|The data consists of two headers, body, and one footer (special symbol “!” exists before header/footer).

Field delimiter is “|”.

Below one can see a typical tasks for such data in our project:

**PREREQUISTE**

1.      Create 3 directories in hdfs:

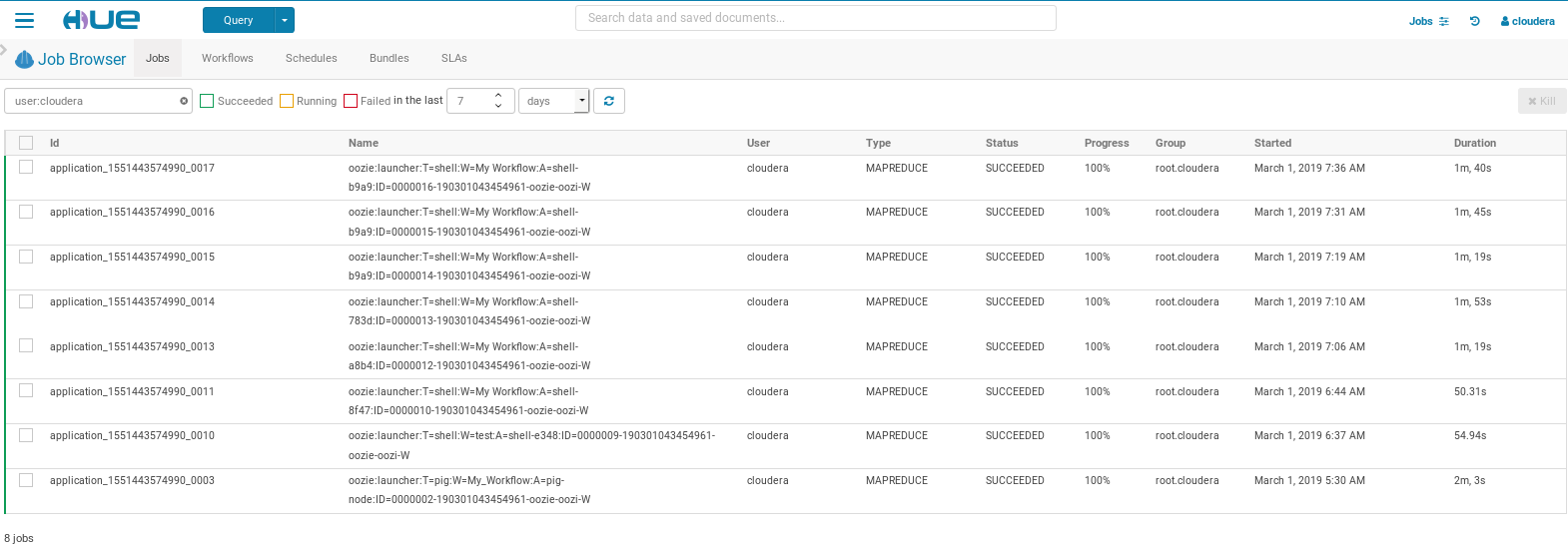
-        STEP1/in: *stores data as it is.*

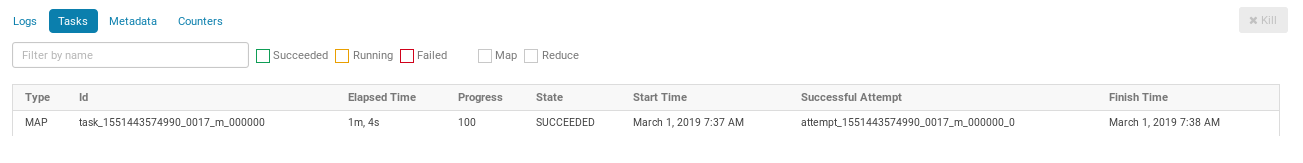
-        STEP2/YYYY/MM/DD/: *stores data after some cleaning, sequential transformations only (without Hadoop parallel transformations).*



I created simple sh app, to remove header and footer,

Later succesfully executed in oozie





-        STEP3/ /: *stores data after Hadoop parallel transformations (prepared for using in Hive).*

2.      Create Hive table

-        With defined partitions (YEAR, MONTH, DAY)

-        Column names should be derived from HEADER2 (you can do it manually, because it should be done only once, it is not a repetitive action)

**SCHEDULING**

3.      Create oozie workflows/coordinators for each transformation steps:

-        **Time based** oozie **coordinator** for STEP1->STEP2.

Its workflow should work with files from STEP1 directory, save result to STEP2 directory and it should be triggered every 5 min.

-        **File based** oozie **coordinator** for STEP2->STEP3.

Its workflow should work with files from STEP2 directory, save result to STEP3 directory and it should be triggered only if in STEP2 exists “\_SUCCESS” file.

**SEQUENTIAL TRANSFORMATIONS FOR 1st WORKFLOW**

4.      Create Bash/Python scripts for sequential transformations (transformation which could be done on one node only)

-        HEADER1 should be deleted.

-        HEADER2 should be deleted.

-        FOOTER1 should be deleted.

-        File content should be validated (number of rows is equal to the value of FOOTER1)

**PARALLEL TRANSFORMATIONS 2nd WORKFLOW**

5.      Create Pig/Hive scripts for Hadoop parallel transformations

-        Replace wrong decimal delimiter (,) to hive style (.) delimiter.

-        Date should be standardized: Wrong date format should be transformed to YYYY-MM-DD in “DATE”.

-        Alter existed Hive table by adding new partition there.

6.      Print all records from Hive using SELECT query

**EVENT STREAMING**

7.      Create flume agent

-        Source: netcat

-        Channel: file

-        Sink: hdfs

**CONFIGURATION**  
  7.1 Create “flume.conf” file, where the following should be defined:

-        Source = “netcat” (as well as all related settings for such source).

-        Channel=”file” (as well as all related settings for such channel).

-        Sink = ‘hdfs” (as well as all related settings for such sink).

*The information about mentioned source, channel and sink exists in official Flume documentation.*

**EXECUTION**

7.2      Run flume agent. (flume-ng … //instruction exists in documentation).

7.3      Run curl telnet://localhost:<port>

7.4      Feed records from test.csv

7.5      After 5th records flume agent should save 1 file (with five records) to hdfs.

7.6      Your regular Data flow process (steps 1-6) should do the rest.

**HTTP SOURCE**

7.7      Create another agent with source = “org.apache.flume.source.http.HTTPSource”

7.8 Send a JSON string to the agent using

curl -X POST -H 'Content-Type: application/json; charset=UTF-8' -d '<JSON string>' [http://localhost:<port>/](http://localhost:%3cport%3e/)

where the JSON string is an array of objects with the parameter “body” that corresponds to one line of the input text:

[

{"body": "!HEADER1"},

{"body": "!DATA1|NAME1|DATA2|DATA3|DOUBLE1|DOUBLE2|INT1|NAME2|DOUBLE3"},

{"body": "2015-09-04|bbb|15-SEP-04|09/04/2015|16,70|5,20||bbb2|81,78|"},

...

]

**TESTING**

8.      Write bash scripts which will test the output results after:

-        FLUME->STEP1

-        STEP1->STEP2

-        STEP2->STEP3