Welcome to Zeppelin Tutorial for Fido Team

# [What is Zeppelin](#_What_is_Zeppelin)

* 1. Definition
  2. Three features

# [How to connect to Zeppelin](#_How_to_connect)

* 1. Fido Zeppelin Notebook Path
  2. Access request
  3. 5 steps to login Zeppelin
  4. Connect Zeppelin to HPCC

# [Spark](#_Spark)

* 1. What is spark
  2. Why spark

# [Example: Import HPCC data via Spark.pyspark](#_Example:_Import_HPCC)

* 1. One example
  2. ClusterURL and thorClusterName
  3. projectList, filter, and path
  4. hpccDataset.Persist() and DataFrame.toPandas()

# [Spark.SQL](#_Spark.SQL)

* 1. What is Spark.SQL?
  2. Syntax requirements
  3. Examples

# What is Zeppelin

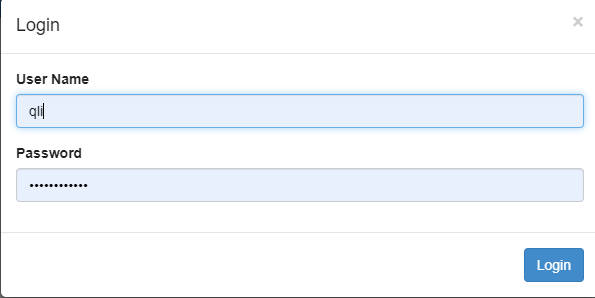
* 1. Definition: Zeppelin is a Web-Based note book that enables data-driven. Interactive data analytics and collaborative documents with Spark, SQL, and Python
  2. Three Powerful features:
     1. Spark: Zeppelin supports Spark, PySpark, Spark R, Spark SQL with dependency loader.
     2. SQL: Zeppelin lets you connect any JDBC data sources seamlessly. Postgresql, Mysql, MariaDB, Redshift, Apache Hive and so on.
     3. Python is supported with Matplotlib, Conda, Pandas SQL and PySpark integrations.

# How to connect with Zeppelin?

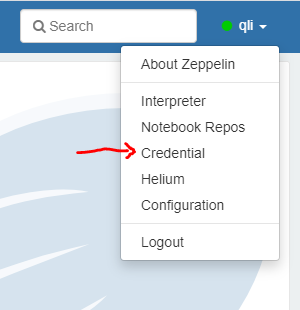
* 1. Fido team zeppelin path: <http://10.194.169.3:8080/>
  2. Access Request: before you login, check if you have access to Fido dev ECL watch:[10.194.169.2:8010](http://10.194.169.2:8010/#/stub/Main-DL/Activity-DL/Grid)
     1. If you do not have the access to Fido dev ECL watch, please request a Jira ticket.
     2. To request access you will need to fill out a ticket in this ISIT queue: <https://isit.lexisnexis.com/Ticket/Create.html?Queue=1086>
  3. Please follow next 5 steps to log in Zeppelin
     1. Go to [Zeppelin page](http://10.194.169.3:8080/)



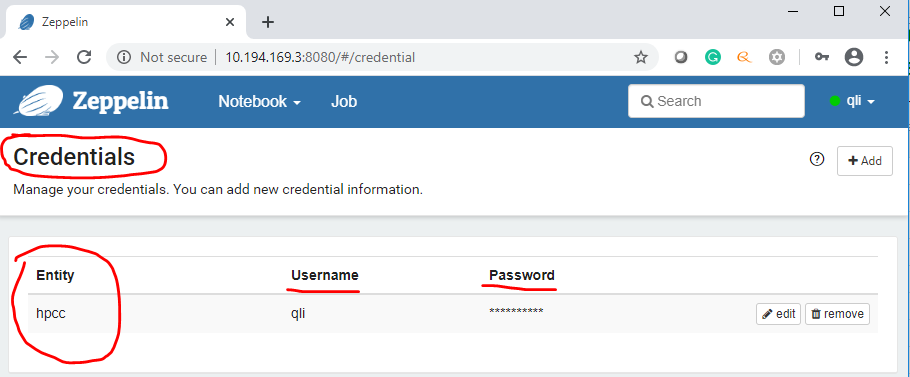
* + 1. Login with your ECL username and password. If you have trouble, please check if you have valid access to ECL



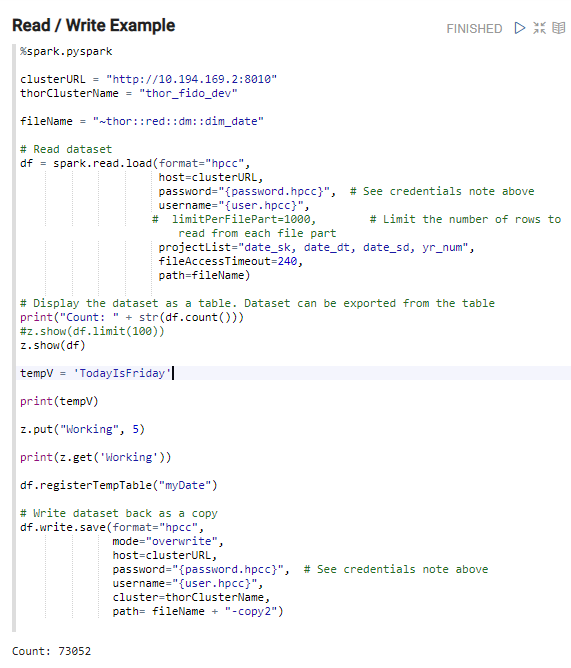
* + 1. Click your username, and click the Credential



* + 1. Set up your credential. Entity is ‘hpcc’, Username is your ECL username, and same for your password.



* + 1. Logout and log back in
  1. How to test if your zeppelin successfully connects to HPCC?
     1. Go to **Example notebook on the Zeppelin front page**, and copy paste below code into a new zeppelin notebook, and click run

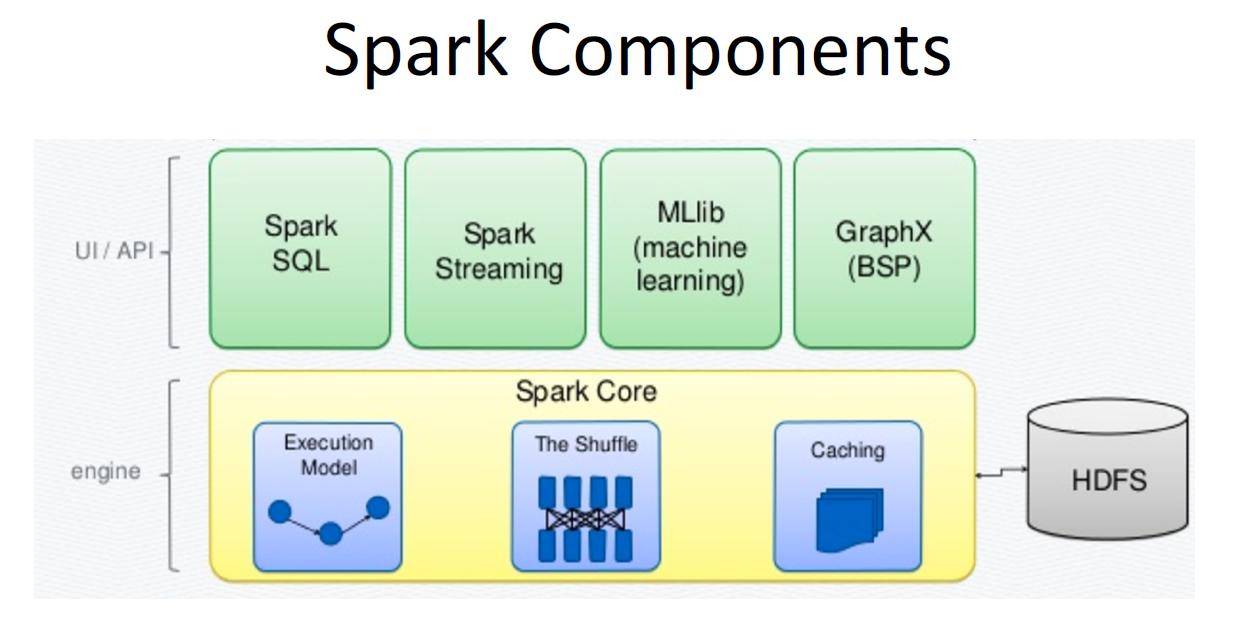


* + 1. If you your code can successfully complied, which means your zeppelin note is good to go!

# Spark

* 1. What is Spark?

Apache Spark is a fast, in-memory data processing engine which allows data workers to efficiently execute streaming, machine learning or SQL workloads that require fast iterative access to dataset.



* 1. Why Spark?
     1. Faster Speed
     2. Generality: Spark makes it easy to combine different processing models seamlessly in the same application.

# Example: Import HPCC data via Spark.pyspark

* 1. If you run below code in the new zeppelin notebook:

%spark.pyspark

import pandas as pd

import numpy as np

#PROD

clusterURL = "http://10.194.93.3:8010"

thorClusterName = "thor\_40\_83"

cust\_acct = spark.read.load(format="hpcc",

host=clusterURL,

password="{password.hpcc}",

username="{user.hpcc}",

projectList="customer\_account\_sk, current\_subaccount\_id, cy\_vertical\_sk, subaccount\_name, src\_create\_dt, src\_cancel\_dt, bip\_prox\_id",

filter="platform\_cd in ('WCI','WCO','WCZ','UA')",

fileAccessTimeout=240,

path="~thor::red::dm::dim\_customer\_account")

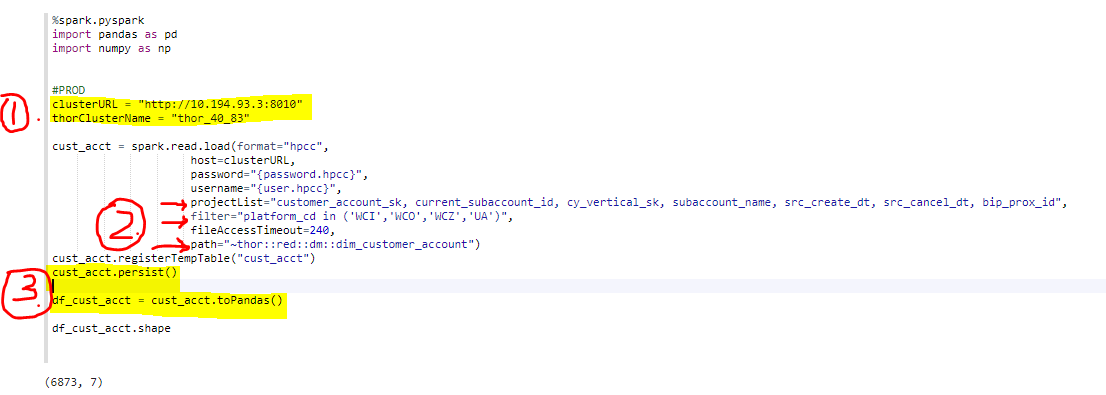
cust\_acct.registerTempTable("cust\_acct")

cust\_acct.persist()

df\_cust\_acct = cust\_acct.toPandas()

df\_cust\_acct.shape

Then you will get similar results as below:



* 1. I would love to go over the above code by 3 parts:

1. **ClusterURL and thorClusterName**: we can choose to work on Dev or Prod.

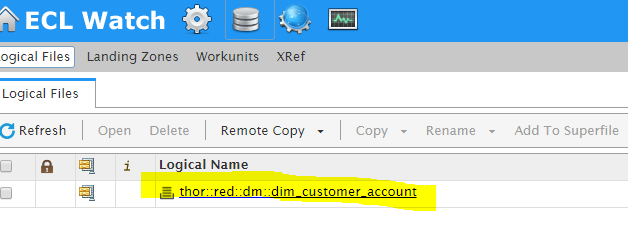
If you choose **dev:**

* clusterURL = "http://10.194.169.2:8010"
* thorClusterName = "thor\_fido\_dev"

If you choose **prod:**

* clusterURL = "http://10.194.93.3:8010"
* thorClusterName = "thor40\_83"

1. **projectList, filter, and path** 
   * **projectList:** list the variables you need from the dataset if you don’t want all the variables. Please note: double check the variable names. Wrong variable names will not produce compiler error, but you will not be able to find the variable that you want in the results.
   * **Filter:** use query language to write your dataset criteria
   * **Path:** the same file path in the ECL watch. For example, if the file path is path="~thor::red::dm::dim\_customer\_account", then you can locate the data file in the ECL Watch by same path



1. **hpccDataset.Persist() and DataFrame.toPandas()**
   * **hpccDataset.Persist():** When you persist an RDD(Resilient Distributed Datasets), the first time it is computed in an action, it will be kept in memory across the nodes.
   * **DataFrame.toPandas():** transform a spark RDD into a pandas data frame so that you can use pandas to do further analysis
     1. **Add one line of magical code: this will enhance the speed**

spark.conf.set("spark.sql.execution.arrow.enabled", "true")

* + 1. **final = final.select("\*").toPandas()**
* **Example:**

final.registerTempTable("final")

spark.conf.set("spark.sql.execution.arrow.enabled", "true")

final.persist()

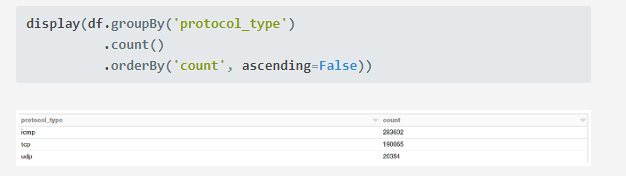
final.registerTempTable("final")

# final = final. toPandas()

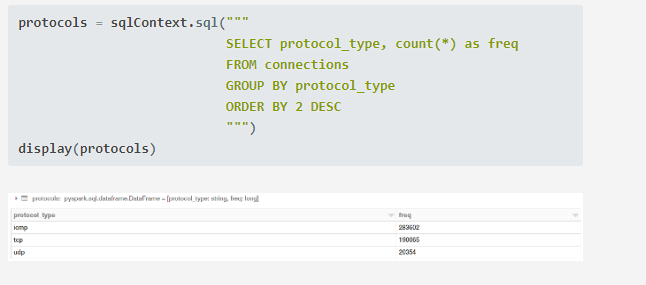
# 

# Spark.SQL

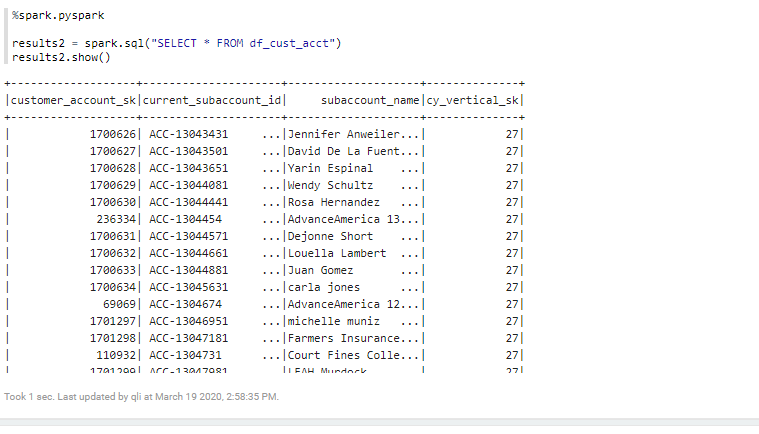
* 1. What is Spark.SQL?
* Spark SQL provides a dataset abstraction that simplifies working with structured datasets
  1. Spark.SQL Syntax requirements?
* There are two ways to perform Spark.SQL: DataFrame DSL syntax and traditional SQL
  1. You can use normal DataFrame DSL syntax to perform aggregations. Please check section d: appendix for more DSL syntax functions.



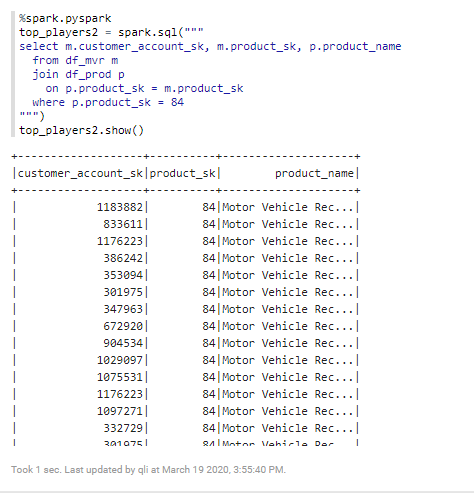
* 1. It will be no problem if you prefer writing SQL, both results will be the same



* 1. Here are some of my Spark.SQL examples in the Zeppelin
     1. SELECT \* FROM



II. SELECT, FROM, LEFT JOIN, WHERE



* 1. Appendix for Spark.SQL DSL syntax

