PROOF OF CONCEPT FOR SQOOP INCREMENTAL
AND
STEPS TO BE TAKEN TO EXECUTE SQOOP JOBS ON OOZIE

- created by arshsheth22@gmail.com

This document attempts to explain how sqoop can be used to provide upserts (update and inserts) in hive from mysql database. Additionally, this document also explains how the entire process can be automated by writing workflow in oozie.

Certain precautions are needed to be observed when executing sqoop jobs in oozie. These are noted down at the end.

Dataset for this document has been borrowed from www.kaggle.com. Please refer database.csv in spacex-missions.zip. First, we will load the data into mysql and perform some initial processing like adding timestamp and primary key

1. Create database and table in mysql

```
CREATE DATABASE spacex; USE spacex;

CREATE TABLE spacex_missions (
Flight_Number VARCHAR(10),
Launch_Date VARCHAR(20),Launch_Time VARCHAR(10),Launch_Site VARCHAR(40),Vehicle_Type
VARCHAR(50),
Payload_Name VARCHAR(50),Payload_Type VARCHAR(50),Payload_Mass_kg
VARCHAR(15),Payload_Orbit VARCHAR(50),
Customer_Name VARCHAR(40),Customer_Type VARCHAR(40),Customer_Country VARCHAR(40),
Mission_Outcome VARCHAR(40),Failure_Reason VARCHAR(40),
Landing_Type VARCHAR(40),Landing_Outcome VARCHAR(40));
```

2. Load data into the table

```
LOAD DATA INFILE '/home/cloudera/Desktop/ajinkya/datasets/spacex-
missions/database.csv'
INTO TABLE spacex_missions
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

3. Create working table

```
CREATE TABLE spacex_missions_w AS
SELECT Flight_Number,
STR_TO_DATE(Concat(Launch_Date," ",Launch_Time),'%d %M %Y %H:%i') as
Launch_DateTime, Launch_Site, Vehicle_Type, Payload_Name, Payload_Type,
CAST(Payload_Mass_kg AS DECIMAL(10,3)) as Payload_Mass_kg,
Payload_Orbit, Customer_Name, Customer_Type, Customer_Country, Mission_Outcome,
Failure_Reason, Landing_Type, Landing_Outcome FROM spacex_missions
```

4. Add timestamp and primarykey to the table

```
ALTER TABLE spacex_missions_w
ADD COLUMN pid INT(5) UNSIGNED PRIMARY KEY AUTO_INCREMENT;
```

```
ALTER TABLE spacex_missions_w
add column Record_TS timestamp default current_timestamp on update
current_timestamp;
```

KNOWN BUG: If Record_TS is not populated with current_timestamp and instead is populated with '0000', truncate spacex_missions_w and write

```
INSERT INTO spacex_missions_w
STR_TO_DATE(Concat(Launch_Date," ",Launch_Time),'%d %M %Y %H:%i') as
Launch_DateTime, Launch_Site, Vehicle_Type, Payload_Name, Payload_Type,
CAST(Payload_Mass_kg AS DECIMAL(10,3)) as Payload_Mass_kg,
Payload_Orbit, Customer_Name, Customer_Type, Customer_Country, Mission_Outcome,
Failure_Reason, Landing_Type, Landing_Outcome,
00, CURRENT_TIMESTAMP FROM spacex_missions
```

'spacex_missions_w' is our working table in mysql; we will create its replica in hive. We will be constantly inserting and updating this table in mysql. After using four step incremental approach, these changes will be reflected in Hive.

6. Replica of mysql table in hive

```
sqoop import \
--connect jdbc:mysql://10.170.245.155:3306/spacex \
--driver com.mysql.jdbc.Driver \
--username root \
--password cloudera \
--table spacex_missions_w \
--hive-import --create-hive-table \
--hive-table spacex.missions_base -m 1
```

```
In case we encounter '0000-00-00 00:00:00' values in timestamp column then we can
use jdbc:mysql://10.170.245.155:3306/spacex?zeroDateTimeBehavior=convertToNull to
convert into to null
```

7. Update mysql data

Now we make changes in mysql table worldbase and load only the updated data into hive using sqoop incremental option. Data in mysql:

```
mysql> select pid,record_ts,flight_number,customer_name,
    -> mission_outcome from spacex_missions_w LIMIT 10;
+------+---------------------+---------------+---------------+-----------------+
| pid  | record_ts           | flight_number | customer_name | mission_outcome |
+------+---------------------+---------------+---------------+-----------------+
| 2018 | 2017-03-22 03:24:16 | F1-1          | DARPA         | Failure         |
| 2019 | 2017-03-22 03:24:16 | F1-2          | DARPA         | Failure         |
| 2020 | 2017-03-22 03:24:16 | F1-3          | ORS           | Failure         |
| 2021 | 2017-03-22 03:24:16 | F1-3          | NASA          | Failure         |
| 2022 | 2017-03-22 03:24:16 | F1-3          | Celestis      | Failure         |
| 2023 | 2017-03-22 03:24:16 | F1-4          |               | Success         |
| 2024 | 2017-03-22 03:24:16 | F1-5          | ATSB          | Success         |
| 2025 | 2017-03-22 03:24:16 | F9-1          |               | Success         |
| 2026 | 2017-03-22 03:24:16 | F9-2          | NASA          | Success         |
| 2027 | 2017-03-22 03:24:16 | F9-3          | NASA          | Success         |
+------+---------------------+---------------+---------------+-----------------+
```

```
UPDATE spacex_missions_w SET customer_name='NA' WHERE pid=2023, pid=2025;
```

Updated mysql:

```
mysql> select pid,record_ts,flight_number,customer_name,
    ->  mission_outcome from spacex_missions_w LIMIT 10;
+------+---------------------+---------------+---------------+-----------------+
| pid  | record_ts           | flight_number | customer_name | mission_outcome |
+------+---------------------+---------------+---------------+-----------------+
| 2018 | 2017-03-22 03:24:16 | F1-1          | DARPA         | Failure         |
| 2019 | 2017-03-22 03:24:16 | F1-2          | DARPA         | Failure         |
| 2020 | 2017-03-22 03:24:16 | F1-3          | ORS           | Failure         |
| 2021 | 2017-03-22 03:24:16 | F1-3          | NASA          | Failure         |
| 2022 | 2017-03-22 03:24:16 | F1-3          | Celestis      | Failure         |
| 2023 | 2017-03-22 03:44:29 | F1-4          | NA            | Success         |
| 2024 | 2017-03-22 03:24:16 | F1-5          | ATSB          | Success         |
| 2025 | 2017-03-22 03:44:29 | F9-1          | NA            | Success         |
| 2026 | 2017-03-22 03:24:16 | F9-2          | NASA          | Success         |
| 2027 | 2017-03-22 03:24:16 | F9-3          | NASA          | Success         |
+------+---------------------+---------------+---------------+-----------------+
```

We can see record_ts has been updated by 20mins for updated columns. We will use that time in sqoop incremental import.

8. Create incremental table

In order to support an on-going reconciliation between current records in HIVE and new change records, two tables should be defined: missions_base and missions_inc

```
sqoop job \
--create spacex_import_job -- import \
--connect jdbc:mysql://quickstart.cloudera:3306/spacex \
--driver com.mysql.jdbc.Driver --username root \
--password cloudera --table spacex_missions_w \
--hive-import --create-hive-table --hive-table spacex.missions_inc -m 1 \
--check-column Record_TS --incremental lastmodified \
--last-value "2017-03-22 03:40:00"

sqoop job --exec spacex_import_job
```

We don't need to reset last-value while running the job next time as sqoop implicitly sets it.

9. Check data in hive

In hive we have two tables missions_base which contains original data and missions_inc which contains updated data.

```
hive> select pid,record_ts,flight_number,customer_name,
    > mission_outcome from missions_base LIMIT 10;
OK
2018    2017-03-22 03:24:16.0   F1-1    DARPA   Failure
2019    2017-03-22 03:24:16.0   F1-2    DARPA   Failure
2020    2017-03-22 03:24:16.0   F1-3    ORS     Failure
2021    2017-03-22 03:24:16.0   F1-3    NASA    Failure
2022    2017-03-22 03:24:16.0   F1-3    Celestis        Failure
2023    2017-03-22 03:24:16.0   F1-4            Success
2024    2017-03-22 03:24:16.0   F1-5    ATSB    Success
2025    2017-03-22 03:24:16.0   F9-1            Success
2026    2017-03-22 03:24:16.0   F9-2    NASA    Success
2027    2017-03-22 03:24:16.0   F9-3    NASA    Success
Time taken: 0.068 seconds, Fetched: 10 row(s)
hive> select pid,record_ts,flight_number,customer_name,
    > mission_outcome from missions_inc LIMIT 10;
OK
2023    2017-03-22 03:44:29.0   F1-4    NA      Success
2025    2017-03-22 03:44:29.0   F9-1    NA      Success
Time taken: 0.065 seconds, Fetched: 2 row(s)
```

10. Reconcile View

This view combines record sets from both the Base (base_table) and Change (incremental_table) tables and is reduced only to the most recent records for each unique "id".
Create reconcile view of missions_base and missions_inc

```
CREATE VIEW missions_reconcile_view AS
SELECT t1.* FROM
    (SELECT * FROM missions_base
     UNION ALL
     SELECT * from missions_inc) t1
JOIN
    (SELECT pid, max(record_ts) max_ts FROM
        (SELECT * FROM missions_base
         UNION ALL
         SELECT * from missions_inc) t_temp
     GROUP BY pid) t2
ON t1.pid = t2.pid AND t1.record_ts = t2.max_ts;
```

11. Compaction of Data

The reconcile_view now contains the most up-to-date set of records and is now synchronized with changes from the RDBMS source system. Before creating this table, any previous instances of the table should be dropped as in the example below.

```
DROP TABLE IF EXISTS missions_reporting;
CREATE TABLE missions_reporting AS
SELECT * FROM missions_reconcile_view;
```

12. Purging incremental table data and reloading data into base table

```
DROP TABLE IF EXISTS missions_inc;
hdfs fs –rm –r /user/hive/warehouse/spacex.db/missions_inc
DROP TABLE missions_base;
CREATE TABLE missions_base LIKE missions_reporting
INSERT OVERWRITE TABLE missions_base SELECT * FROM missions_reporting;
```

```
hive> select pid,record_ts,flight_number,customer_name,
    > mission_outcome from missions_base LIMIT 10;
OK
2018    2017-03-22 03:24:16.0   F1-1    DARPA   Failure
2019    2017-03-22 03:24:16.0   F1-2    DARPA   Failure
2020    2017-03-22 03:24:16.0   F1-3    ORS     Failure
2021    2017-03-22 03:24:16.0   F1-3    NASA    Failure
2022    2017-03-22 03:24:16.0   F1-3    Celestis
2023    2017-03-22 03:44:29.0   F1-4    NA      Success
2024    2017-03-22 03:24:16.0   F1-5    ATSB    Success
2025    2017-03-22 03:44:29.0   F9-1    NA      Success
2026    2017-03-22 03:24:16.0   F9-2    NASA    Success
2027    2017-03-22 03:24:16.0   F9-3    NASA    Success
Time taken: 0.069 seconds, Fetched: 10 row(s)
```

This entire process can be automated by designing a oozie workflow. We will desgin oozie workflow in Hue.

At the time of writing this document, there are three strategies known so as to automate incremental updates on oozie.

1) Using SQOOP metastore

When SQOOPing updated data from saved job into HIVE, it was observed that SQOOP automatically updated timestamp parameter for `--last-value` for next incremental import.
Now we may ask this question where does SQOOP store this timestamp?
The answer to this question is sqoop metastore. A particular SQOOP job is stored on a metastore of a particular node. There is a difference between invoking SQOOP command via CLI on an edgenode and invoking SQOOP via OOZIE. OOZIE may invoke SQOOP commands on any node and not necessarily on the node which has SQOOP job and metastore saved. This problem can be tackled by using `--meta-store` option in SQOOP command. This sets a single meta-store to be used for OOZIE. Additional research may be needed.

2) Calculating latest timestamp using Hive Query

In this strategy, before SQOOPing data, we calculate the latest timestamp of base_table in HIVE using SELECT MAX() query. The output is echoed in shell. And by using capture-output facility of OOZIE we directly pass this value into `--last-value` of SQOOP command. We will be using this strategy in this document.

3) Using additional column "process_date"

Review needed.

Designing the DAG on OOZIE:
Description of columns present in the scripts below-

record_ts : timestamp of the record
pid : primary key

1) Shell Action

Execute IMPALA query and echo the output

```
if [ "$(whoami)" = "yarn" ]; then
 export USER=yarn
 export PYTHON_EGG_CACHE=/tmp/impala-shell-python-egg-cache-${USER}
fi
```

```
Q0=$(impala-shell -i "localhost" -q "invalidate metadata;" )
echo "$Q0"

QUERY=$(impala-shell -i "localhost" -q "select max(record_ts) from
spacex.missions_base;")
Q1=`echo "$QUERY" | egrep -o "[0-9][0-9][0-9][0-9]-[0-2][0-9]-[0-3][0-9] [0-2][0-9]:
[0-6][0-9]:[0-9][0-9]"`
echo "MAXTS=$Q1"
```

Explanation of script:

We are setting environment variables to yarn because OOZIE executes its actions as YARN and not as CLOUDERA. There may be a conflict between `whoami` and `$USER`.

```
[cloudera@quickstart ~]$ sudo -u yarn whoami
yarn
[cloudera@quickstart ~]$ whoami
cloudera
[cloudera@quickstart ~]$ sudo -u yarn echo $USER
cloudera
[cloudera@quickstart ~]$ echo $USER
cloudera
```

Secondly, we need to set up PYTHON_EGG_CACHE as Impala is written in Python and might need a temporary cache directory.

Possible enhancement: invalidate metadata might be replaced by refresh as refresh is less resource consuming and faster.

Query output:

```
+-----------------------+
| max(record_ts)        |
+-----------------------+
| 2017-04-07 03:14:29.0 |
+-----------------------+
```

Therefore we use a **regex** expression to extract timestamp.
And echo the output as

```
MAXTS=2017-04-07 03:14:29.0
```

2) SQOOP Action

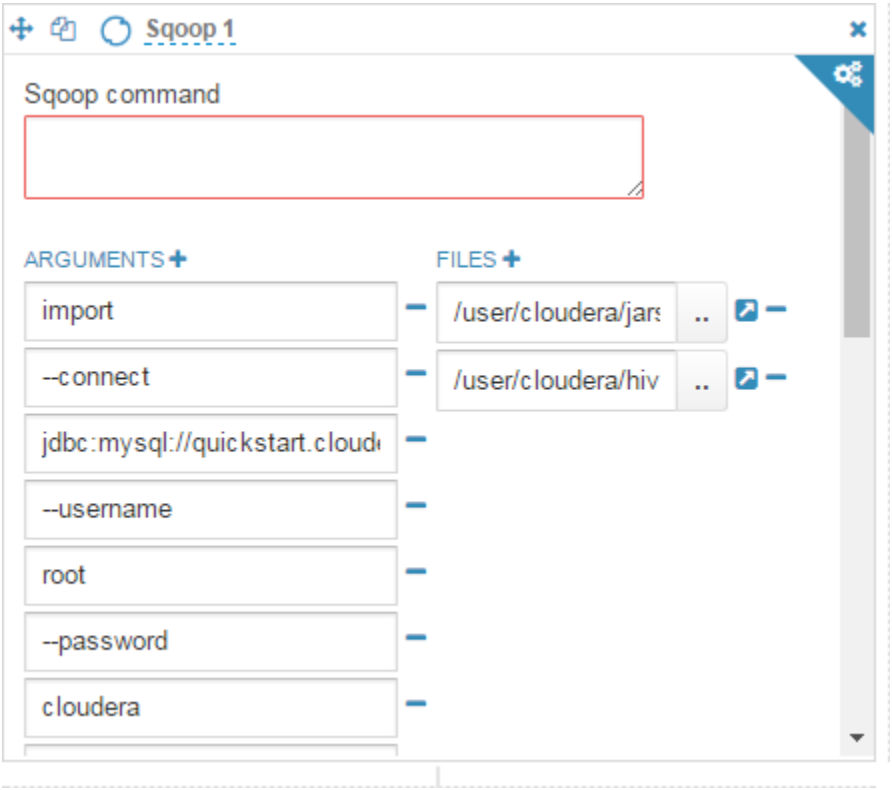
All the options will have to be entered as argument in Sqoop action

```
import
--connect
jdbc:mysql://quickstart.cloudera:3306/spacex
--username
root
--password
cloudera
--table
spacex_missions_w
-m
1
--check-column
```

```
Record_TS
--incremental
lastmodified
--last-value
${wf:actionData("shell-1860")["MAXTS"]}
--merge-key
pid
--split-by
pid
--target-dir
/user/cloudera/hive/external/missions_inc
```

Files selected under SQOOP action

```
/user/cloudera/jars/mysql-connector-java-5.1.34-bin.jar
/user/cloudera/hive/hive-site.xml
```



It must be noted that create-hive-table doesn't work with OOZIE. However, hive-import may work and we can use this facility. However, here we have uploaded data into HDFS and we will create an external table on top of it. Specifying, --merge-key primary_key is required so as to execute incremental update the second time. (Meaning first incremental update may work without specifying merge-key). Everything else is same as SQOOP command mentioned in Hortonworks 4 Steps Incrmental Guide.

3) HIVE Action to create incremental table

```
create external table spacex.missions_inc (
    flight_number   string,
    launch_datetime string,
      launch_site   string,
      vehicle_type  string,
      payload_name  string,
```

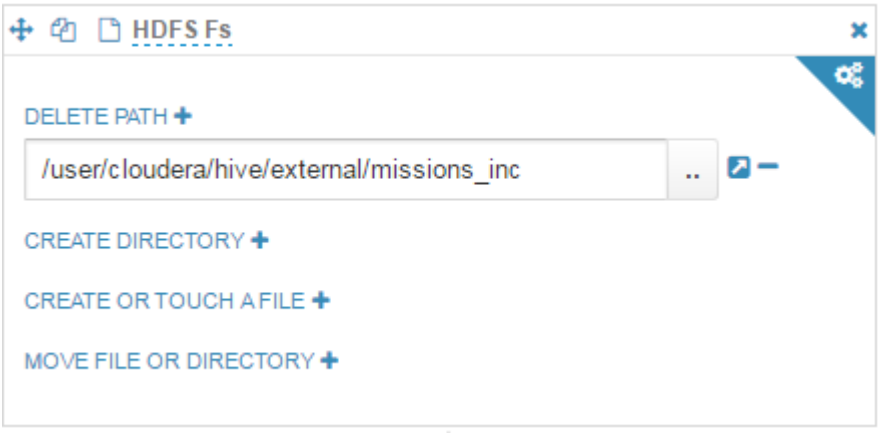
```
        payload_type string,
        payload_mass_kg     double,
        payload_orbit       string,
        customer_name       string,
        customer_type       string,
        customer_country    string,
        mission_outcome     string,
        failure_reason      string,
        landing_type string,
        landing_outcome     string,
        pid    bigint,
        record_ts    string)
    ROW FORMAT delimited
    FIELDS TERMINATED BY ','
    STORED AS TEXTFILE
    LOCATION '/user/cloudera/hive/external/missions_inc/';
```

4) HIVE Action to reconcile and purge

```
    --RECONCILE AND PURGE HIVE TABLES
    DROP TABLE IF EXISTS spacex.missions_reporting;
    CREATE TABLE spacex.missions_reporting AS
    SELECT * FROM spacex.missions_reconcile_view;
    DROP TABLE IF EXISTS spacex.missions_inc;
    --DROP TABLE IF EXISTS spacex.missions_base;
    --CREATE TABLE spacex.missions_base LIKE spacex.missions_reporting;
    INSERT OVERWRITE TABLE spacex.missions_base SELECT * FROM spacex.missions_reporting;
```

Instead of deleting Base table, it might be a good idea to overwrite it.

5) HDFS Action



Possible Enhancement: All the hive actions mentioned above can be executed in IMPALA and performance can be improved especially those which involve JOINS