https://hive.apache.org/

Master IPS 2021-2022

Need for High-Level Languages:

- Hadoop is great for large-data processing!
  - But writing Java programs for everything is verbose and slow
  - Not everyone wants to (or can) write Java code
- Solution: develop higher-level data processing languages
  - Hive: HQL is like SQL
  - Pig: Pig Latin is a bit like Perl

Introduction

- **Hive is an ETL and data warehouse tool on top of Hadoop ecosystem and used for processing structured and semi structured data.**
  - Rich data types (structs, lists and maps)
  - Efficient implementations of SQL filters, joins and group-by's on top of map reduce
  - Query language is HQL, variant of SQL
  - Tables stored on HDFS as flat files
  - Developed by Facebook, now open source

- **Hive is a database present in Hadoop ecosystem performs DDL and DML operations, and it provides flexible query language such as HQL for better querying and processing of data.**

- **It provides so many features compared to RDMS which has certain limitations.**

Different modes of Hive

Hive can operate in two modes depending on the size of data nodes in Hadoop. These modes are :

1. **Local mode**
2. **Map reduce mode**

When to use Local mode:

- If the Hadoop installed under pseudo mode (having one data node)
- If the data size is smaller (limited to a single local machine).
- Processing will be very fast on smaller data sets present in the local machine.

When to use Map reduce mode:

- If Hadoop is having multiple data nodes and data is distributed across different node .
- It will perform on large amount of data sets and query will execute in a parallel way.
- Processing of large data sets with better performance can be achieved through this mode.

By default, Hive will work on Map Reduce mode and for local mode you can have the following setting :
To work in local mode set :
SET mapred.job.tracker=local;
From the Hive version 0.7 it supports a mode to run map reduce jobs in local mode automatically.

Requirements

- Java 1.7

  Note: Hive versions 1.2 onward require Java 1.7 or newer. Hive versions 0.14 to 1.1 work with Java 1.6 as well. Users are strongly advised to start moving to Java 1.8 (see HIVE-8607).
- Hadoop 2.x (preferred), 1.x (not supported by Hive 2.0.0 onward). Hive versions up to 0.13 also supported Hadoop 0.20.x, 0.23.x.
- Hive is commonly used in production Linux and Windows environment. Mac is a commonly used development environment.

Installing Hive from a Stable Release

- Start by downloading the most recent stable release of Hive from one of the Apache download mirrors (https://hive.apache.org/downloads.html).
- Next you need to unpack the tarball. This will result in the creation of a subdirectory named hive-x.y.z (where x.y.z is the release number):

```
$ tar -xzvf hive-x.y.z.tar.gz
```

the environment variable `HIVE_HOME` to point to the installation directory:

```
$ cd hive-x.y.z
$ export HIVE_HOME={{pwd}}
```

ally, add `$HIVE_HOME/bin` to your `PATH`:

```
$ export PATH=$HIVE_HOME/bin:$PATH
```

Hive Architecture

- Metastore: stores system catalog
- Driver: manages life cycle of HiveQL query as it moves thru' HIVE; also manages session handle and session statistics
- Query compiler: Compiles HiveQL into a directed acyclic graph of map/reduce tasks
- Execution engines: The component executes the tasks in proper dependency order; interacts with Hadoop
- HiveServer: provides Thrift interface and JDBC/ODBC for integrating other applications.
- Client components: CLI, web interface, jdbc/odbc inteface
- Extensibility interface include SerDe, User Defined Functions and User Defined Aggregate Function.

Hive Architecture

In Hive, the data is stored in HDFS and the table, database, schema, and other HQL definitions are stored in a metastore. The metastore could be any RDBMS database, such as MySQL or Oracle. Hive creates a database and a set of tables in metastore to store HiveQL definitions.

There are three modes of configuring a Metastore :

1. Embedded
2. Local
3. Remote

Internal Vs External Tables in Hive

- "Internal" or "Hive managed" tables are the default
  - Stored at HDFS directory/hive/warehouse by default
  - Hive assumes no other system is using **This DATA**
  - So a "drop table" statement removes the metadata **AND the underlying HDFS data**
- External tables can be specified
  - Point "LOCATION" to some HDFS directory path
  - "drop table" now only removes the metadata

Data & Metadata

Id	TagName	Count
23	hadoop	2325
24	java	213123
27	solr	32313
28	json	92223

Data and MetaData

Data

23	hadoop	2325
24	java	213123
27	solr	32313
28	json	92223

Metadata

Id	TagName	Count

Overview

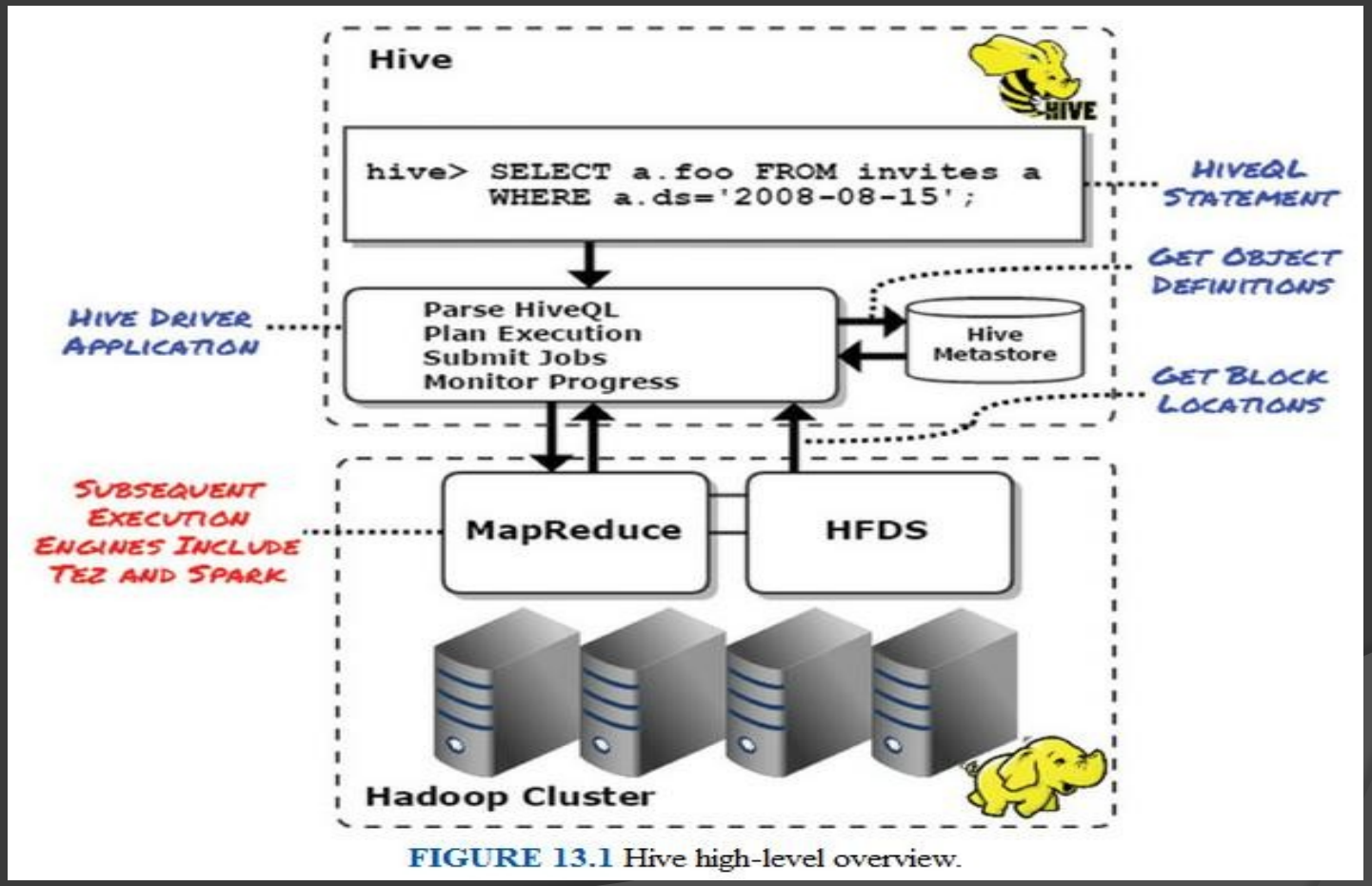


FIGURE 13.1 Hive high-level overview.

- Data Model:
- Tables
  - Typed columns (int, float, string, boolean)
  - Also, list: map (for JSON-like data)
- Partitions
  - For example, range-partition tables by date
- Buckets
  - Hash partitions within ranges (useful for sampling, join optimization)

Metastore:

- Database: namespace containing a set of tables
- Holds table definitions (column types, physical layout)
- Holds partitioning information
- Can be stored in Derby, MySQL, and many other relational databases

Physical Layout:

- Warehouse directory in HDFS
  - E.g., /user/hive/warehouse
- Tables stored in subdirectories of warehouse
  - Partitions form subdirectories of tables
- Actual data stored in flat files
  - Control char-delimited text, or SequenceFiles
  - With custom SerDe, can use arbitrary format

Example:

- Hive looks similar to an SQL database
- Relational join on two tables:
  - Table of word counts from Shakespeare collection
  - Table of word counts from the bible

- SELECT s.word, s.freq, k.freq FROM shakespeare s
- JOIN bible k ON (s.word = k.word) WHERE s.freq >= 1 AND k.freq >= 1 ORDER BY s.freq DESC LIMIT 10;
- the 25848 62394
- I 23031 8854
- and 19671 38985
- to 18038 13526
- of 16700 34654
- a 14170 8057
- you 12702 2720
- my 11297 4135
- in 10797 12445
- is 8882 6884

Group By, Join, Multitable Insert …

GROUP BY

```
hive> FROM invites a INSERT OVERWRITE TABLE events SELECT a.bar, count(*) WHERE a.foo > 0 GROUP BY a.bar;
hive> INSERT OVERWRITE TABLE events SELECT a.bar, count(*) FROM invites a WHERE a.foo > 0 GROUP BY a.bar;
```

Note that for versions of Hive which don't include HIVE-287, you'll need to use `COUNT(1)` in place of `COUNT(*)`.

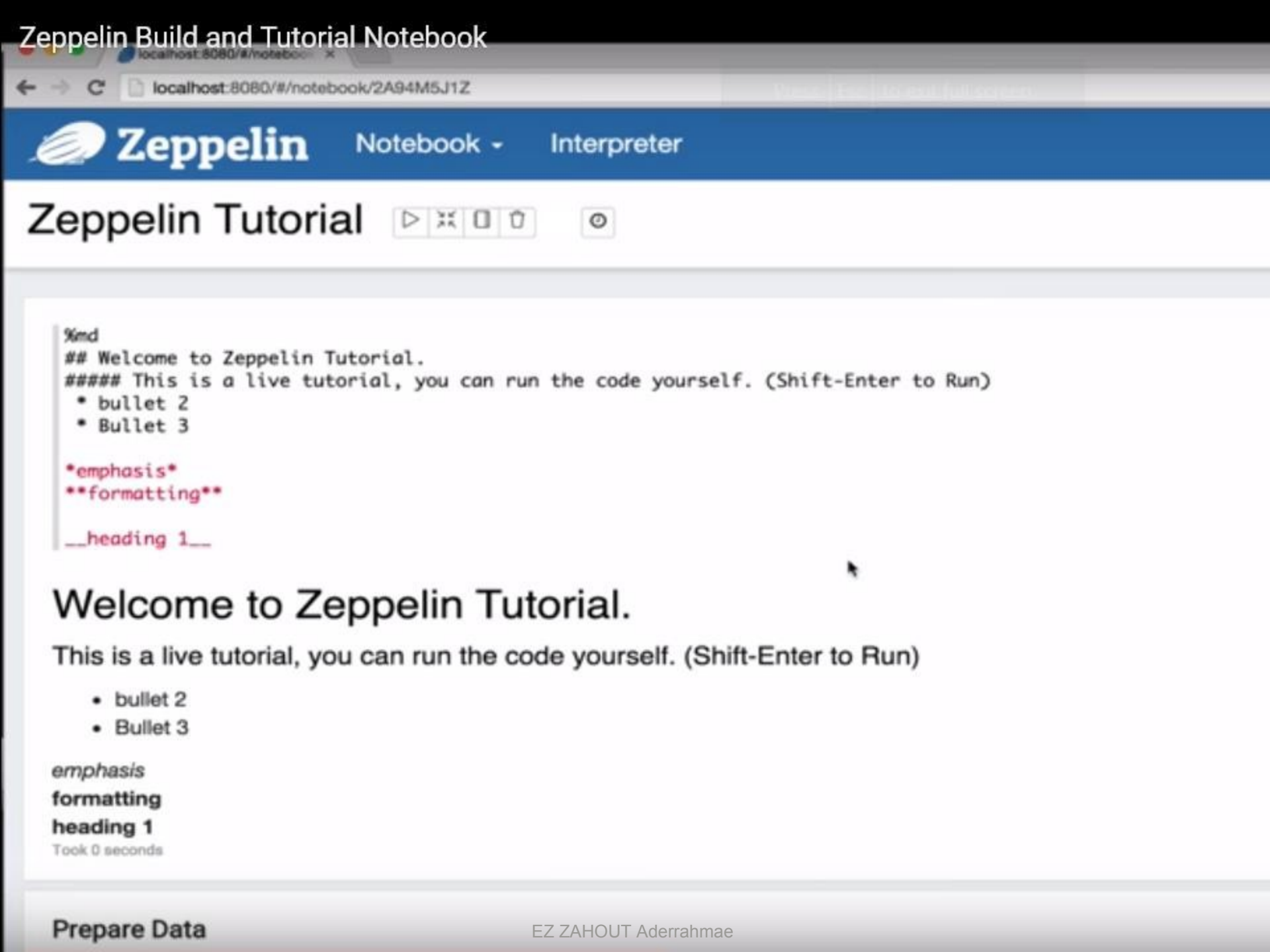
JOIN

```
hive> FROM pokes t1 JOIN invites t2 ON (t1.bar = t2.bar) INSERT OVERWRITE TABLE events SELECT t1.bar, t1.foo, t2.foo;
```

MULTITABLE INSERT

```
FROM src
INSERT OVERWRITE TABLE dest1 SELECT src.* WHERE src.key < 100
INSERT OVERWRITE TABLE dest2 SELECT src.key, src.value WHERE src.key >= 100 and src.key < 200
INSERT OVERWRITE TABLE dest3 PARTITION(ds='2008-04-08', hr='12') SELECT src.key WHERE src.key >= 200 and src.key < 300
INSERT OVERWRITE LOCAL DIRECTORY '/tmp/dest4.out' SELECT src.value WHERE src.key >= 300;
```

Zeppelin Build and Tutorial Notebook



METADATA

Function	Hive
Selecting a database	`USE database;`
Listing databases	`SHOW DATABASES;`
Listing tables in a database	`SHOW TABLES;`
Describing the format of a table	`DESCRIBE (FORMATTED|EXTENDED) table;`
Creating a database	`CREATE DATABASE db_name;`
Dropping a database	`DROP DATABASE db_name (CASCADE);`

CURRENT SQL COMPATIBILITY

Hive SQL Datatypes
INT
TINYINT/SMALLINT/BIGINT
BOOLEAN
FLOAT
DOUBLE
STRING
TIMESTAMP
BINARY
ARRAY, MAP, STRUCT, UNION
DECIMAL
CHAR
VARCHAR
DATE

Hive SQL Semantics
SELECT, LOAD, INSERT from query
Expressions in WHERE and HAVING
GROUP BY, ORDER BY, SORT BY
Sub-queries in FROM clause
GROUP BY, ORDER BY
CLUSTER BY, DISTRIBUTE BY
ROLLUP and CUBE
UNION
LEFT, RIGHT and FULL INNER/OUTER JOIN
CROSS JOIN, LEFT SEMI JOIN
Windowing functions (OVER, RANK, etc.)
INTERSECT, EXCEPT, UNION DISTINCT
Sub-queries in WHERE (IN/NOT IN, EXISTS/NOT EXISTS
Sub-queries in HAVING

Hive 0.10
Hive 0.11
Future

RETRIEVING

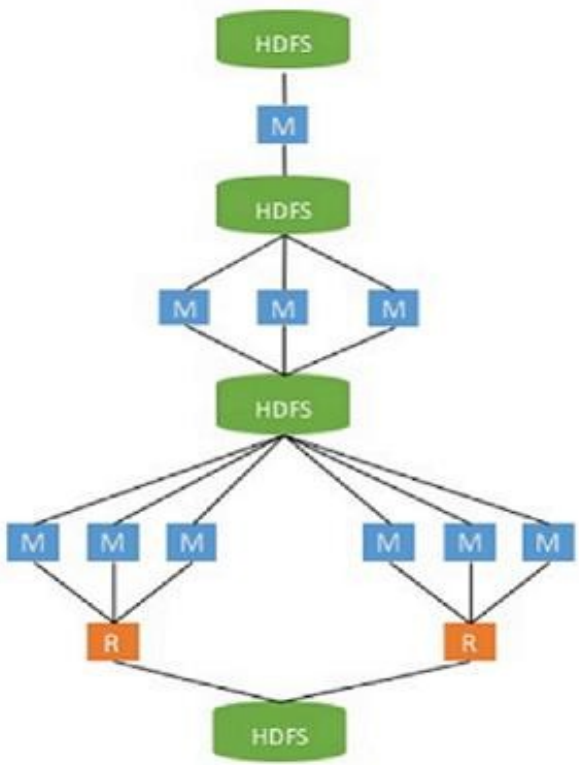
Function	MySQL	Hive
Retrieving Information (General)	`SELECT from_columns FROM table WHERE conditions;`	`SELECT from_columns FROM table WHERE conditions;`
Retrieving All Values	`SELECT * FROM table;`	`SELECT * FROM table;`
Retrieving Some Values	`SELECT * FROM table WHERE rec_name = "value";`	`SELECT * FROM table WHERE rec_name = "value";`
Retrieving With Multiple Criteria	`SELECT * FROM TABLE WHERE rec1 = "value1" AND rec2 = "value2";`	`SELECT * FROM TABLE WHERE rec1 = "value1" AND rec2 = "value2";`
Retrieving Specific Columns	`SELECT column_name FROM table;`	`SELECT column_name FROM table;`
Retrieving Unique Output	`SELECT DISTINCT column_name FROM table;`	`SELECT DISTINCT column_name FROM table;`

RETRIEVING INFORMATION

Sorting	`SELECT col1, col2 FROM table ORDER BY col2;`	`SELECT col1, col2 FROM table ORDER BY col2;`
Sorting Reverse	`SELECT col1, col2 FROM table ORDER BY col2 DESC;`	`SELECT col1, col2 FROM table ORDER BY col2 DESC;`
Counting Rows	`SELECT COUNT(*) FROM table;`	`SELECT COUNT(*) FROM table;`
Grouping With Counting	`SELECT owner, COUNT(*) FROM table GROUP BY owner;`	`SELECT owner, COUNT(*) FROM table GROUP BY owner;`
Maximum Value	`SELECT MAX(col_name) AS label FROM table;`	`SELECT MAX(col_name) AS label FROM table;`
Selecting from multiple tables (Join same table using alias w/"AS")	`SELECT pet.name, comment FROM pet, event WHERE pet.name = event.name;`	`SELECT pet.name, comment FROM pet JOIN event ON (pet.name = event.name)`

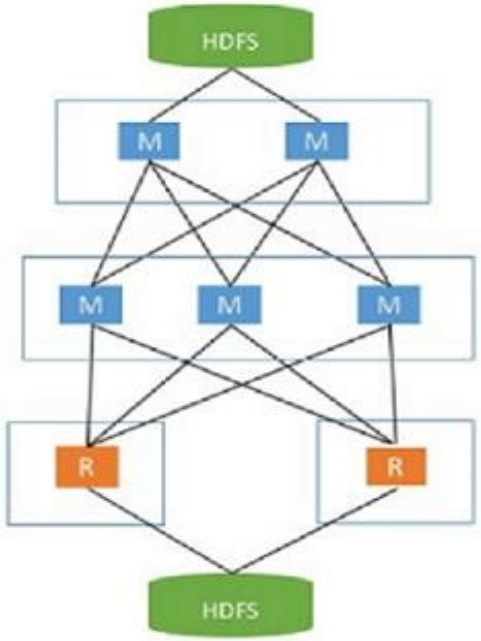
MapReduce and DAG engine

MapReduce



DAG engine

(Tez / Spark)



No intermediate DFS reads/writes!

HiveQL

DDL :

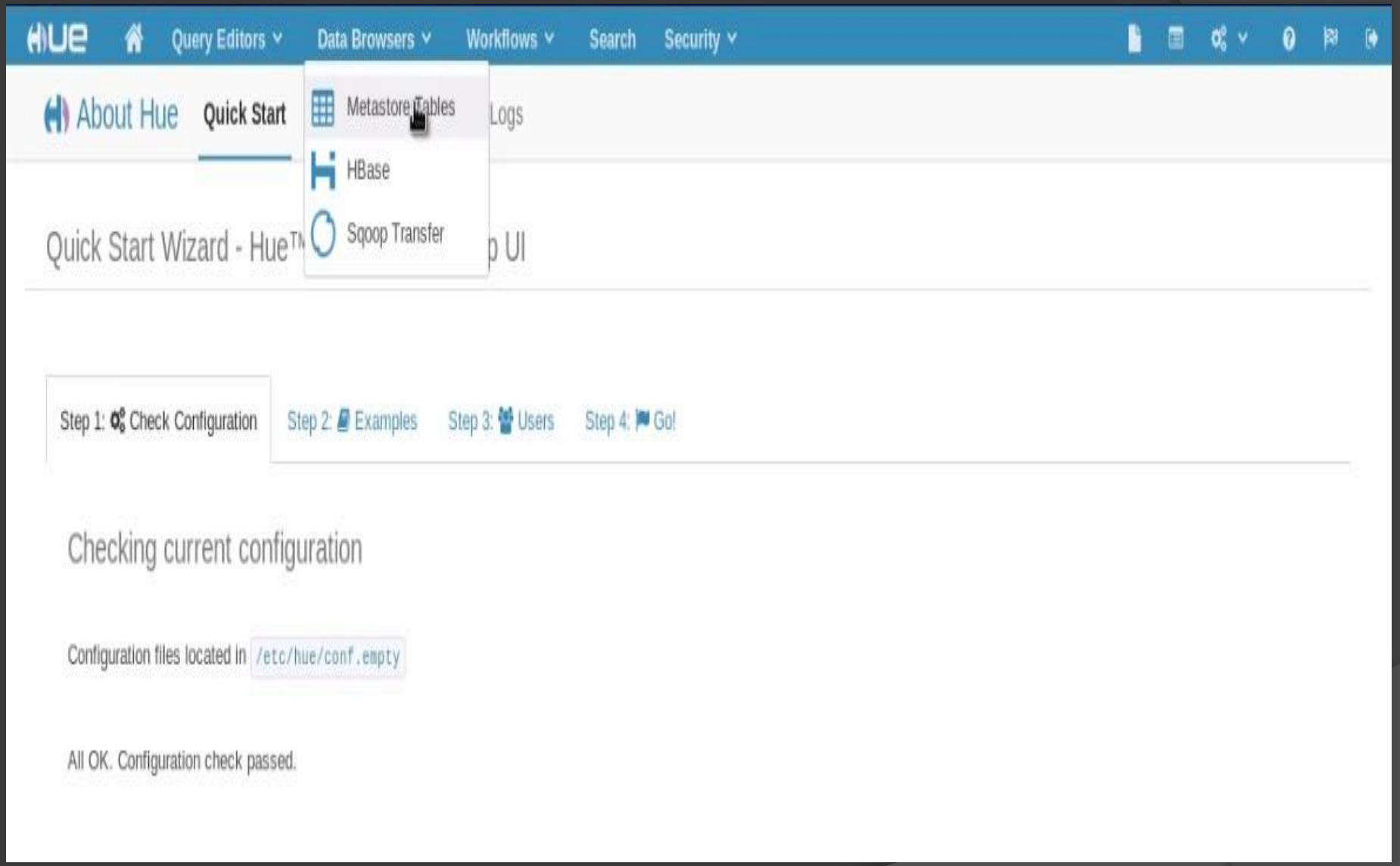
- CREATE DATABASE
- CREATE TABLE
- ALTER TABLE
- SHOW TABLE
- DESCRIBE

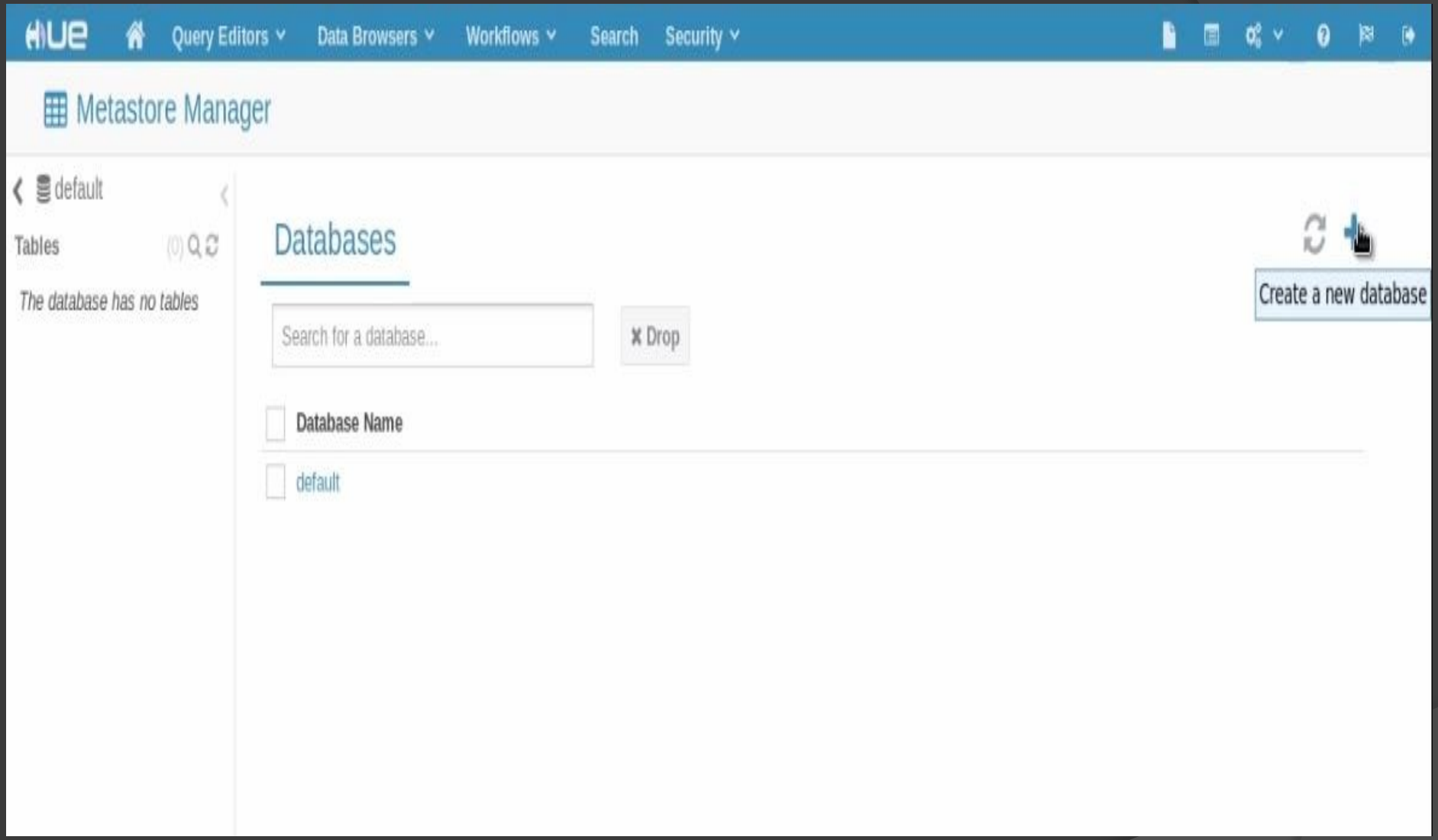
DML:

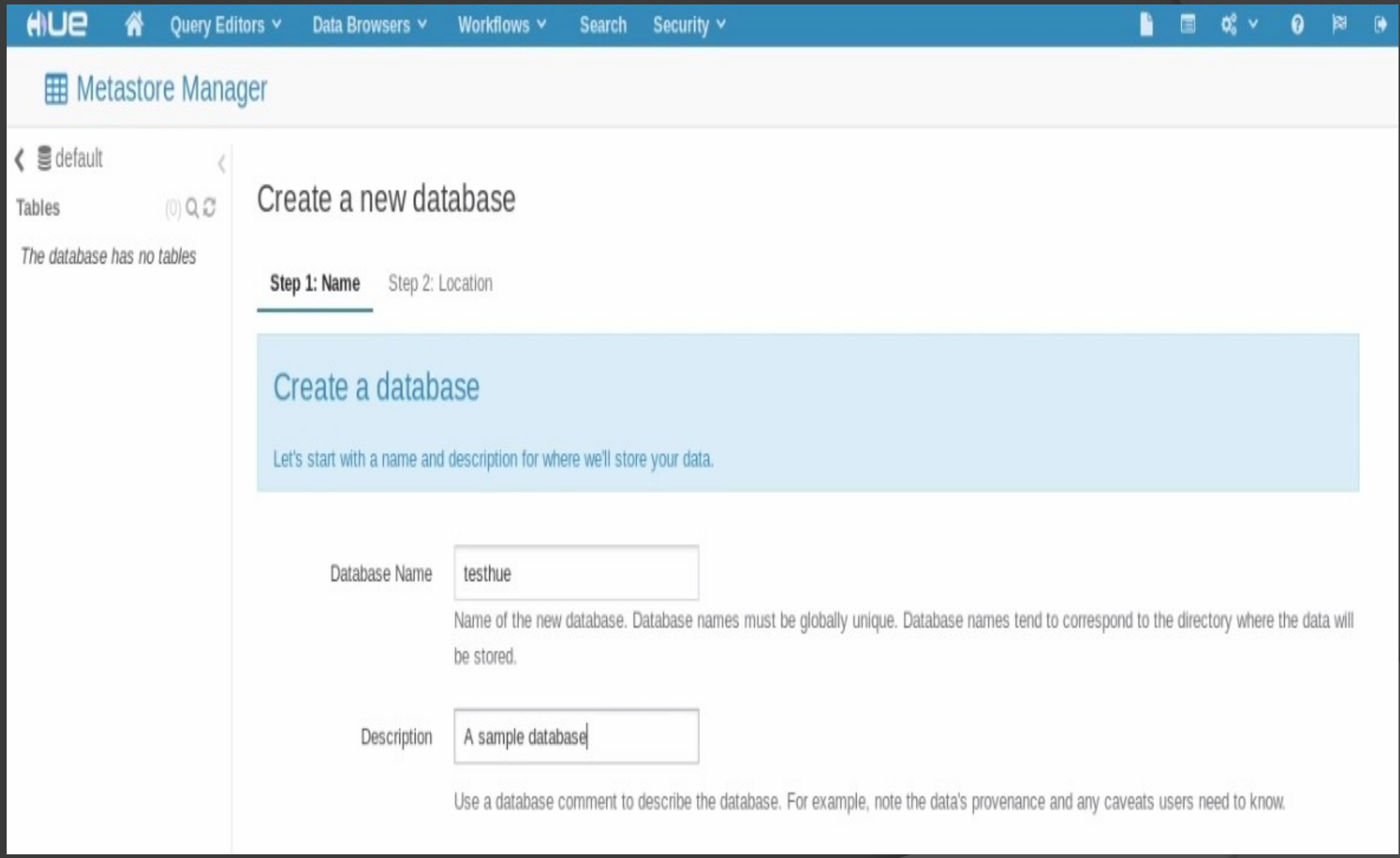
- LOAD TABLE
- INSERT

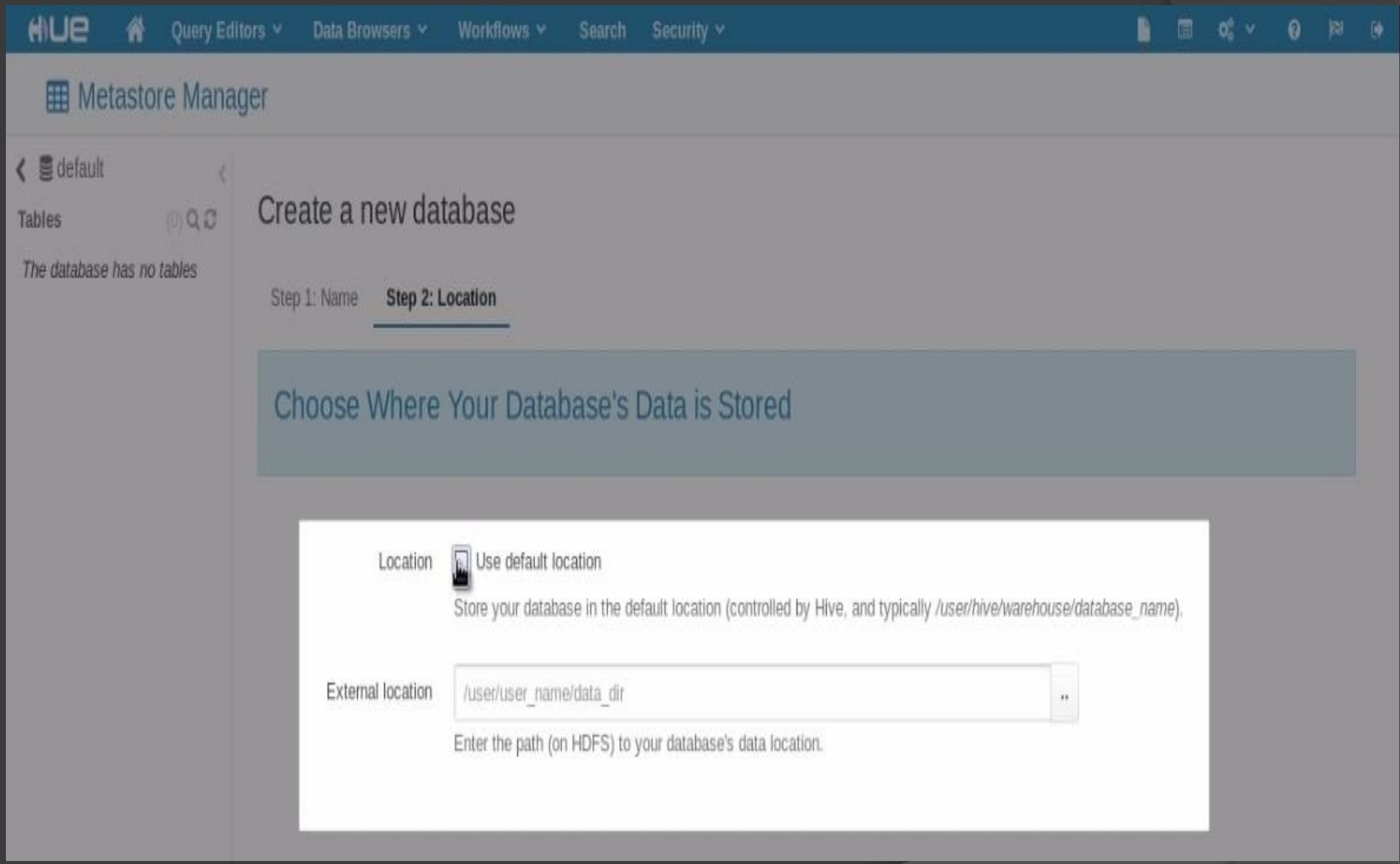
QUERY:

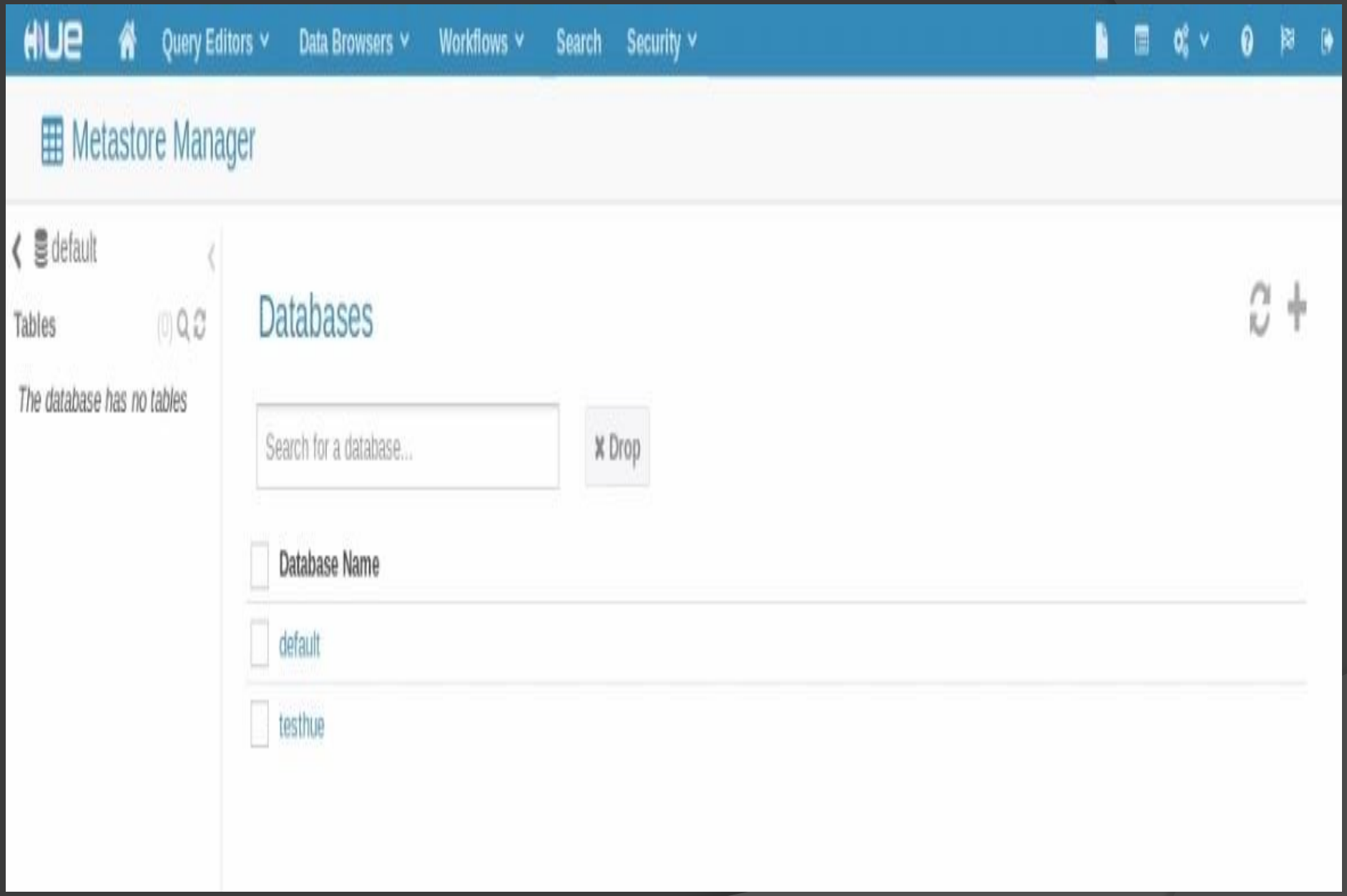
- SELECT
- GROUP BY
- JOIN
- MULTI TABLE INSERT

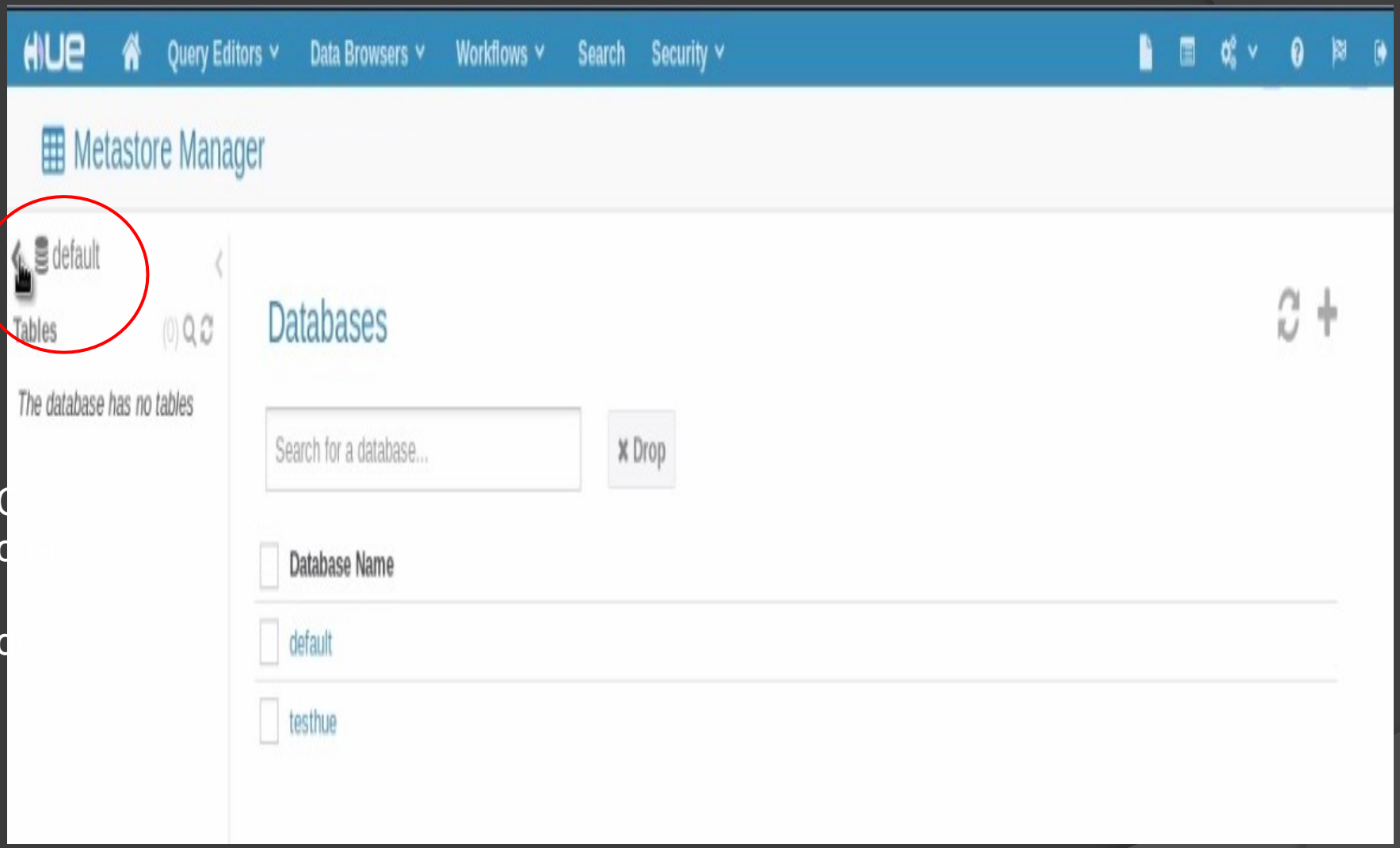

HUE
Query Editors
Data Browsers
Workflows
Search
Security
About Hue
Quick Start
Metastore Tables
HBase
Sqoop Transfer
Logs
Quick Start Wizard - Hue
p UI
Step 1: Check Configuration
Step 2: Examples
Step 3: Users
Step 4: Go!
Checking current configuration
Configuration files located in /etc/hue/conf.empty
All OK. Configuration check passed.

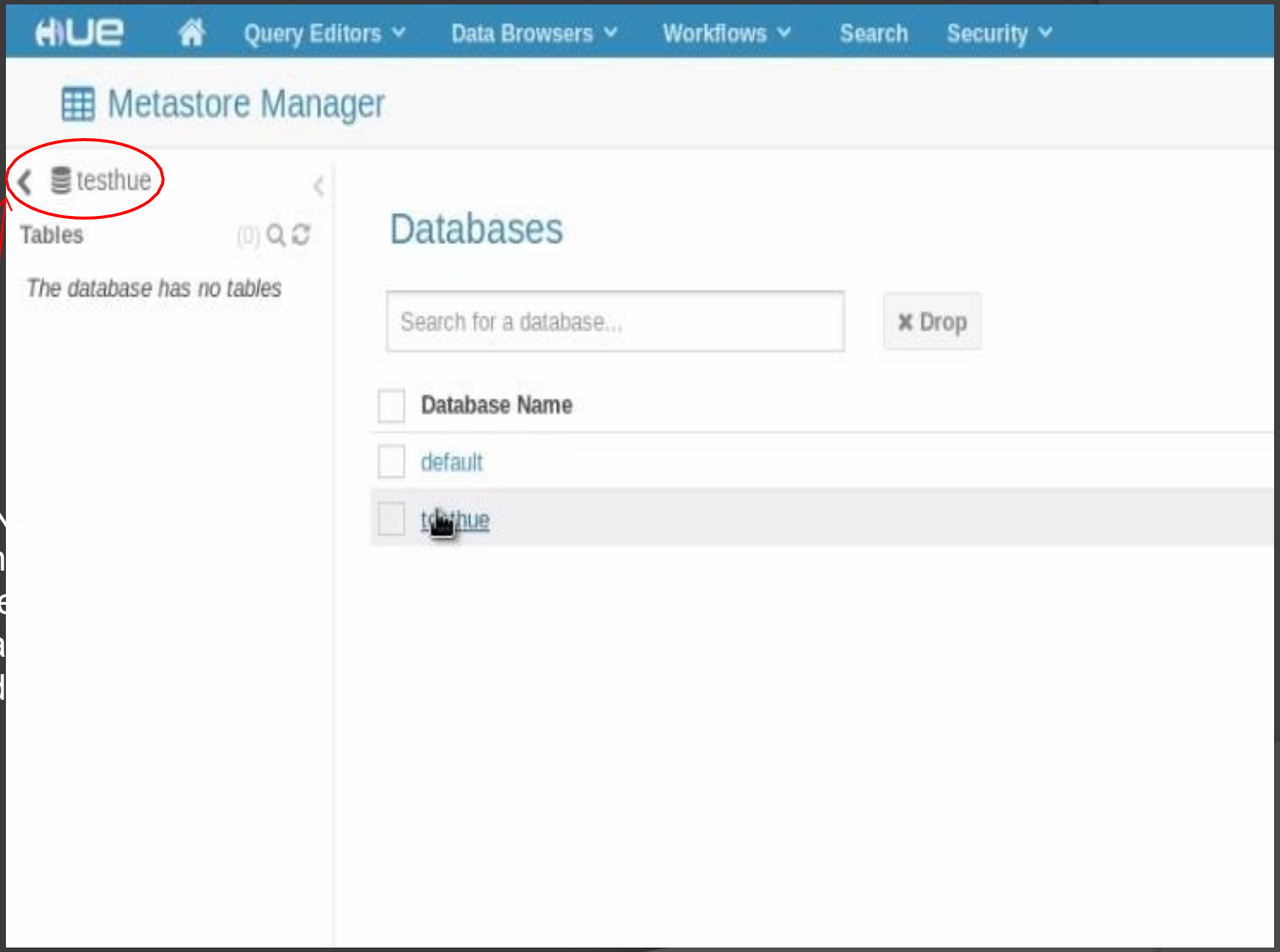

HUE
Query Editors
Data Browsers
Workflows
Search
Security
Metastore Manager
default
Tables
(0)
The database has no tables
Databases
Search for a database...
Drop
Create a new database
Database Name
default


HUE
Query Editors
Data Browsers
Workflows
Search
Security
Metastore Manager
default
Tables
(0)
The database has no tables
Create a new database
Step 1: Name
Step 2: Location
Create a database
Let's start with a name and description for where we'll store your data.
Database Name
testhue
Name of the new database. Database names must be globally unique. Database names tend to correspond to the directory where the data will be stored.
Description
A sample database
Use a database comment to describe the database. For example, note the data's provenance and any caveats users need to know.

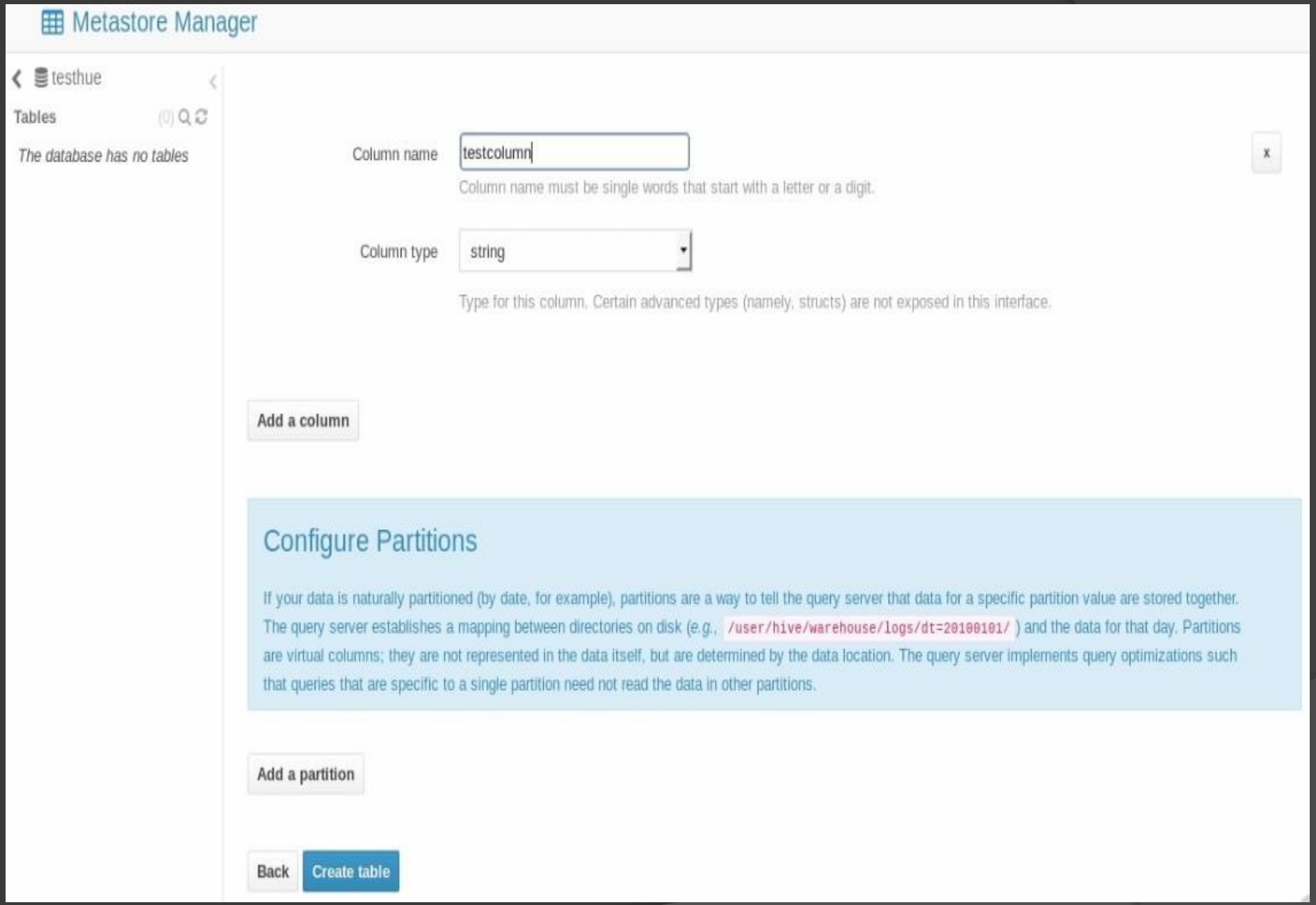

HUE
Query Editors
Data Browsers
Workflows
Search
Security
Metastore Manager
default
Tables
The database has no tables
Create a new database
Step 1: Name
Step 2: Location
Choose Where Your Database's Data is Stored
Location
Use default location
Store your database in the default location (controlled by Hive, and typically /user/hive/warehouse/database_name).
External location
/user/user_name/data_dir
Enter the path (on HDFS) to your database's data location.


HUE
Query Editors
Data Browsers
Workflows
Search
Security
Metastore Manager
default
Tables
The database has no tables
Databases
Search for a database...
Drop
Database Name
default
testhue


HUE
Query Editors
Data Browsers
Workflows
Search
Security
Metastore Manager
default
Tables
(0)
The database has no tables
Databases
Search for a database...
Drop
Database Name
default
testhue


HUE
Query Editors
Data Browsers
Workflows
Search
Security
Metastore Manager
testhue
Tables
(0)
The database has no tables
Databases
Search for a database...
Drop
Database Name
default
testhue

N
h
te
a
d

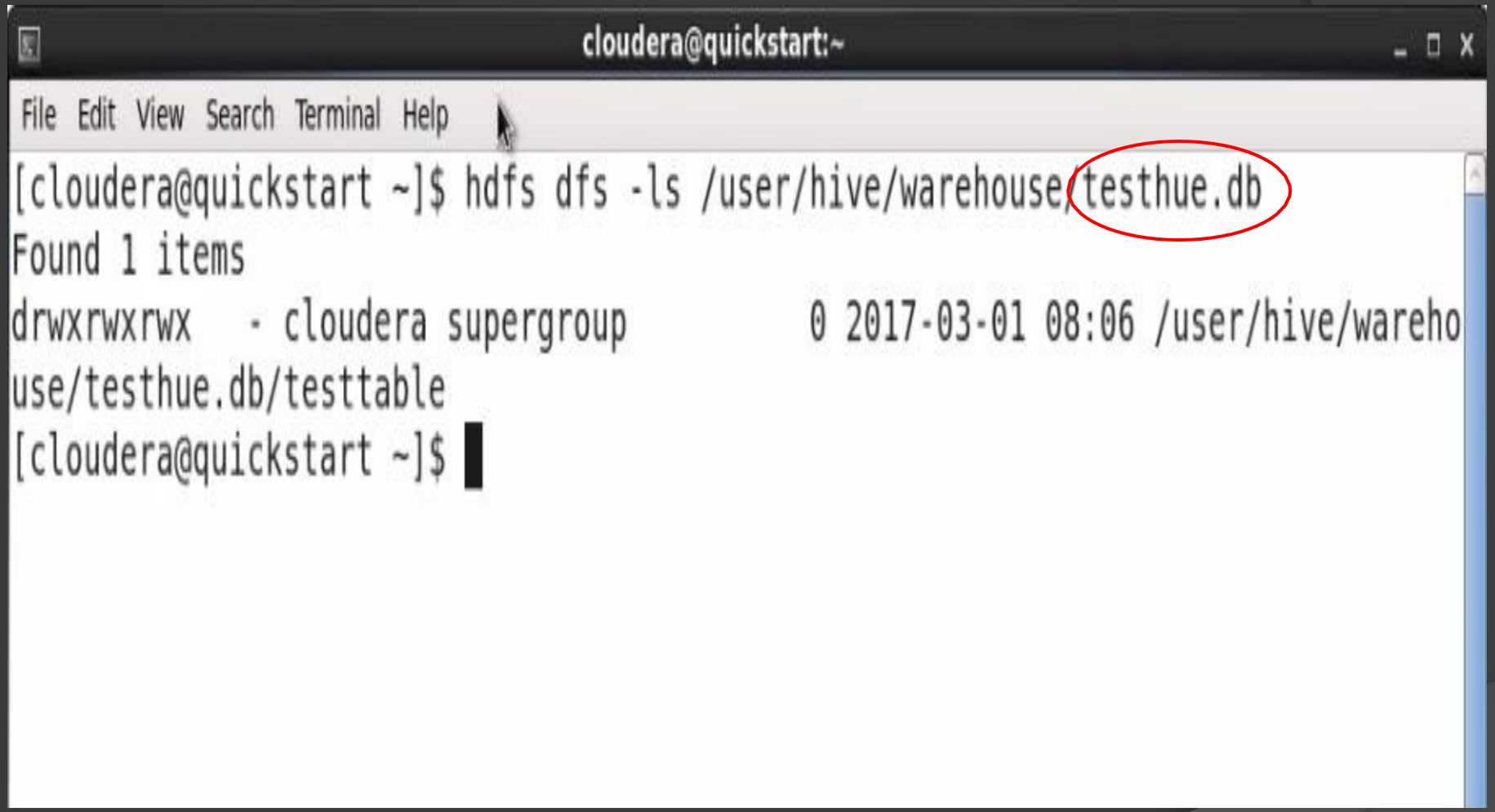

Metastore Manager
testhue
Tables (0)
The database has no tables
Column name
testcolumn
Column name must be single words that start with a letter or a digit.
x
Column type
string
Type for this column. Certain advanced types (namely, structs) are not exposed in this interface.
Add a column
Configure Partitions
If your data is naturally partitioned (by date, for example), partitions are a way to tell the query server that data for a specific partition value are stored together. The query server establishes a mapping between directories on disk (e.g., /user/hive/warehouse/logs/dt=20100101/) and the data for that day. Partitions are virtual columns; they are not represented in the data itself, but are determined by the data location. The query server implements query optimizations such that queries that are specific to a single partition need not read the data in other partitions.
Add a partition
Back
Create table

Examples

Complex operators are used to access the elements of complex type.

The following table describes complex operators available in Hive:

Operator	Operand type	Description
`A[i]`	`A` is an array object, and `i` is the index of an element in the array.	It will return the element at the `i` index of array.
`M[key]`	`M` is the map object, that is, **key-value** pair.	It will return the value corresponding to the specified `key` in the map.
`S.a`	`S` is the `Struct` object.	Returns the `a` field of `S`.


cloudera@quickstart:~
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/testhue.db
Found 1 items
drwxrwxrwx - cloudera supergroup 0 2017-03-01 08:06 /user/hive/wareho
use/testhue.db/testtable
[cloudera@quickstart ~]$