Hortonworks Data Platform

Data Integration Services with HDP

(Feb 6, 2013)

docs.hortonworks.com

Hortonworks Data Platform : Data Integration Services with HDP

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, Zookeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain free and open source.

Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

Table of Contents

1. Using Data Integration Services Powered by Talend

1.1. Using Data Integration Services Powered by Talend

Talend Open Studio for Big Data is a powerful and versatile open source data integration solution. It enables an enterprise to work with existing data and existing systems, and use Hadoop to power large scale data analysis across the enterprise.

Talend Open Studio (TOS) is distributed as an add-on for Hortonworks Data Platform (HDP). TOS uses the following HDP components:

- Enable users to read/write from/to Hadoop as a data source/sink.
- HCatalog Metadata services enable users to import raw data into Hadoop (HBase and HDFS), create, and manage schemas.
- Pig and Hive for analyzing these data sets.
- Enable users to schedule these ETL jobs on a recurring basis on a Hadoop Cluster using Oozie.

In this section:

1.2. Prerequisites

- Ensure that you have deployed HDP for all the nodes in your cluster. For instructions on deploying HDP, see “Getting Ready to Install” in *Installing HDP Using Apache Ambari* here.
- Ensure that you create a home directory for the user launching the TOS in the HDFS cluster.

  EXAMPLE: If `hdptestuser` is responsible for launching TOS, then execute the following command on the gateway machine as the administrator user (HDFS user) to create home directory:

  ```
  % hadoop dfs -mkdir /user/hdptestuser
  ```

- Ensure the user launching the TOS has appropriate permissions on the HDP cluster.

  EXAMPLE: If `hdptestuser` is responsible for launching TOS, then execute the following command on the gateway machine as the administrator user (HDFS user) to provide the required permissions:

  ```
  % hadoop dfs –chown hdptestuser:hdptestuser /user/hdptestuser
  ```

1.3. Instructions

This section provides you instructions on the following:

- Deploying Talend Open Studio
- Writing the first Talend job for data import
- Modifying the job to perform data analysis

1.3.1. Deploying Talend Open Studio

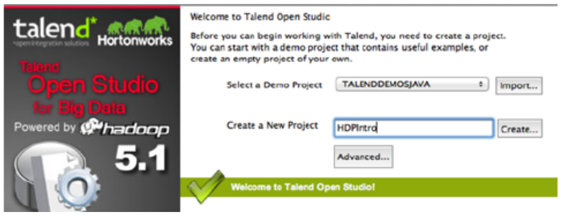
This section provides a high-level description of setting up and using Talend Open Studio.

1.3.1.1. Step 1: Download and launch the application

- Download the Talend Open Studio add-on for HDP from here.
- After the download is complete, unzip the contents in an install location.
- Invoke the executable file corresponding to your operating system.
- Read and accept the end user license agreement.

1.3.1.2. Step 2: Create a new project

- Provide a project name (for example, HDPIntro) and click the “Create” button.



Welcome to Talend Open Studio

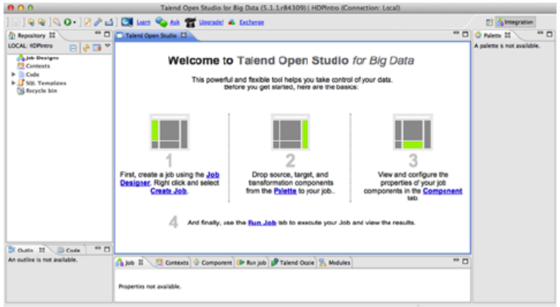
Before you can begin working with Talend, you need to create a project. You can start with a demo project that contains useful examples, or create an empty project of your own.

Select a Demo Project: TALENDDEMOSJAVA

Create a New Project: HDPIntro

Welcome to Talend Open Studio!

- Click “Finish” on the “New Project” dialog
- Select the newly created project and click “Open”.
- The “Connect To TalendForge” dialog appears, you can choose to register or click “Skip” to continue.
- You should now see the progress information bar and a welcome window. Wait for the application to initialize and then click “Start now!” to continue. Talend Open Studio (TOS) main window appears and is now ready for use.



Welcome to Talend Open Studio *for Big Data*

This powerful and flexible tool helps you take control of your data.
Before you get started, here are the basics:

1 First, create a job using the Job Designer. Right click and select Create Job.

2 Drop source, target, and transformation components from the Palette to your job..

3 View and configure the properties of your job components in the Component tab.

4 And finally, use the Run Job tab to execute your Job and view the results.

A palette is not available.

An outline is not available.

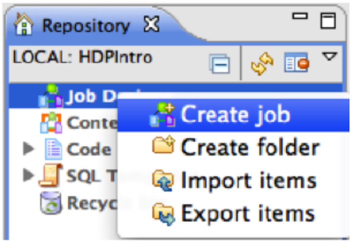
Properties not available.

1.3.2. Writing first Talend job for data import

This section provides instructions on designing a simple job of importing a file into the Hadoop cluster.

1.3.2.1. Step 1: Create a new job

- In the Repository tree view, right-click the “Job Designs” node.
- From the contextual menu, select “Create job”.



- In the “New Job” wizard provide a name (for example HDPJob) and click “Finish”.
- An empty design workspace corresponding to the Job name opens up.

1.3.2.2. Step 2: Create a sample input file

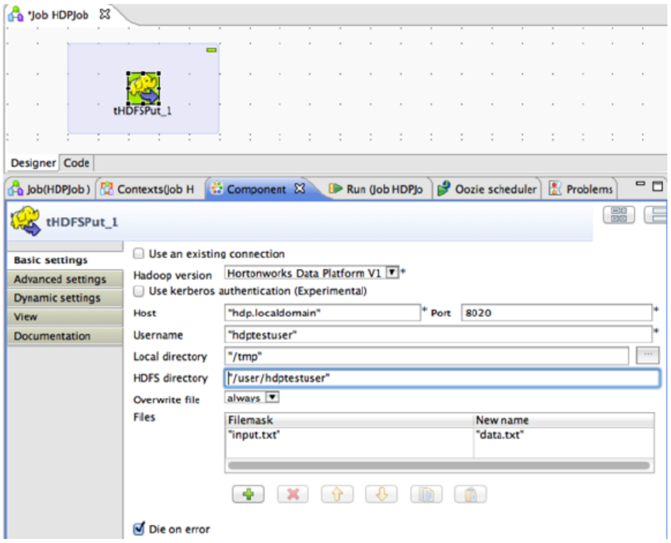
- Under the /tmp directory of your TOS master deployment machine, create a text file (for example: input.txt) with the following contents:

```
101;Adam;Wiley;Sales
102;Brian;Chester;Service
103;Julian;Cross;Sales
104;Dylan;Moore;Marketing
105;Chris;Murphy;Service
106;Brian;Collingwood;Service
107;Michael;Muster;Marketing
108;Miley;Rhodes;Sales
109;Chris;Coughlan;Sales
110;Aaron;King;Marketing
```

1.3.2.3. Step 3: Build the job

Jobs are composed of components that are available in the Palette.

- Expand the “Big Data” tab in the Palette.
- Click on the component `tHDFSPut` and click on the design workspace to drop this component.
- Double-click `tHDFSPut` to define component in its Basic Settings view.
- Set the values in the Basic Settings corresponding to your HDP cluster (see the screenshot given below):



tHDFSPut_1

☐ Use an existing connection

Hadoop version: Hortonworks Data Platform V1

☐ Use kerberos authentication (Experimental)

Host: "hdp.localdomain" Port: 8020

Username: "hdptestuser"

Local directory: "/tmp"

HDFS directory: "/user/hdptestuser"

Overwrite file: always

Files

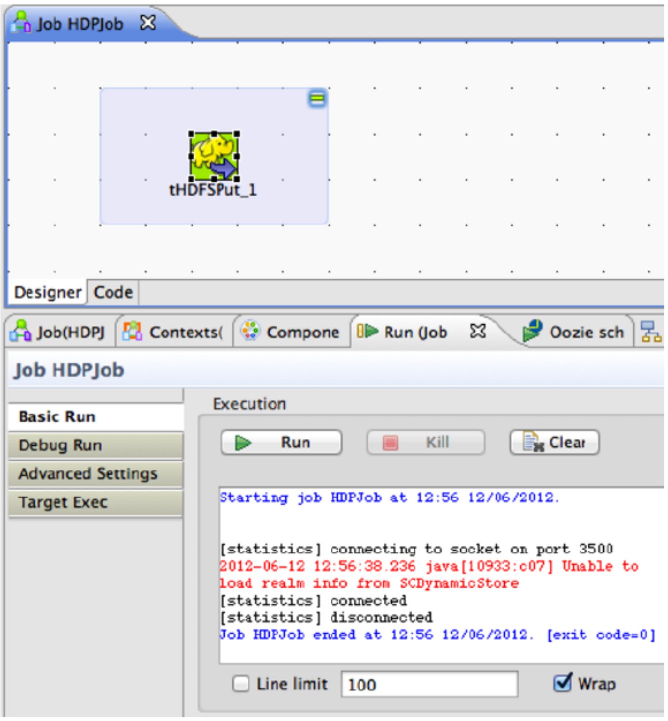
Filemask	New name
"input.txt"	"data.txt"

☑ Die on error

1.3.2.4. Step 4: Run the job

You now have a working job. You can run it by clicking the green play icon.

You should see the following:



```
Starting job HDPJob at 12:56 12/06/2012.


[statistics] connecting to socket on port 3500
2012-06-12 12:56:38.236 java[10933:c07] Unable to
load realm info from SCDynamicStore
[statistics] connected
[statistics] disconnected
Job HDPJob ended at 12:56 12/06/2012. [exit code=0]
```

1.3.2.5. Step 5: Verify the import operation

- From the gateway machine or the HDFS client, open a console window and execute the following command:

```
[hdptestuser@hdp ~]$ hadoop dfs -ls /user/hdptestuser/data.txt
```

- You should see the following result on your terminal window:

```
Found 1 items
-rw-r--r-- 3 hdptestuser hdptestuser
252 2012-06-12 12:52 /user/
hdptestuser/data.txt
```

- This message indicates that the local file was successfully created in your Hadoop cluster.

1.3.3. Modifying the job to perform data analysis

This section provides instructions on using Apache Pig to aggregate the data.

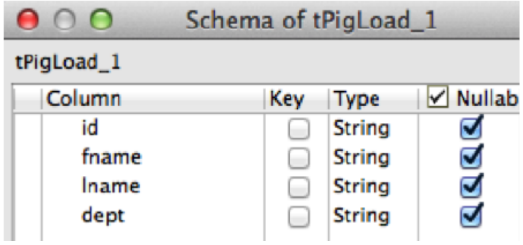
1.3.3.1. Step 1: Add Pig component from the Big Data Palette

- Expand the “Pig” tab in the Big Data Palette.
- Click on the component `tPigLoad` and place it in the design workspace.

1.3.3.2. Step 2: Define basic settings for Pig component

- Double-click `tPigLoad` component to define its Basic Settings.

- Click the “Edit Schema” button (“...” button). Define the schema of the input data as shown below and click OK:



Schema of tPigLoad_1

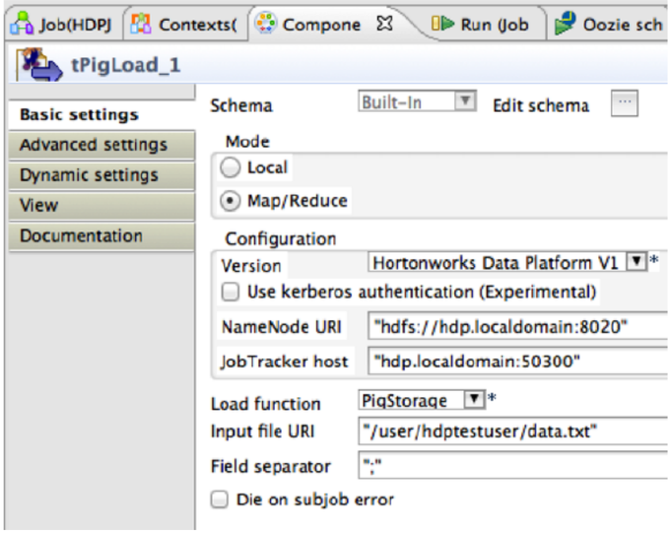
tPigLoad_1

Column	Key	Type	☑ Nullab
id	□	String	☑
fname	□	String	☑
lname	□	String	☑
dept	□	String	☑

- Provide the values for the mode, configuration, NameNode URI, JobTracker host, load function, and input file URI fields as shown.

Important

Ensure that the NameNode URI and the JobTracker host corresponds to accurate values available in your HDP cluster. The Input File URI corresponds to the path where we imported the `input.txt` file in previous section.



tPigLoad_1

Schema Built-In Edit schema ...

Mode
○ Local
● Map/Reduce

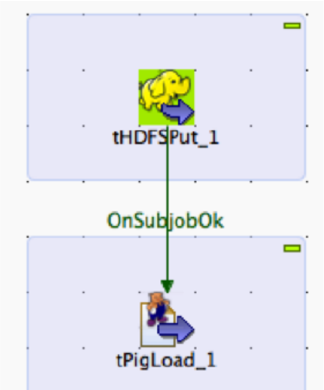
Configuration
Version Hortonworks Data Platform V1
□ Use kerberos authentication (Experimental)
NameNode URI "hdfs://hdp.localdomain:8020"
JobTracker host "hdp.localdomain:50300"

Load function PigStorage
Input file URI "/user/hdptestuser/data.txt"
Field separator ";"
□ Die on subjob error

1.3.3.3. Step 3: Connect the Pig and HDFS components

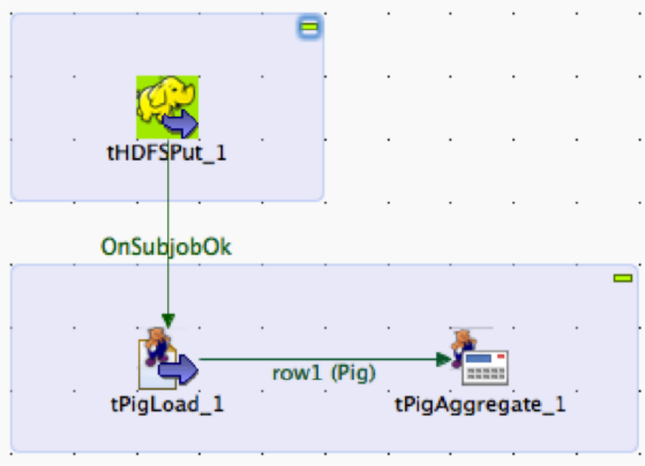
Connect the Pig and HDFS components to define the workflow. To connect the two components:

- Right-click the source component (`tHDFSPut`) on your design workspace.
- From the contextual menu, select Trigger -> On Subjob Ok.
- Click the target component (`tPigLoad`).



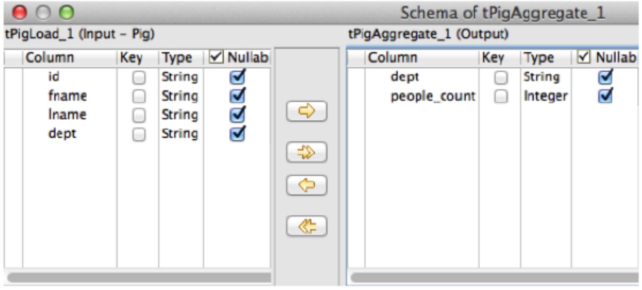
1.3.3.4. Step 4: Add and connect Pig aggregate component

- Add the component `tPigAggregate` next to `tPigLoad`.
- From the contextual menu, right-click on `tPigLoad` and select Row -> Pig Combine.
- Click on `tPigAggregate`.



1.3.3.5. Step 5: Define basic settings for Pig Aggregate component

- Double-click `tPigAggregate` to define the component in its Basic Settings.
- Click on the “Edit schema” button and define the output schema as shown below:



Schema of tPigAggregate_1

tPigLoad_1 (Input - Pig)

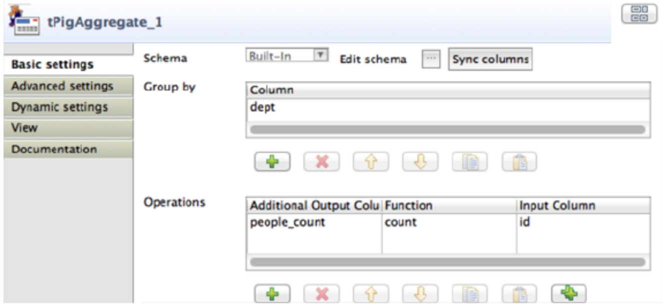
Column	Key	Type	☑ Nullab
id	□	String	☑
fname	□	String	☑
lname	□	String	☑
dept	□	String	☑

tPigAggregate_1 (Output)

Column	Key	Type	☑ Nullab
dept	□	String	☑
people_count	□	Integer	☑

1.3.3.6. Step 6: Define aggregation function for the data

- Add a column to “Group by”, choose “dept”.
- In the Operations table, choose the “people_count” in the Additional Output column, function as “count” and input column “id” as shown:



Schema: Built-In

Group by:

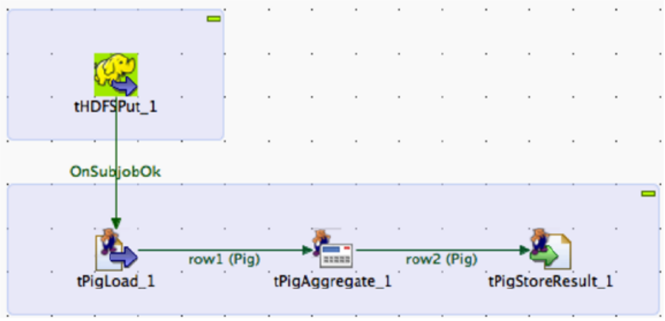
Column
dept

Operations:

Additional Output Colu	Function	Input Column
people_count	count	id

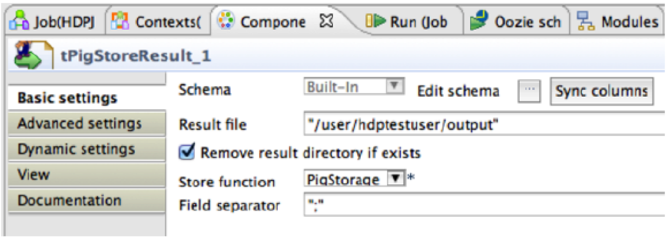
1.3.3.7. Step 7: Add and connect Pig data storage component

- Add the component `tPigStoreResult` next to `tPigAggregate`.
- From the contextual menu, right-click on `tPigLoad`, select Row -> Pig Combine and click on `tPigStoreResult`.



1.3.3.8. Step 8: Define basic settings for data storage component

- Double-click `tPigStoreResult` to define the component in its Basic Settings view.
- Specify the result directory on HDFS as shown:



Schema: Built-In

Result file: "/user/hdptestuser/output"

☑ Remove result directory if exists

Store function: PigStorage *

Field separator: ";"

1.3.3.9. Step 9: Run the modified Talend job

- The modified Talend job is ready for execution. Save the job and click the play icon to run as instructed in Step 4 of previous section.

1.3.3.10. Step 10: Verify the results

- From the gateway machine or the HDFS client, open a console window and execute the following command:

```
[hdptestuser@hdp ~]$ hadoop dfs -cat /user/hdptestuser/output/part-r-00000
```

- You should see the following output:

```
Sales;4
Service;3
Marketing;3
```

2. Using HDP for Metadata Services (HCatalog)

Hortonworks Data Platform deploys Apache HCatalog to manage the metadata services for your Hadoop cluster.

Apache HCatalog is a table and storage management service for data created using Apache Hadoop. This includes:

- Providing a shared schema and data type mechanism.
- Providing a table abstraction so that users need not be concerned with where or how their data is stored.
- Providing interoperability across data processing tools such as Pig, MapReduce, and Hive.

For more details, see the following resources:

- HCatalog Overview
- Installation From Tarball
- Load and Store Interfaces
- Input and Output Interfaces
- Reader and Writer Interfaces
- Command Line Interface
- Storage Formats
- Dynamic Partitioning
- Notification
- Authorization
- API Documentation

For more details on the Apache HCatalog project, use the following resources:

- HCatalog Wiki
- HCatalog Mailing Lists

2.1. Using WebHCat

WebHCat provides a REST-like web API for HCatalog and related Hadoop components.

For more details, see the following resources:

3. Using Apache Hive

Hortonworks Data Platform deploys Apache Hive for your Hadoop cluster.

Hive is a data warehouse infrastructure built on top of Hadoop. It provides tools to enable easy data ETL, a mechanism to put structures on the data, and the capability for querying and analysis of large data sets stored in Hadoop files.

Hive defines a simple SQL query language, called HiveQL, that enables users familiar with SQL to query the data. At the same time, this language also allows programmers who are familiar with the MapReduce framework to plug in their custom mappers and reducers to perform more sophisticated analysis that may not be supported by the built-in capabilities of the language.

Hive Documentation

Documentation for Hive release 0.10.0 can be found in multiple places.

1. The Hive wiki is organized in these sections:
   - General Information about Hive
   - User Documentation
   - Administrator Documentation
   - Resources for Contributors
2. Supplementary documentation describes new features and bug fixes, including:
   - HiveServer2 JDBC
   - Decimal data type
   - Metastore server security
   - Secure cluster configuration (JDBC client setup)
3. Javadocs describe the Hive API. The supplementary documentation includes a complete set of Javadocs for this release.

Note

Previous Hive releases included a small documentation set (xdocs). That information is now available in the Hive wiki.

Hive JIRAs

Issue tracking for Hive bugs and improvements can be found here: Hive JIRAs.

Hive ODBC Driver

Hortonworks provides a Hive ODBC driver that allows you to connect popular Business Intelligence (BI) tools to query, analyze and visualize data stored within the Hortonworks Data Platform.

- Download the Hortonworks Hive ODBC driver from here.
- The instructions on installing and using this driver are available here.

4. Using HDP for Workflow and Scheduling (Oozie)

Hortonworks Data Platform deploys Apache Oozie for your Hadoop cluster.

Oozie is a server-based workflow engine specialized in running workflow jobs with actions that execute Hadoop jobs, such as MapReduce, Pig, Hive, Sqoop, HDFS operations, and sub-workflows. Oozie supports coordinator jobs, which are sequences of workflow jobs that are created at a given frequency and start when all of the required input data is available. A command-line client and a browser interface allow you to manage and administer Oozie jobs locally or remotely.

For additional Oozie documentation, use the following resources:

- Quick Start Guide
- Developer Documentation
  - Oozie Workflow Overview
  - Running the Examples
  - Workflow Functional Specification
  - Coordinator Functional Specification
  - Bundle Functional Specification
  - EL Expression Language Quick Reference
  - Command Line Tool
  - Workflow Rerun
  - Kerberos Authentication
  - Email Action
  - Writing a Custom Action Executor
  - Oozie Client Javadocs
  - Oozie Core Javadocs
  - Oozie Web Services API
- Administrator Documentation
  - Oozie Installation and Configuration
  - Oozie Monitoring
  - Command Line Tool

5. Using Apache Sqoop

5.1. Apache Sqoop

Hortonworks Data Platform deploys Apache Sqoop for your Hadoop cluster.

Sqoop is a tool designed to transfer data between Hadoop and relational databases. You can use Sqoop to import data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS), transform the data in Hadoop MapReduce, and then export the data back into an RDBMS. Sqoop automates most of this process, relying on the database to describe the schema for the data to be imported. Sqoop uses MapReduce to import and export the data, which provides parallel operation as well as fault tolerance.

For additional information see the Sqoop documentation, including these sections in the User Guide:

- Basic Usage
- Sqoop Tools
- Troubleshooting

5.1.1. Sqoop Connectors

Sqoop uses a connector based architecture which supports plugins that provide connectivity to new external systems. Using specialized connectors, Sqoop can connect with external systems that have optimized import and export facilities, or do not support native JDBC. Connectors are plugin components based on Sqoop's extension framework and can be added to any existing Sqoop installation.

HDP provides the following connectors for Sqoop:

- **Sqoop connector for Teradata:** This connector can be downloaded from here.
- **MySQL connector:** This connector is provided with the Hortonworks repository and the instructions on configuring this connector are available here.
- **Oracle JDBC connector:** The instructions on configuring this connector are available here.

A Sqoop connector for SQL Server is available from Microsoft:

- **SQL Server R2 connector:** This connector and its documentation can be downloaded from here.

5.1.2. Sqoop Import Table Commands

When connecting to an Oracle database, the Sqoop `import` command requires case-sensitive table names and usernames (typically uppercase). Otherwise the import fails with error message "Attempted to generate class with no columns!"

Prior to the resolution of SQOOP-741, `import-all-tables` would fail for an Oracle database. See the jira for more information.

The `import-all-tables` command has additional restrictions. See Chapter 8 in the Sqoop User Guide.

6. Installing and Configuring Flume in HDP

This document describes the process of manually installing and configuring Apache Flume for use with the Hortonworks Data Platform (HDP). Flume is the 1.x release line of the Flume project.

6.1. Installing and Configuring Flume in HDP

Use the following links to install and configure Flume for HDP

- Understand Flume
- Install Flume
- Configure Flume
- HDP and Flume
- A Simple Example

What follows is a very high level description of the mechanism. For much greater detail, see the Flume html doc-set that is installed with Flume. Once you have installed Flume, the doc-set can be accessed at `file:///usr/lib/flume/docs/index.html`. The *"Flume User Guide"* is available at `file:///usr/lib/flume/docs/FlumeUserGuide.html`. The same docs are also available at the Flume website, flume.apache.org.

6.2. Understand Flume

Flume is a top level project at the Apache Software Foundation. While it can function as a general-purpose event queue manager, in the context of Hadoop it is most often used as a log aggregator, collecting log data from many diverse sources and moving them to a centralized data store.

Note

What follows is a very high level description of the mechanism. For much greater detail, see the Flume html doc-set that is installed with Flume. Once you have installed Flume, the doc-set can be accessed at `file:///usr/lib/flume/docs/index.html`. The *"Flume User Guide"* is available at `file:///usr/lib/flume/docs/FlumeUserGuide.html`.

6.2.1. Flume Components

A Flume data flow is made up of five main components: Events, Sources, Channels, Sinks, and Agents.

6.2.1.1. Events

An event is the basic unit of data that is moved using Flume. It is similar to a message in JMS and is generally small. It is made up of headers and a byte-array body.

6.2.1.2. Sources

The source receives the event from some external entity and stores it in a channel. The source must understand the type of event that is sent to it: an Avro event requires an Avro source.

6.2.1.3. Channels

A channel is an internal passive store with certain specific characteristics. An in-memory channel, for example, can move events very quickly, but does not provide persistence. A file based channel provides persistence. A source stores an event in the channel where it stays until it is consumed by a sink. This temporary storage allows source and sink to run asynchronously.

6.2.1.4. Sinks

The sink removes the event from the channel and forwards it on either to a destination, like HDFS, or to another agent/dataflow. The sink must output an event that is appropriate to the destination.

6.2.1.5. Agents

An agent is the container for a Flume data flow. It is any physical JVM running Flume. The same agent can run multiple sources, sinks, and channels. A particular data flow path is set up through the configuration process.

6.3. Install Flume

Flume is included in the HDP-1.1.0.15 repository, but it is not installed automatically as part of the standard HDP installation process.

6.3.1. Prerequisites

1. You must have installed at least core Hadoop on your system using one of the available methods. See HDP Deployment Options for more information on your options.

2. You must have set up your `JAVA_HOME` environment variable per your operating system. See the various installation guides above for more information.

6.3.2. Installation

To install Flume, from a terminal window type:

[For RHEL or CentOS]

```
yum install flume
```

or

[For SLES]

```
zypper install flume
```

6.3.3. Users

The installation process automatically sets up the appropriate `flume` user and `flume` group in the operating system.

6.3.4. Directories

The main Flume files are located in `/usr/lib/flume` and the main configuration files are located in `/etc/flume/conf`.

6.4. Configure Flume

You configure Flume by using a properties file, which is specified on Flume start-up. There is a template for this file in the configuration directory: `/etc/flume/conf/flume-conf.properties.template`.

Note

There is also a file `flume-env.sh.template` in the `conf` directory. This file can be used to set environment variables automatically at start-up.

6.5. HDP and Flume

Flume ships with many source, channel, and sink types. For use with HDP the following types have been thoroughly tested:

6.5.1. Sources

- Exec (basic, restart)
- Syslogtcp
- Syslogudp

6.5.2. Channels

- Memory
- File

6.5.3. Sinks

- HDFS: secure, nonsecure

- HBase

6.6. A Simple Example

The following snippet shows some of the kinds of properties that can be set using the properties file. For more detailed information, see the *“Flume User Guide”*.

```
agent.sources = pstream
agent.channels = memoryChannel
agent.channels.memoryChannel.type = memory
agent.sources.pstream.channels = memoryChannel
agent.sources.pstream.type = exec
agent.sources.pstream.command = python print_stream.py 200 10
agent.sinks = hdfsSink
agent.sinks.hdfsSink.type = hdfs
agent.sinks.hdfsSink.channel = memoryChannel
agent.sinks.hdfsSink.hdfs.path = hdfs://hdp/user/root/flumetest
agent.sinks.hdfsSink.hdfs.fileType = SequenceFile
agent.sinks.hdfsSink.hdfs.writeFormat = Text
```

The source here is defined as an `exec` source, which simply means that the agent runs a given command on start-up which streams data to `stdout`, where the source gets it. In this case, the command is a Python script written for test purposes. The channel is defined as an in-memory channel and the sink is an HDFS sink.