

Migrating Data Using Sqoop

Date published: 2021-02-22

Date modified: 2021-06-08



Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Data migration to Apache Hive.....	4
Moving data from databases to Apache Hive.....	4
Create a Sqoop import command.....	4
Import RDBMS data into Hive.....	6
Import command options.....	7

Data migration to Apache Hive

To migrate data from an RDBMS, such as MySQL, to Hive, you should consider Apache Sqoop in CDP with the Teradata connector. Apache Sqoop client CLI-based tool transfers data in bulk between relational databases and HDFS or cloud object stores including Amazon S3 and Microsoft ADLS.

The source of legacy system data that needs to undergo an extract, transform, and load (ETL) process typically resides on the file system or object store. You can also import data in delimited text (default) or SequenceFile format, and then convert data to ORC format recommended for Hive. Generally, for querying the data in Hive, ORC is the preferred format because of the performance enhancements ORC provides.

Using Sqoop with the Teradata Connector

CDP does not support the Sqoop exports using the Hadoop jar command (the Java API). The connector documentation from Teradata includes instructions that include the use of this API. CDP users have reportedly mistaken the unsupported API commands, such as `-forcestage`, for the supported Sqoop commands, such as `--staging-force`. Cloudera supports the use of Sqoop only with commands documented in [Using the Cloudera Connector Powered by Teradata](#). Cloudera does not support using Sqoop with Hadoop jar commands, such as those described in the Teradata Connector for Hadoop Tutorial.

Apache Sqoop documentation on the Cloudera web site

To access the latest Sqoop documentation, go to [Sqoop Documentation 1.4.7.7.1.6.0](#).

Moving data from databases to Apache Hive

You can use Sqoop to import data from a relational database into for use with Hive. You can import the data directly to Hive. Assuming the Sqoop client service is available in your cluster, you can issue import and export commands from the command line.

Related Information

[Apache Sqoop Documentation \(v1.4.7.7.1.6\)](#)

Create a Sqoop import command

You create a single Sqoop import command that imports data from diverse data sources, such as a relational database on a different network, into Apache Hive using Apache Sqoop.

About this task

You enter the Sqoop import command on the command line of your Hive cluster to import data from a data source into the cluster file system and Hive. The import can include the following information, for example:

- Database connection information: database URI, database name, and connection protocol, such as `jdbc:mysql:`
- The data to import
- Parallel processing directives for fast data transfer
- Destination for imported data

Sqoop is tested to work with Connector/J 5.1. If you have upgraded to Connector/J 8.0, and want to use the `zeroDateTimeBehavior` property to handle values of '0000-00-00' in DATE columns, explicitly specify `zeroDateTimeBehavior=CONVERT_TO_NULL` in the connection string. For example, `--connect jdbc:mysql://<MySQL host>/<DB>?zeroDateTimeBehavior=CONVERT_TO_NULL`.

Procedure

1. Create an import command that specifies the Sqoop connection to the RDBMS.

- To enter a password for the data source on the command line, use the -P option in the connection string.
- To specify a file where the password is stored, use the --password-file option.

Password on command line:

```
sqoop import --connect jdbc:mysql://db.foo.com:3306/bar \  
<data to import> \  
--username <username> \  
-P
```

Specify password file:

```
sqoop import --connect jdbc:mysql://db.foo.com:3306/bar \  
--table EMPLOYEES \  
--username <username> \  
--password-file ${user.home}/.password
```

2. Specify the data to import in the command.

- Import an entire table.
- Import a subset of the columns.
- Import data using a free-form query.

Entire table:

```
sqoop import \  
--connect jdbc:mysql://db.foo.com:3306/bar \  
--table EMPLOYEES
```

Subset of columns:

```
sqoop import \  
--connect jdbc:mysql://db.foo.com:3306/bar \  
--table EMPLOYEES \  
--columns "employee_id,first_name,last_name,job_title"
```

Free-form query to import the latest data:

```
sqoop import \  
--connect jdbc:mysql://db.foo.com:3306/bar \  
--table EMPLOYEES \  
--where "start_date > '2018-01-01'"
```

3. Optionally, specify write parallelism in the import statement to run a number of map tasks in parallel:

- Set mappers: If the source table has a primary key, explicitly set the number of mappers using --num-mappers.
- Split by: If primary keys are not evenly distributed, provide a split key using --split-by
- Sequential: If you do not have a primary key or split key, import data sequentially using --num-mappers 1 or --autoreset-to-one-mapper in query.
- Set mappers:

```
sqoop import --connect jdbc:mysql://db.foo.com:3306/bar \  
--table EMPLOYEES \  
--num-mappers 1
```

```
--num-mappers 8 \
```

- Split by:

```
sqoop import --connect jdbc:mysql://db.foo.com:3306/bar \  
--table EMPLOYEES \  
--split-by dept_id
```

- Setting mappers evenly splits the primary key range of the source table.
 - Split by evenly splits the data using the split key instead of a primary key.
4. Specify importing the data into Hive using Hive default delimiters `--hive-import`.
 5. Specify the Hive destination of the data.
 - If you think the table does not already exist in Hive, name the database and table, and use the `--create-hive-table` option.
 - If you want to insert the imported data into an existing Hive external table, name the database and table, but do not use the `--create-hive-table` option.

This command imports the MySQL EMPLOYEES table to a new Hive table named in the warehouse.

```
sqoop import --connect jdbc:mysql://db.foo.com:3306/corp \  
--table EMPLOYEES \  
--hive-import \  
--create-hive-table \  
--hive-database 'mydb' \  
--hive-table 'newtable'
```

This command imports the MySQL EMPLOYEES table to an external table in HDFS.

```
sqoop import --connect jdbc:mysql://db.foo.com:3306/corp \  
--table EMPLOYEES \  
--hive-import \  
--hive-database 'mydb' \  
--hive-table 'myexternaltable'
```

Specify the database and table names, enclosed in single quotation marks, on separate lines (recommended) as shown above. Alternatively specify the database and table names on one line, and enclose the database and table names in backticks.

```
--hive-table `mydb`.`myexternaltable`
```

Due to the Hive-16907 bug fix, Hive rejects ``db.table`` in SQL queries. A dot (.) is no longer allowed in table names. You need to change queries that use such references to prevent Hive from interpreting the entire `db.table` string as the table name.

Related Information

[Apache Sqoop Documentation \(v1.4.7.1.6\)](#)

Import RDBMS data into Hive

You can test the Apache Sqoop import command and then run the command to import relational database tables into Apache Hive.

About this task

You enter the Sqoop import command on the command line of your Hive cluster to import data from a data source to Hive. You can test the import statement before actually executing it.

Before you begin

- The Apache Sqoop client service is available.
- The Hive Metastore and Hive services are available.

Procedure

1. Optionally, test the import command before execution using the eval option.

```
sqoop eval --connect jdbc:mysql://db.foo.com/bar \
--query "SELECT * FROM employees LIMIT 10"
```

The output of the select statement appears listing 10 rows of data from the RDBMS employees table.

2. Run a Sqoop import command that specifies the Sqoop connection to the RDBMS, the data you want to import, and the destination Hive table name.

This command imports the MySQL EMPLOYEES table to a new Hive table named in the warehouse.

```
sqoop import --connect jdbc:mysql://db.foo.com:3306/corp \
--table EMPLOYEES \
--hive-import \
--create-hive-table \
--hive-table mydb.newtable
```

Related Information

[Apache Sqoop Documentation \(v1.4.7.1.6\)](#)

Import command options

You can use Sqoop command options to import data into Apache Hive.

Table 1: Sqoop Command Options for Importing Data into Hive

Sqoop Command Option	Description
--hive-home <directory>	Overrides \$HIVE_HOME.
--hive-import	Imports tables into Hive using Hive's default delimiters if none are explicitly set.
--hive-overwrite	Overwrites existing data in the Hive table.
--create-hive-table	Creates a hive table during the operation. If this option is set and the Hive table already exists, the job will fail. Set to false by default.
--hive-table <table_name>	Specifies the table name to use when importing data into Hive.
--hive-drop-import-delims	Drops the delimiters \n, \r, and \01 from string fields when importing data into Hive.
--hive-delims-replacement	Replaces the delimiters \n, \r, and \01 from strings fields with a user-defined string when importing data into Hive.
--hive-partition-key	Specifies the name of the Hive field on which a sharded database is partitioned.
--hive-partition-value <value>	A string value that specifies the partition key for data imported into Hive.
--map-column-hive <map>	Overrides the default mapping from SQL type to Hive type for configured columns.