Talend Open Studio for Big Data

Getting Started Guide

6.3.0

Adapted for v6.3.0. Supersedes previous releases.

Publication date: October 27, 2016

Copyleft

This documentation is provided under the terms of the Creative Commons Public License (CCPL).

For more information about what you can and cannot do with this documentation in accordance with the CCPL, please read: http://creativecommons.org/licenses/by-nc-sa/2.0/

Notices

Talend is a trademark of Talend, Inc.

All brands, product names, company names, trademarks and service marks are the properties of their respective owners.

License Agreement

The software described in this documentation is licensed under the Apache License, Version 2.0 (the "License"); you may not use this software except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0.html. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

This product includes software developed at AOP Alliance (Java/J2EE AOP standards), ASM, Amazon, AntlR, Apache ActiveMQ, Apache Ant, Apache Avro, Apache Axiom, Apache Axis, Apache Axis 2, Apache Batik, Apache CXF, Apache Cassandra, Apache Chemistry, Apache Common Http Client, Apache Common Http Core, Apache Commons, Apache Commons Bcel, Apache Commons JxPath, Apache Commons Lang, Apache Datafu, Apache Derby Database Engine and Embedded JDBC Driver, Apache Geronimo, Apache HCatalog, Apache Hadoop, Apache Hbase, Apache Hive, Apache HttpClient, Apache HttpComponents Client, Apache JAMES, Apache Log4j, Apache Lucene Core, Apache Neethi, Apache Oozie, Apache POI, Apache Parquet, Apache Pig, Apache PiggyBank, Apache ServiceMix, Apache Sqoop, Apache Thrift, Apache Tomcat, Apache Velocity, Apache WSS4J, Apache WebServices Common Utilities, Apache Xml-RPC, Apache Zookeeper, Box Java SDK (V2), CSV Tools, Cloudera HTrace, ConcurrentLinkedHashMap for Java, Couchbase Client, DataNucleus, DataStax Java Driver for Apache Cassandra, Ehcache, Ezmorph, Ganymed SSH-2 for Java, Google APIs Client Library for Java, Google Gson, Groovy, Guava: Google Core Libraries for Java, H2 Embedded Database and JDBC Driver, Hector: A high level Java client for Apache Cassandra, Hibernate BeanValidation API, Hibernate Validator, HighScale Lib, HsqlDB, Ini4j, JClouds, JDO-API, JLine, JSON, JSR 305: Annotations for Software Defect Detection in Java, JUnit, Jackson Java JSON-processor, Java API for RESTful Services, Java Agent for Memory Measurements, Jaxb, Jaxen, JetS3T, Jettison, Jetty, Joda-Time, Json Simple, LZ4: Extremely Fast Compression algorithm, LightCouch, MetaStuff, Metrics API, Metrics Reporter Config, Microsoft Azure SDK for Java, Mondrian, MongoDB Java Driver, Netty, Ning Compression codec for LZF encoding, OpenSAML, Paraccel JDBC Driver, Parboiled, PostgreSQL JDBC Driver, Protocol Buffers - Google's data interchange format, Resty: A simple HTTP REST client for Java, Riak Client, Rocoto, SDSU Java Library, SL4J: Simple Logging Facade for Java, SQLite JDBC Driver, Scala Lang, Simple API for CSS, Snappy for Java a fast compressor/decompresser, SpyMemCached, SshJ, StAX API, StAXON - JSON via StAX, Super SCV, The Castor Project, The Legion of the Bouncy Castle, Twitter4J, Uuid, W3C, Windows Azure Storage libraries for Java, Woden, Woodstox: High-performance XML processor, Xalan-J, Xerces2, XmlBeans, XmlSchema Core, Xmlsec - Apache Santuario, YAML parser and emitter for Java, Zip4J, atinject, dropbox-sdk-java: Java library for the Dropbox Core API, google-guice. Licensed under their respective license.

Table of Contents

Chapter 1. Introduction to Talend Open Studio for Big Data

Talend Open Studio for Big Data provides unified development and management tools to integrate and process all of your data with an easy to use, visual designer.

Built on top of *Talend* data integration solutions, *Talend* Big Data solutions provide a powerful tool set that enables users to access, transform, move and synchronize big data by leveraging the Apache Hadoop Big Data Platform and makes the Hadoop platform ever so easy to use.

1.1. Functional architecture of Talend Big Data solutions

The Talend Open Studio for Big Data functional architecture is an architectural model that identifies Talend Open Studio for Big Data functions, interactions and corresponding IT needs. The overall architecture has been described by isolating specific functionalities in functional blocks.

The following chart illustrates the main architectural functional blocks.
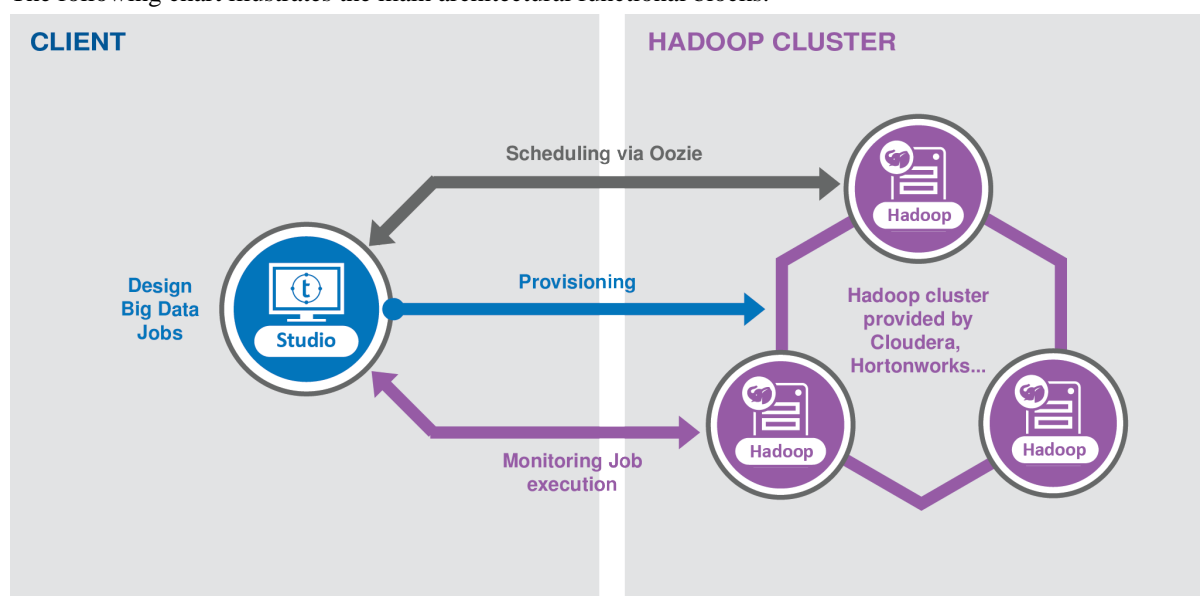
The different types of functional blocks are:

- From *Talend Studio*, you design and launch Big Data Jobs that leverage a Hadoop cluster to handle large data sets. Once launched, these Jobs are sent to, deployed on and executed on this Hadoop cluster.

- The Oozie workflow scheduler system is integrated within the Studio through which you can deploy, schedule, and execute Big Data Jobs on a Hadoop cluster and monitor the execution status and results of these Jobs.

- A Hadoop cluster independent of the *Talend* system to handle large data sets.

Chapter 2. Prerequisites to using Talend products

This chapter provides basic software and hardware information required and recommended to get started with your *Talend* product:

- *Memory requirements*.

- *Software requirements*.

It also guides you to install and configure required and recommended third-party tools:

- *Installing Java*.

- *Setting up the Java environment variable on Windows* or *Setting up the Java environment variable on Linux*.

- *Installing 7-Zip (Windows)*.

To successfully install the software, you need administrative access to your computer. To get administrative access, contact your Administrator.

2.1. Memory requirements

To make the most out of your *Talend* product, please consider the following memory and disk space usage:

Memory usage	3GB minimum, 4 GB recommended
Disk space	3GB

2.2. Software requirements

To make the most out of your *Talend* product, please consider the following system and software requirements:

Required software

- Operating System for Talend Studio:

Support type	Operating System	Version	Processor
Recommended	Microsoft Windows Professional	7	64-bit
Recommended	Linux Ubuntu	14.04	64-bit
Supported	Apple OS X	El Capitan/10.11	64-bit
		Yosemite/10.10	64-bit
		Mavericks/10.9	64-bit

- Java 8 JRE Oracle. See *Installing Java*.

- A properly installed and configured Hadoop cluster.

  You have ensured that the client machine on which the *Talend Studio* is installed can recognize the host names of the nodes of the Hadoop cluster to be used. For this purpose, add the IP address/hostname mapping entries for the services of that Hadoop cluster in the *hosts* file of the client machine.

  For example, if the host name of the Hadoop Namenode server is *talend-cdh550.weave.local* and its IP address is *192.168.x.x*, the mapping entry reads *192.168.x.x talend-cdh550.weave.local*.

Optional software

- 7-Zip. See *Installing 7-Zip (Windows)*.

2.3. Installing Java

To use your *Talend* product, you need Oracle Java Runtime Environment installed on your computer.

1. From the Java SE Downloads page, under **Java Platform, Standard Edition**, click the **JRE Download**.

2. From the **Java SE Runtime Environment 8 Downloads** page, click the radio button to **Accept License Agreement**.

3. Select the appropriate download for your Operating System.

4. Follow the Oracle installation steps to install Java.

When Java is installed on your computer, you need to set up the `JAVA_HOME` environment variable. For more information, see:

- *Setting up the Java environment variable on Windows*.

- *Setting up the Java environment variable on Linux*.

2.4. Setting up the Java environment variable on Windows

Prior to installing your *Talend* product, you have to set the `JAVA_HOME` and `Path` environment variables:

1. Go to the **Start Menu** of your computer, right-click on **Computer** and select **Properties**.

2. In the **[Control Panel Home]** window, click **Advanced system settings**.

3. In the **[System Properties]** window, click **Environment Variables...**.

4. Under **System Variables**, click **New...** to create a variable. Name the variable `JAVA_HOME`, enter the path to the Java 8 JRE, and click **OK**.

    Example of default JRE path: *C:\Program Files\Java\jre1.8.0_77*.

5. Under **System Variables**, select the **Path** variable and click **Edit...** to add the previously defined `JAVA_HOME` variable at the end of the `Path` environment variable, separated with semi colon.

    Example: *`<PathVariable>;%JAVA_HOME%\bin`*.

2.5. Setting up the Java environment variable on Linux

Prior to installing your *Talend* product, you have to set the `JAVA_HOME` and `Path` environment variables:

1. Find the JRE installation home directory.

    Example: `/usr/lib/jvm/jre1.8.0_65`

2. Export it in the `JAVA_HOME` environment variable.

    Example:

    ```
    export JAVA_HOME=/usr/lib/jvm/jre1.8.0_65
    export PATH=$JAVA_HOME/bin:$PATH
    ```

3. Add these lines at the end of the user profiles in the *~/.profile* file or, as a superuser, at the end of the global profiles in the */etc/profile* file.

4. Log on again.

2.6. Installing 7-Zip (Windows)

Talend recommends to install 7-Zip and to use it to extract the installation files: http://www.7-zip.org/download.html.

1.   Download the 7-Zip installer corresponding to your Operating System.

2.   Navigate to your local folder, locate and double-click the 7z exe file to install it.

The download will start automatically.

1.   Download the 7-Zip installer corresponding to your Operating System.

Chapter 3. Downloading and installing Talend Open Studio for Big Data

Talend Open Studio for Big Data is easy to install. After downloading it from *Talend*'s Website, a simple unzipping will install it on your computer.

This chapter provides basic information useful to download and install it.

3.1. Downloading Talend Open Studio for Big Data

Talend Open Studio for Big Data is a free open source product that you can download directly from *Talend*'s Website:

1. Go to *Talend Open Studio for Big Data* Download page.

2. Click **DOWNLOAD FREE TOOL**.

The download will start automatically.

3.2. Installing Talend Open Studio for Big Data

Installation is done by unzipping the TOS_BD zip file previously downloaded.

This can be done either by using:

*   7Zip (Windows recommended): *Extracting via 7-Zip (Windows recommended)*.

*   Windows default unzipper: *Extracting via Windows default unzipping tool*.

*   Linux default unzipper (for a Linux based Operating System): *Extracting via the Linux GUI unzipper*.

3.2.1. Extracting via 7-Zip (Windows recommended)

For Windows, *Talend* recommends you to install 7-Zip and use it to extract files. For more information, see *Installing 7-Zip (Windows)*.

To install the studio, follow the steps below:

1. Navigate to your local folder, locate the **TOS** zip file and move it to another location with a path as short as possible and without any space character.

    Example: *C:/Talend/*

2. Unzip it by right-clicking on the compressed file and selecting **7-Zip** > **Extract Here**.

3.2.2. Extracting via Windows default unzipping tool

If you do not want to use 7-Zip, you can use Windows default unzipping tool:

1. Unzip it by right-click the compressed file and select, **Extract All**.

2. Click on **Browse** and navigate to the *C: drive*.

3. Select **Make new folder** and name the folder *Talend*. Click **OK**.

4.   Click on **Extract** to begin the installation.

3.2.3. Extracting via the Linux GUI unzipper

To install the studio, follow the steps below:

1.   Navigate to your local folder, locate the **TOS** zip file and move it to another location with a path as short as possible and without any space character.

Example: *home/user/talend/*

2.   Unzip it by right-clicking on the compressed file and selecting **Extract Here**.

Chapter 4. Configuring and setting up your Talend product

This chapter provides basic information required to configure and set up your *Talend* product, including:

- *Launching the Studio for the first time*

- *Logging in to the Studio*

- *Installing additional packages*

- *Setting up Hadoop connection manually*

- *Setting up connection to HDFS*

- *Uploading files to HDFS*

- *Preparing file metadata*

4.1. Launching the Studio for the first time

The Studio installation directory contains binaries for several platforms including Mac OS X and Linux/Unix.

To open the *Talend Studio* for the first time, do the following:

1. Double-click the executable file corresponding to your operating system, for example:

    *   TOS_*-win-x86_64.exe, for Windows.

    *   TOS_*-linux-gtk-x86_64, for Linux.

    *   TOS_*-macosx-cocoa.app, for Mac.

2. In the **[User License Agreement]** dialog box that opens, read and accept the terms of the end user license agreement to proceed.

4.2. Logging in to the Studio

To log in to the *Talend Studio* for the first time, do the following:

1. In the *Talend Studio* login window, select **Create a new project**, specify the project name: *getting_started* and click **Finish** to create a new local project.

2. Depending on the product you are using, either of the following opens:

    *   the Quick Tour. Play it to get more information on the User Interface of the Studio, and click **Stop** to end it.

    *   the Welcome page. Follow the links to get more information about the Studio, and click **Start Now!** to close the page and continue opening the Studio.

Now you have successfully logged in to the *Talend Studio*. Next you need to install additional packages required for the *Talend Studio* to work properly.

4.3. Installing additional packages

Talend recommends that you install additional packages, including third-party libraries and database drivers, as soon as you log in to your *Talend Studio* to allow you to fully benefit from the functionalities of the Studio.

1. When the **[Additional Talend Packages]** wizard opens, install additional packages by selecting the **Required** and **Optional third-party libraries** check boxes and clicking **Finish**.

    This wizard opens each time you launch the studio if any additional package is available for installation unless you select the **Do not show this again** check box. You can also display this wizard by selecting **Help** > **Install Additional Packages** from the menu bar.

    For more information, see the section about installing additional packages in the *Talend Installation and Upgrade Guide*.

2. In the **[Download external modules]** window, click the **Accept all** button at the bottom of the wizard to accept all the licenses of the external modules used in the studio.

Depending on which libraries you selected, you may need to accept their license more than once.

Wait until all the libraries are installed before starting to use the studio.

3.   If required, restart your *Talend Studio* for certain additional packages to take effect.

4.4. Setting up Hadoop connection manually

Setting up the connection to a given Hadoop distribution in **Repository** allows you to avoid configuring that connection each time when you need to use the same Hadoop distribution.

Prerequisites:

• You have ensured that the client machine on which the *Talend Studio* is installed can recognize the host names of the nodes of the Hadoop cluster to be used. For this purpose, add the IP address/hostname mapping entries for the services of that Hadoop cluster in the *hosts* file of the client machine.

For example, if the host name of the Hadoop Namenode server is *talend-cdh550.weave.local* and its IP address is *192.168.x.x*, the mapping entry reads *192.168.x.x talend-cdh550.weave.local*.

• The Hadoop cluster to be used has been properly configured and is running.

The Cloudera Hadoop cluster to be used in this example is of the CDH V5.5 in the Yarn mode and applies the default configuration of the distribution without enabling the Kerberos security. For further information about the default configuration of the CDH V5.5 distribution, see Deploy CDH 5 on a cluster and Default ports used in CDH5.

1.   In the **Repository** tree view of your studio, expand **Metadata** and then right-click **Hadoop cluster**.

2.   Select **Create Hadoop cluster** from the contextual menu to open the **[Hadoop cluster connection]** wizard.

3.   Fill in generic information about this connection, such as **Name** and **Description** and click **Next** to open the **[Hadoop configuration import wizard]** wizard that helps you import the ready-for-use configuration if any.

4.   Select the **Enter manually Hadoop services** check box to manually enter the configuration information for the Hadoop connection being created.
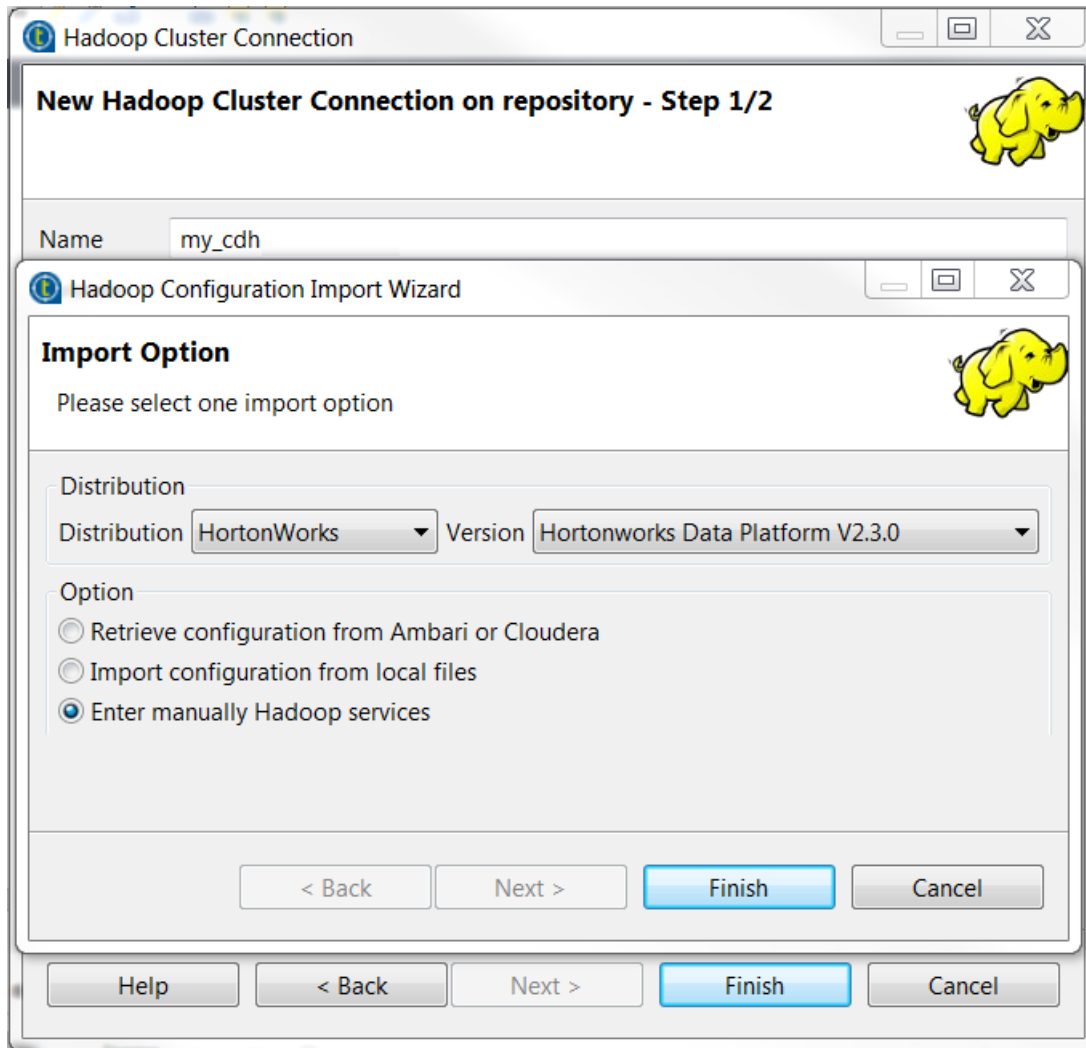
5. Click **Finish** to close this import wizard.

6. From the **Distribution** list, select **Cloudera** and then from the **Version** list, select **Cloudera CDH5.5 (YARN mode)**.

7. In the **Namenode URI** field, enter the URI pointing to the machine used as the NameNode service of the Cloudera Hadoop cluster to be used.

    The NameNode is the master node of a Hadoop system. For example, assume that you have chosen a machine called *machine1* as the NameNode, then the location to be entered is *hdfs://machine1:portnumber*.

    On the cluster side, the related property is specified in the configuration file called *core-site.xml*. If you do not know what URI is to be entered, check the *fs.defaultFS* property in the *core-site.xml* file of your cluster.

8. In the **Resource manager** field and the **Resource manager scheduler** field, enter the URIs pointing to these two services, respectively.

    On the cluster side, these two services share the same host machine but use different default port numbers. For example, if the machine hosting them is *resourcemanager.company.com*, the location of Resource manager is *resourcemanager.company.com:8032* and the location of Resource manager scheduler is *resourcemanager.company.com:8030*.

    If you do not know the name of the hosting machine of these services, check the *yarn.resourcemanager.hostname* property in the configuration file called *yarn-site.xml* of your cluster.

9. In the **Job history** field, enter the location of the JobHistory service. This service allows the metrics information of the current Job to be stored in the JobHistory server.

   The related property is specified in the configuration file called *mapred-site.xml* of your cluster. For the value you need to put in this field, check the *mapreduce.jobhistory.address* property in this *mapred-site.xml* file.

10. In the **Staging directory** field, enter this directory defined in your Hadoop cluster for temporary files created by running programs.

    The related property is specified in the *mapred-site.xml* file of your cluster. For further information, check the *yarn.app.mapreduce.am.staging-dir* property in this *mapred-site.xml* file.

11. Select the **Use datanode hostname** check box to allow the Studio to access each Datanode of your cluster via their host names.

    This actually sets the *dfs.client.use.datanode.hostname* property of your cluster to *true*.

12. In the **User name** field, enter the user authentication name you want the Studio to use to connect to your Hadoop cluster.

13. Since the Hadoop cluster to be connected to is using the default configuration, leave the other fields or check boxes in this wizard as they are because they are used to define any custom Hadoop configuration.

14. Click the **Check services** button to verify that the Studio can connect to the NameNode and the ResourceManager services you have specified.

    A dialog box pops up to indicate the checking process and the connection status.

    If the connection fails, you can click **Error log** at the end of each progress bar to diagnose the connection issues.

15. Once this check indicates that the connection is successful, click **Finish** to validate your changes and close the wizard.

The new connection, called *my_cdh* in this example, is displayed under the **Hadoop cluster** folder in the **Repository** tree view.

You can then continue to create the child connections to different Hadoop elements such as HDFS or Hive based on this connection.

4.5. Setting up connection to HDFS

A connection to HDFS in **Repository** allows you to reuse this connection in related Jobs.

Prerequisites:

• The connection to the Hadoop cluster hosting the HDFS system to be used has been set up from the **Hadoop cluster** node in the **Repository**.

   For further information about how to create this connection, see *Setting up Hadoop connection manually*.

• The Hadoop cluster to be used has been properly configured and is running and you have the proper access permission to that distribution and its HDFS.

• You have ensured that the client machine on which the *Talend Studio* is installed can recognize the host names of the nodes of the Hadoop cluster to be used. For this purpose, add the IP address/hostname mapping entries for the services of that Hadoop cluster in the *hosts* file of the client machine.
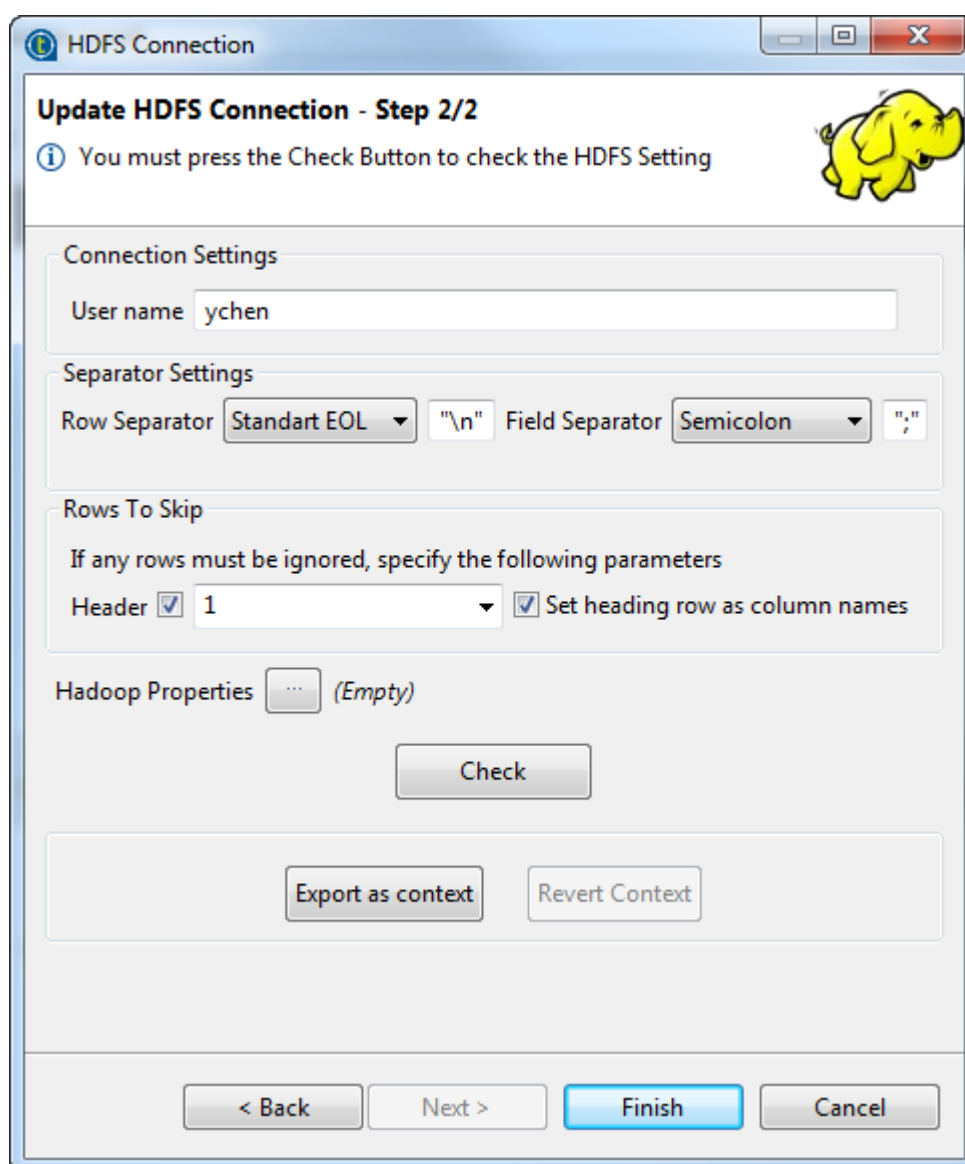
For example, if the host name of the Hadoop Namenode server is *talend-cdh550.weave.local* and its IP address is *192.168.x.x*, the mapping entry reads *192.168.x.x talend-cdh550.weave.local*.

1.   Expand the **Hadoop cluster** node under **Metadata** in the **Repository** tree view, right click the Hadoop connection to be used and select **Create HDFS** from the contextual menu.

2.   In the connection wizard that opens up, fill in the generic properties of the connection you need create, such as **Name**, **Purpose** and **Description**.

3.   Click **Next** when completed. The second step requires you to fill in the HDFS connection data.

The **User name** property is automatically pre-filled with the value inherited from the Hadoop connection you selected in the previous steps.

The **Row separator** and the **Field separator** properties are using the default values.

4. Select the **Set heading row as column names** check box to use the data in the heading rows of the HDFS file to be used to define the column names of this file.

   The **Header** check box is then automatically selected and the **Header** field is filled with *1*. This means that the first row of the file will be ignored as data body but used as column names of the file.

5. Click **Check** to verify your connection.

   A message pops up to indicate whether the connection is successful.

6. Click **Finish** to validate these changes.

The new HDFS connection is now available under the **Hadoop cluster** node in the **Repository** tree view. You can then use it to define and centralize the schemas of the files stored in the connected HDFS system in order to reuse these schemas in a *Talend* Job.

4.6. Uploading files to HDFS

Uploading a file to HDFS allows the Big Data Jobs to read and process it.

Prerequisites:

- The connection to the Hadoop cluster to be used and the connection to the HDFS system of this cluster have been set up from the **Hadoop cluster** node in the **Repository**.

  If you have not done so, see *Setting up Hadoop connection manually* and then *Setting up connection to HDFS* to create these connections.

- The Hadoop cluster to be used has been properly configured and is running and you have the proper access permission to that distribution and the HDFS folder to be used.

- You have ensured that the client machine on which the *Talend* Jobs are executed can recognize the host names of the nodes of the Hadoop cluster to be used. For this purpose, add the IP address/hostname mapping entries for the services of that Hadoop cluster in the *hosts* file of the client machine.

  For example, if the host name of the Hadoop Namenode server is *talend-cdh550.weave.local* and its IP address is *192.168.x.x*, the mapping entry reads *192.168.x.x talend-cdh550.weave.local*.

In this procedure, you will create a Job that writes data in the HDFS system of the Cloudera Hadoop cluster to which the connection has been set up in the **Repository** as explained in *Setting up Hadoop connection manually*. This data is needed for the use case described in . The files needed for the use case can be download here.

1. In the **Repository** tree view, right-click the **Job Designs** node and select **Create folder** from the contextual menu.

2. In the **[New Folder]** wizard, name your Job folder *getting_started* and click **Finish** to create your folder.

3. Right-click the *getting_started* folder and select **Create job** from the contextual menu.
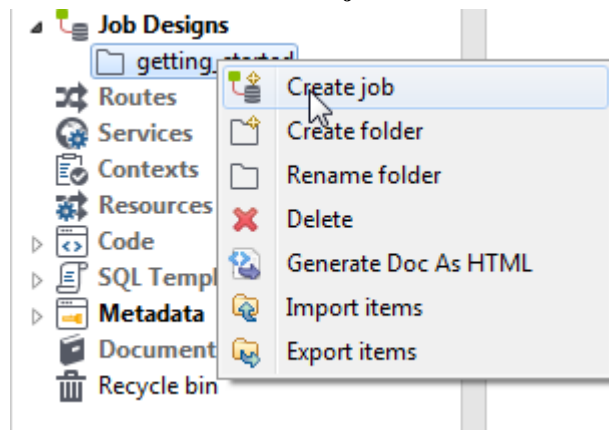
4. In the **[New Job]** wizard, enter a name for the Job to be created and other useful information.

   For example, enter *write_to_hdfs* in the **Name** field.

   In this step of the wizard, **Name** is the only mandatory field. The information you provide in the **Description** field will appear as a tooltip when you move your mouse pointer over the Job in the **Repository** tree view.

5. Click **Finish** to create your Job.

   An empty Job is opened in the Studio.

6. Expand the **Hadoop cluster** node under **Metadata** in the **Repository** tree view.

7. Expand the Hadoop connection you have created and then the **HDFS** folder under it. In this example, it is the *my_cdh* Hadoop connection.

8. Drop the HDFS connection from the **HDFS** folder into the workspace of the Job you are creating. This connection is *cdh_hdfs* in this example.

The **[Components]** window is displayed to show all the components that can directly reuse this HDFS connection in a Job.
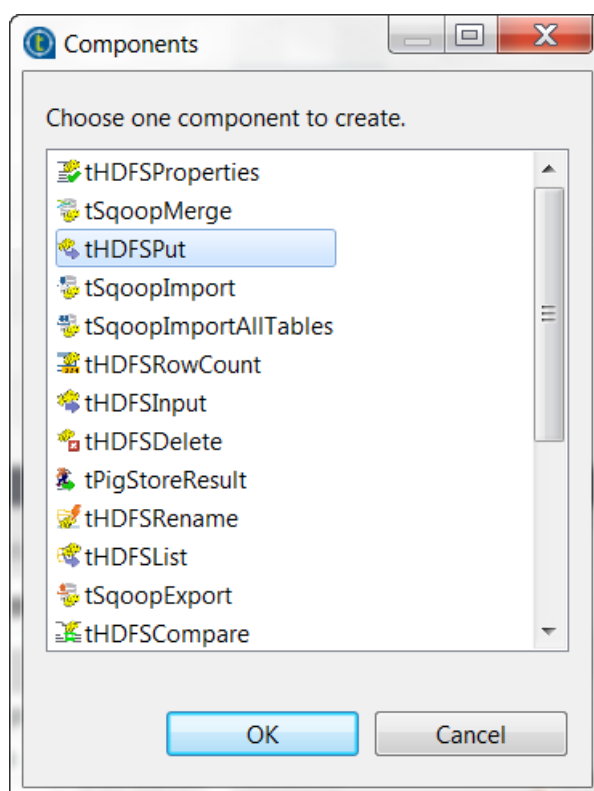
9. Select **tHDFSPut** and click **OK** to validate your choice.

This **[Components]** window is closed and a **tHDFSPut** component is automatically placed in the workspace of the current Job, with this component having been labelled using the name of the HDFS connection mentioned in the previous step.

10. Double-click **tHDFSPut** to open its **Component** view.

The connection to the HDFS system to be used has been automatically configured by using the configuration of the HDFS connection you have set up and stored in the **Repository**. The related parameters in this tab therefore becomes read-only. These parameters are: **Distribution**, **Version**, **NameNode URI**, **Use Datanode Hostname**, **User kerberos authentication** and **Username**.

11. In the **Local directory** field, enter the path, or browse to the folder in which the files to be copied to HDFS are stored.

    The files about movies and their directors are stored in this directory.

12. In the **HDFS directory** field, enter the path, or browse to the target directory in HDFS to store the files.

    This directory is created on the fly if it does not exist.

13. From the **Overwrite file** drop-down list, select **always** to overwrite the files if they already exist in the target directory in HDFS.

14. In the **Files** table, add one row by clicking the **[+]** button in order to define the criteria to select the files to be copied.

15. In the **Filemask** column, enter an asterisk (*) within the double quotation marks to make **tHDFSPut** select all the files stored in the folder you specified in the **Local directory** field.
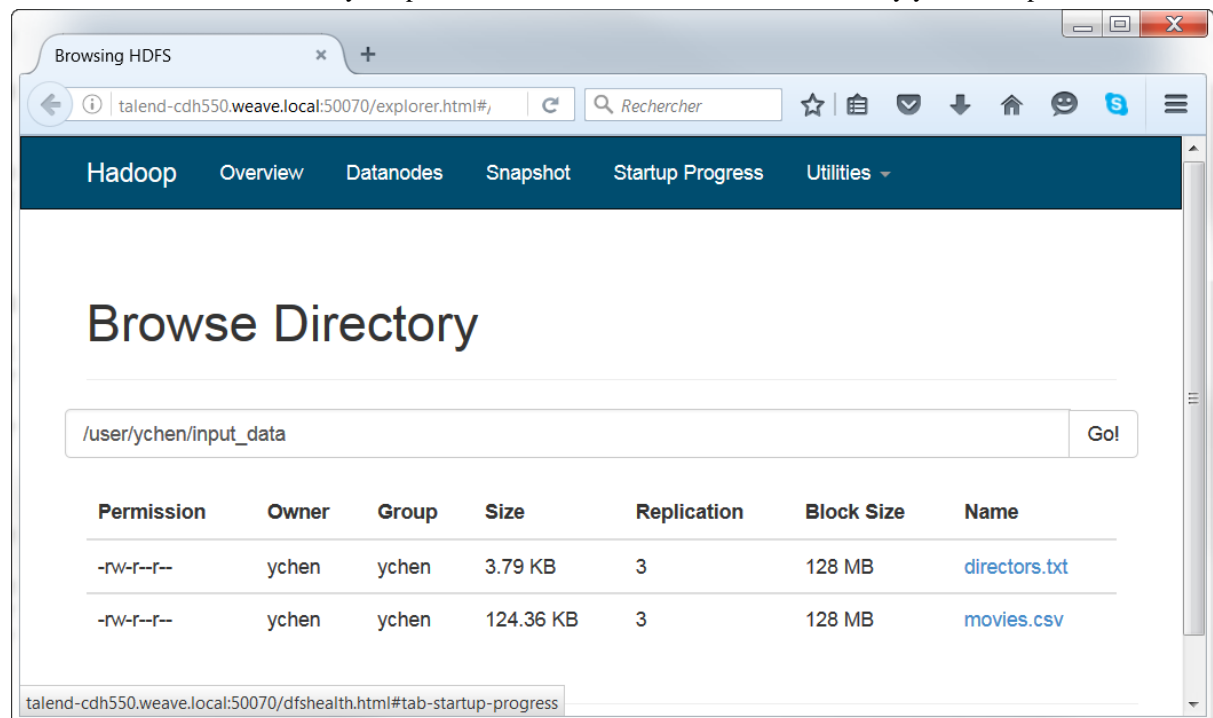
16. Leave the **New name** column empty, that is to say, keep the default double quotation marks as is, so as to make the name of the files unchanged after being uploaded.

17. Press **F6** to run the Job.

    The **Run** view is opened automatically. It shows the progress of this Job.
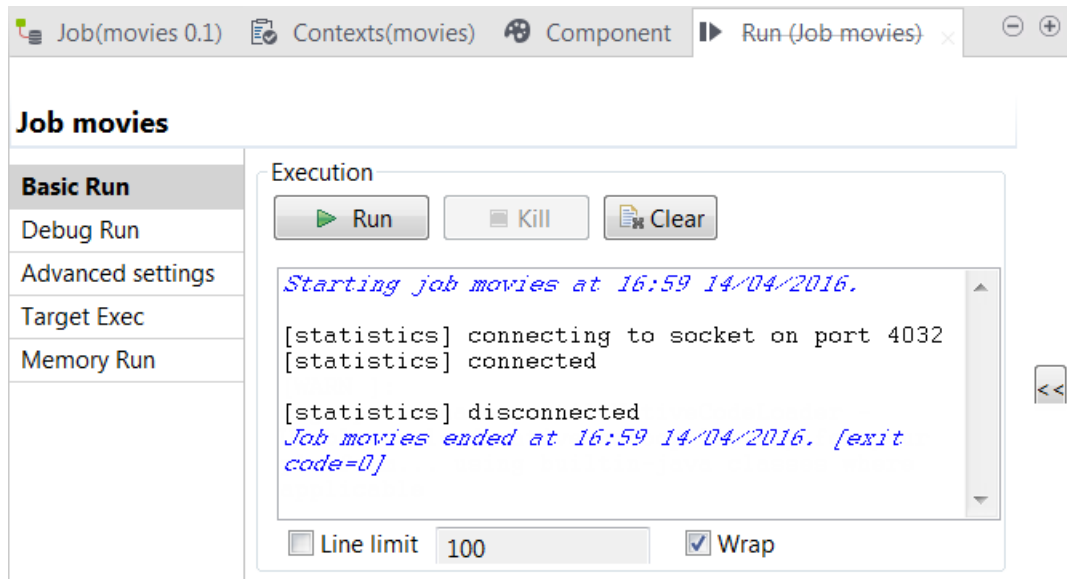
When the Job is done, the files you uploaded can be found in HDFS in the directory you have specified.

4.7. Preparing file metadata

In the **Repository**, setting up the metadata of a file stored in HDFS allows you to directly reuse its schema in a related Big Data component without having to define each related parameter manually.

Prerequisites:

• You have launched your **Talend Studio** and opened the **Integration** perspective.

• The source files *movies.csv* and *directors.txt* have been uploaded into HDFS as explained in *Uploading files to HDFS*.

- The connection to the Hadoop cluster to be used and the connection to the HDFS system of this cluster have been set up from the **Hadoop cluster** node in the **Repository**.

  If you have not done so, see *Setting up Hadoop connection manually* and then *Setting up connection to HDFS* to create these connections.

- The Hadoop cluster to be used has been properly configured and is running and you have the proper access permission to that distribution and the HDFS folder to be used.

- You have ensured that the client machine on which the *Talend Studio* is installed can recognize the host names of the nodes of the Hadoop cluster to be used. For this purpose, add the IP address/hostname mapping entries for the services of that Hadoop cluster in the *hosts* file of the client machine.

  For example, if the host name of the Hadoop Namenode server is *talend-cdh550.weave.local* and its IP address is *192.168.x.x*, the mapping entry reads *192.168.x.x talend-cdh550.weave.local*.

Since the *movies.csv* file you need to process has been stored in the HDFS system being used, you can retrieve its schema to set up its metadata in the **Repository**.

The schema of the *directors.txt* file can also be retrieved, but is intentionally ignored in the retrieval procedure explained below, because in this scenario, this *directors.txt* file is used to demonstrate how to manually define a schema in a Job.

1. Expand the **Hadoop cluster** node under **Metadata** in the **Repository** tree view.

2. Expand the Hadoop connection you have created and then the **HDFS** folder under it.

   In this example, it is the *my_cdh* Hadoop connection.

3. Right click the HDFS connection in this **HDFS** folder and from the contextual menu, select **Retrieve schema**.

   In this scenario, this HDFS connection has been named to *cdh_hdfs*.

   A **[Schema]** wizard is displayed, allowing you to browse to files in HDFS.
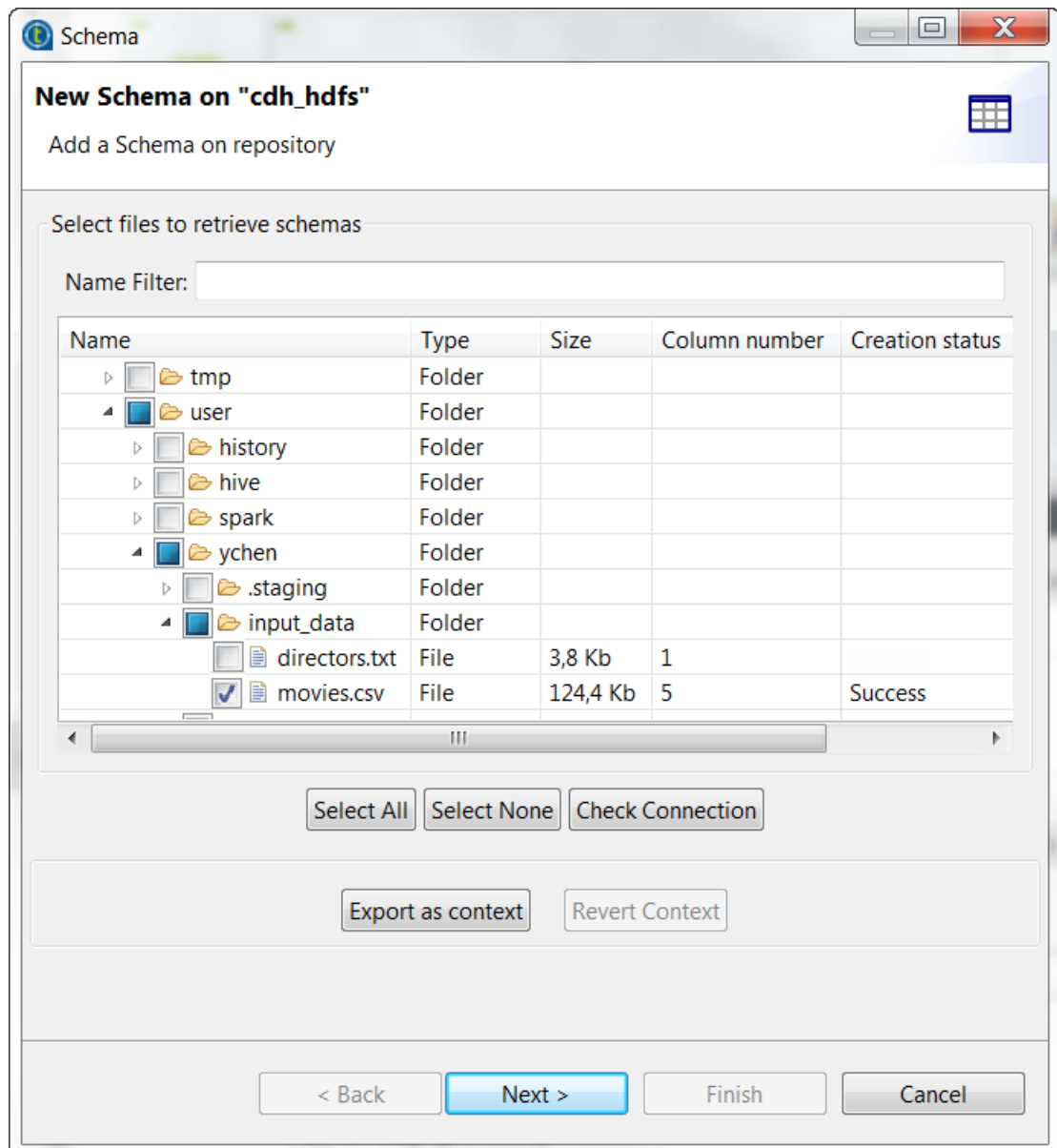
4. Expand the file tree to show the *movies.csv* file, from which you need to retrieve the schema, and select it.

   In this scenario, the *movies.csv* file is stored in the following directory: */user/ychen/input_data*.

5. Click **Next** to display the retrieved schema in the wizard.

   The schema of the movie data is displayed in the wizard and the first row of the data is automatically used as the column names.

If the first row of the data you are using is not used this way, you need to review how you set the **Header** configuration when you were creating the HDFS connection as explained in *Setting up connection to HDFS*.

6. Click **Finish** to validate these changes.

You can now see the file metadata under the HDFS connection you are using in the **Repository** tree view.

- ▲ 🐘 **Hadoop Cluster**
  - ▲ 🐘 my_cdh 0.1
    - ▲ ☐ HDFS(1)
      - ▲ 🗋 cdh_hdfs 0.1
        - ▲ ▦ movies
          - ▲ ☐ Columns(5)
            - ▤ directorID
            - ▤ movieID
            - ▤ releaseYear
            - ▤ title
            - ▤ url
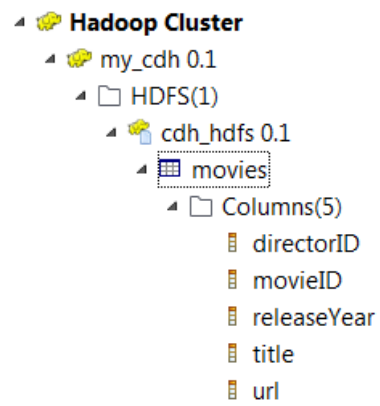
Chapter 5. Performing data integration tasks for Big Data

This chapter takes the example of a company that provides movie rental and streaming video services, and shows how such a company could make use of Talend Open Studio for Big Data.

You will work with data about movies and directors and data about your customers as you learn how to:

- upload data stored in a local file system to the HDFS file system of the company's Hadoop cluster

- join the director data to the movie data to produce a new dataset and store this dataset in the HDFS system too.

5.1. Joining movie and director information

This scenario demonstrates:

1. How to create a *Talend* Job. See *Creating the Job* for details.

2. How to drop and link the components to be used in a Job. See *Dropping and linking components* for details.

3. How to configure the input components using the related metadata from the **Repository**. See *Configuring the input data* for details.

4. How to configure the transformation to join the input data. See *Configuring the data transformation* for details.

5. How to write the transformed data to HDFS. See *Writing the output to HDFS* for details.

5.1.1. Creating the Job

A *Talend* Job allows you to access and use the *Talend* components to design technical processes to read, transform or write data.

Prerequisites:

• You have launched your *Talend Studio* and opened the **Integration** perspective.

Proceed as follows to create the Job:

1. In the **Repository** tree view, right-click the **Job Designs** node and select **Create folder** from the contextual menu.

2. In the **[New Folder]** wizard, name your Job folder *getting_started* and click **Finish** to create your folder.

3. Right-click the *getting_started* folder and select **Create job** from the contextual menu.
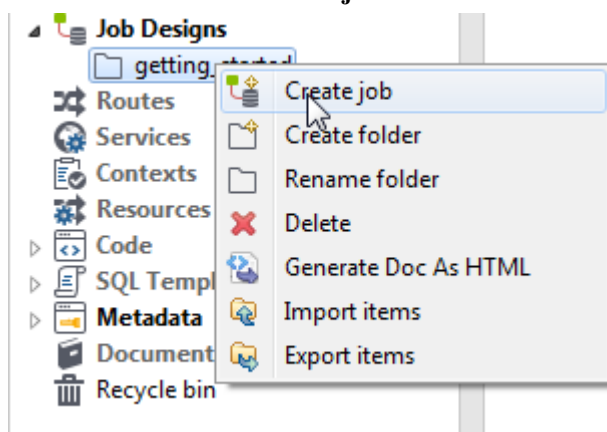
4. In the **[New Job]** wizard, enter a name for the Job to be created and other useful information.

   For example, enter *aggregate_movie_director* for example in the **Name** field.

   In this step of the wizard, **Name** is the only mandatory field. The information you provide in the **Description** field will appear as a tooltip when you move your mouse pointer over the Job in the **Repository** tree view.

5. Click **Finish** to create your Job.

An empty Job is opened in the Studio.

The component **Palette** is now available in the Studio. You can start to design the Job by leveraging this **Palette** and the **Metadata** node in the **Repository**.

5.1.2. Dropping and linking components

The Pig components to be used are orchestrated in the Job workspace to compose a Pig process for data transformation.

Prerequisites:

• You have launched your **Talend Studio** and opened the **Integration** perspective.

• An empty Job has been created as described in *Creating the Job* and is open in the workspace.

Proceed as follows to add and connect the components:

1. In the Job, enter the name of the component to be used and select this component from the list that appears. In this scenario, the components are two **tPigLoad** components, a **tPigMap** component and two **tPigStoreResult** components.

    • The two **tPigLoad** components are used to load the movie data and the director data, respectively, from HDFS into the data flow of the current Job.

    • The **tPigMap** component is used to transform the input data.

    • The **tPigStoreResult** components write the results into given directories in HDFS.

2. Double-click the label of one of the **tPigLoad** component to make this label editable and then enter *movie* to change the label of this **tPigLoad**.

3. Do the same to label another **tPigLoad** component to *director*.

4. Right click the **tPigLoad** component that is labelled *movie*, then from the contextual menu, select **Row > Pig combine** and click **tPigMap** to connect this **tPigLoad** to the **tPigMap** component.

    This is the main link through which the movie data is sent to **tPigMap**.

5. Do the same to connect the *director* **tPigLoad** component to **tPigMap** using the **Row > Pig combine** link.

    This is the **Lookup** link through which the director data is sent to **tPigMap** as lookup data.

6. Do the same to connect the **tPigMap** component to **tPigStoreResult** using the **Row > Pig combine** link, then in the pop-up wizard, name this link to *out1* and click **OK** to validate this change.

7. Repeat these operations to connect the **tPigMap** component to another **tPigStoreResult** component using the **Row > Pig combine** link and name it to *reject*.

Now the whole Job looks as follows in the workspace:
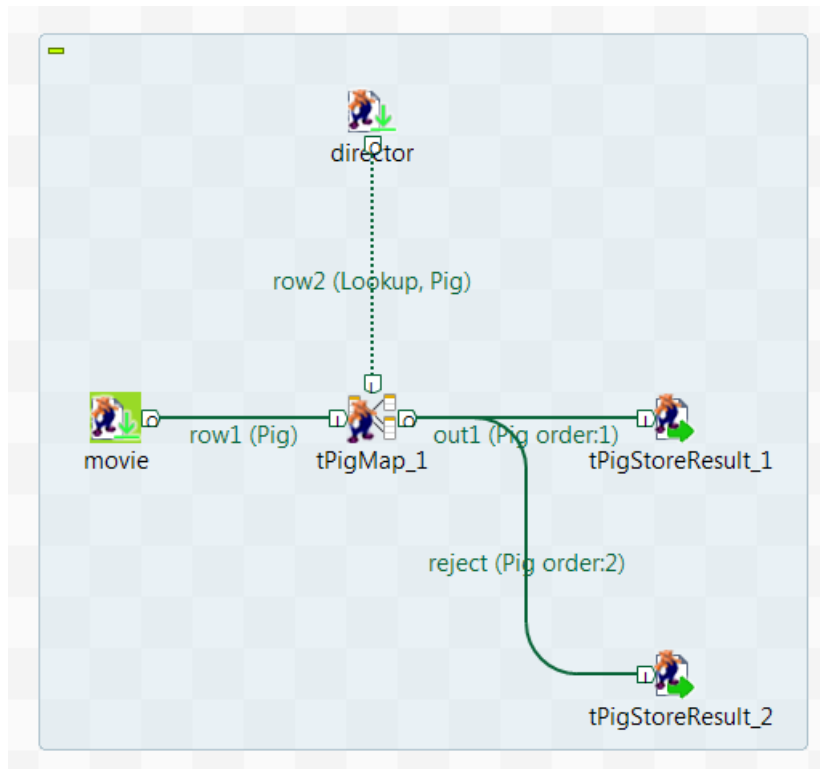
5.1.3. Configuring the input data

Two **tPigLoad** components are configured to load data from HDFS into the Job.

Prerequisites:

- The source files, *movies.csv* and *directors.txt* have been uploaded into HDFS as explained in *Uploading files to HDFS*.

- The metadata of the movie.csv file has been set up in the HDFS folder under the **Hadoop cluster** node in the **Repository**.

  If you have not done so, see *Preparing file metadata* to create the metadata.

Once the Job has been created with all the Pig components to be used being present in the Job and linked together, you need to configure the **tPigLoad** components to properly read data from HDFS.

1. Expand the **Hadoop cluster** node under the **Metadata** node in the **Repository** and then the *my_cdh* Hadoop connection node and its child node to display the *movies* schema metadata node you have set up under the **HDFS** folder as explained in *Preparing file metadata*.

2. Drop this schema metadata node onto the *movie* **tPigLoad** component in the workspace of the Job.

3. Double-click the *movie* **tPigLoad** component to open its **Component** view.

   This **tPigLoad** has automatically reused the HDFS configuration and the movie metadata from the **Repository** to define the related parameters in its **Basic settings** view.

4. From the **Load function** drop-down list, select **PigStorage** to use the PigStorage function, a built-in function from Pig, to load the movie data as a structured text file.

   For further information about the PigStorage function of Pig, see PigStorage.

5. From the Hadoop connection node called *my_cdh* in the **Repository**, drop the *cdh_hdfs* HDFS connection node under the **HDFS** folder onto the **tPigLoad** component labelled *director* in the workspace of the Job.

   This applies the configuration of the HDFS connection you have created in the **Repository** on the HDFS-related settings in the current **tPigLoad** component.

6. Double-click the *director* **tPigLoad** component to open its **Component** view.

   This **tPigLoad** has automatically reused the HDFS configuration from the **Repository** to define the related parameters in its **Basic settings** view.

7. Click the **[...]** button next to **Edit schema** to open the schema editor.

8. Click the **[+]** button twice to add two rows and in the **Column** column, rename them to *ID* and *Name*, respectively.


9. Click **OK** to validate these changes and accept the propagation prompted by the pop-up dialog box.

10. From the **Load function** drop-down list, select **PigStorage** to use the PigStorage function.

11. In the **Input file URI** field, enter the directory where the data about the director data is stored. As is explained in *Uploading files to HDFS*, this data has been written in */user/ychen/input_data/directors.txt*.

12. Click the **Field separator** field to open the **[Edit parameter using repository]** dialog box to update the field separator.
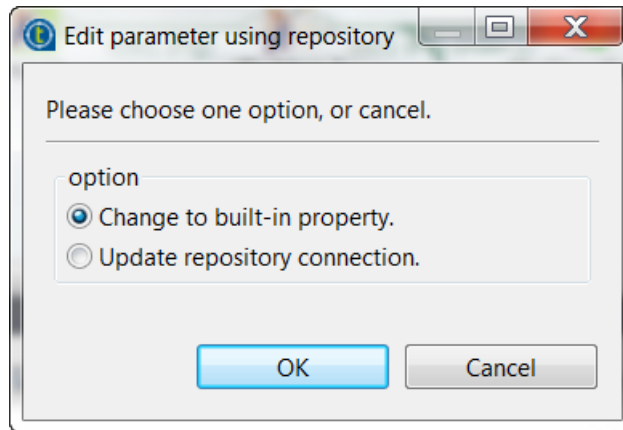
You need to change this field separator because this **tPigLoad** is reusing the default separator, a semicolon (;), you have defined for the HDFS metadata while the director data is actually using a coma (,) as separator.

13. Select **Change to built-in property** and click **OK** to validate your choice.

The **Field separator** field becomes editable.
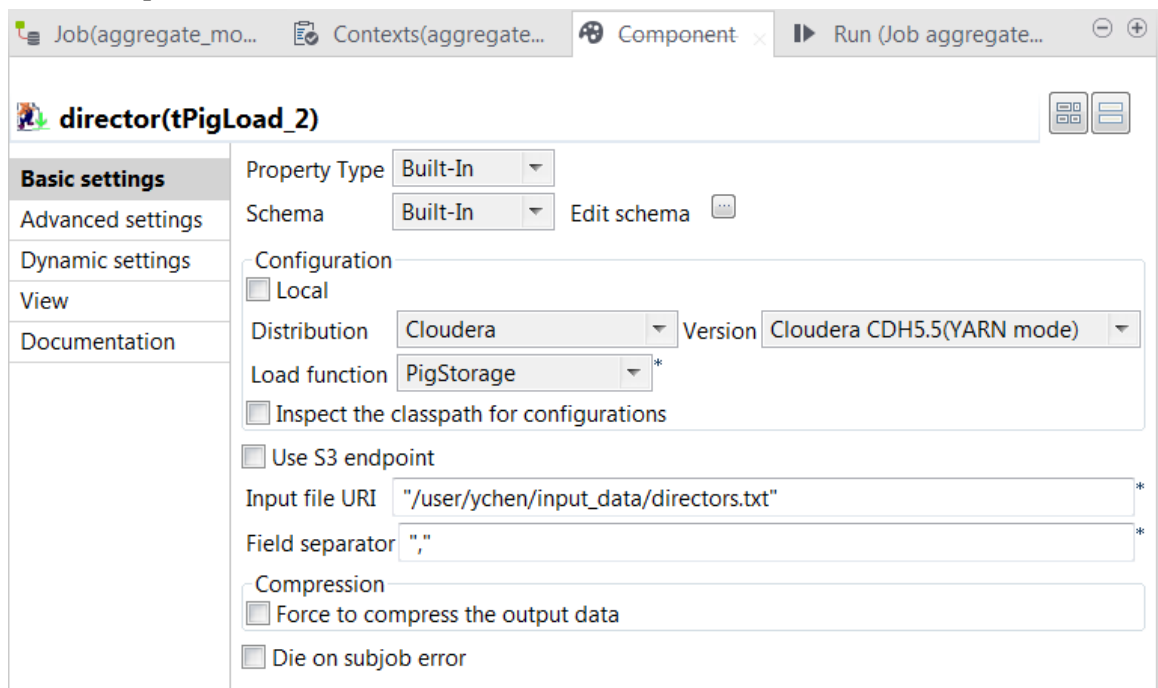

14. Enter a coma within double quotation marks.

The **tPigLoad** components are now configured to load the movie data and the director data to the Job.

5.1.4. Configuring the data transformation

The **tPigMap** component is configured to join the movie data and the director data.

Once the movie data and the director data are loaded into the Job, you need to configure the **tPigMap** component to join them to produce the output you expect.

1. Double-click **tPigMap** to open its **Map Editor** view.
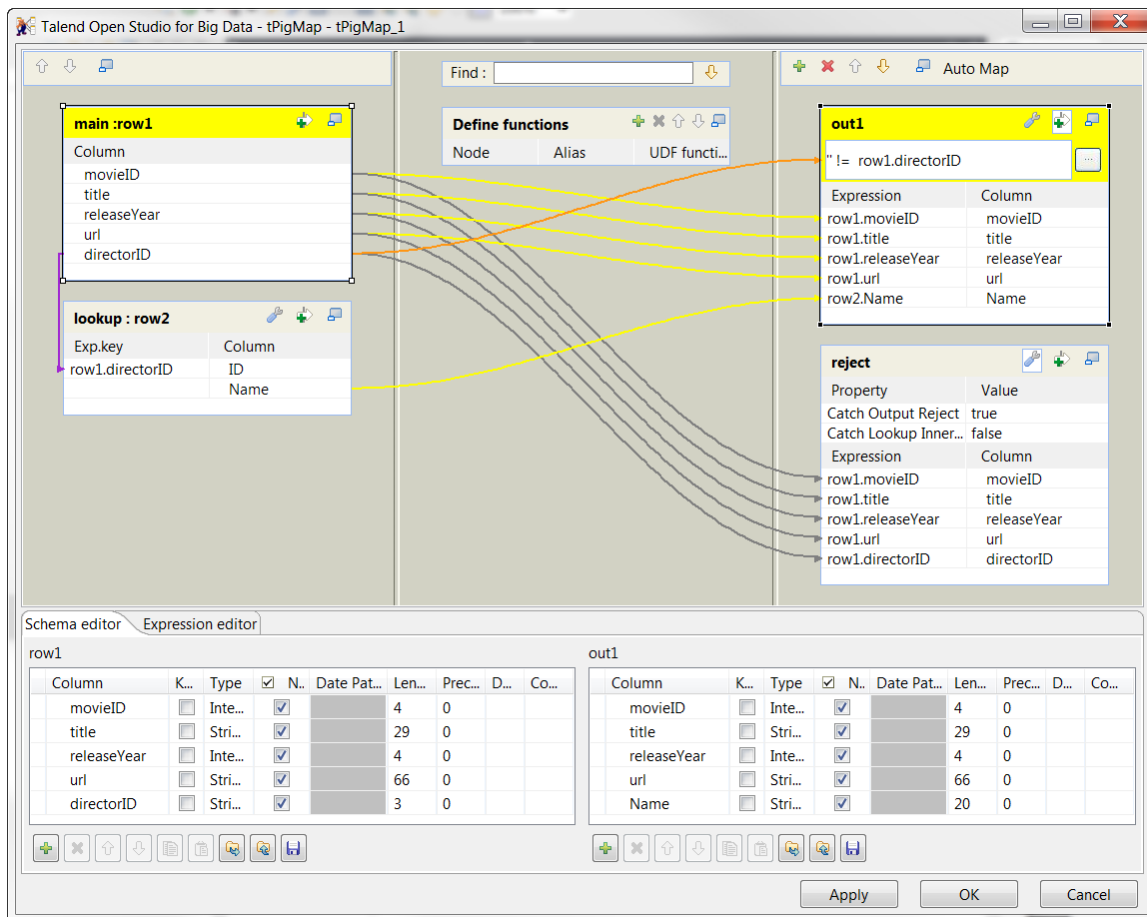
2.   Drop the *movieID* column, the *title* column, the *releaseYear* column and the *url* column from the left side onto each of the output flow table.

     On the input side (left side) of the **Map Editor**, each of the two tables represents one of the input flow, the upper one for the main flow and the lower one for the lookup flow.

     On the output side (right side), the two tables represent the output flows that you named to *out1* and *reject* when you linked **tPigMap** to **tPigStoreResult** in *Dropping and linking components*.

3.   On the input side, drop the *directorID* column from the main flow table to the **Expr.key** column of the *ID* row in the lookup flow table.

     This way, the join key between the main flow and the lookup flow is defined.

4.   Drop the *directorID* column from the main flow table to the *reject* table on the output side and drop the *Name* column from the lookup flow table to the *out1* table.

     The configuration in the previous two steps describes how the columns of the input data are mapped to the columns of the output data flow.

     From the **Schema editor** view in the lower part of the editor, you can see the schemas on both sides have been automatically completed.

5.
     On the *out1* output flow table, click the ⬇ button to display the editing field for the filter expression.

6.   Enter

```
''!= row1.directorID
```

This allows **tPigMap** to output only the movie records in each of which the *directorID* field is not empty. A record with an empty *directorID* field is filtered out.

7. On the *reject* output flow table, click the [icon] button to open the settings panel.

8. In the **Catch Output Reject** row, select **true** to output the records with empty *directorID* fields in the *reject* flow.

9. Click **Apply**, then click **OK** to validate these changes and accept the propagation prompted by the pop-up dialog box.

The transformation is now configured to complete the movie data with the names of their directors and write the movie records that do not contain any director data into a separate data flow.

5.1.5. Writing the output to HDFS

Two **tPigStoreResult** components are configured to write the expected movie data and the rejected movie data to different directories in HDFS.

Prerequisites:

• You have ensured that the client machine on which the *Talend* Jobs are executed can recognize the host names of the nodes of the Hadoop cluster to be used. For this purpose, add the IP address/hostname mapping entries for the services of that Hadoop cluster in the *hosts* file of the client machine.

For example, if the host name of the Hadoop Namenode server is *talend-cdh550.weave.local* and its IP address is *192.168.x.x*, the mapping entry reads *192.168.x.x talend-cdh550.weave.local*.

• The Hadoop cluster to be used has been properly configured and is running.

After the movie data and the director data have been transformed by **tPigMap**, you need to configure the two **tPigStoreResult** components to write the output into HDFS.

1. Double-click the **tPigStoreResult** which receives the *out1* link.

Its **Basic settings** view is opened in the lower part of the Studio.
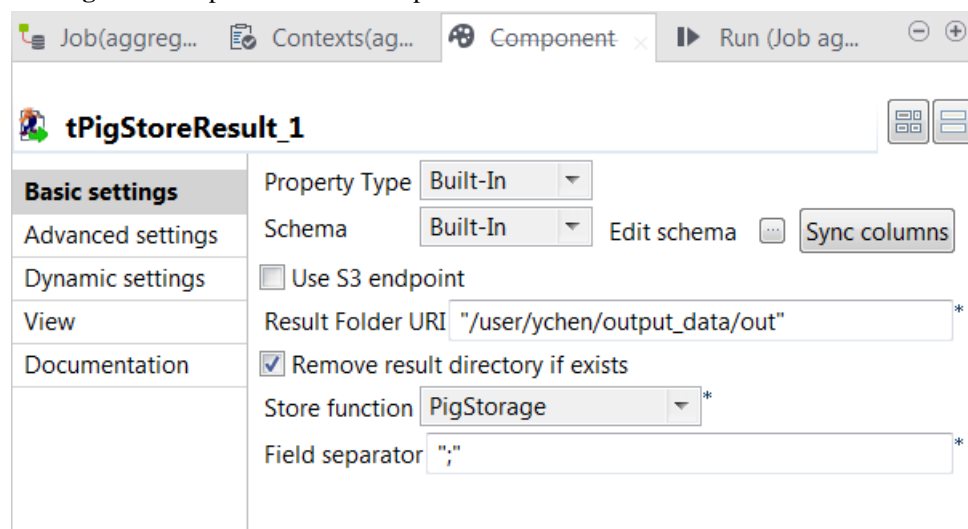
2. In the **Result file** field, enter the directory you need to write the result in. In this scenario, it is */user/ychen/output_data/out*, which receives the records that contain the names of the movie directors.

3. Select **Remove result directory if exists** check box.

4. In the **Store function** list, select **PigStorage** to write the records in human-readable UTF-8 format.

5. In the **Field separator** field, enter *;* within double quotation marks.

6. Repeat the same operations to configure the **tPigStoreResult** that receives the *reject* link, but set the directory, in the **Result file** field, to */user/ychen/output_data/reject*.

7. Press **F6** to run the Job.

    The **Run** view is automatically opened in the lower part of the Studio and shows the execution progress of this Job.

Once done, you can check, for example in the web console of your HDFS system, that the output has been written in HDFS.
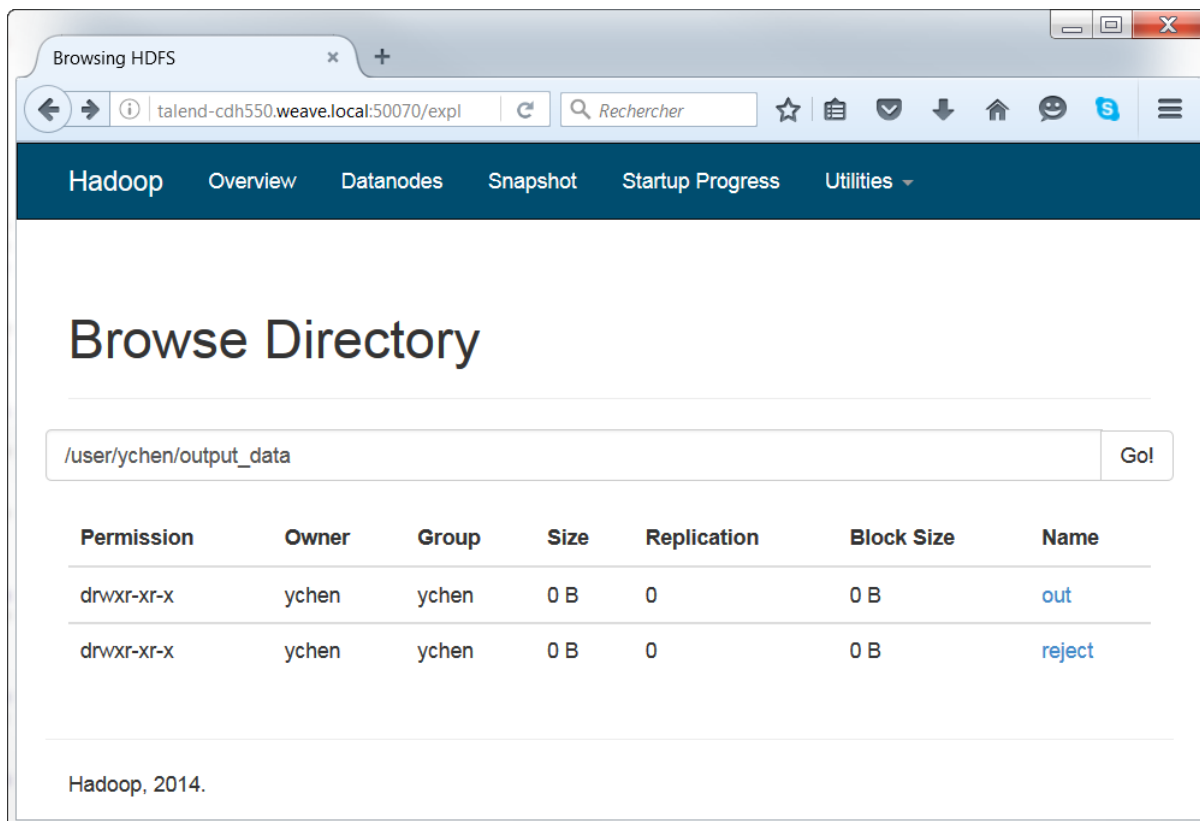
5.2. What's next?

You have seen how *Talend Studio* helps you manage your big data using *Talend* Jobs. You have learned how to access and move your data to a given Hadoop cluster via *Talend* Jobs, filter and transform your data, and store the filtered and transformed data in the HDFS system of the Hadoop cluster. Along the way, you have learned how to centralize frequently used Hadoop connections in the **Repository** and easily reuse these connections in your Jobs.

To learn more about *Talend Studio*, see:

• *Talend Studio User Guide*

• *Talend Open Studio for Big Data Components Reference Guide*

To ensure that your data is clean, you can try *Talend Open Studio for Data Quality* and *Talend Data Preparation*.

To learn more about *Talend* products and solutions, visit www.talend.com.