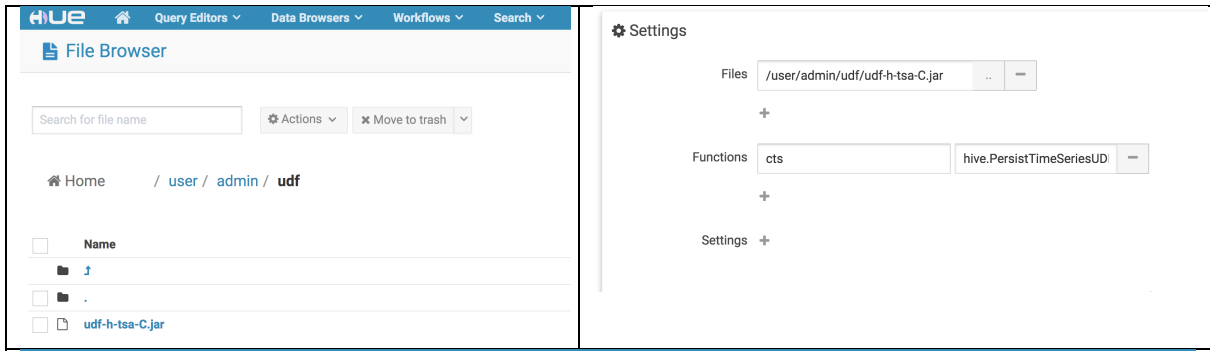
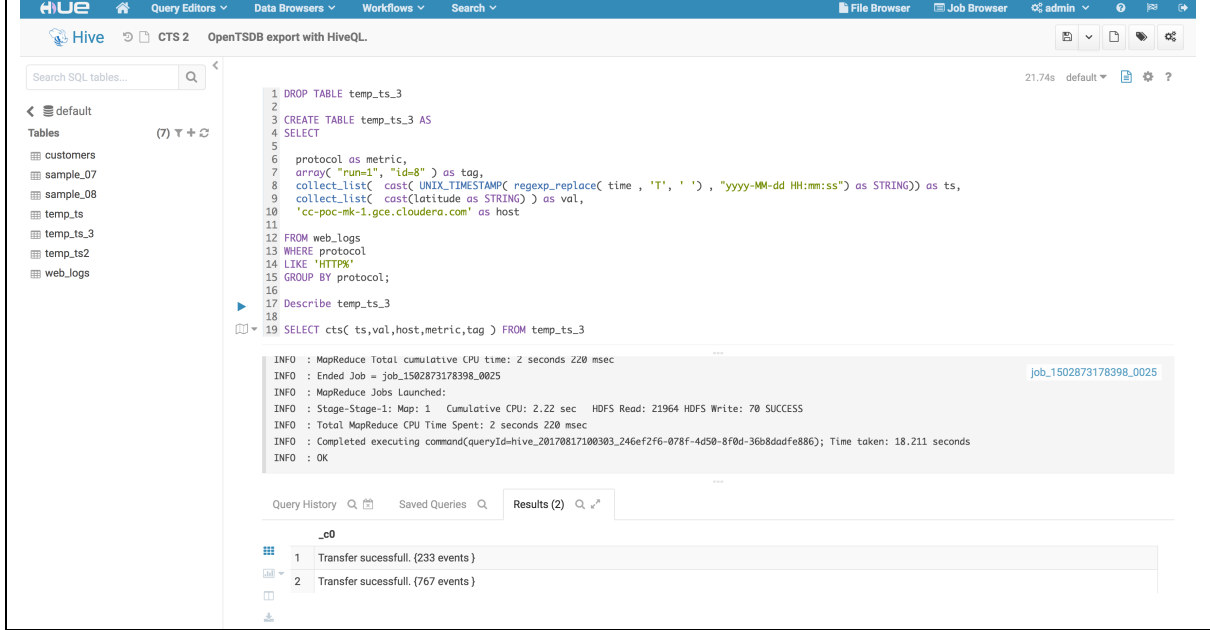


HiveQL to OpenTSDB (Custom UDF)

1. Build and Upload UBER-Jar to HDFS
2. Define Function in Hive
3. Group data and persist in temporary table
4. Inspect data using HiveQL
5. Export time series to OpenTSDB
6. Use OpenTSDB visualization and export functionality



The HUE interface is shown with the File Browser and Settings panels. The File Browser displays the path /user/admin/udf and lists files . and udf-h-tsa-C.jar. The Settings panel shows the file path /user/admin/udf/udf-h-tsa-C.jar and the function hive.PersistTimeSeriesUD.



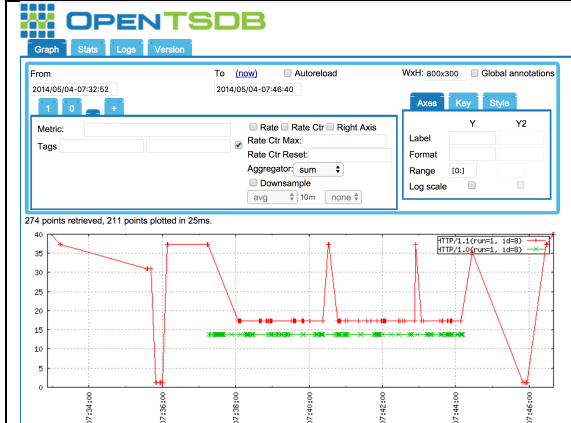
The HUE interface shows the HiveQL query editor and the query results. The query is a HiveQL statement that creates a temporary table temp_ts_3 and exports data to OpenTSDB. The results show two transfer events.

```
1 DROP TABLE temp_ts_3
2
3 CREATE TABLE temp_ts_3 AS
4 SELECT
5   protocol as metric,
6   array( "run=1", "id=8" ) as tag,
7   collect_list( cast( UNIX_TIMESTAMP( regexp_replace( time , 'T', ' ' ) , "yyy-MM-dd HH:mm:ss" ) as STRING ) ) as ts,
8   collect_list( cast( latitude as STRING ) ) as val,
9   'cc-poc-mk-1.gce.cloudera.com' as host
10
11
12 FROM web_logs
13 WHERE protocol
14 LIKE 'HTTP%'
15 GROUP BY protocol;
16
17 Describe temp_ts_3
18
19 SELECT cts( ts,val,host,metric,tag ) FROM temp_ts_3
```

INFO : MapReduce Total cumulative CPU time: 2 seconds 220 msec
INFO : Ended Job = job_1502873178398_0025
INFO : MapReduce Jobs Launched:
INFO : Stage-Stage-1: Map: 1 Cumulative CPU: 2.22 sec HDFS Read: 21964 HDFS Write: 70 SUCCESS
INFO : Total MapReduce CPU Time Spent: 2 seconds 220 msec
INFO : Completed executing command(queryId=hive_20170817100303_246ef2f6-078f-4d50-8f8d-36b8dadfe886); Time taken: 18.211 seconds
INFO : OK

Query History Saved Queries Results (2)

id	event
1	Transfer successful. {233 events}
2	Transfer successful. {767 events}



The OpenTSDB interface shows a time series graph. The graph displays data points over time, with a red line representing the metric and green dots representing the tags.

Next Steps:

- Configure OpenTSDB Host as parameter of UDF
- Calculate search query to retrieve the particular time series object via OpenTSDB API
- Define a TSA-Function for on-the-fly episode profiling => return episode profile as JSON for Solr import