Setting up Pentaho Data Integration 5.4.1 with Hadoop Cluster (Clouder Manager)

Pentaho Data Integration (`PDI`) is a suite of open source Business Intelligence (BI) products which provide data integration, OLAP services, reporting, dashboarding, data mining and ETL capabilities. Here we are setting up pentaho server has the below steps. Will be referring to `Pentaho Data Integration` as **PDI** from now on.

1. Install PDI.
2. Make ETL (PDI) as the Gateway for HDFS, HIVE, YARN, HBASE from Cloudera Manager.
    1. Test Access to Hadoop from PDI Server.
3. Updating configuration in PDI for Hadoop Setup.
    1. Test PDI with Hadoop.

Installing Pentaho Server 5.4.1 Community Edition.

Installation is very straight forward, we get a `pdi.zip` which needs to be extracted in the location required. Here we will be extracting the file into `/opt` directory on our RHEL ETL PDI Server.

Extract and move the directory to `/opt`

```
[root@pentaho-server ~]# unzip pdi.zip
[root@pentaho-server ~]# mv data-integration /opt
```

Create a soft Link `/opt` directory, we have used `kettle` / `pentaho` as the softlink.

```
[root@pentaho-server ~]# cd opt
[root@pentaho-server opt]# ln -s data-integration pentaho
[root@pentaho-server opt]# ln -s data-integration kettle
[root@pentaho-server opt]# ls -l
total 12
drwxr-xr-x  12 root         root         4096 Sep 14 11:43 data-integration
lrwxrwxrwx   1 root         root           16 Sep  7 14:37 kettle -> data-integration
lrwxrwxrwx   1 root         root           16 Sep 15 10:17 pentaho -> data-integration
drwxr-xr-x.  2 root         root         4096 May 17  2013 rh
[root@pentaho-server opt]#
```

Thats it Our, Installation is complete. :)

Create ETL (PDI) Server Gateway for `hdfs`, `hive`, `yarn`, `hbase` from Cloudera Manager.

We are doing this so for two reasons.

1. PDI Hadoop cloudera configurations can be managed from the cloudera-manager.
2. Also we make PDI server communicate to Hadoop cluster directly.

Before we add the gateway, this server should be part of the hosts on the cloudera manager, so that all the packages are installed on this server.

Steps to add a host on Cloudera Manager.

NOTE : Before we start make sure the `/etc/hosts` directory is similar to what we have on the `hadoop` cluster, and ETL PDI server, so that we are able to communicate with the `hadoop` cluster from ETL PDI Server.

1. Click on `Hosts`.
2. Add a **new host to Cluster**.
3. Add a Host to cluster wizard.
4. Enter the IP address or FQDN for PDI server.
5. Install JDK and JCL Policy both as we have `kerberos` enabled cluster.
6. Complete the wizard, and you will see the host added to the cloudera manager. Just install all the components without setting any roles to the host.
7. You will see that the host will have no roles assigned to it.

Now we are ready to make PDI server a gateway.

Steps to create a Gateway in Cloudera Manager. We will take `hdfs` as an example. But the steps are identical for all other services.

NOTE : we can add all the roles to the host, and `deploy-client` configuration all at once.

1. Login as an Admin user.
2. Got to the service we want to create a Gateway for, we can create gateway for `hdfs`, `yarn`, `hbase`, `hive`, `impala`.
3. Click `hdfs` > `Instances` tab > `Add Role to Instance` button > Select `gateway` and add the server in the text box.
4. `Continue` and `Finish`, this will add the configuration information in the cloudera manager, but next we need to `deploy-client` configuration
5. And we are done.

Testing Access to Hadoop from ETL PDI Server.

Now we have make the server a gateway, now lets test the configuration works.

Login to the hadoop user.

```
[root@pentaho-server ~]# su hadoop-user
[hadoop-user@pentaho-server ~]$ cd ~
[hadoop-user@pentaho-server ~]$ pwd
/home/hadoop-user
```

Check if we are able to access the HDFS directories. All good.

```
[hadoop-user@pentaho-server ~]$ hadoop fs -ls /
Found 6 items
drwxr-xr-x   - hdfs  supergroup          0 2015-05-29 15:32 /benchmarks
drwxr-xr-x   - hbase hbase               0 2015-09-14 15:10 /hbase
drwxrwxr-x   - solr  solr                0 2015-05-29 11:49 /solr
drwxrwxrwx   - hdfs  supergroup          0 2015-09-14 17:12 /tmp
drwxr-xr-x   - hdfs  supergroup          0 2015-05-29 16:22 /use
drwxrwxr-x   - hdfs  supergroup          0 2015-09-15 16:30 /user
```

Lets run a test Job, Should run successfully.

```
[hadoop-user@pentaho-server pentaho_test]$ hadoop jar \
            /opt/cloudera/parcels/CDH/lib/hadoop-0.20-mapreduce/hadoop-examples.jar pi 10 1000
...
verbose
...
MR JOB Completed successfully
```

Once we know that we are able to running the hadoop command and run a MR Job, we are done.

Updating PDI configuration to communicate to Hadoop Setup.

Before we go into this we need to get the configuration from `clouder-manager`.

1. Goto `hdfs` server.
2. On the right top corner Click `Action` > `Download Client Configuration`

Do this for `hdfs`, `hive`, `yarn`. Then move these files to PDI server, so that we can configure it.

Now lets continue to work on updating the config on PDI server.

```
[root@pentaho-server opt]# ls -l
total 12
drwxr-xr-x   4 cloudera-scm cloudera-scm 4096 Sep 14 13:12 cloudera
drwxr-xr-x  12 root         root         4096 Sep 14 11:43 data-integration
lrwxrwxrwx   1 root         root           16 Sep  7 14:37 kettle -> data-integration
lrwxrwxrwx   1 root         root           16 Sep 15 10:17 pentaho -> data-integration
drwxr-xr-x.  2 root         root         4096 May 17  2013 rh
```

Change Directory to `cd /opt/pentaho/plugins/pentaho-big-data-plugin/`

```
[root@pentaho-server opt]# cd /opt/pentaho/plugins/pentaho-big-data-plugin/
[root@pentaho-server pentaho-big-data-plugin]# ls -l
total 96640
drwxr-xr-x 7 root root     4096 Jun 14 14:35 hadoop-configurations
drwxr-xr-x 2 root root     4096 Jun 14 14:35 lib
-rw-r--r-- 1 root root   901809 Jun 14 12:55 pentaho-big-data-plugin-5.4.0.1-130.jar
-rw-r--r-- 1 root root 98034663 Jun 14 12:55 pentaho-mapreduce-libraries.zip
-rw-r--r-- 1 root root     2383 Sep 15 10:01 plugin.properties
drwxr-xr-x 2 root root     4096 Jun 14 12:55 plugins
```

Update the `plugin.properties` with `active.hadoop.configuration=cdh53` leave the other parameters as it is.

We are using `cdh53` as we are using `cdh5.4` in our setup. `cdh53` works for `cdh5.3` and `cdh5.4`.

```
[root@pentaho-server pentaho-big-data-plugin]# cat plugin.properties
# The Hadoop Configuration to use when communicating with a Hadoop cluster.
#        This is used for all Hadoop client tools
# including HDFS, Hive, HBase, and Sqoop.
# For more configuration options specific to the Hadoop configuration choosen
# here see the config.properties file in that configuration's directory.
active.hadoop.configuration=cdh53

# Path to the directory that contains the available Hadoop configurations
hadoop.configurations.path=hadoop-configurations
```

```
# Version of Kettle to use from the Kettle HDFS installation directory.
#                   This can be set globally here or overridden per job
# as a User Defined property. If not set we will use the version of
#                    Kettle that is used to submit the Pentaho MapReduce job.
pmr.kettle.installation.id=

# Installation path in HDFS for the Pentaho MapReduce Hadoop Distribution
# The directory structure should follow this structure where {version}
#                     can be configured through the Pentaho MapReduce
# User Defined properties as kettle.runtime.version
#
pmr.kettle.dfs.install.dir=/opt/pentaho/mapreduce

# Enables the use of Hadoop's Distributed Cache to
#               store the Kettle environment required to execute Pentaho MapReduce
# If this is disabled you must configure all TaskTracker
#               nodes with the Pentaho for Hadoop Distribution
# @deprecated This is deprecated and is provided as a migration path for existing installations.
pmr.use.distributed.cache=true

# Pentaho MapReduce runtime archive to be preloaded into
#               kettle.hdfs.install.dir/pmr.kettle.installation.id
pmr.libraries.archive.file=pentaho-mapreduce-libraries.zip

# Additional plugins to be copied when Pentaho MapReduce's
#               Kettle Environment does not exist on DFS. This should be a comma-separated
# list of plugin folder names to copy.
# e.g. pmr.kettle.additional.plugins=my-test-plugin,steps/DummyPlugin
pmr.kettle.additional.plugins=
```

Now lets go into `hadoop-configurations > cdh53` to update configuration from `cloudera`.

```
[root@pentaho-server pentaho-big-data-plugin]# ls
hadoop-configurations  lib  pentaho-big-data-plugin-5.4.0.1-130.jar
pentaho-mapreduce-libraries.zip  plugin.properties  plugins
[root@pentaho-server pentaho-big-data-plugin]# cd hadoop-configurations/
[root@pentaho-server hadoop-configurations]# ls -l
total 20
drwxr-xr-x 3 root root 4096 Sep 16 11:46 cdh53
drwxr-xr-x 3 root root 4096 Jun 14 14:35 emr34
drwxr-xr-x 3 root root 4096 Sep 15 10:01 hadoop-20
drwxr-xr-x 3 root root 4096 Jun 14 14:35 hdp22
drwxr-xr-x 3 root root 4096 Jun 14 14:35 mapr401
[root@pentaho-server hadoop-configurations]# cd cdh53/
```

In `cdh53` update all the configuration with `cloudera-manager` configurations.

```
[root@pentaho-server cdh53]# ls -l
total 212
-rw-r--r-- 1 root root    835 Jun 14 12:49 config.properties
-rwxr-xr-x 1 root root   3798 Sep 16 11:16 core-site.xml
-rwxr-xr-x 1 root root   2546 Sep 16 11:16 hadoop-env.sh
-rwxr-xr-x 1 root root   3568 Sep 16 11:16 hdfs-site.xml
```

```
-rwxr-xr-x 1 root root   1126 Sep 16 11:46 hive-env.sh
-rwxr-xr-x 1 root root   2785 Sep 16 11:46 hive-site.xml
drwxr-xr-x 4 root root   4096 Jun 14 14:35 lib
-rwxr-xr-x 1 root root    314 Sep 16 11:16 log4j.properties
-rwxr-xr-x 1 root root   4678 Sep 16 11:16 mapred-site.xml
-rw-r--r-- 1 root root 128746 Jun 14 12:49 pentaho-hadoop-shims-cdh53-54.2015.06.01.jar
-rw-r--r-- 1 root root   6871 Jun 14 12:49 pentaho-hadoop-shims-cdh53-hbase-comparators-54.jar
-rwxr-xr-x 1 root root    315 Sep 16 11:16 ssl-client.xml
-rwxr-xr-x 1 root root   1141 Sep 16 11:16 topology.map
-rwxr-xr-x 1 root root   1510 Sep 16 11:16 topology.py
-rwxr-xr-x 1 root root   6208 Sep 16 11:16 yarn-site.xml
[root@pentaho-server cdh53]#
```

Add the following files as below. You can get these files from the `clouder-manager`.

Example : On Cloudera Manager, `Clusters` -> `Services` -> `hdfs` -> `Action` (On the right top corner) -> `download-client` configuration. Do the same for `yarn` and other services.

From `hdfs` configuration.

```
core-site.xml
hadoop-env.sh
hdfs-site.xml
log4j.properties
ssl-client.xml
topology.map
topology.py
```

From `hive` configuration.

```
hive-env.sh
hive-site.xml
```

From `yarn` configuration.

```
mapred-site.xml
yarn-site.xml
```

We are done with configuration, now lets test it.

Testing PDI Hadoop.

We will do a simple test by copying file from local drive to Hadoop cluster.

`load_hdfs` job information. Its a simple Job to test connectivity between PDI and Hadoop cluster.

Important part of the file is where we give the location for `src` and `dest`.

```
<field>
    <source_filefolder>
        file:///home/hadoop-user/pentaho_test/update_raw.txt
    </source_filefolder>
    <destination_filefolder>
        hdfs://nameservice-ha/user/hadoop-user/weblogs/raw
    </destination_filefolder>
    <wildcard>^.*\.txt</wildcard>
</field>
```

Here is the complete file `load_hdfs.kjb`.

```
[hadoop-user@pentaho-server pentaho_test]$ cat load_hdfs.kjb
<?xml version="1.0" encoding="UTF-8"?>
<job>
   <name>load_hdfs</name>
   <description />
   <extended_description />
   <job_version />
   <job_status>0</job_status>
   <directory>/</directory>
   <created_user>-</created_user>
   <created_date>2011/12/19 13:48:36.419</created_date>
   <modified_user>-</modified_user>
   <modified_date>2011/12/19 13:48:36.419</modified_date>
   <parameters />
   <slaveservers />
   <job-log-table>
      <connection />
      <schema />
      <table />
      <size_limit_lines />
      <interval />
      <timeout_days />
      <field>
         <id>ID_JOB</id>
         <enabled>Y</enabled>
         <name>ID_JOB</name>
      </field>
      <field>
         <id>CHANNEL_ID</id>
         <enabled>Y</enabled>
         <name>CHANNEL_ID</name>
      </field>
      <field>
         <id>JOBNAME</id>
         <enabled>Y</enabled>
         <name>JOBNAME</name>
      </field>
      <field>
         <id>STATUS</id>
         <enabled>Y</enabled>
         <name>STATUS</name>
      </field>
      <field>
         <id>LINES_READ</id>
         <enabled>Y</enabled>
         <name>LINES_READ</name>
      </field>
      <field>
         <id>LINES_WRITTEN</id>
         <enabled>Y</enabled>
         <name>LINES_WRITTEN</name>
      </field>
      <field>
```

```
      <id>LINES_UPDATED</id>
      <enabled>Y</enabled>
      <name>LINES_UPDATED</name>
    </field>
    <field>
      <id>LINES_INPUT</id>
      <enabled>Y</enabled>
      <name>LINES_INPUT</name>
    </field>
    <field>
      <id>LINES_OUTPUT</id>
      <enabled>Y</enabled>
      <name>LINES_OUTPUT</name>
    </field>
    <field>
      <id>LINES_REJECTED</id>
      <enabled>Y</enabled>
      <name>LINES_REJECTED</name>
    </field>
    <field>
      <id>ERRORS</id>
      <enabled>Y</enabled>
      <name>ERRORS</name>
    </field>
    <field>
      <id>STARTDATE</id>
      <enabled>Y</enabled>
      <name>STARTDATE</name>
    </field>
    <field>
      <id>ENDDATE</id>
      <enabled>Y</enabled>
      <name>ENDDATE</name>
    </field>
    <field>
      <id>LOGDATE</id>
      <enabled>Y</enabled>
      <name>LOGDATE</name>
    </field>
    <field>
      <id>DEPDATE</id>
      <enabled>Y</enabled>
      <name>DEPDATE</name>
    </field>
    <field>
      <id>REPLAYDATE</id>
      <enabled>Y</enabled>
      <name>REPLAYDATE</name>
    </field>
    <field>
      <id>LOG_FIELD</id>
      <enabled>Y</enabled>
      <name>LOG_FIELD</name>
    </field>
  </job-log-table>
```

```
<jobentry-log-table>
   <connection />
   <schema />
   <table />
   <timeout_days />
   <field>
      <id>ID_BATCH</id>
      <enabled>Y</enabled>
      <name>ID_BATCH</name>
   </field>
   <field>
      <id>CHANNEL_ID</id>
      <enabled>Y</enabled>
      <name>CHANNEL_ID</name>
   </field>
   <field>
      <id>LOG_DATE</id>
      <enabled>Y</enabled>
      <name>LOG_DATE</name>
   </field>
   <field>
      <id>JOBNAME</id>
      <enabled>Y</enabled>
      <name>TRANSNAME</name>
   </field>
   <field>
      <id>JOBENTRYNAME</id>
      <enabled>Y</enabled>
      <name>STEPNAME</name>
   </field>
   <field>
      <id>LINES_READ</id>
      <enabled>Y</enabled>
      <name>LINES_READ</name>
   </field>
   <field>
      <id>LINES_WRITTEN</id>
      <enabled>Y</enabled>
      <name>LINES_WRITTEN</name>
   </field>
   <field>
      <id>LINES_UPDATED</id>
      <enabled>Y</enabled>
      <name>LINES_UPDATED</name>
   </field>
   <field>
      <id>LINES_INPUT</id>
      <enabled>Y</enabled>
      <name>LINES_INPUT</name>
   </field>
   <field>
      <id>LINES_OUTPUT</id>
      <enabled>Y</enabled>
      <name>LINES_OUTPUT</name>
   </field>
```

```
    <field>
      <id>LINES_REJECTED</id>
      <enabled>Y</enabled>
      <name>LINES_REJECTED</name>
    </field>
    <field>
      <id>ERRORS</id>
      <enabled>Y</enabled>
      <name>ERRORS</name>
    </field>
    <field>
      <id>RESULT</id>
      <enabled>Y</enabled>
      <name>RESULT</name>
    </field>
    <field>
      <id>NR_RESULT_ROWS</id>
      <enabled>Y</enabled>
      <name>NR_RESULT_ROWS</name>
    </field>
    <field>
      <id>NR_RESULT_FILES</id>
      <enabled>Y</enabled>
      <name>NR_RESULT_FILES</name>
    </field>
    <field>
      <id>LOG_FIELD</id>
      <enabled>N</enabled>
      <name>LOG_FIELD</name>
    </field>
    <field>
      <id>COPY_NR</id>
      <enabled>N</enabled>
      <name>COPY_NR</name>
    </field>
  </jobentry-log-table>
  <channel-log-table>
    <connection />
    <schema />
    <table />
    <timeout_days />
    <field>
      <id>ID_BATCH</id>
      <enabled>Y</enabled>
      <name>ID_BATCH</name>
    </field>
    <field>
      <id>CHANNEL_ID</id>
      <enabled>Y</enabled>
      <name>CHANNEL_ID</name>
    </field>
    <field>
      <id>LOG_DATE</id>
      <enabled>Y</enabled>
      <name>LOG_DATE</name>
```

```
    </field>
    <field>
      <id>LOGGING_OBJECT_TYPE</id>
      <enabled>Y</enabled>
      <name>LOGGING_OBJECT_TYPE</name>
    </field>
    <field>
      <id>OBJECT_NAME</id>
      <enabled>Y</enabled>
      <name>OBJECT_NAME</name>
    </field>
    <field>
      <id>OBJECT_COPY</id>
      <enabled>Y</enabled>
      <name>OBJECT_COPY</name>
    </field>
    <field>
      <id>REPOSITORY_DIRECTORY</id>
      <enabled>Y</enabled>
      <name>REPOSITORY_DIRECTORY</name>
    </field>
    <field>
      <id>FILENAME</id>
      <enabled>Y</enabled>
      <name>FILENAME</name>
    </field>
    <field>
      <id>OBJECT_ID</id>
      <enabled>Y</enabled>
      <name>OBJECT_ID</name>
    </field>
    <field>
      <id>OBJECT_REVISION</id>
      <enabled>Y</enabled>
      <name>OBJECT_REVISION</name>
    </field>
    <field>
      <id>PARENT_CHANNEL_ID</id>
      <enabled>Y</enabled>
      <name>PARENT_CHANNEL_ID</name>
    </field>
    <field>
      <id>ROOT_CHANNEL_ID</id>
      <enabled>Y</enabled>
      <name>ROOT_CHANNEL_ID</name>
    </field>
  </channel-log-table>
  <pass_batchid>N</pass_batchid>
  <shared_objects_file />
  <entries>
    <entry>
      <name>START</name>
      <description />
      <type>SPECIAL</type>
      <start>Y</start>
```

```
      <dummy>N</dummy>
      <repeat>N</repeat>
      <schedulerType>0</schedulerType>
      <intervalSeconds>0</intervalSeconds>
      <intervalMinutes>60</intervalMinutes>
      <hour>12</hour>
      <minutes>0</minutes>
      <weekDay>1</weekDay>
      <DayOfMonth>1</DayOfMonth>
      <parallel>N</parallel>
      <draw>Y</draw>
      <nr>0</nr>
      <xloc>68</xloc>
      <yloc>88</yloc>
   </entry>
   <entry>
      <name>Copy Files</name>
      <description />
      <type>COPY_FILES</type>
      <copy_empty_folders>Y</copy_empty_folders>
      <arg_from_previous>N</arg_from_previous>
      <overwrite_files>N</overwrite_files>
      <include_subfolders>N</include_subfolders>
      <remove_source_files>N</remove_source_files>
      <add_result_filesname>N</add_result_filesname>
      <destination_is_a_file>N</destination_is_a_file>
      <create_destination_folder>Y</create_destination_folder>
      <fields>
         <field>
            <source_filefolder>
                 file:///home/hadoop-user/pentaho_test/update_raw.txt
            </source_filefolder>
            <destination_filefolder>
                 hdfs://nameservice-ha/user/hadoop-user/weblogs/raw
             </destination_filefolder>
            <wildcard>^.*\.txt</wildcard>
         </field>
      </fields>
      <parallel>N</parallel>
      <draw>Y</draw>
      <nr>0</nr>
      <xloc>191</xloc>
      <yloc>88</yloc>
   </entry>
</entries>
<hops>
   <hop>
      <from>START</from>
      <to>Copy Files</to>
      <from_nr>0</from_nr>
      <to_nr>0</to_nr>
      <enabled>Y</enabled>
      <evaluation>Y</evaluation>
      <unconditional>Y</unconditional>
   </hop>
```

```
   </hops>
   <notepads />
</job>
[hadoop-user@pentaho-server pentaho_test]$
```

Lets create a location for the file to be put.

```
[hadoop-user@pentaho-server pentaho_test]$ hadoop fs -mkdir -p /user/hadoop-user/weblogs/raw
[hadoop-user@pentaho-server pentaho_test]$ ls -l
total 83124
-rw-r--r-- 1 hadoop-user hadoop-user     6160 Sep 16 11:26 load_hdfs.kjb
-rwxr-xr-x 1 hadoop-user hadoop-user 77908174 Jan 10  2012 weblogs_rebuild.txt
```

We have kettle job will will load the `weblogs_rebuild.txt` file to Hadoop cluster, location created earlier `/user/hadoop-user/weblogs/raw`.

```
[hadoop-user@pentaho-server pentaho_test]$ sh /opt/pentaho/kitchen.sh -file=/home/hadoop-user/pentaho_te
2015/09/16 11:26:37 - Kitchen - Start of run.
2015/09/16 11:26:38 - load_hdfs - Start of job execution
2015/09/16 11:26:38 - load_hdfs - Starting entry [Copy Files]
2015/09/16 11:26:38 - Copy Files - Starting ...
2015/09/16 11:26:38 - Copy Files - Processing row source
        File/folder source : [file:///home/hadoop-user/pentaho_test/weblogs_rebuild.txt] ...
        destination file/folder : [hdfs://nameservice-ha/user/hadoop-user/weblogs/raw]...
        wildcard : [^.*\.txt]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in
    [jar:file:/opt/data-integration/launcher/../lib/slf4j-log4j12-1.7.5.jar!\
                                                /org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/data-integration/plugins/pentaho-big-data-plugin/\
                         lib/slf4j-log4j12-1.7.3.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2015/09/16 11:26:40 - load_hdfs - Finished job entry [Copy Files] (result=[true])
2015/09/16 11:26:40 - load_hdfs - Job execution finished
2015/09/16 11:26:40 - Kitchen - Finished!
2015/09/16 11:26:40 - Kitchen - Start=2015/09/16 11:26:37.578, Stop=2015/09/16 11:26:40.932
2015/09/16 11:26:40 - Kitchen - Processing ended after 3 seconds.
```

Job completed successfully, now lets check the hadoop location.

```
[hadoop-user@pentaho-server pentaho_test]$ hadoop fs -ls  /user/hadoop-user/weblogs/raw
Found 1 items
-rw-r--r--   3 hadoop-user hadoop-user   77908174 2015-09-16 11:26 \
                                            /user/hadoop-user/weblogs/raw/weblogs_rebuild.txt
[hadoop-user@pentaho-server pentaho_test]$ ls -l
```

Looks like we have the file copied to HDFS. So we are good. Few links which are helpful.

```
http://wiki.pentaho.com/display/BAD/Configuring+Pentaho+for+your+Hadoop+Distro+and+Version
http://wiki.pentaho.com/display/BAD/Additional+Configuration+for+YARN+Shims
```