Cloudera Runtime 7.2.18

Indexing Data Using Spark-Solr Connector

Date published: 2015-05-05
Date modified: 2024-04-03

https://docs.cloudera.com/

Legal Notice

Contents

[Technical Preview] Setting up the Hive-Solr connector

Hive-Solr integration allows you to index Hive data into Solr and to query Solr data from Hive. It also enables Kerberos support, securing communication between Hive and Solr.

Important:

The Hive Solr connector is provided by Cloudera as Technical Preview in already released Cloudera versions. Cloudera does not recommend its use in production environments. This feature has many known limitations, including the lack of predicate pushdown, resulting in bad performance especially for SELECT queries. Cloudera currently does not plan on making this feature generally available (GA).

This feature may be deprecated in a future release.

Prerequisites

- Deployed Solr
- Deployed Hive (make sure the hive command line tool works by running the hive command as a 'hive' user.

Tip: If the command execution fails, check sslTrustStore in beeline-site.xml.

Workflow

1. Create a collection in Solr where you want to index data using Hive.
2. Register the Hive-Solr connector in Hive.
3. Create a solr-jaas.conf file, using a principal with read and write rights on both Hive and Solr (must be different from the Solr principal).

   For example:

   ```
   Client {
     com.sun.security.auth.module.Krb5LoginModule required
     useKeyTab=true
     keyTab="/data/solr-indexer.keytab"
     storeKey=true
     useTicketCache=true
     debug=true
     principal="solr-indexer@EXAMPLE.COM";
   };
   ```

   Copy the keytab and the solr-jaas.conf to all nodes.
4. Create an external table in Hive. If Solr ZooKeeper TLS is enabled, include the lww.truststore, lww.truststore.password, lww.jaas.file (Kerberized clusters), and lww.zookeeper.secure table properties. See Enabling SSL for Solr ZooKeeper.
5. Insert data to Solr through the Hive external table.
6. Query Solr data from Hive.

Register the Hive-Solr connector in Hive

The Hive-Solr SerDe JAR is part of the Cloudera Search package. If you want to start using Hive for indexing data into Solr, you have to make it available for the Hive service in your cluster. To do so, you create a symlink to the solr-hive-serde-*[***VERSION***]*.jar file in the auxlib directory of the Hive node.

Procedure

1. Log in to the Hive node.
2. Create a symbolic link to solr-hive-serde-*[***VERSION***]*.jar

   Replace *[***VERSION***]* with the actual version of the JAR file you want to link.

   For example:

   ```
   ln -s /opt/cloudera/parcels/CDH/lib/search/solr-hive-serde-4.0.0.7.2.10.
   0-79.jar /opt/cloudera/parcels/CDH/lib/hive/auxlib/
   ```
3. Restart Hive:

   In Cloudera Manager, locate the Hive service then click Actions Restart

Create an external Hive table for Solr

Indexing data to Solr requires creating a Hive external table, enabling Solr to use (read or write) data in Hive.

Before you begin

Your user must have the right to create tables in Hive.

Create an external table in Hive using the CREATE EXTERNAL TABLE command.

```
CREATE EXTERNAL TABLE [***TABLE NAME***] (
[***FIELD NAME***]_[***SUFFIX***] [***FIELD TYPE***], [***FIELD
 NAME***]_[***SUFFIX***] [***FIELD TYPE***], [***FIELD
 NAME***]_[***SUFFIX***] [***FIELD TYPE***],  solr_query string)
ROW FORMAT SERDE 'com.lucidworks.hadoop.hive.LWSerDe'
STORED BY 'com.lucidworks.hadoop.hive.LWStorageHandler'
WITH SERDEPROPERTIES ('serialization.format'='1')
TBLPROPERTIES (
    'solr.collection'='[***COLLECTION NAME***]',
    'solr.zkhost'='[***HOST***]:2181, [***HOST 2***]:2181, [***HOST
 N***]:2181/solr',
    'solr.query'='*:*');
```

where:

[TABLE NAME***]***

is the name of the table that you want to create

[FIELD NAME***]***

is the name of a field you want to add to the table

[SUFFIX***]***

defines the field type for Solr

FIELD_TYPE, used to define the type of a table field in Hive is not sent to Solr when indexing data from a Hive table. Field types must match or be compatible, however, for queries to complete properly.

By default, Solr uses its schema guessing feature to determine field types for incoming data. If the field type assigned by Solr is incompatible with the field type defined in Hive, it causes a ClassCas tException in the response to subsequent to queries.

To avoid this problem, use the dynamic fields Solr feature. These direct Solr to use specific field types based on a prefix or suffix found on an incoming field name, which overrides Solr guessing at the type. Solr includes by default dynamic field rules for nearly all types it supports, so you

only need to use the same suffix on your field names in your Hive tables for the correct type to be defined.

FIELD_TYPE defines the type of the field for Hive. This information is not forwarded to Solr. This is why you need to add a *[***SUFFIX***]* to every *[***FIELD NAME***]*.

[FIELD TYPE***]***

used to define the type of a table field in Hive. It is not sent to Solr when indexing data from a Hive table. The field types must match or be compatible, however, for queries to complete properly. This is why you need to add a *[***SUFFIX***]* to every *[***FIELD NAME***]* that defines its field type for Solr.

solr_query

Adding the optional solr_query string column allows you to specify query parameters for Solr. If you use standard 'WHERE' conditions in Hive queries, Hive first reads all the documents from Solr and executes the filter only after that, which can be very inefficient. By enabling this feature adding the special solr_query column, you can use Solr to filter the results.

ROW FORMAT SERDE

com.lucidworks.hadoop.hive.LWSerDe

STORED BY

defines a custom storage handler, for example, com.lucidworks.hadoop.hive.LWStorageHandler which is one of the classes included with the Hive SerDe jar.

solr.collection

Replace *[***COLLECTION NAME***]* with the name of the Solr collection for this table. This parameter is mandatory. If not defined, an exception is thrown.

solr.zkhost

The location of the ZooKeeper quorum if using Cloudera Search in SolrCloud mode. If this property is set along with the solr.server.url property, the solr.server.url property takes precedence.

solr.query

The specific Solr query to execute to read this table. If not defined, it defaults to *:*. This property is not necessary when loading data to a table, but is mandatory when defining the table so Hive can later read the table.

Secure ZooKeeper table properties

When Solr uses TLS-enabled ZooKeeper, add the following table properties to TBLPROPERTIES. For information about enabling ZooKeeper TLS for Solr, see Enabling SSL for Solr ZooKeeper.

lww.truststore

The full path to the JKS truststore used for TLS connections from the Hive-Solr connector to Solr and ZooKeeper. Use a path that is valid on every node where Hive map/reduce tasks run, such as the Cloudera Manager AutoTLS global truststore.

lww.truststore.password

The password for the truststore file defined in lww.truststore. This field can be left blank if the truststore does not use a password.

lww.jaas.file

Required when indexing to or reading from a Kerberized Solr cluster. This property defines the path to a JAAS file that contains a service principal and keytab location for a user who is authorized to read from and write to Solr and Hive. You may need this property in addition to the secure ZooKeeper settings on Kerberized clusters.

Note: The JAAS configuration file must be copied to the same path on every node where a Node Manager is running (that is, every node where map/reduce tasks are executed).

lww.zookeeper.secure

Set to true when the Hive-Solr connector must connect to Solr through a TLS-enabled ZooKeeper quorum. Use together with lww.truststore and lww.truststore.password.

If Solr ZooKeeper TLS is enabled, include the following properties in TBLPROPERTIES:

```
'lww.truststore'='[***PATH/TO/TRUSTSTORE***]',
    'lww.truststore.password'='[***TRUSTSTORE PASSWORD***]',
    'lww.jaas.file'='[***PATH/TO/JAAS.CONF***]',
    'lww.zookeeper.secure'='true'
```

Example

In this example, the claim_id_s, line_id_s, and member_id_s are defined as strings, and service_dt_i as an integer. In Solr's default schema, there is a dynamic field rule that any field with an _s suffix should be a string. Similarly, there is another rule that any field with _i as a suffix should be an integer (pint). This allows you to make sure the field types in Solr and Hive match.

```
CREATE EXTERNAL TABLE solr (claim_id_s string, line_id_s string, member_id_s
 string, service_dt_i int)
ROW FORMAT SERDE 'com.lucidworks.hadoop.hive.LWSerDe'
STORED BY 'com.lucidworks.hadoop.hive.LWStorageHandler'
WITH SERDEPROPERTIES ('serialization.format'='1')
TBLPROPERTIES ('solr.collection'='example', 'solr.zkhost'='samplehost1:21
81,samplehost2:2181,samplehost3:2181/solr', 'solr.query'='*:*');
```

Example

The following example adds secure ZooKeeper table properties for a Kerberized cluster with TLS-enabled Solr ZooKeeper:

```
CREATE EXTERNAL TABLE solr (claim_id_s string, line_id_s string, member_id_s
 string, service_dt_i int)
ROW FORMAT SERDE 'com.lucidworks.hadoop.hive.LWSerDe'
STORED BY 'com.lucidworks.hadoop.hive.LWStorageHandler'
WITH SERDEPROPERTIES ('serialization.format'='1')
TBLPROPERTIES (
    'solr.collection'='example',
    'solr.zkhost'='samplehost1:2181,samplehost2:2181,samplehost3:2181/solr',
    'solr.query'='*:*',
    'lww.truststore'='/var/lib/cloudera-scm-agent/agent-cert/cm-auto-global
_truststore.jks',
    'lww.truststore.password'='',
    'lww.jaas.file'='/tmp/solr-jaas.conf',
    'lww.zookeeper.secure'='true');
```

Related Information

Dynamic Fields

Schemaless Mode

Field Types Included with Solr

Drop an external table along with data

Insert data from Hive into Solr

You can insert data to Solr through an external Hive table, using the INSERT Hive statement.

Procedure

- To insert data, use the name of the Solr table as the target for the Hive INSERT statement:

```
INSERT INTO [***TARGET TABLE***]
      SELECT id, field1_s, field2_i FROM [***SOURCE TABLE***];
```

For example, to insert

```
INSERT INTO solr SELECT claim_id_s, line_id_s, member_id_s, service_dt_i
 FROM sometable
```

Run queries from Hive on data indexed to Solr

Once a Hive external table is configured, any syntactically correct Hive query will be able to query the generated Solr index.

Procedure

- To query fields from a table:

```
hive> SELECT [***FIELD NAME***], [***FIELD NAME***], [***FIELD NAME***]
 FROM [***TABLE NAME***];
```

For example, to select three fields named "claim_id_s", "line_id_s", and "service_dt_i" from the "solr" table, run the following query:

```
hive> SELECT claim_id_s, line_id_s, service_dt_i FROM solr;
```

- To filter for rows:

You can either use the standard WHERE conditions. In this case, Hive reads all the documents from Solr and executes the filter after that. This can be very inefficient.

```
hive> SELECT * FROM [***TABLE***] WHERE [***FIELD NAME***]='somestring';
```

If you have added the special solr_query column during table creation, it can be used to filter results by specifying query parameters for Solr.

```
hive> SELECT * FROM [***TABLE***] WHERE solr_query='[***SOLR QUERY***]'
```

You can specify multiple query parameters using the & delimiter.

Note:

Do not mix the two filtering options (using the standard columns and using the solr_query column) in the same query.

For example:

```
hive> SELECT * FROM solr WHERE solr_query='fq=service_dt_s:1*'
```

```
hive> SELECT * FROM solr WHERE solr_query='q=(1d)&df=service_dt_s'
```

```
hive> SELECT * FROM solr WHERE solr_query="fq={!geofilt sfield=gps_p}&p
t=46.9,16.9&d=11111"
```

Related Information

Apache Hive query basics