Lab: Data Sources in DSXL

*Jun 2018*

*Author: Parminder Kaur parminder.kaur@ibm.com*

**Table of contents**

Contents

[Overview 1](#_Toc515985148)

[What You Should Be Able to Do 1](#_Toc515985149)

[Required software, access, and files 1](#_Toc515985150)

[Part 1: Add dashdb built-in/packaged data source 1](#_Toc515985151)

[Part 2: Add PostgreSQL custom data source 2](#_Toc515985152)

[Part 3: Add HDFS and Hive HDP data sources 3](#_Toc515985153)

# Overview

In this lab, you will learn how to create data source and remote dataset definition that can be used to pull data into DSXL.

DSXL v1.2 packages Big SQL, Db2, Db2 for z/OS, dashdb, Hive, HDFS, Informix, Netezza and Oracle databases. However, customer can also work with other JDBC databases that are not prepackaged by using custom JDBC data source feature. In this lab, you will learn how to work with a packaged data source, custom JDBC and HDP data sources.

This lab has been divided into three parts:

Part 1 - where you will add a dashdb built-in/packaged data source to DSXL

Part 2 – where you will add a PostgreSQL custom data source to DSXL

Part 3 – where you will add HDP data sources to DSXL

# What You Should Be Able to Do

* Pull data from external data sources explicitly supported or packaged by DSXL
* Configure DSXL to work with custom JDBC data sources
* Share data sources and remote datasets definitions with project collaborators
* Browse and Preview HDP and Hive remote datasets
* Use insert-to-code to read/fetch data from defined remote datasets

# Required software, access, and files

• To complete this lab, you will need access to a DSX Local cluster v1.2.0.2 with Internet connectivity.

# Part 1: Add dashdb built-in/packaged data source

1. Create a new blank project, follow instruction in documentation <https://content-dsxlocal.mybluemix.net/docs/content/local/projects.html#create-a-project>.
2. Create dashdb data source and remote data set definitions by following instructions in documentation <https://content-dsxlocal.mybluemix.net/docs/content/local/createdatasources.html>.

You can use below mentioned dashdb test server for this lab.

JDBC URL: jdbc:db2://dashdb-entry-yp-dal09-07.services.dal.bluemix.net:50000/BLUDB

Username: dash10765

Password: j1S(m{sWv7ZB

schema: SAMPLES

table: EDUCATION

Note that, there is a Shared checkbox option in add data source page. This option allows you to share your data source credential information with all the project collaborators. However, if you want to keep your credential private, leave it unchecked. The collaborators may see the data source definition and related dataset, but they would not see credential information and should provide a valid credential to access the data source.

1. Pull data from above created dashdb remote asset using insert-to-code.
   1. Create a blank Jupyter notebook. Go to **Notebooks -> add notenook**. Select Jupyter with Python 2.7, Scala 2.11, R 3.4.3, Spark 2.0.2.
   2. Click on Find data icon [creen shot 2018-06-01 at 4 26 09 pm](https://media.github.ibm.com/user/54527/files/84f286ae-65b8-11e8-9432-a844cfb7ab12) . Select Remote and click inset to code on your dashdb remote dataset. Select Insert Pandas DataFrame from the dropdown, this will insert a code snippet in a new jupyter cell.
   3. Run the cell by clicking **Cell -> Run All**. After successful cell execution, you would see data frame from your dashdb table as output.

# Part 2: Add PostgreSQL custom data source

To work with custom JDBC, admin needs to upload JDBC driver jar to DSXL so that DSX users can add custom data sources and remote data sets and access data from them.

Follow instructions in documentation <https://content-dsxlocal.mybluemix.net/docs/content/local/drivers.html> to first import PostgresSQL driver jar, create a custom data source and a remote data set, and finally access data using Insert Pandas DataFrame insert-to-code option from a Jupyter with Python 2.7, Scala 2.11, R 3.4.3, Spark 2.0.2 notebook.

For this lab, you can download PostgresSQL driver jar from <https://ibm.box.com/s/cjdhf403snni14dvo6bu02ijx038x4ge> and use following PostgreSQL test server.

Data source type: custom jdbc

JDBC URL: jdbc:postgresql://sl-us-south-1-portal.8.dblayer.com:27422/compose

JDBC driver class name: org.postgresql.Driver

Username: admin

Password: TMFZJHGBDSLWMXIU

schema: public

table: cars

# Part 3: Add HDFS and Hive HDP data sources

Pre-requites:

* You need to have a HDP cluster with DSXHI (DSX Hadoop Integration Service) installed on Hadoop edge node. Already covered in `Hadoop Integration` training session. Here is the documentation <https://content-dsxlocal.mybluemix.net/docs/content/local/hdp.html>.
* If you choose not to install DSXHI on your HDP cluster, then follow steps mentioned in documentation <https://content-dsxlocal.mybluemix.net/docs/content/local/hdp.html#option-2-set-up-a-hdp-cluster-without-dsxhi>. (also follow steps from 3.2)

3.1. For HDP DSXHI cluster,

1. Register DSXHI cluster to DSXL. Follow documentation <https://content-dsxlocal.mybluemix.net/docs/content/local/hadoopintegration.html>.
2. Once you are done with registration, the HDFS and Hive data sources will automatically get populated in your existing project or new project. You can browse HDFS and Hive data sources and preview their remote data sets.
3. Access data using Insert Pandas DataFrame insert-to-code option on HDFS data set from a Jupyter with Python 2.7, Scala 2.11, R 3.4.3, Spark 2.0.2 notebook. Note that insert-to-code is not supported for Hive remote data sets. Documentation on Hive <https://content-dsxlocal.mybluemix.net/docs/content/local-dev/hadoop-secure_hive_data.html> (beyond the scope of this lab).
   1. For HDP non-DSXHI cluster,
4. Download topology xml file from <https://ibm.box.com/s/d14hcxc8qgzur13tugwgztobpxirjq6v>.
5. Replace `knox.token.verification.pem` property value with the token generated by your DSXL cluster. To retrieve token, run `curl -k https://<your-cluster-ip>/auth/jwtcert` on command line or terminal.



Give your xml file a unique name so that it does not conflict with others lab work.

1. Copy your topology xml file to HDP cluster. For this lab, use following cluster: (Note this cluster would expire on Jun14)

prmndr-hdp1-fyre.ibm.com

password: Temp4lab!

// \*\*\*use below command for copy\*\*\*

scp ./dsx-yourname.xml root@prmndr-hdp1.fyre.ibm.com:/usr/hdp/current/knox-server/conf/topologies

1. Once you have configured your HDP cluster by adding a new topology for DSX, create HDFS and Hive data sources and remote data sets manually.

Data source type: HDFS - HDP

HDFS host: prmndr-hdp1-fyre.ibm.com

HDFS port: 8020

WebHDFS URL: https://prmndr-hdp1.fyre.ibm.com:8443/gateway/<dsx-yourname>/webhdfs/v1

Data source type: Hive - HDP

WebHCat URL: https://prmndr-hdp1.fyre.ibm.com:8443/gateway/<dsx-yourname>/templeton/v1

WebHDFS URL: [https://prmndr-hdp1.fyre.ibm.com:8443/gateway/<dsx-yourname>/webhdfs/v1](https://prmndr-hdp1.fyre.ibm.com:8443/gateway/%3cdsx-yourname%3e/webhdfs/v1)

Livy URL: https://prmndr-hdp1.fyre.ibm.com:8443/gateway/<dsx-yourname>/livy2/v1

1. You can browse and preview on HDFS and Hive assets. Also, you can access data from insert-to-code. Note, insert-to-code on Hive remote data set is not yet supported.