LEARNING sqoop

#sqoop

Table of Contents

About 1

Chapter 1: Getting started with sqoop 2

Remarks 2

Examples 2

Installation or Setup 2

Install Dependencies for SQOOP 2

Start and Stop Sqoop Server Services 3

Sqoop Client Configuration steps 3

Chapter 2: Connecting Sqoop to other databases/datastores 4

Introduction 4

Examples 4

Load JDBC Driver 4

Validate the connection 4

Import table to new catalog 4

Import the results of a query from a relational database into HDFS: 5

Import data directly into Hive Warehouse 5

Import data from RDBMS to HBase table 5

Chapter 3: merge data-sets imported via incremental import using Sqoop 6

Remarks 6

Examples 6

Import New Data - append mode 6

Import New as well as Updated Data - lastmodified mode 6

Chapter 4: Sqoop Export 8

Examples 8

Sqoop Export basic example 8

Chapter 5: Sqoop Import 9

Syntax 9

Remarks 9

Examples 9

Import RDBMS Table to HDFS 9

Import to a particular directory in HDFS 10

Specify parent HDFS directory for Sqoop job 10

Import subset of RDBMS Table to HDFS 10

Using --where tag 10

Using Free Form Query 11

Import-All-Tables 11

Credits 12

About

Chapter 1: Getting started with sqoop

Remarks

SQOOP Server configuration files are stored in server/config directory of distributed artifact along side with other configuration files of Tomcat (to host SQOOP server).

File `sqoop_bootstrap.properties` specifies which configuration provider should be used for loading configuration for rest of Sqoop server. Default value PropertiesConfigurationProvider should be sufficient.

Second configuration file `sqoop.properties` contains remaining configuration properties that can affect Sqoop server. File is very well documented, so check if all configuration properties fits your environment. Default or very little tweaking should be sufficient most common cases.

Examples

Installation or Setup

Sqoop ships as one binary package however it’s compound from two separate parts client and server. You need to install server on single node in your cluster. This node will then serve as an entry point for all connecting Sqoop clients. Server acts as a mapreduce client and therefore Hadoop must be installed and configured on machine hosting Sqoop server. Clients can be installed on any arbitrary number of machines. Client is not acting as a mapreduce client and thus you do not need to install Hadoop on nodes that will act only as a Sqoop client.

Copy Sqoop artifact on machine where you want to run Sqoop server. This machine must have installed and configured Hadoop. You don’t need to run any Hadoop related services there, however the machine must be able to act as an Hadoop client.

```
# Extract Sqoop  tar
tar -xf sqoop-<version>-bin-hadoop<hadoop-version>.tar.gz

# Move decompressed content to any location
  (you can also setup soft links to sqoop directory)
mv sqoop-<version>-bin-hadoop<hadoop version>.tar.gz /opt/apache/sqoop

# Change working directory
cd /opt/apache/sqoop
```

Install Dependencies for SQOOP

You need to install Hadoop libraries into Sqoop server war file. Sqoop provides convenience script addtowar.sh to do so.

If you have installed Hadoop in usual location in /usr/lib and executable hadoop is in your path, you can use automatic Hadoop installation procedure:

```
./bin/addtowar.sh -hadoop-auto
```

In case that you have Hadoop installed in different location, you will need to manually specify Hadoop version and path to Hadoop libraries. You can use parameter -hadoop-version for specifying Hadoop major version,

```
./bin/addtowar.sh -hadoop-version 2.0 -hadoop-path /usr/lib/hadoop-
common:/usr/lib/hadoop-hdfs:/usr/lib/hadoop-yarn
```

- Installed required JDBC jars for sqoop to connect to database

```
./bin/addtowar.sh -jars /path/to/jar/mysql-connector-java-*-bin.jar
```

Start and Stop Sqoop Server Services

```
./bin/sqoop.sh server start
./bin/sqoop.sh server stop
```

Sqoop Client Configuration steps

Copy Sqoop distribution artifact on target machine and unzip it in desired location. You can start client with following command:

```
bin/sqoop.sh client
```

Sqoop 2 client have ability to load resource files similarly as other command line tools. At the beginning of execution Sqoop client will check existence of file .sqoop2rc in home directory of currently logged user. If such file exists, it will be interpreted before any additional actions. This file is loaded in both interactive and batch mode. It can be used to execute any batch compatible commands.

Example resource file:

```
# Configure our Sqoop 2 server automatically
set server --host sqoop2.company.net

# Run in verbose mode by default
set option --name verbose --value true
```

Read Getting started with sqoop online: https://riptutorial.com/sqoop/topic/1050/getting-started-with-sqoop

Chapter 2: Connecting Sqoop to other databases/datastores

Introduction

Shows how a sqoop script could be used to import data from various datastores/databases.

Examples

Load JDBC Driver

For accessing the MS SQL Server database Sqoop requires an additional JDBC driver which can be downloaded from Microsoft. The following steps will install MSSQL Server JDBC driver to Sqoop:

```
wget 'http://download.microsoft.com/download/0/2/A/02AAE597-3865-456C-
AE7F-613F99F850A8/sqljdbc_4.0.2206.100_enu.tar.gz'

tar -xvzf sqljdbc_4

cp sqljdbc_4.0/enu/sqljdbc4.jar /usr/hdp/current/sqoop-server/lib/
```

Validate the connection

To check that the connection to the server is valid:

```
sqoop list-tables --connect "jdbc:sqlserver://<server_ip>:1433;database=<database_name>" --
                  username <user_name>
                  --password <password>
```

Before doing this it is recommended to check if SqlServer's configuration allows remote access to 1433 port.

Open **Configuration Manager** => **SQL Server Network configuration** => **Protocols for MSSQLSERVER** and check that Protocol is enabled and the needed IP and port is enabled and is active.

Import table to new catalog

To import data from SQL Server to Hadoop:

```
sqoop import --table TestTable
             --connect "jdbc:sqlserver://192.168.1.100:1433;database=Test_db"
             --username user
             --password password
             --split-by id
             --target-dir /user/test
```

- split-by – used mandatory if no primary key
- target-dir – new catalog, which does not exist yet

Import the results of a query from a relational database into HDFS:

Query can be used instead of table in import operation:

```
sqoop import --query 'select Id,Message from TestTable where
             $CONDITIONS' --where 'id>100'
             --connect "jdbc:sqlserver://192.168.1.100:1433;database=Test_db
             --username user
             --password password
             --split-by id
             --target-dir /user/test/
             --fields-terminated-by '\t'
```

- where $CONDITIONS - mandatory even if where condition does not exist
- split-by - mandatory, specifies the column for the split operation. Used to split tasks in import MapReduce job

Import data directly into Hive Warehouse

The data can be imported directly into Hive:

```
sqoop import --hive-import
             --table EventLog
             --connect "jdbc:sqlserver://192.168.1.99:1433;database=Test_db"
             --username user
             --password password
             --split-by id
             --fields-terminated-by '\t'
```

Import data from RDBMS to HBase table

The following sqoop command will be used to import the data from RDBMS table into HBase table, if the table does not exists in HBase it will create a new table and import the data into this table

```
sqoop import \
        --query 'select emp_id, emp_name, emp_sal from employee where $CONDITIONS' \
        --connect "jdbc:sqlserver://192.168.1.99:1433;database=test_db" \
        --username username \
        --password password \
        --hbase-create-table \
        --hbase-table employee_table \
        --hbase-row-key emp_id
```

Read Connecting Sqoop to other databases/datastores online:
https://riptutorial.com/sqoop/topic/3930/connecting-sqoop-to-other-databases-datastores

Chapter 3: merge data-sets imported via incremental import using Sqoop

Remarks

Sqoop incremental import comes into picture because of a phenomenon called CDC i.e. **Change Data Capture.** Now what is CDC?

> CDC is a design pattern that captures individual data changes instead of dealing with the entire data. Instead of dumping our entire database, using CDC, we could capture just the data changes made to the master database.

For example : If we are dealing with a data problem, say, 1 lakh data entries coming into the RDBMS daily and we have to get this data in Hadoop on a daily basis then we would want to just get the newly added data, as importing the complete RDBMS data daily to Hadoop will be an overhead and delays the availability of data also. For a detailed explanation go through this link.

Examples

Import New Data - append mode

If you are only adding new rows in your RDBMS (*not updating existing data*)

You need two additional parameters:

- `--check-column` : A column name that should be checked for newly appended data. `Integer` would be a suitable data type for this column.
- `--last-value` : The last value that successfully imported into Hadoop. All the newly added data after this value will be imported.

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/testdb \
--username sqoop \
--password sqoop \
--table employee \
--incremental append \
--check-column id \
--last-value 100
```

Import New as well as Updated Data - lastmodified mode

If you are adding new rows and updating existing data.

You need two additional parameters:

- `--check-column` : A column name that should be checked for newly appended and updated data. `date`, `time`, `datetime` and `timestamp` are suitable data types for this column.

- `--last-value` : The last value that successfully imported into Hadoop. All the newly added and updated data after this value will be imported.

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/testdb \
--username sqoop \
--password sqoop \
--table employee \
--incremental lastmodified \
--check-column last_update_date \
--last-value "2015-10-20 06:00:01"
```

Read merge data-sets imported via incremental import using Sqoop online: https://riptutorial.com/sqoop/topic/5673/merge-data-sets-imported-via-incremental-import-using-sqoop

Chapter 4: Sqoop Export

Examples

Sqoop Export basic example

The export tool exports a set of files from HDFS back to an RDBMS. The target table must already exist in the database. The input files are read and parsed into a set of records according to the user-specified delimiters.

Example :

```
sqoop export \
--connect="jdbc:<databaseconnector>" \
--username=<username> \
--password=<password> \
--export-dir=<hdfs export directory> \
--table=<tablename>
```

Read Sqoop Export online: https://riptutorial.com/sqoop/topic/6999/sqoop-export

Chapter 5: Sqoop Import

Syntax

- `<rdbms-jdbc-url>` // RDBMS JDBC URL
- `<username>` // Username of the RDBMS database
- `<password>` // Password of the RDBMS database
- `<table-name>` // RDBMS database table
- `<hdfs-home-dir>` // HDFS home directory
- `<condition>` // Condition that can be expressed in the form of a SQL query with a WHERE clause.
- `<sql-query>` // SQL Query
- `<target-dir>` // HDFS Target Directory

Remarks

Sqoop is a Hadoop Command Line tool that imports table from an RDBMS data source to HDFS and vice versa. It generates a Java class which allows us to interact with the imported data. Each row from a table is saved as a separate record in HDFS. Records can be stored as text files or in binary representation as Avro or Sequence Files. There are 2 versions of sqoop :

> Sqoop1 and Sqoop2

Sqoop1 is the widely accepted tool and is recommended for production environments. Find the comparison between Sqoop1 and Sqoop2 as stated on Cloudera's website.

Examples

Import RDBMS Table to HDFS

```
sqoop import \
--connect <rdbms-jdbc-url> \
--username <username> \
--password <password> \
--table <table-name>
```

Example with Mysql:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/testdb \
--username root \
--password root \
--table employees
```

CSV file with the imported data will be created under **employees** directory under home directory.

Inspect using command:

```
hadoop fs -cat <hdfs-home-dir>/employees/part-m-*
```

Import to a particular directory in HDFS

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/testdb \
--username root \
--password root \
--table emplyoees \
--target-dir /dev/data/employees
```

This will generate CSV file under `/dev/data/employees` directory.

Specify parent HDFS directory for Sqoop job

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/testdb \
--username root \
--password root \
--table emplyoees \
--warehouse-dir /dev/warehouse/
```

`--warehouse-dir` tag in above command will change your home directory to `/dev/warehouse/`

Import subset of RDBMS Table to HDFS

Using `--where` tag

```
sqoop import \
--connect <rdbms-jdbc-url> \
--username <username> \
--password <password> \
--table <table-name> \
--where "<condition>"
```

Example with Mysql:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/testdb \
--username root \
--password root \
--table employees \
--where "country = 'INDIA'"
```

Any special functions, conversions, or even user-defined functions can be used in `--where` clause.

Using Free Form Query

```
sqoop import \
--connect <rdbms-jdbc-url> \
--username <username> \
--password <password> \
--query <sql-query> \
--target-dir <target-dir>
```

Example with Mysql:

```
sqoop import \
--connect jdbc:mysql://mysql.example.com/testdb \
--username root \
--password root \
--query "SELECT emplyoees.name, \
                address.city \
                FROM emplyoees \
                JOIN address USING(emp_id) \
                WHERE \$CONDITION" \
--taget-dir /dev/data/employees
```

Import-All-Tables

SQOOP provides facility to import all tables

```
sqoop import-all-tables \
--connect <rdbms-jdbc-url> \
--username <username> \
--password <password> \
--hive-import \
--create-hive-table \
--hive-database <dbname> \
--warehouse-dir <warehouse-dir>
```

Important points to note on differences between import and import-all-tables:

Need to provide --warehouse-dir=//stage.db database name as input parameter to download all tables into a database. In sqoop import we will be providing only --target-dir not the --warehouse-dir

Example:

```
sqoop import-all-tables --connect="jdbc:mysql://serverip:3306/dbname"
--username=xxx --password=yyy
-m 1 --hive-import
--hive-overwrite
--create-hive-table
--hive-database dbname
--hive-home /user/hive/warehouse
--warehouse-dir=/user/hive/warehouse/retail_stage.db
```

Read Sqoop Import online: https://riptutorial.com/sqoop/topic/5057/sqoop-import

Credits

S. No	Chapters	Contributors
1	Getting started with sqoop	Community, dev ツ, Venkata Karthik
2	Connecting Sqoop to other databases/datastores	Alex, Ani Menon, Prasad Khode
3	merge data-sets imported via incremental import using Sqoop	dev ツ, NeoWelkin
4	Sqoop Export	Kannan Kandasamy
5	Sqoop Import	dev ツ, Kannan Kandasamy, NeoWelkin