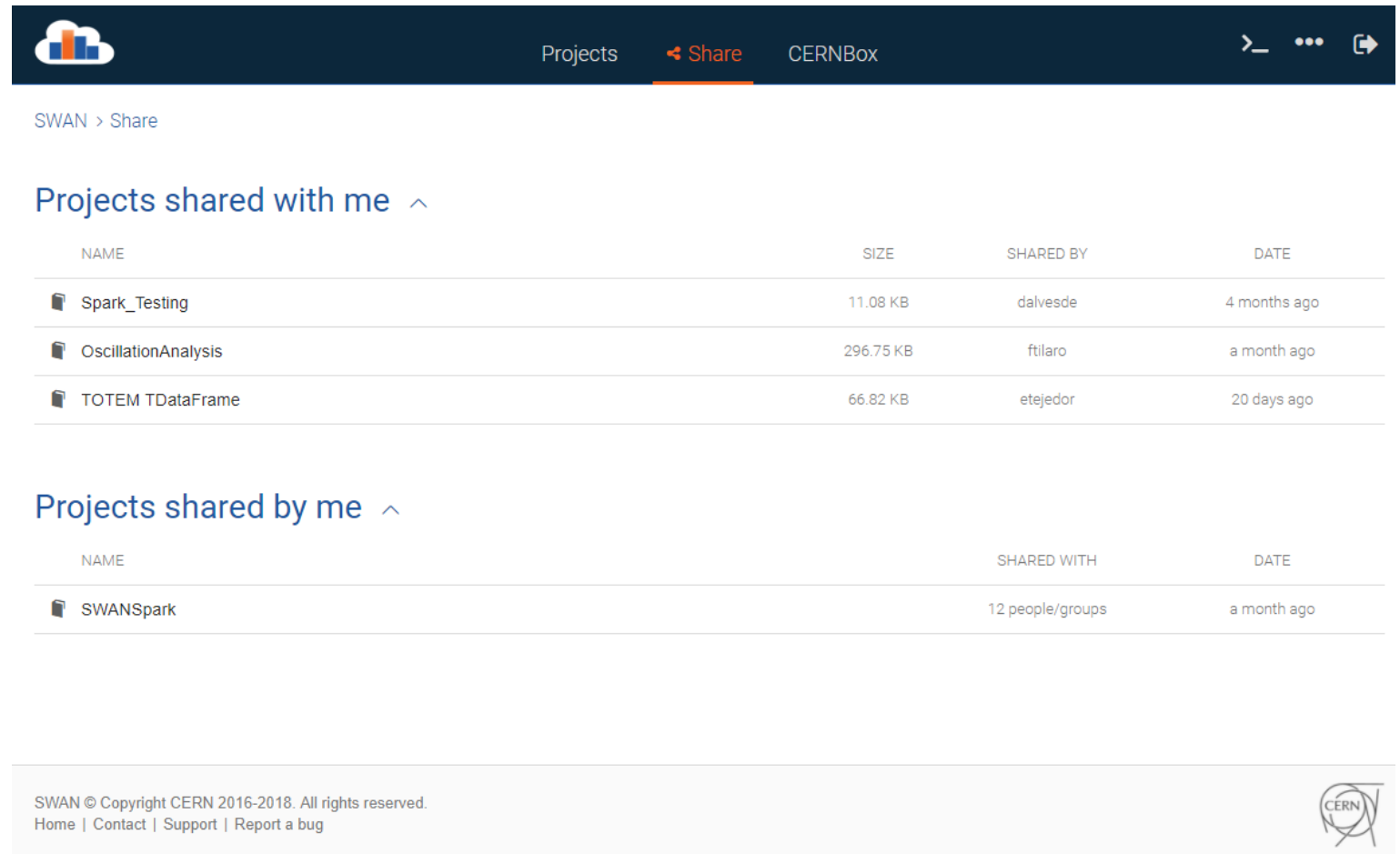


Authentication and Encryption

- Authentication
 - `spark.authenticate` : authentication via shared secret, ensures that all the actors (driver, executor, AppMaster) share the same secret
- Encryption
 - encryption is enabled for all spark application services (block transfer, RPC etc)
- Further details on SWAN_Spark security model
 - https://gitlab.cern.ch/dmaas/security/blob/master/swan_security.pdf

Collaboration and Sharing

- Concept of *project* to share notebooks, code and data
- Users can *clone* a *shared project* directly from the interface
 - Jupyter doesn't allow concurrent editing



The screenshot shows the 'Share' page in the SWAN interface. The breadcrumb path is 'SWAN > Share'. The page is divided into two sections: 'Projects shared with me' and 'Projects shared by me'. The 'Projects shared with me' section contains a table with three rows of shared projects. The 'Projects shared by me' section contains a table with one row of a project shared by the user.

NAME	SIZE	SHARED BY	DATE
Spark_Testing	11.08 KB	dalvesde	4 months ago
OscillationAnalysis	296.75 KB	ftilaro	a month ago
TOTEM TDataFrame	66.82 KB	etejedor	20 days ago

NAME	SHARED WITH	DATE
SWANSpark	12 people/groups	a month ago

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

Sharing made easy

- Sharing from inside SWAN interface
- Users can share “Projects”
 - Special kind of folder that contains notebooks and other files, i.e. input data

The screenshot displays the SWAN interface with a 'Share Project' modal window open. The background interface shows a navigation bar with 'Projects', 'Share', and 'CERNBox' tabs. Below the navigation bar, the breadcrumb 'SWAN > My Projects' is visible. The 'My Projects' section contains a list of project folders: 'analytix-hostmetrics-example', 'analytix-hostmetrics-example.org', 'OscillationAnalysis', 'physics_analysis_using_swan_spark_template', 'SWANspark', 'SWANspark_Demo', and 'TOTEM TDataFrame'. The 'Share Project' dialog box is positioned on the right side of the screen. It has a title bar with a close button (X). The dialog content includes: 'You are sharing: SWANspark', 'You can share with people. Your contacts will be able to see your project, including all the files inside it, and clone it. You can prefix the search by "a:" to also look into secondary and service accounts.', a search input field with the placeholder text 'Start typing to add names...', a list of users under the heading 'Shared with' (aimar, giffels, Imagnoni, emotes, bgarrido, jlunadur, ugentile, pmrowczy, zbaranow, prodrigu, valya), and two buttons at the bottom: 'Stop Sharing' (red) and 'Update' (blue).

Target User Communities

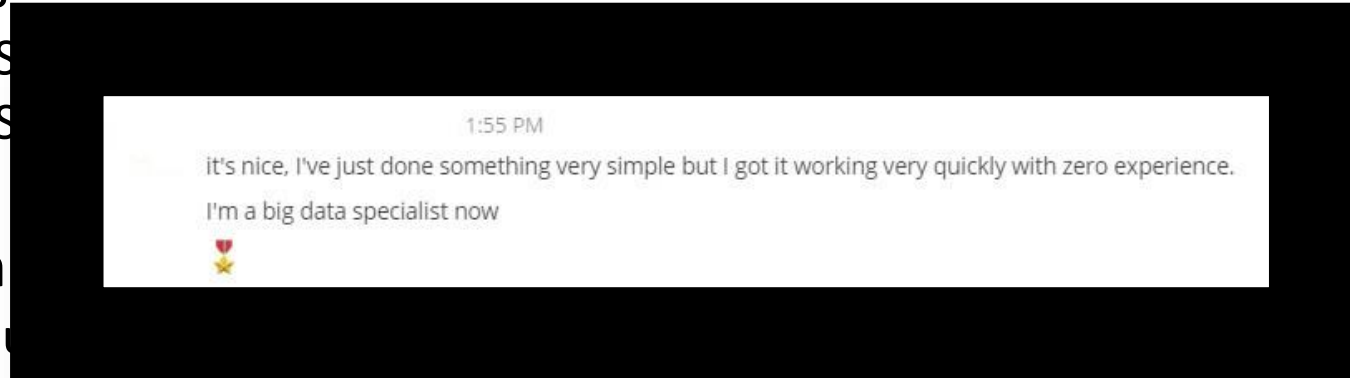
- BE-NXCALS
 - will offer their users SWAN_Spark as key entry point for analysis

- Physics

- CMS
- CMS

- IT Mon

- For t



- IT Security

- Features with the goal of lowering the barrier for large scale distributed analysis with Apache Spark (PySpark)

Industry focus



Databricks Unified Platform
- Simplifying Big Data and AI



Cloudera Data Science Workbench
- Enables fast, easy and secure self-service data science



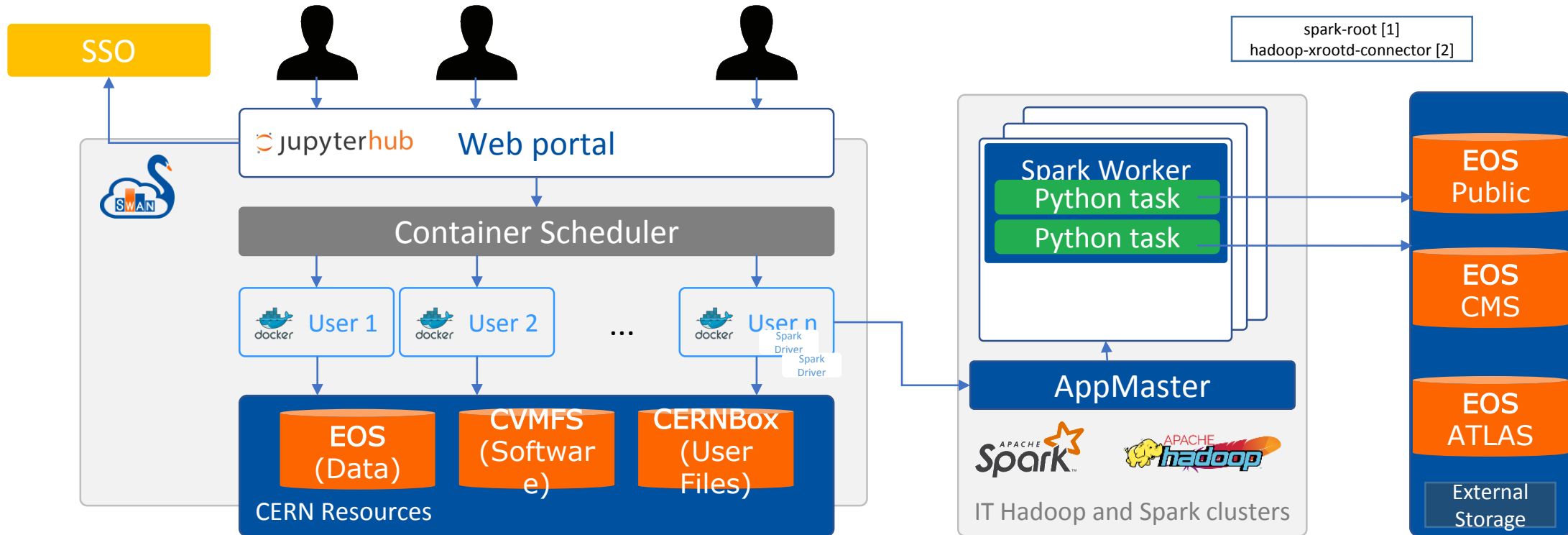
Google Colaboratory 

Colaboratory is a research tool for machine learning education and research. It's a Jupyter notebook environment that requires no setup to use.

 Seattle, WA  <https://research.google.com/colaborato...>

SWAN_Spark – Demo

SWAN_Spark – ArchitectureX



[1] <https://github.com/diana-hep/spark-root>

[2] <https://github.com/cerndb/hadoop-xrootd>



Integration of SWAN with Spark clusters

The current setup allows to execute PySpark operations on CERN Hadoop and Spark clusters. This notebook illustrates the use of Spark in SWAN to analyze the monitoring data available on HDFS and plots a heatmap of loadAvg across machines in a particular service.

Connect to the cluster

To connect to a cluster, click on the star button on the top and follow the instructions

- The star button only appears if you have selected a SPARK cluster in the configuration
- The star button is active after the notebook kernel is ready

Import necessary spark and python stuff

```
In [1]: from pyspark.sql.functions import from_unixtime, when, col
        from pyspark.sql.types import *
        from pyspark.sql.functions import from_json
```

```
[2]: %matplotlib inline
      import pandas as pd
```


Points to remember

1. By default spark driver memory is 2GB
 - Increase it by setting `spark.driver.memory` if you are going to collect more data into the driver
2. Spark dynamic scaling (allocation) is enabled only on *analytix*
3. Static executor allocation on *NXCALS* and *hadalytic*
 - *spark.executor.instance*
 - *spark.executor.memory*
 - *spark.executor.cores*
4. Python is sourced from CVMFS
 - Details of software included in LCG release - <http://lcginfo.cern.ch/>

Future work and enhancements

1. Avoid typing password to access spark clusters
 - Automatic generation of credentials (service ticket) is DONE
 - Integration with HADOOP security layer is IN PROGRESS
2. Multiple spark connections per user session
 - memory per user is restricted to 8GB – 10 GB
3. Longevity of swan user session?
 - currently its 6 hrs

Future work and enhancements

4. HDFS browser

- ability to browse HDFS from SWAN

5. Datasets

- abstraction to create and share datasets

6. Job submission

- SWAN user session is a full-fledged Hadoop-Spark client

SWAN_Spark

- SWAN_Spark is fully available Hadoop and Spark users
 - URL – <http://swan.cern.ch>
 - Example Notebooks - <[analytix-example](#)>, <[gallery](#)>
 - Note: Request an Hadoop account through SNOW
- Support ticket via SNOW, general feedback welcome to
 - ai-hadoop-admins@cern.ch
 - swan-admins@cern.ch