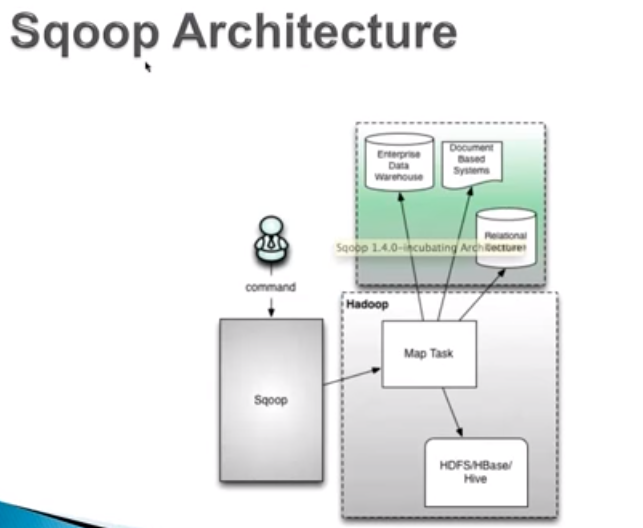
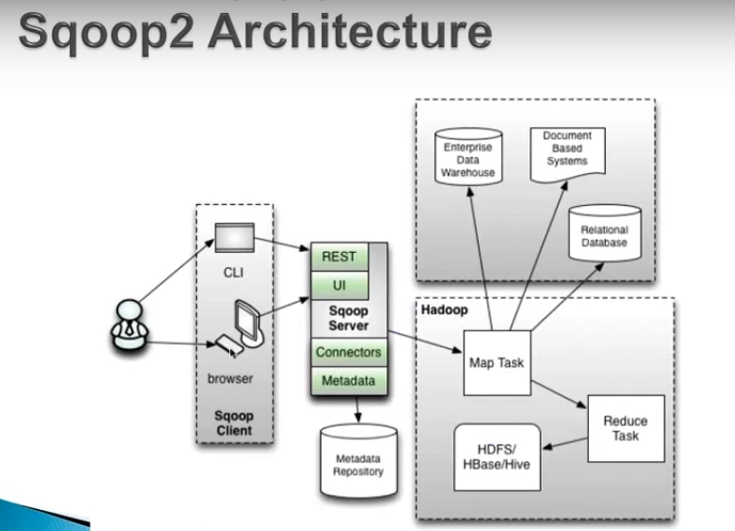
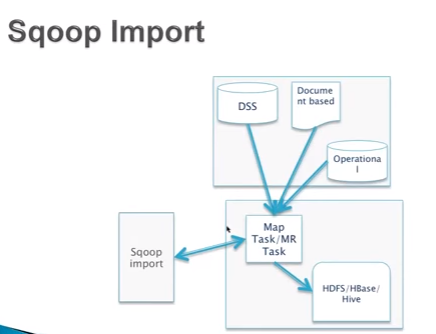
Quick Ref : Sqoop







|  |  |  |
| --- | --- | --- |
| **S.No** | **Topic** | **Desc** |
|  | **Web Reference** |  |
|  | Sqoop vs Sqoop 2 | <http://www.cloudera.com/documentation/enterprise/5-4-x/topics/cdh_ig_sqoop_vs_sqoop2.html> |
|  | Sqoop version 1.4 guide | <https://sqoop.apache.org/docs/1.4.2/SqoopUserGuide.html> |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | **General Info** |  |
|  | **Pre-request**  <check the MySQL availability> | > mysql -h ip-172-31-13-154 -u sqoopuser -p |
|  | **Pre-request**  <JDBC Jar connector> | # Pre-request to export/import data between MySQL and HDFS. Setup MySQL JDBC connector and assign JDBC Connector to sqoop libary   1. JDBC (MySQL, Oracle, etc) Jar and DB username & Password 2. For import/export between HDFS and RDBMS using Sqoop, make sure Sqoop **version is compatible** with MySQL Connector jar version otherwise, it will not work 3. Ex: sqoop 1.4.2 will work with below MySQL jar version 4. >ls –ltr /usr/share/java/mysql-connector-java-5.1.17.jar |
|  | **Pre-request**  <JDBC Jar connector> | # how to setup soft link for my-sql-connector   1. >ls –ltr /usr/share/java/mysql-connector-java.jar # To check the soft link for the mysql jar version 2. If you have in-compatiable version then download the correct version and ‘unlink’ the old version of mysql-connector-jar version and connect the new version of mysql-connector 3. >unlink /usr/share/java/mysql-connector-java.jar 4. >ln –s /usr/share/java/mysql-connector-java-<verno>-bin.jar /usr/share/java/mysql-connector-java.jar 5. >ls –ltr /usr/share/java/mysql-connector-java.jar   <https://www.youtube.com/watch?v=G2fnCH_QU_U&index=51&list=PLf0swTFhTI8qKo5rHQ5XmcAWjkMZnfo6I> |
|  | **Pre-request**  **<** Sqoop libraries> | # how to setup soft link for MySQL JDBC Connector with Sqoop libraries  # After this we will have MySQL connector in the sqoop libraries   1. >ls –ltr /usr/hdp/2.3.0.0-2557/sqoop/lib/mysql-connector-java.jar 2. >ls –s /usr/share/java/mysql-connector-java.jar /usr/hdp/2.3.0.0-2557/sqoop/lib/mysql-connector-java.jar 3. >ls –ltr /usr/hdp/2.3.0.0-2557/sqoop/lib/mysql-connector-java.jar |
| 1 | **Sqoop vs Sqoop2** | **Sqoop:**   1. Client Only. (No client server). 2. Keep the JDBC jar in client, wherever you run the job 3. It supports Command Line Only (CLI Only) 4. It uses Map Only (No Reducer). Mapper is responsible for both reading and writing data into source & target. 5. Not secured: Sqoop command uses human readable RDBMS user ID and password. if one has access to sqoop command, he will have access to all JDBC jars and RDBMS 6. Not easily extensible and no separation of duties: Both read from source and write to target uses only Mapper (No Reducer)   **Sqoop2:**   1. Sqoop 2: Client/server architecture. It requires Sqoop server and Sqoop Client. 2. Developer use clients 3. Admin setup server and keep JDBC jar files in server. (Jar files will reside in servers) 4. Sqoop2 supports both Command Line Interface (CLI) and web UI (like Hue, etc). 5. It uses both Mapper and Reducer. Mapper is responsible to read the data from source and Reducer is responsible to write data into target 6. Secured: Sqoop 2 uses ‘Metadata Repository’ to store all the configuration. It brings more security compare to Sqoop1 7. Secured: Administrator will setup ‘connector’ in ‘Metadata Repository’ and share the connector name. so that those who are using sqoop command can use connector to import/export. We cannot get Login credential from the connector. 8. Extensible and separation of duties. |
| 2 | **About Sqoop1** | 1. Sqoop is export & Import utility between Hadoop and ‘any’ RDBMS. Sometime Hadoop admin has to perform this tasks 2. Sqoop uses MapReduce as a framework to interact with HDFS/Hadoop. It uses only Map (not Reduce) 3. Sqoop uses JDBC connectivity to interact with RDBMS |
| 3 | The term ‘source’ & ‘sync’ | 1. Source means source DB (it can be RDBMS or HDFS) 2. Sync means Target DB (it can be RDBMS or HDFS) 3. Source & Sync connected by JDBC |
| 4 | Parameter & log file | Sqoop doesn’t store log files anywhere. There is no parameter file and log file. Since it is using MapReduce, we can get required info from mapreduce log |
|  | HCatalog | Might not be able to read HCatalog |
|  | Delta | Combination of Hql and Sqoop to get delta |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | **General Command** |  |
| 1 | Prerequest for the script  <port> | # 3306  - for JDBC Listener – To export/import data . For example, if moving data from MySQL, TCP port 3306 must be open  # 16000 - Connection between Sqoop and the metastore |
| 2 | **About sqoop import** | 1. Sqoop import is to get data from conventional databases (RDBMS), NoSQL, document based databases into HDFS eco system. 2. HDFS eco system could be a Hbase, Casendra, Hive tables or HDFS directory 3. It uses MapReduce (Map) framework to load data in parallel. The default mapper is 4 |
| 2.1 | Run Sqoop Import | # Importing from RDBMS to HDFS  # To go to new line in the script. Use ‘\’ before you press ‘enter’  # Directory should not already exists  # Port # 3306 is optional with ip, if 3306 is default and already port is open. Usually default port will open after hive configuration  # demo is the database name, cards is a table name, suit is a column name  # The below script will import the table ‘cards’ from DB ‘demo’ to target directory with following delimiters ‘|’  **Option1 (for password): use –P (Capital P), it will prompt for password:**  sqoop list-databases "jdbc:mysql://ip-10-0-0-15/demo" \  --username demo\_user –P \  **Option2 (for password) – directly give password:**  sqoop import --connect "jdbc:mysql://ip-10-0-0-15/demo" \  --username demo\_user \  --password hadoop20 \  --table cards \  --target-dir /user/hduser/sqoop\_cards/ \  --input-fields-terminated-by '|' \  --input-lines-terminated-by '\n' \  --num-mappers 4 \  --split-by suit \  --outdir java\_files  A different example, consider the below ip has pwd less access to my sql then use command as follows   * sqoop import --connect jdbc:mysql://ip-10-0-0-15/demo – table cards –m 1   <https://www.youtube.com/watch?v=6K3aHZQavkQ> |
|  | Practical 1  *<To list all the Databases>* | # We can set parameter instead of hard code username, connector, etc  # sqoopuser is the username.  # As soon as you enter the command 1, it will prompt for password then show all the databases (from RDBMS) for which sqoopuser has access  # Because you are using ‘list-databases’ command  **Command1:**  >sqoop list-databases  --connect "jdbc:mysql://ip-172-31-13-154.ec2.internal:3306"  --username sqoopuser -P |
|  | Practical 2  *<To list all tables from RDBMS databse>* | # sqoopuser is the username, sqoopex is the DB name  # As soon as you enter the command 1, it will prompt for password then show all the tables under sqoopex Database:  # Because you are using ‘list-tables’ command  **Command1:**  >sqoop list-tables \  --connect "jdbc:mysql://ip-172-31-13-154.ec2.internal:3306/sqoopex" \  --username sqoopuser -P |
|  | Practical 3  *<Create table in Hive to import data from RDBMS>* | # Do NOT Create table in Hive. Sqoop will automatically create table in hive during Import. Also it uses MySQL datatypes  # describe formatted <hive\_table>; -- To get the location of table/directory  # Use hive-import and hive-table options. Otherwise even import succeeded, you can see the data in CLI but not from Hive/Hue  # kumar123 is DB, hive\_salary is table name.  >sqoop import \  --connect "jdbc:mysql://ip-172-31-13-154.ec2.internal:3306/sqoopex" \  --username sqoopuser \  --password NHkkP876rp \  --table salary \  --hive-import \  --hive-table kumar123.hive\_salary \  --target-dir hdfs://ip-172-31-53-48.ec2.internal:8020/apps/hive/warehouse/kumar123.db/hive\_salary \  --fields-terminated-by '|' \  --lines-terminated-by '\n' \  --num-mappers 1 \ |
| 2.2 | Execution steps | 1. Generate custom DBWritable class reading metadata of table 2. Connect to database – default 4 concurrent connections 3. Read and split the data using custom DBWritable class 4. Load data into HDFS |
| 2.3 | Split logic | # Usually split-by should be primary key in the import script, if no primary key, sqoop will use only one mapper by default, it will screw the performance   1. Uses primary key or unique key 2. Get minimum and maximum value 3. Based on minimum and maximum value, it will Compute range based on number of map tasks(default is 4) 4. Process mutual exclusive data in parallel 5. Without primary/unique keys import process only uses one map tasks |
|  |  |  |
| 3 | Run Sqoop export  <pending> | #Data will be copied from hive table cards.single\_deck to mysql table demo.cards  sqoop export --connect "jdbc:mysql://ip-10-0-0-15/demo" \  --username demo\_user \  --password hadoop20 \  --table cards \  --export-dir /user/hive/warehouse/cards.db/single\_deck \  --input-fields-terminated-by '|' \  --input-lines-terminated-by '\n' \  --num-mappers 2 \  --batch \  --outdir java\_files |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. Your company stores user profile records in an OLTP databases. You want to join these records

with web server logs you have already ingested into the Hadoop file system.

What is the best way to obtain and ingest these user records?

1. Ingest with Hadoop streaming
2. Ingest using Hive's IQAD DATA command
3. **Ingest with sqoop import**
4. Ingest with Pig's LOAD command
5. Ingest using the HDFS put command