PXF Build and Install

How to Build

PXF uses Gradle. For simplicity we've encapsulated it with simple make commands. Make sure Java is installed prior to building PXF.

```
# Clone hawq repository if you haven't previously done
git clone https://git-wip-us.apache.org/repos/asf/hawq.git

# Head to PXF code
cd hawq/pxf

# Compile & Test PXF
make

# Simply Run unittest
make unittest
```

Setup Prerequisites

If you simply need to test PXF, you can do so using the Demo profile which doesn't require any requisites as all it does is test against static data from PXF.

Setup Hadoop

Please follow the steps here: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html

Note:

- you might need to build hadoop from source on Red Hat/CentOS 6.x if the downloaded hadoop package has higher glibc version requirement. When that happens, you will probably see the warning below when running start-dfs.sh." WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform"
- You will also need to set the port for fs.defaultFS to 8020 in etc/hadoop/core-site.xml (The example above set it as 9000.)
- HDFS is a must, but YARN is optional. YARN is only needed when you want to use YARN as the global resource manager.
- must setup passphraseless ssh, otherwise there will be some problems of "hawq init cluster" in the following step.

Your need to verify your HDFS works.

```
# start HDFS
start-dfs.sh

# Do some basic tests to make sure HDFS works
echo "test data" >> ./localfile
hadoop fs -mkdir /test
hadoop fs -put ./localfile /test
hadoop fs -ls /
hadoop fs -get /test/localfile ./hdfsfile
```

Setup HAWQ (If you use HAWQ)

Please follow the steps here to Setup HAWQ

Setup Hive (If you need Hive)

Hive needs to be installed only if you wish to run HAWQ against Hive tables

Download any hive 1.x release from one of the Apache download mirrors http://www.apache.org/dyn/closer.cgi/hive/

```
# Extract Hive tarbal
tar -xzvf apache-hive-x.y.z-bin.tar.gz

# Set Hadoop path
export HADOOP_HOME=<your hadoop deployment location>
export HADOOP_USER_CLASSPATH_FIRST=true
# Set Hive path
cd apache-hive-x.y.z-bin
export HIVE_HOME=$PWD

# Create HDFS temp directory
$HADOOP_HOME/bin/hadoop fs -mkdir /tmp
$HADOOP_HOME/bin/hadoop fs -chmod g+w   /tmp

# Create Hive Warehouse
$HADOOP_HOME/bin/hadoop fs -mkdir -p /user/hive/warehouse
$HADOOP_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse

# Set Metastore URI in $HIVE_HOME/conf/hive-site.xml
<configuration>
    <property>
        <name>hive.metastore.uris</name>
        <value>thrift://localhost:9083</value>
    </property>
</configuration>

# Start Hive Metastore as a background process
$HIVE_HOME/bin/hive --service metastore &
# Use Hive CLI/Shell
$HIVE_HOME/bin/hive
hive> CREATE TABLE hivetest (id INT, name STRING);
hive> SHOW TABLES;
hive> DESCRIBE hivetest;
```

Install PXF

```
cd $CODE_BASE/pxf

# Set PXF home directory.
# GPHOME must have been set previously as per the HAWQ Build instructions
export PXF_HOME=$GPHOME/pxf

# If you wish to install PXF for GPDB, please refer to Build PXF for other databases section below instead.
make install
# This would create the necessary artifacts under PXF_HOME
```

PXF RPM (Optional)

If you prefer using the rpm approach of installing PXF on an environment that has the required rpm dependancies installed, you can do the following

```
# Create rpm artifacts
make rpm
# Install apache tomcat rpm
rpm -ivh distribution/apache-tomcat-*.rpm
# Install the pxf rpms
rpm -ivh build/distribution/pxf*
```

Configure PXF

You will see the PXF configuration files in $PXF_HOME/conf

Update the following files based on your environment and hadoop directly layout.

1. pxf-env.sh
    a. Set LD_LIBRARY_PATH to ${HADOOP_HOME}/lib/native
    b. Set PXF_LOGDIR to ${PXF_HOME}/logs
    c. Set PXF_RUNDIR to ${PXF_HOME}
    d. Set PXF_USER to your username
2. pxf-log4j.properties
    a. Set log4j.appender.ROLLINGFILE.File to the expanded path of $PXF_HOME/logs/pxf-service.log. (Don't use the environment variable in this file)
3. pxf-private.classpath
    a. Update the library and configuration paths of hadoop,hive,pxf, etc. Use only absolute paths without referring to environment variables

Init/Start/Stop PXF

```
# Deploy PXF
$PXF_HOME/bin/pxf init
# If you get an error "WARNING: instance already exists in ..." make sure you clean up pxf-service directory
under $PXF_HOME/bin/pxf and rerun init

# Create PXF Log Dir
mkdir $PXF_HOME/logs

# Start PXF
$PXF_HOME/bin/pxf start

# Check Status
$PXF_HOME/bin/pxf status
# You can also check if the service is running by using the following request to check API version
curl "localhost:51200/pxf/ProtocolVersion"

# To stop PXF $PXF_HOME/bin/pxf stop

## Note: If you see a failure
```

Test PXF

Below are steps which demonstrates accessing a HDFS file from HAWQ.

```
# Create an HDFS directory for PXF example data files
$HADOOP_HOME/bin/hadoop fs -mkdir -p /data/pxf_examples

# Create a delimited plain text data file named pxf_hdfs_simple.txt:
echo 'Prague,Jan,101,4875.33' > /tmp/pxf_hdfs_simple.txt
echo 'Rome,Mar,87,1557.39' >> /tmp/pxf_hdfs_simple.txt
echo 'Bangalore,May,317,8936.99' >> /tmp/pxf_hdfs_simple.txt
echo 'Beijing,Jul,411,11600.67' >> /tmp/pxf_hdfs_simple.txt

# Add the data file to HDFS:
$HADOOP_HOME/bin/hadoop fs -put /tmp/pxf_hdfs_simple.txt /data/pxf_examples/

#Display the contents of the pxf_hdfs_simple.txt file stored in HDFS:
$HADOOP_HOME/bin/hadoop fs -cat /data/pxf_examples/pxf_hdfs_simple.txt
```

Now you can access the hdfs file from HAWQ using the HdfsTextSimple profile as shown below.

```
postgres=# CREATE EXTERNAL TABLE pxf_hdfs_textsimple(location text, month text, num_orders int, total_sales
float8)
            LOCATION ('pxf://localhost:51200/data/pxf_examples/pxf_hdfs_simple.txt?PROFILE=HdfsTextSimple')
          FORMAT 'TEXT' (delimiter=E',');
postgres=# SELECT * FROM pxf_hdfs_textsimple;

   location    | month | num_orders | total_sales
---------------+-------+------------+-------------
 Prague        | Jan   |        101 |     4875.33
 Rome          | Mar   |         87 |     1557.39
 Bangalore     | May   |        317 |     8936.99
 Beijing       | Jul   |        411 |    11600.67
(4 rows)
```

Below are steps which demonstrates accessing a Hive table from HAWQ

```
# Create a Hive table to expose our sample data set.
hive> CREATE TABLE sales_info (location string, month string,
        number_of_orders int, total_sales double)
        ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
        STORED AS textfile;

# Load the pxf_hive_datafile.txt sample data file into the sales_info table you just created:
hive> LOAD DATA LOCAL INPATH '/tmp/pxf_hive_datafile.txt'
        INTO TABLE sales_info;

# Perform a query from hive on sales_info to verify that the data was loaded successfully:
hive> SELECT * FROM sales_info;

# Query the table from HAWQ to access the hive table
postgres=# SELECT * FROM hcatalog.default.sales_info

   location    | month | num_orders | total_sales
---------------+-------+------------+-------------
 Prague        | Jan   |        101 |     4875.33
 Rome          | Mar   |         87 |     1557.39
 Bangalore     | May   |        317 |     8936.99
 ...
```

Build PXF for other databases

PXF can be deployed to different environments, for different databases. Thus it's convenient to tailor PXF build for some specific default configuration parameters, such as - default PXF user, default log and run directories.

All supported databases are stored in hawq/pxf/gradle/profiles. By default, HAWQ databases is used.

To build PXF bundle for GPDB:

```
make install DATABASE=gpdb
```