Cloudera Runtime 7.1.0

Cloudera Search Tutorial

Date published: 2019-11-19
Date modified:

https://docs.cloudera.com/

Legal Notice

Contents

Cloudera Search Tutorial

This tutorial introduces you to Cloudera Search and demonstrates some of its basic capabilities to help you become familiar with the concepts and components involved in Search. It demonstrates creating a simple test collection to validate your Cloudera Search installation, and then continues on to more advanced use cases of batch indexing.

The topics in this tutorial assume you have deployed Cloudera Search. The examples in this tutorial use two shards, so make sure that your deployment includes at least two Solr servers.

This tutorial uses modified schema.xml and solrconfig.xml configuration files. In the versions of these files included with the tutorial, unused fields have been removed for simplicity. Original versions of these files include many additional options. For information on all available options, see the Apache Solr wiki:

- SchemaXml
- SolrConfigXml

Validating the Cloudera Search Deployment

Important: This tutorial does not account for Apache Ranger authorization. If you are using Ranger authorization, you must create policies to allow the users and actions described in the tutorial.

After installing and deploying Cloudera Search, you can validate the deployment by indexing and querying sample documents. You can think of this as a type of "Hello, World!" for Cloudera Search to make sure that everything is installed and working properly.

Before beginning this process, make sure you have access to the Apache Solr admin web console. If your cluster is Kerberos-enabled, make sure you have access to the solr@EXAMPLE.COM Kerberos principal (where EXAMPLE.COM is your Kerberos realm name).

Create a Test Collection

Procedure

1. If you are using Kerberos, kinit as a user that has privileges to create the collection:

   ```
   kinit solr@EXAMPLE.COM
   ```

   Replace EXAMPLE.COM with your Kerberos realm name.
2. If you are using Kerberos, create a JAAS configuration file in your home directory ($HOME/jaas.conf) as follows:

   ```
   Client {
    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=false
    useTicketCache=true
    principal="solr@EXAMPLE.COM";
    };
   ```

3. Make sure that the SOLR_ZK_ENSEMBLE environment variable is set in /etc/solr/conf/solr-env.sh. For example:

```
cat /etc/solr/conf/solr-env.sh
```

```
export SOLR_ZK_ENSEMBLE=zk01.example.com:2181,zk02.example.com:2181,zk03
.example.com:2181/solr
```

This is automatically set on hosts with a Solr Server or Gateway role in Cloudera Manager.

4. Generate configuration files for the collection:

```
solrctl instancedir --generate $HOME/test_collection_config
```

5. Upload the configuration to ZooKeeper:

- Security enabled:

```
solrctl --jaas $HOME/jaas.conf instancedir --create test_collection_conf
ig $HOME/test_collection_config
```

- Nonsecure:

```
solrctl instancedir --create test_collection_config $HOME/test_collectio
n_config
```

6. Create a new collection with two shards (specified by the -s parameter) using the named configuration (specified by the -c parameter).

```
solrctl collection --create test_collection -s 2 -c test_collection_config
```

Index Sample Data

About this task

Cloudera Search includes sample data for testing and validation. Run the relevant command to index this data for searching. Replace *search01.example.com* in the example below with the name of any host running the Solr Server process.

Procedure

- Security enabled:

```
cd /opt/cloudera/parcels/CDH/share/doc/solr-doc*/example/exampledocs
                     find *.xml -exec curl -i -k --negotiate -u: https:
//search01.example.com:8985/solr/test_collection/update -H "Content-Type:
text/xml" --data-binary @{} \;
```

- Nonsecure:

```
cd /opt/cloudera/parcels/CDH/share/doc/solr-doc*/example/exampledocs
                java -Durl=http://search01.example.com:8983/solr/test_coll
ection/update -jar post.jar *.xml
```

Query Sample Data

Run a query to verify that the sample data is successfully indexed and that you are able to search it.

Procedure

1. Open the Solr admin web interface in a browser by accessing the relevant URL:
   - TLS Enabled: https://*search01.example.com*:8985/solr
   - TLS Disabled: http://*search01.example.com*:8983/solr

   Replace *search01.example.com* with the name of any host running the Solr Server process. If you have security enabled on your cluster, enter the credentials for the solr@EXAMPLE.COM principal when prompted.
2. Select Cloud from the left panel.
3. From the Collection Selector drop-down menu in the left panel, select the test_collection collection.
4. Select Query from the left panel and click Execute Query. If you see results such as the following, indexing was successful:


```
"response":{"numFound":32,"start":0,"maxScore":1.0,"docs":[
      {
        "id":"SP2514N",
        "name":["Samsung SpinPoint P120 SP2514N - hard drive - 250 GB -
ATA-133"],
        "manu":["Samsung Electronics Co. Ltd."],
        "manu_id_s":"samsung",
        "cat":["electronics",
          "hard drive"],
        "features":["7200RPM, 8MB cache, IDE Ultra ATA-133",
          "NoiseGuard, SilentSeek technology, Fluid Dynamic Bearing (FDB)
 motor"],
        "price":[92.0],
        "popularity":[6],
        "inStock":[true],
        "manufacturedate_dt":"2006-02-13T15:26:37Z",
        "store":["35.0752,-97.032"],
        "manu_str":["Samsung Electronics Co. Ltd."],
        "_version_":1650678864820568064,
        "cat_str":["electronics",
          "hard drive"],
        "name_str":["Samsung SpinPoint P120 SP2514N - hard drive - 250 GB
 - ATA-133"],
        "features_str":["7200RPM, 8MB cache, IDE Ultra ATA-133",
          "NoiseGuard, SilentSeek technology, Fluid Dynamic Bearing (FDB)
 motor"],
        "store_str":["35.0752,-97.032"]},
```


Indexing Sample Tweets with Cloudera Search

After you have verified that Cloudera Search is installed and running properly, you can experiment with other methods of ingesting and indexing data. This tutorial uses tweets to demonstrate batch indexing. Continue on to the next portion of the tutorial:

- Preparing to Index Sample Tweets with Cloudera Search on page 6

To learn more about Solr, see the Apache Solr Tutorial.

Preparing to Index Sample Tweets with Cloudera Search

In this section of the Cloudera Search tutorial, you will create a collection for tweets. The remaining examples in the tutorial use the same collection, so make sure that you follow these instructions carefully.

Create a Collection for Tweets

Procedure

1. On a host with Solr Server installed, make sure that the SOLR_ZK_ENSEMBLE environment variable is set in /etc/solr/conf/solr-env.sh.

   For example:

   ```
   cat /etc/solr/conf/solr-env.sh
   ```

   ```
   export SOLR_ZK_ENSEMBLE=zk01.example.com:2181,zk02.example.com:2181,zk03
   .example.com:2181/solr
   ```

   This is automatically set on hosts with a Solr Server or Gateway role in Cloudera Manager.
2. If you are using Kerberos, kinit as the user that has privileges to create the collection:

   ```
   kinit solr@EXAMPLE.COM
   ```

   Replace EXAMPLE.COM with your Kerberos realm name.
3. Generate the configuration files for the collection, including the tweet-specific schema.xml:

   ```
   solrctl instancedir --generate $HOME/cloudera_tutorial_tweets_config
                   cp /opt/cloudera/parcels/CDH/share/doc/search*/search-
   crunch/solr/collection1/conf/schema.xml $HOME/cloudera_tutorial_tweets_c
   onfig/conf
   ```
4. Upload the configuration to ZooKeeper:

   - Security Enabled:

     ```
     solrctl --jaas $HOME/jaas.conf instancedir --create cloudera_tutorial_tw
     eets_config $HOME/cloudera_tutorial_tweets_config
     ```
   - Security Disabled:

     ```
     solrctl instancedir --create cloudera_tutorial_tweets_config $HOME/cloud
     era_tutorial_tweets_config
     ```
5. Create a new collection with two shards (specified by the -s parameter) using the named configuration (specified by the -c parameter):

   ```
   solrctl collection --create cloudera_tutorial_tweets -s 2 -c cloudera_tu
   torial_tweets_config
   ```
6. Verify that the collection is live. Open the Solr admin web interface in a browser by accessing the relevant URL:

   - TLS Enabled: https://*search01.example.com*:8985/solr/#/~cloud
   - TLS Disabled: http://*search01.example.com*:8983/solr/#/~cloud

   If you have Kerberos authentication enabled on your cluster, enter the credentials for the solr@EXAMPLE.COM principal when prompted. Replace *search01.example.com* with the name of any host running the Solr Server process. Look for the cloudera_tutorial_tweets collection to verify that it exists.
7. Prepare the configuration for use with MapReduce:

   ```
   cp -r $HOME/cloudera_tutorial_tweets_config $HOME/cloudera_tutorial_twee
   ts_mr_config
   ```

Copy Sample Tweets to HDFS

Procedure

1. Copy the provided sample tweets to HDFS. These tweets will be used to demonstrate the batch indexing capabilities of Cloudera Search:
   - Security Enabled:

     ```
     kinit hdfs@EXAMPLE.COM
     ```

     ```
     hdfs dfs -mkdir -p /user/jdoe
     ```

     ```
     hdfs dfs -chown jdoe:jdoe /user/jdoe
     ```

     ```
     kinit jdoe@EXAMPLE.COM
     ```

     ```
     hdfs dfs -mkdir -p /user/jdoe/indir
     ```

     ```
     hdfs dfs -put /opt/cloudera/parcels/CDH/share/doc/search*/examples/test-documents/sample-statuses-*.avro /user/jdoe/indir/
     ```

     ```
     hdfs dfs -ls /user/jdoe/indir
     ```
   - Security Disabled:

     ```
     sudo -u hdfs hdfs dfs -mkdir -p /user/jdoe
     ```

     ```
     sudo -u hdfs hdfs dfs -chown jdoe:jdoe /user/jdoe
     ```

     ```
     hdfs dfs -mkdir -p /user/jdoe/indir
     ```

     ```
     hdfs dfs -put /opt/cloudera/parcels/CDH/share/doc/search*/examples/test-documents/sample-statuses-*.avro /user/jdoe/indir/
     ```

     ```
     hdfs dfs -ls /user/jdoe/indir
     ```
2. Ensure that outdir is empty and exists in HDFS:

   ```
   hdfs dfs -rm -r -skipTrash /user/jdoe/outdir
   ```

   ```
   hdfs dfs -mkdir /user/jdoe/outdir
   ```

   ```
   hdfs dfs -ls /user/jdoe/outdir
   ```

What to do next

The sample tweets are now in HDFS and ready to be indexed. Continue to the next section to index the sample tweets.

Using MapReduce Batch Indexing to Index Sample Tweets

The following sections include examples that illustrate how to use MapReduce for batch indexing. Batch indexing is useful for periodically indexing large amounts of data, or for indexing a dataset for the first time. Before continuing, make sure that you have completed the procedures earlier in the tutorial.

Batch Indexing into Online Solr Servers Using GoLive

About this task

Warning: Batch indexing into offline Solr shards is not supported in environments in which batch indexing into online Solr servers using GoLive occurs.

MapReduceIndexerTool is a MapReduce batch job driver that creates a set of Solr index shards from a set of input files and writes the indexes into HDFS in a flexible, scalable, and fault-tolerant manner. Using GoLive, MapReduceIndexerTool also supports merging the output shards into a set of live customer-facing Solr servers. The following steps demonstrate these capabilities.

Procedure

1. If you are working with a secured cluster, configure your client JAAS file ($HOME/jaas.conf) as follows:

```
Client {
 com.sun.security.auth.module.Krb5LoginModule required
 useKeyTab=false
 useTicketCache=true
 principal="solr@EXAMPLE.COM";
 };
```

2. If you are using Kerberos, kinit as a user that has privileges to update the collection:

```
kinit solr@EXAMPLE.COM
```

   Replace EXAMPLE.COM with your Kerberos realm name.

3. Delete any existing documents in the cloudera_tutorial_tweets collection. If your cluster does not have security enabled, run the following commands as the solr user by adding sudo -u solr before the command:

```
solrctl collection --deletedocs cloudera_tutorial_tweets
```

4. Run the MapReduce job with the --go-live parameter. Replace *nn01.example.com* and *zk01.example.com* with your NameNode and ZooKeeper hostnames, respectively.

   - Security Enabled:

```
YARN_OPTS="-Djava.security.auth.login.config=/path/to/jaas.conf" yarn
 jar /opt/cloudera/parcels/CDH/lib/solr/contrib/mr/search-mr-job.jar org
.apache.solr.hadoop.MapReduceIndexerTool -D 'mapred.child.java.opts=-
Xmx500m' -D 'mapreduce.job.user.classpath.first=true' --log4j /opt/clou
dera/parcels/CDH/share/doc/search*/examples/solr-nrt/log4j.properties
 --morphline-file /opt/cloudera/parcels/CDH/share/doc/search*/examples/
solr-nrt/test-morphlines/tutorialReadAvroContainer.conf --output-dir hdf
s://nn01.example.com:8020/user/jdoe/outdir --verbose --go-live --zk-hos
t zk01.example.com:2181/solr --collection cloudera_tutorial_tweets hdf
s://nn01.example.com:8020/user/jdoe/indir
```

   - Security Disabled:

```
yarn jar /opt/cloudera/parcels/CDH/lib/solr/contrib/mr/search-mr-job.jar
 org.apache.solr.hadoop.MapReduceIndexerTool -D 'mapred.child.java.opts=
-Xmx500m' -D 'mapreduce.job.user.classpath.first=true' --log4j /opt/clou
dera/parcels/CDH/share/doc/search*/examples/solr-nrt/log4j.properties
 --morphline-file /opt/cloudera/parcels/CDH/share/doc/search*/examples/
solr-nrt/test-morphlines/tutorialReadAvroContainer.conf --output-dir hdf
s://nn01.example.com:8020/user/jdoe/outdir --verbose --go-live --zk-hos
```

```
t zk01.example.com:2181/solr --collection cloudera_tutorial_tweets hdf
s://nn01.example.com:8020/user/jdoe/indir
```

This command requires a morphline file, which must include a SOLR_LOCATOR directive. Any CLI parameters for --zkhost and --collection override the parameters of the solrLocator. The SOLR_LOCATOR directive might appear as follows:

```
SOLR_LOCATOR : {
  # Name of solr collection
  collection : collection_name
  # ZooKeeper ensemble
  zkHost : "zk01.example.com:2181,zk02.example.com:2181,zk03.example.com
:2181/solr"
}

morphlines : [
  {
    id : morphline1
    importCommands : ["org.kitesdk.**", "org.apache.solr.**"]
    commands : [
      { generateUUID { field : id } }

      { # Remove record fields that are unknown to Solr schema.xml.
        # Recall that Solr throws an exception on any attempt to load a
document that
        # contains a field that isn't specified in schema.xml.
        sanitizeUnknownSolrFields {
          solrLocator : ${SOLR_LOCATOR} # Location from which to fetch
Solr schema
        }
      }

      { logDebug { format : "output record: {}", args : ["@{}"] } }

      {
        loadSolr {
          solrLocator : ${SOLR_LOCATOR}
        }
      }
    ]
  }
]
```

For help on how to run the MapReduce job, run the following command:

```
yarn jar /opt/cloudera/parcels/CDH/lib/solr/contrib/mr/search-mr-*-job.jar
 org.apache.solr.hadoop.MapReduceIndexerTool --help
```

For development purposes, you can use the --dry-run option to run in local mode and print documents to stdout instead of loading them into Solr. Using this option causes the morphline to run locally without submitting a job to MapReduce. Running locally provides faster turnaround during early trial and debug sessions.

To print diagnostic information, such as the content of records as they pass through the morphline commands, enable TRACE log level diagnostics by adding the following entry to your log4j.properties file:

```
log4j.logger.org.kitesdk.morphline=TRACE
```

The log4j.properties file location can be specified by using the MapReduceIndexerTool --log4j /path/to/log4j.properties command-line option.

5. Check the job status at:

```
http://rm01.example.com:8088/ui2/#/yarn-apps/apps
```

   For secure clusters, replace http with https and port 8088 with 8090.

6. When the job is complete, run a Solr query. For example, for a Solr server running on search01.example.com, go to one of the following URLs in a browser, depending on whether you have enabled security on your cluster:

   - Security Enabled: https://search01.example.com:8985/solr/cloudera_tutorial_tweets/select?q=*:*
   - Security Disabled: http://search01.example.com:8983/solr/cloudera_tutorial_tweets/select?q=*:*

   If indexing was successful, this page displays the first 10 query results.

Batch Indexing into Offline Solr Shards

About this task

Running the MapReduce job without GoLive causes the job to create a set of Solr index shards from a set of input files and write the indexes to HDFS. You can then explicitly point each Solr server to one of the HDFS output shard directories.

Batch indexing into offline Solr shards is mainly intended for offline use-cases by advanced users. Use cases requiring read-only indexes for searching can be handled by using batch indexing without the --go-live option. By not using GoLive, you can avoid copying datasets between segments, thereby reducing resource utilization.

Procedure

1. If you are working with a secured cluster, configure your client JAAS file ($HOME/jaas.conf) as follows:

```
Client {
 com.sun.security.auth.module.Krb5LoginModule required
 useKeyTab=false
 useTicketCache=true
 principal="solr@EXAMPLE.COM";
 };
```

2. If you are using Kerberos, kinit as the user that has privileges to update the collection:

```
kinit jdoe@EXAMPLE.COM
```

   Replace EXAMPLE.COM with your Kerberos realm name.

3. Delete any existing documents in the cloudera_tutorial_tweets collection. If your cluster does not have security enabled, run the following commands as the solr user by adding sudo -u solr before the command:

```
solrctl collection --deletedocs cloudera_tutorial_tweets
```

4. Delete the contents of the outdir directory:

```
hdfs dfs -rm -r -skipTrash /user/jdoe/outdir/*
```

5. Run the MapReduce job as follows, replacing *nn01.example.com* in the command with your NameNode hostname.

   - Security Enabled:

```
YARN_OPTS="-Djava.security.auth.login.config=/path/to/jaas.conf" yarn
 jar /opt/cloudera/parcels/CDH/lib/solr/contrib/mr/search-mr-*-job.jar
 org.apache.solr.hadoop.MapReduceIndexerTool -D 'mapred.child.java.opt
s=-Xmx500m' -D 'mapreduce.job.user.classpath.first=true' --log4j /opt/
cloudera/parcels/CDH/share/doc/search*/examples/solr-nrt/log4j.propert
ies --morphline-file /opt/cloudera/parcels/CDH/share/doc/search*/exa
```

```
mples/solr-nrt/test-morphlines/tutorialReadAvroContainer.conf --outp
ut-dir hdfs://nn01.example.com:8020/user/jdoe/outdir --verbose --zk-h
ost zk01.example.com:2181/solr --collection cloudera_tutorial_tweets --s
hards 2 hdfs://nn01.example.com:8020/user/jdoe/indir
```

- Security Disabled:

```
yarn jar /opt/cloudera/parcels/CDH/lib/solr/contrib/mr/search-mr-*-job.j
ar org.apache.solr.hadoop.MapReduceIndexerTool -D 'mapred.child.java
.opts=-Xmx500m' -D 'mapreduce.job.user.classpath.first=true' --log4j /
opt/cloudera/parcels/CDH/share/doc/search*/examples/solr-nrt/log4j.pro
perties --morphline-file /opt/cloudera/parcels/CDH/share/doc/search*/
examples/solr-nrt/test-morphlines/tutorialReadAvroContainer.conf --outp
ut-dir hdfs://nn01.example.com:8020/user/jdoe/outdir --verbose --zk-h
ost zk01.example.com:2181/solr --collection cloudera_tutorial_tweets --s
hards 2 hdfs://nn01.example.com:8020/user/jdoe/indir
```

6. Check the job status at:

```
http://rm01.example.com:8088/ui2/#/yarn-apps/apps
```

For secure clusters, replace http with https and port 8088 with 8090.

7. After the job is completed, check the generated index files. Individual shards are written to the results directory with names of the form part-00000, part-00001, part-00002, and so on. This example has two shards:

```
hdfs dfs -ls /user/jdoe/outdir/results
```

```
hdfs dfs -ls /user/jdoe/outdir/results/part-00000/data/index
```

8. In the Cloudera Manager web console for the cluster, stop the Solr service (Solr service Actions Stop).
9. Identify the paths to each Solr core:

```
hdfs dfs -ls /solr/cloudera_tutorial_tweets
```

```
Found 2 items
drwxr-xr-x   - solr solr          0 2017-03-13 06:20 /solr/cloudera_tutori
al_tweets/core_node1
drwxr-xr-x   - solr solr          0 2017-03-13 06:20 /solr/cloudera_tut
orial_tweets/core_node2
```

10. Move the index shards into place.
    a) (Kerberos only) Switch to the solr user:

```
kinit solr@EXAMPLE.COM
```

    b) Remove outdated files. If your cluster does not have security enabled, run the following commands as the solr user by adding sudo -u solr before the command:

```
hdfs dfs -rm -r -skipTrash /solr/cloudera_tutorial_tweets/core_node1/dat
a/index
hdfs dfs -rm -r -skipTrash /solr/cloudera_tutorial_tweets/core_node1/dat
a/tlog
hdfs dfs -rm -r -skipTrash /solr/cloudera_tutorial_tweets/core_node2/
data/index
hdfs dfs -rm -r -skipTrash /solr/cloudera_tutorial_tweets/core_node2/
data/tlog
```

c) Change ownership of the results directory to solr. If your cluster has security enabled, kinit as the HDFS superuser (hdfs by default) before running the following command. If your cluster does not have security enabled, run the command as the HDFS superuser by adding sudo -u hdfs before the command:

```
hdfs dfs -chown -R solr /user/jdoe/outdir/results
```

d) (Kerberos only) Switch to the solr user:

```
kinit solr@EXAMPLE.COM
```

e) Move the two index shards into place. If your cluster does not have security enabled, run the following commands as the solr user by adding sudo -u solr before the command:

```
hdfs dfs -mv /user/jdoe/outdir/results/part-00000/data/index /solr/clou
dera_tutorial_tweets/core_node1/data
```

```
hdfs dfs -mv /user/jdoe/outdir/results/part-00001/data/index /solr/clou
dera_tutorial_tweets/core_node2/data
```

11. In the Cloudera Manager web console for the cluster, start the Solr service (Solr service Actions Start).

12. Run some Solr queries. For example, for a Solr server running on search01.example.com, go to one of the following URLs in a browser, depending on whether you have enabled security on your cluster:

- Security Enabled: https://search01.example.com:8985/solr/cloudera_tutorial_tweets/select?q=*:*
- Security Disabled: http://search01.example.com:8983/solr/cloudera_tutorial_tweets/select?q=*:*

If indexing was successful, this page displays the first 10 query results.