Integración
HBASE - HIVE

¿Y podemos utilizar HBase desde Hive?

- Como es habitual… a través de SerDe
- Definamos una tabla t1 con una columna family cf1. Desde la shell de hbase:

  create 't1', 'cf1'

- Añadamos datos a esa tabla registro a registro:

  put 't1', '1', 'cf1:col_1', '11'
  put 't1', '1', 'cf1:col_2', '22'

  O hagámoslo a través de la utilidad de carga masiva (ver ejemplo de comandos en los ficheros de ayuda subidos a Moodle de hbase)

- La definición en Hive tiene que ser acorde a lo que esperamos almacenar (o hemos almacenado ya) en Hbase (atención a este condicionante, el schemaless de Hbase puede ser un inconveniente)

En Hive, hemos actualizado su configuración

- Los dockers de la plataforma comparten cierta configuración (HDFS y YARN) pero el Docker de Hive no tiene configurado, por defecto, como acceder al master ni a los región servers de Hbase
- Hemos resuelto este punto copiando en el Docker de hive el fichero de configuración principal de Hbase
- Así, el docker de Hive tiene la información necesaria de red, puertos y acceso para acceder a HBase

Definimos ya la tabla

- Una vez que el entorno de hive ya está configurado (la versión #3.0 ya lo está) podremos crear la tabla como externa utilizando las funcionalidades del SerDe correspondiente.

```
create database hbase;
use hbase;
create external table hbase_t1(key int, col_1 int, col_2 int)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key, cf1:col_1, cf1:col_2")
TBLPROPERTIES ("hbase.table.name" = "t1", "hbase.mapred.output.outputable" = "t1");
exit;
```

```
0: jdbc:hive2://localhost:10000> create database hbase;
No rows affected (0.192 seconds)
0: jdbc:hive2://localhost:10000> use hbase;
No rows affected (0.075 seconds)
0: jdbc:hive2://localhost:10000> create external table hbase_t1(key int, col_1 int, col_2 int)
. . . . . .> STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
. . . . . .> WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key, cf1:col_1, cf1:col_2")
. . . . . .> TBLPROPERTIES ("hbase.table.name" = "t1", "hbase.mapred.output.outputable" = "t1");
No rows affected (0.516 seconds)
0: jdbc:hive2://localhost:10000> exit;
```

Consultamos la tabla Hbase desde Hive

- Consultamos la tabla hbase como una tabla más.

```
use hbase;
select * from hbase_t1;
exit;
```

```
root@hive-server:~/hive_hbase# beeline -u jdbc:hive2://localhost:10000 -f query_test_hbase.hql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 2.3.2)
Driver: Hive JDBC (version 2.3.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://localhost:10000> use hbase;
No rows affected (0.205 seconds)
0: jdbc:hive2://localhost:10000>
0: jdbc:hive2://localhost:10000> select * from hbase_t1;
+---------------+----------------+----------------+
| hbase_t1.key  | hbase_t1.col_1 | hbase_t1.col_2 |
+---------------+----------------+----------------+
| 1             | 1135           | 1246           |
| 2             | 2135           | 2246           |
| 3             | 3135           | NULL           |
+---------------+----------------+----------------+
3 rows selected (1.025 seconds)
```