Using Hive Data Connectors to support External Data Sources in Data Warehouse Public Cloud (Preview)

Date published: 2023-11-20
Date modified: 2023-11-20

Legal Notice

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

Using Hive data connectors to support external data sources

Cloudera Data Warehouse allows you to use Hive data connectors to map databases present in external data sources to a local Hive Metastore (HMS). The external data sources can be of different types, such as MySQL, PostgreSQL, Oracle, Redshift, or Derby. You can create external tables to represent the data, and then query the tables.

***Note**: This feature is in technical preview and not recommended for use in production deployments. Cloudera recommends that you try this feature in test and development environments.*

What is a data connector

Data connectors in Hive are objects that are defined using a set of properties, such as hostname, user credentials, and type that are required to connect Hive to the external data source. You define a connector and then use the same connector to map multiple databases from the remote data source to the local HMS.

Benefits of using data connectors

Currently, you can use a JDBCStorageHandler to connect Hive to external data sources. Users define a table in the HMS for which data resides in a remote JDBC data store. The Hive table's metadata is persisted locally in the HMS. When HiveServer (HS2) runs a query against this table, data is retrieved from the remote JDBC table. However, there are certain limitations which make it difficult to map databases having a large number of tables.

- Each remote table in the remote data source has to be individually mapped to a local Hive table. It is cumbersome when you have to map an entire database having a lot of tables.
- New tables created in the remote data source are not automatically visible and should be manually mapped in Hive.
- The metadata for the mapped table is static. It does not track changes to the remote table. If the remote table is modified (added columns or dropped columns), the mapped Hive table has to be dropped and recreated. This is not feasible for organizations where tables are constantly changing.

With data connectors, you can map a database or schema in the remote data source to a Hive database. Such databases are referred to as 'Remote' databases in Hive. The benefits of creating remote databases using data connectors are as follows:

- Tables within a mapped database are automatically visible from Hive. The metadata for these tables are not persisted in the HMS. They are retrieved at runtime through an active connector.

- New tables created are automatically visible in Hive.
- The columns and their data types are mapped to compatible Hive data types during runtime. Therefore, metadata changes to tables in the remote datasource are immediately visible in hive.
- The same connector can be used to map another database to a Hive database. All the connection information is shared between these remote databases.

To create a remote database, you must first define a data connector with the required properties and then use this connector to create and map a remote database in an external data source to Hive.

Hive currently supports the following data connector types - 'mysql', 'postgres', 'oracle', 'derby', 'mssql', and 'hivejdbc'. Use the appropriate data connector type to map databases present in the corresponding external data sources.

The 'hivejdbc' connector helps you achieve SQL query federation between two distinct Hive clusters or between Hive and Hive-like compute engines, for example, Amazon EMR. For more information, see Data Connector for Hive and Hive-like engines.

Creating a remote database using a data connector

Learn how to create a Hive data connector, which you can then use to create and map a remote database to Hive. The remote database can reside in an external data source, such as MySQL, PostGreSQL, Oracle, Redshift, or Derby.

Prerequisites:
You must ensure that the `hive.security.temporary.authorization.for.data.connectors` property is set to "true".
In the Hive virtual warehouse details page, go to **Configurations** > **Hiveserver2** > **hive-site** configuration file and click ✚ to add this property as a custom configuration.

Steps:

1. Create a data connector using the following syntax:

   Syntax:

   ```
   CREATE CONNECTOR [IF NOT EXISTS] connector_name
          [TYPE datasource_type]
          [URL datasource_url]
          [COMMENT connector_comment]
          [WITH DC PROPERTIES (property_name=property_value, ...)];
   ```

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided 'as is' without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Example:

If you are using a cleartext password:

```
CREATE CONNECTOR postgres_local TYPE 'postgres' URL
'jdbc:postgresql://localhost:5432' WITH DC PROPERTIES
("hive.sql.dbcp.username"="postgres",
"hive.sql.dbcp.password"="postgres");
```

If you are using a Java keystore instead of a cleartext password:

```
CREATE CONNECTOR postgres_local_ks TYPE 'postgres' URL
'jdbc:postgresql://localhost:5432' WITH DC PROPERTIES
("hive.sql.dbcp.username"="postgres",
"hive.sql.dbcp.password.keystore"="jceks://app/local/hive/secrets.jc
eks",
"hive.sql.dbcp.password.key"="postgres.credential");
```

Note: The example provided above shows a data connector created for a Postgres data source type. Hive currently supports the following data connector types - 'mysql', 'postgres', 'oracle', 'derby', 'mssql', and 'hivejdbc'. Use the appropriate data connector type to map databases present in the corresponding external data sources.

2. Create a REMOTE database in Hive using the data connector created in the previous step.

   Syntax:

   ```
   CREATE [REMOTE] (DATABASE) [IF NOT EXISTS] database_name
         [COMMENT database_comment]
         [USING connector_name]
         [WITH DBPROPERTIES (property_name=property_value, ...)];
   ```

   Example:

   ```
   CREATE REMOTE DATABASE postgres_hive_test USING postgres_local WITH
   DBPROPERTIES ("connector.remoteDbName"="remote_db_test");
   ```

   This statement maps a remote database named "remote_db_test" to a Hive database named "postgres_hive_test" in Hive.

3. You can now use the tables that are available in the remote database.

   Example:

   ```
   USE remote_db_test;
   SHOW TABLES;
   ```

```
DESCRIBE [formatted] <tablename>;
SELECT <col1> from <tablename> where <filter1> and <filter2>;
```

Important:

- The metadata for these tables are not persisted in HMS.
- CREATE, ALTER, and DROP table DDLs are currently not supported in remote databases.