Big Data Analytics Introduction to Hive

Lecture #1

Introduction to Hive

Hive Module Outline

- What is Hive?
- Hive Components
- What is Hive Data Model?
- Underlying Hive Architecture
- Using Hive in Practice

What is Hive?

Data warehouse infrastructure build on top of Hadoop for querying and managing large data sets

Why Hive?

Hadoop is great!

MapReduce is very low level
Lack of expressiveness

Higher level data processing
languages are needed

What is Hive?

A system for managing and querying unstructured data as if it were structured

- Stores schema in Database
- Uses Map-Reduce for execution
- HDFS for Storage

Hive Features

Designed for OLAP

On Line Analytical Processing

SQL type language for querying

HiveQL or HQL

It is familiar, fast, scalable, and extensible

Can plug in map/reduce scripts in language of choice

Hive is NOT

Relational database

Designed for Online Transaction Processing (OLTP)

Language for real-time queries and row-level updates

*Online transaction processing, or **OLTP**, is a class of information systems that facilitate and manage transaction-oriented applications, typically for data entry and retrieval transaction processing (Wikipedia)*

History

ETL =
Extract, Transform and Load

Early Hive development work started at Facebook in 2007

Hive is an Apache project under Hadoop

http://hive.apache.org

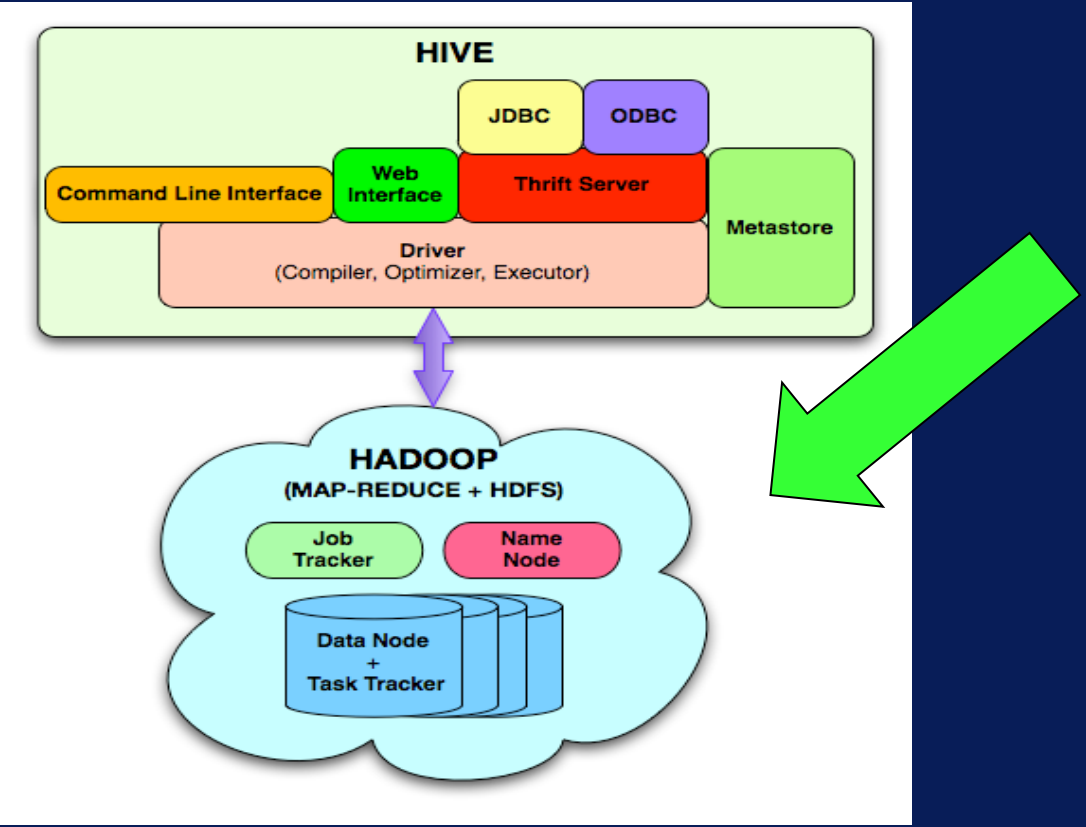
Data Warehouse Infrastructure for Hadoop

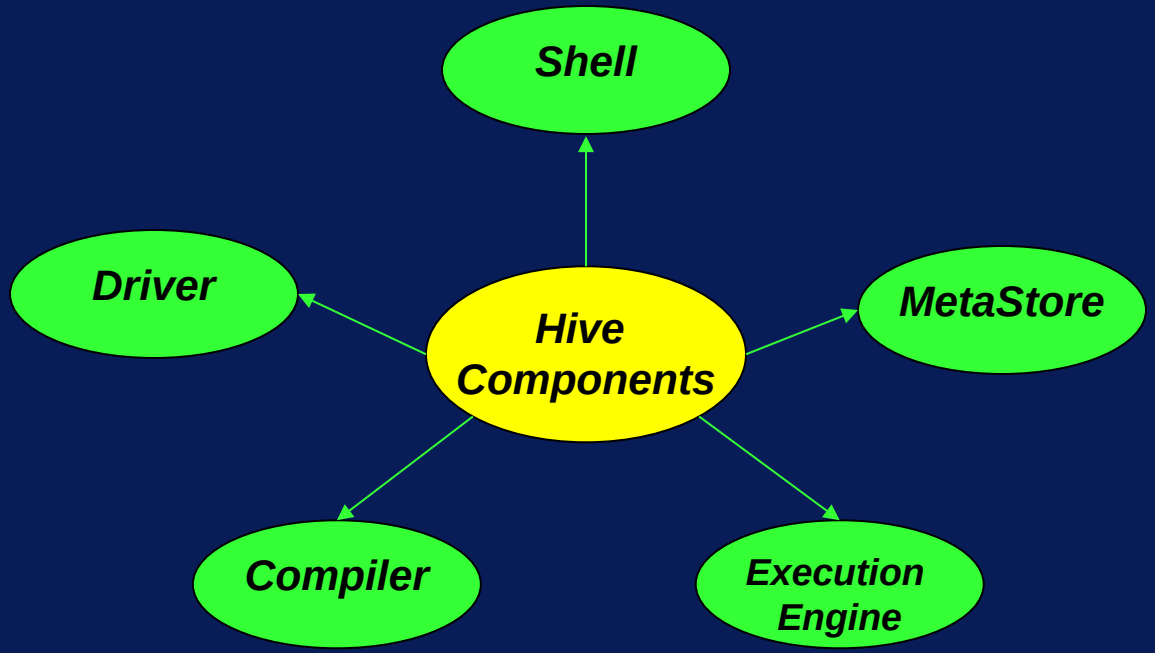
SQL-like query language (QL)

Enables developers to utilize custom mappers and reducers

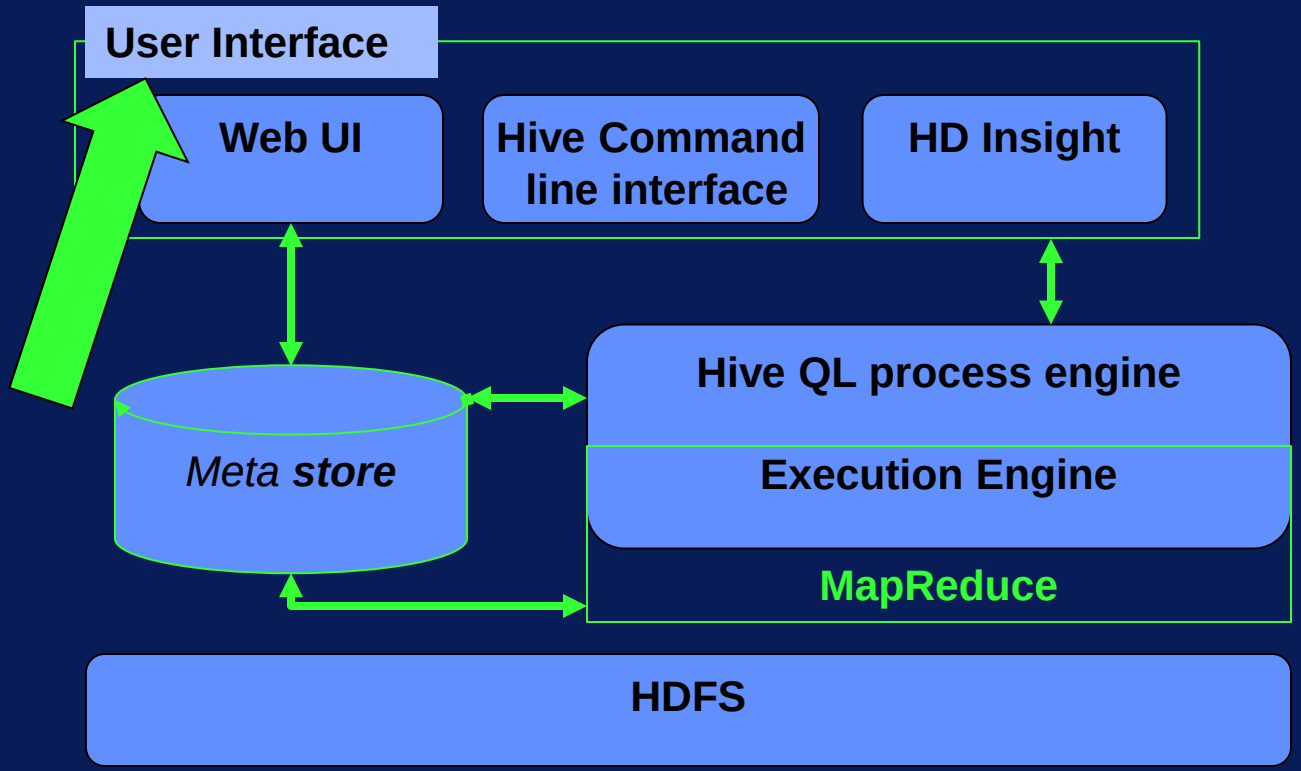
Provides tools to enable ETL on large data

Hive Architecture and Components

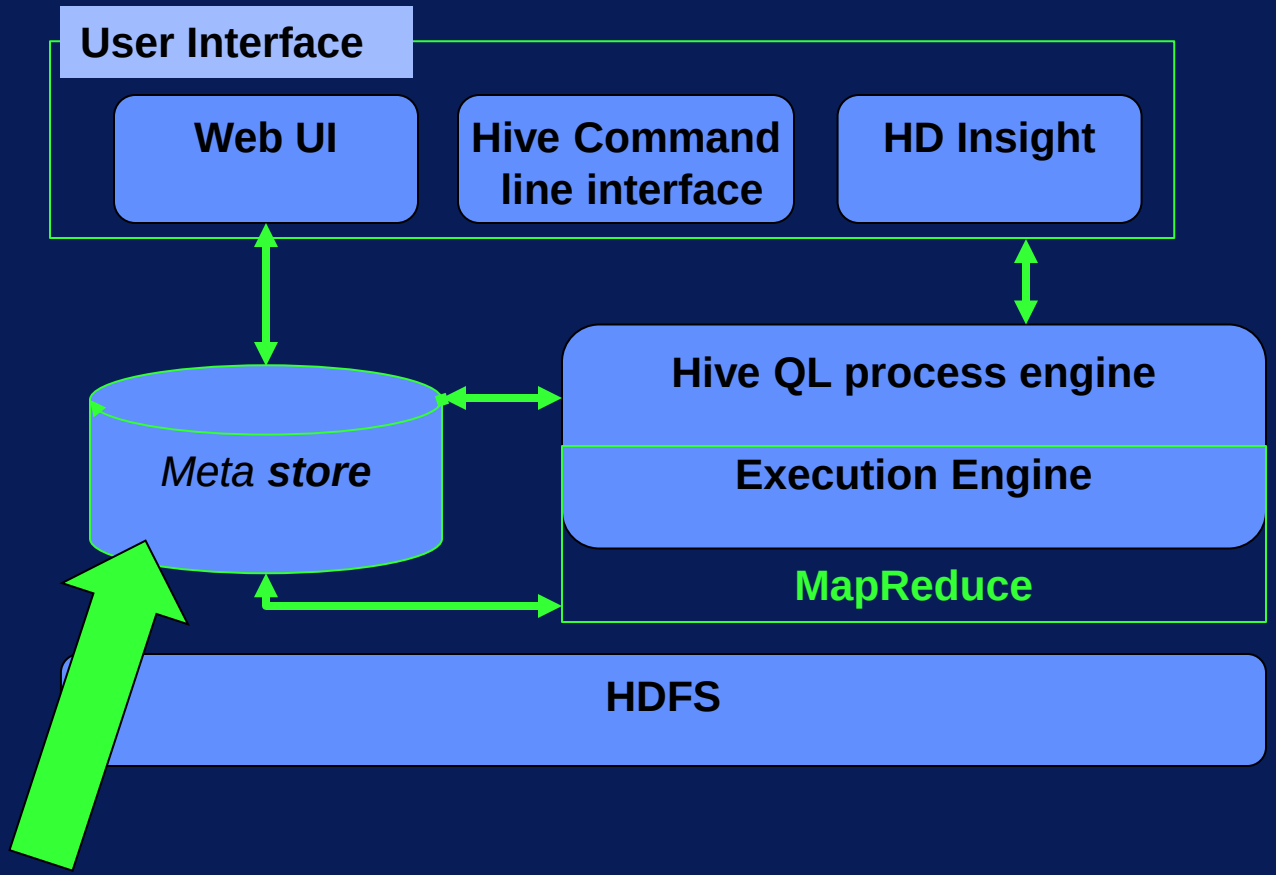



Shell
Driver
Hive Components
MetaStore
Compiler
Execution Engine

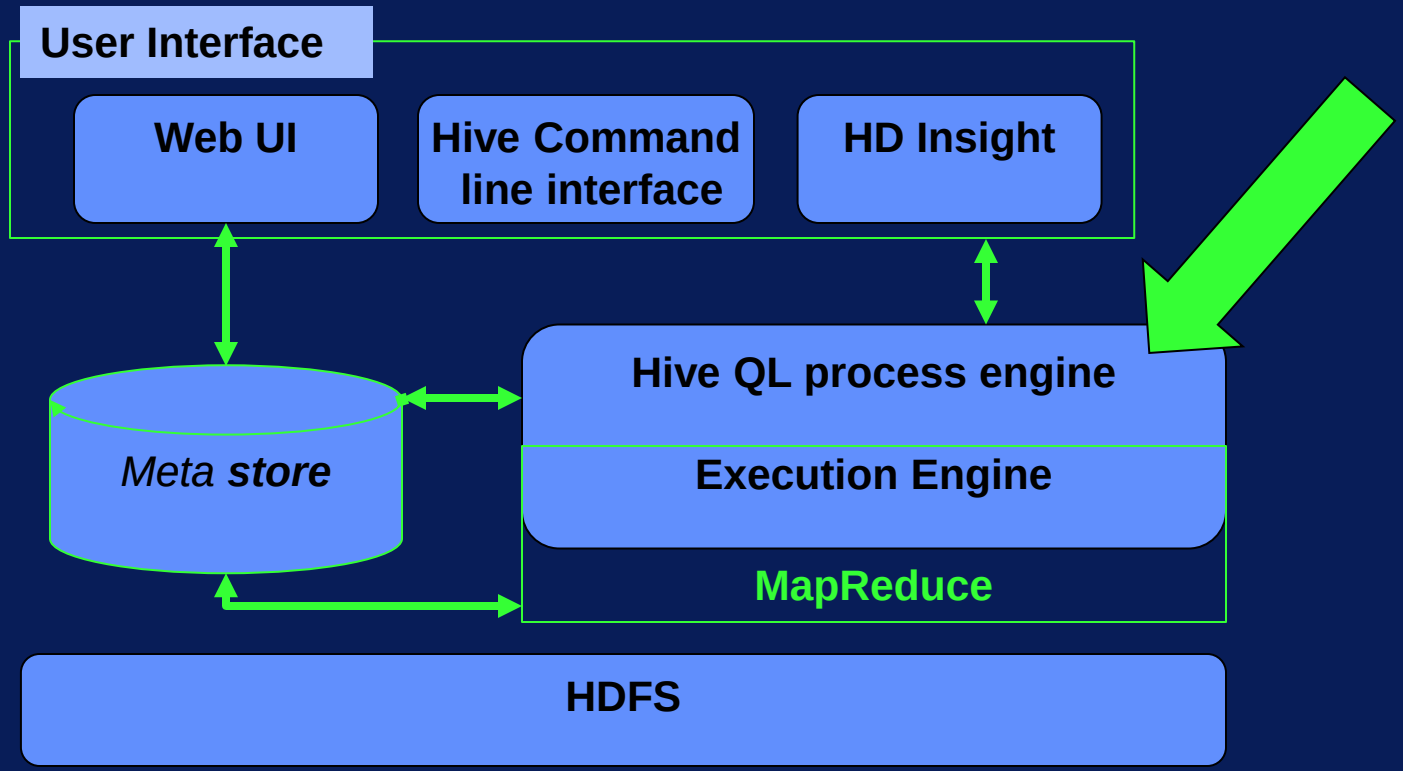
Hive Architecture



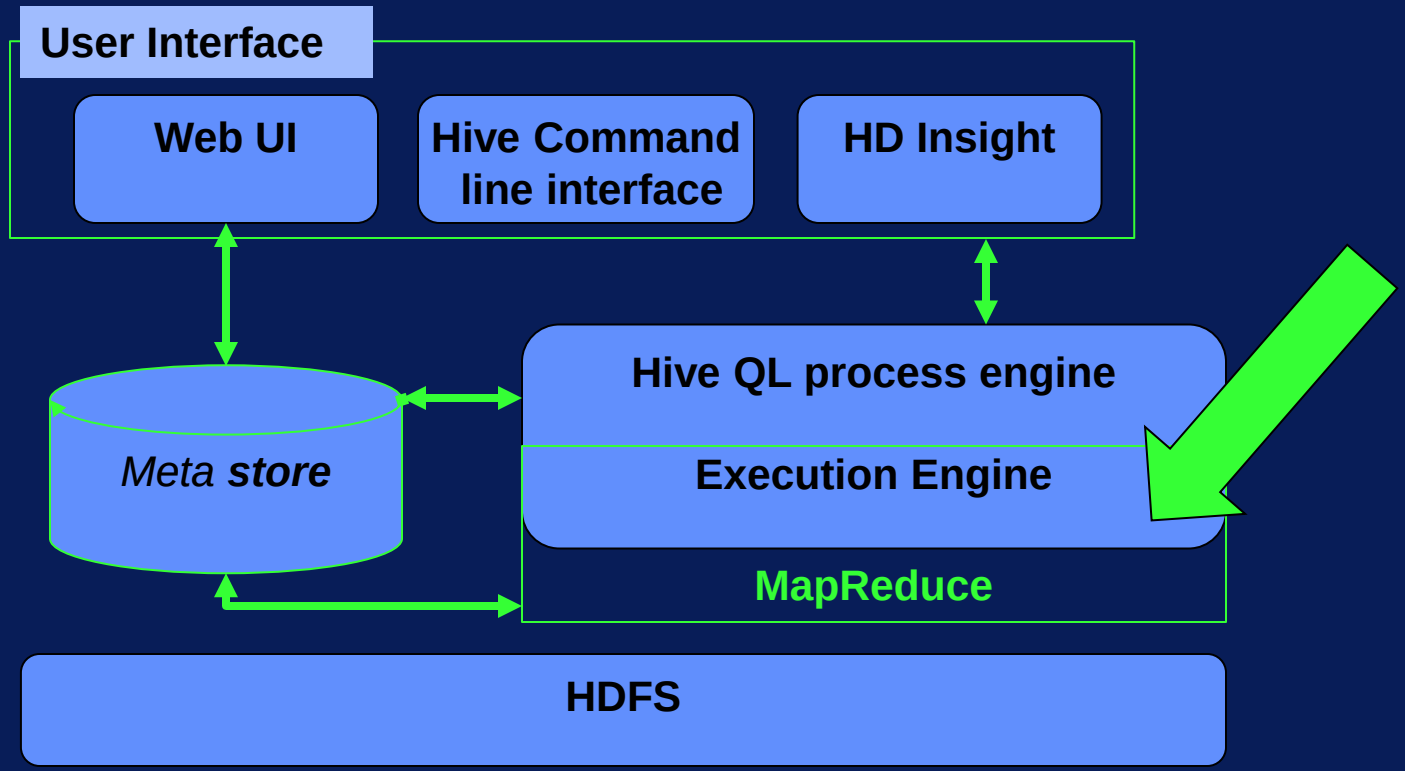
Hive Architecture

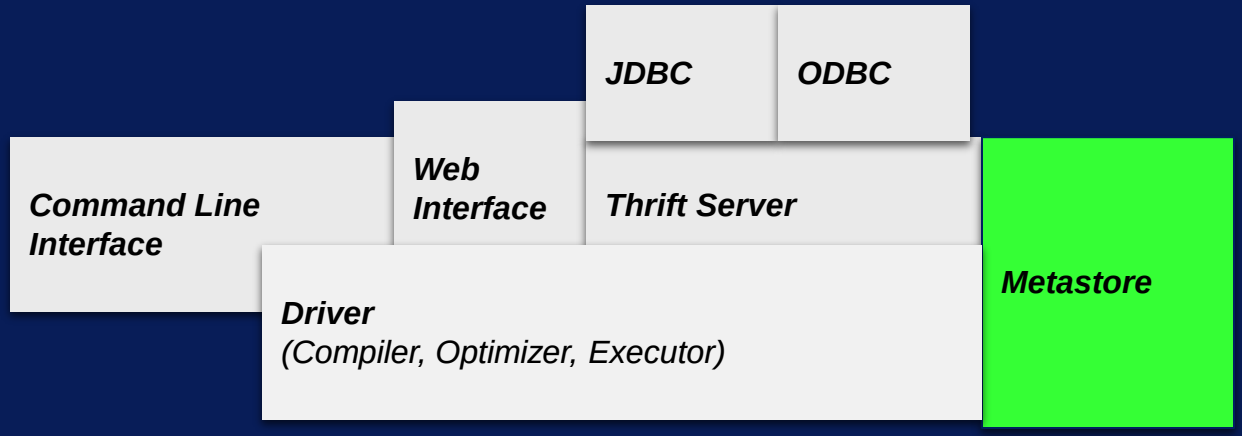


Hive Architecture



Hive Architecture

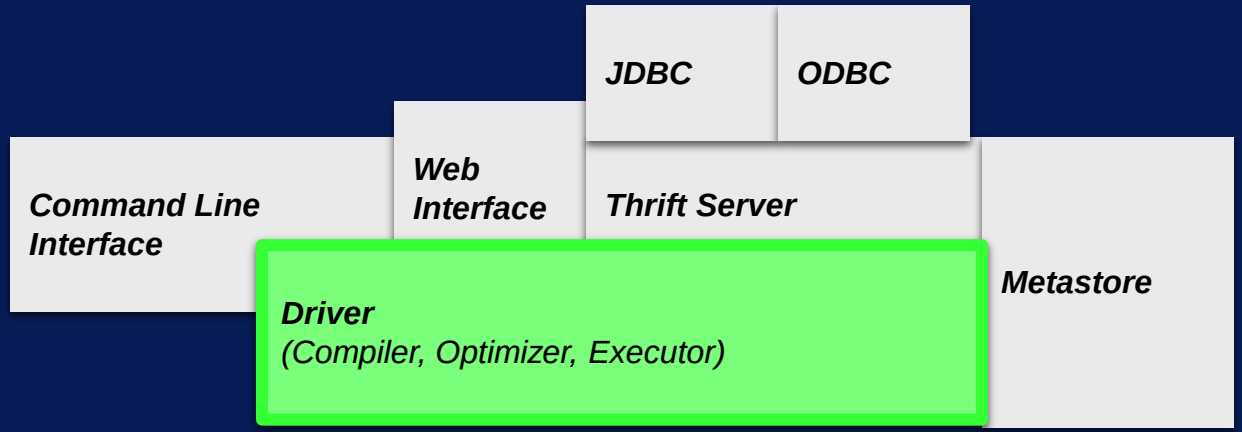




Metastore

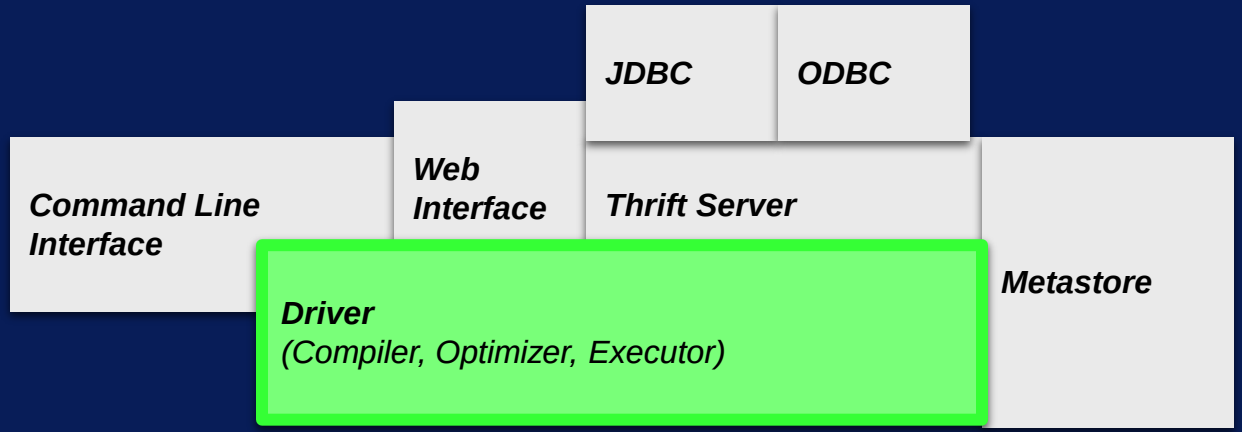
Stores the system catalog and meta data about tables, columns, partitions etc.

Stored on a traditional RDBMS



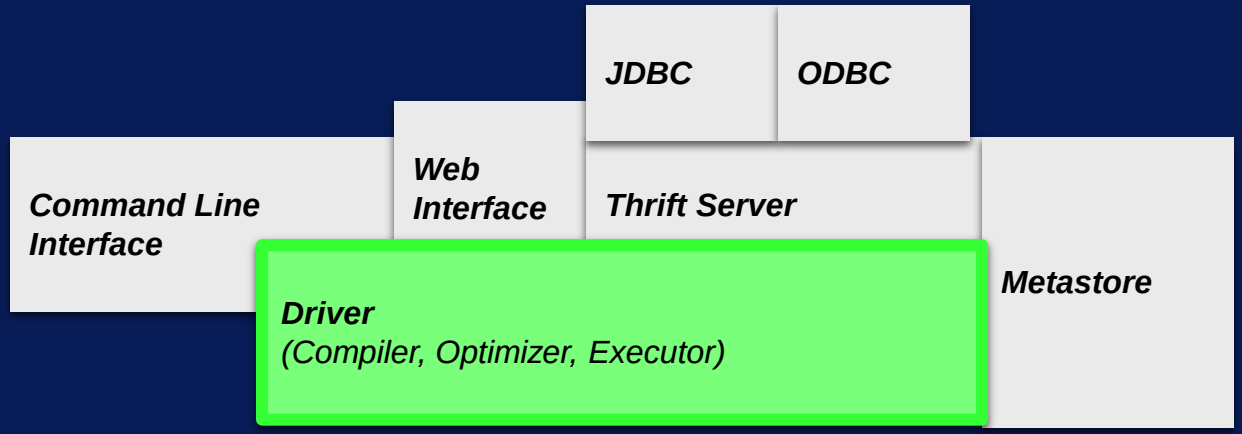
Driver

- Manages the lifecycle of a HiveQL statement
- Maintains a session handle and any session statistics



Query Compiler

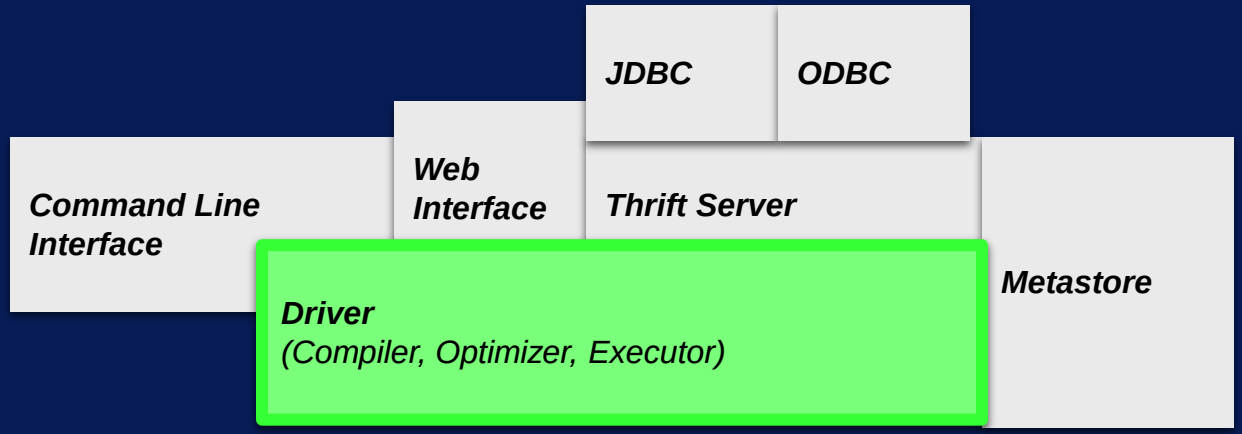
The component that compiles HiveQL into a directed acyclic graph of map/reduce tasks



Optimizer

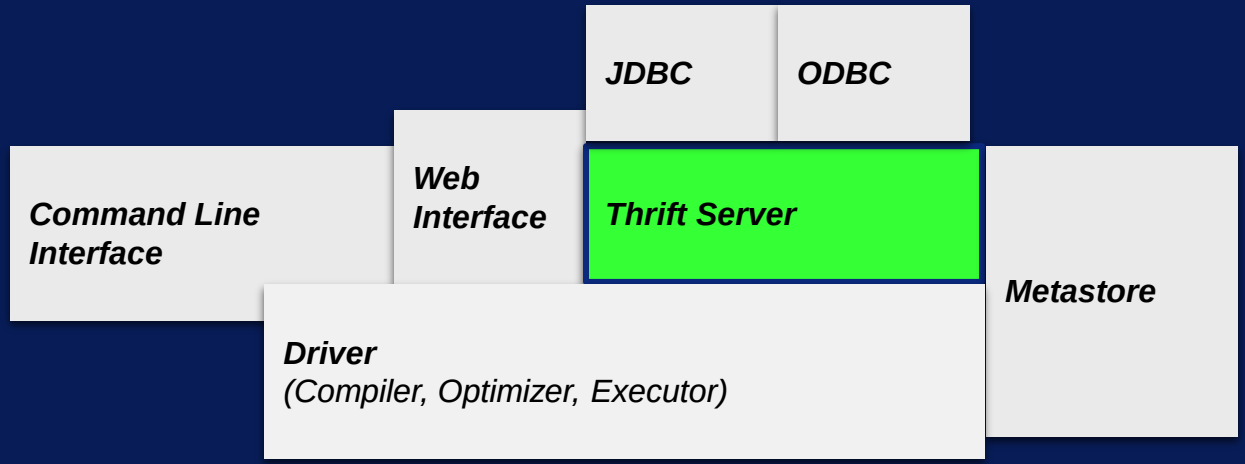
Consists of a chain of transformations

Performs Column Pruning , Partition Pruning, Repartitioning of Data



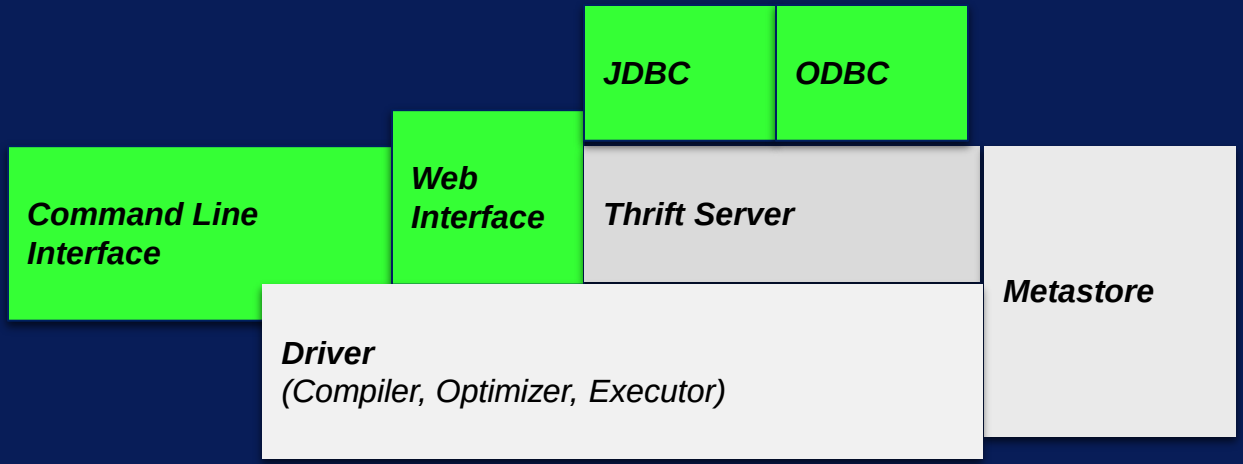
Execution Engine

- Executes the tasks produced by the compiler in proper dependency order
- Interacts with the underlying Hadoop instance



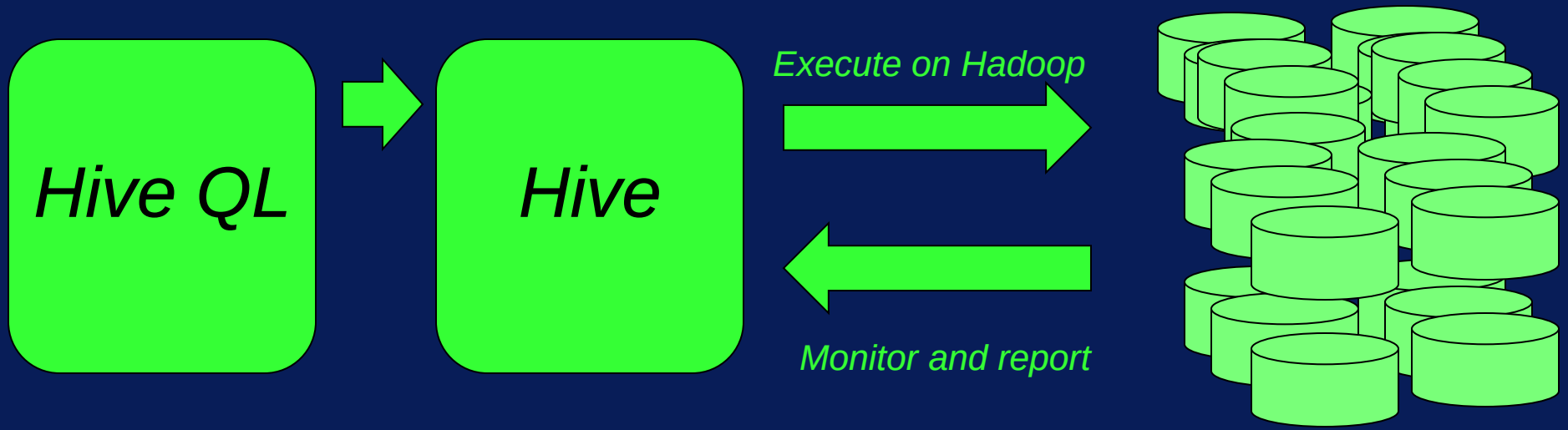
HiveServer

Provides a Thrift interface and a JDBC/ODBC server Enables Hive integration with other applications

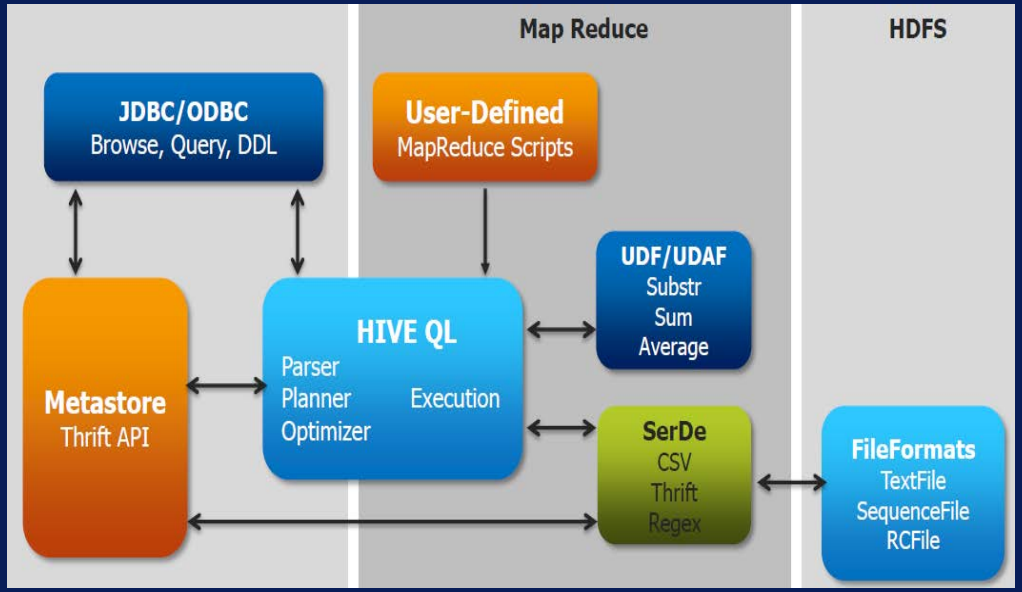


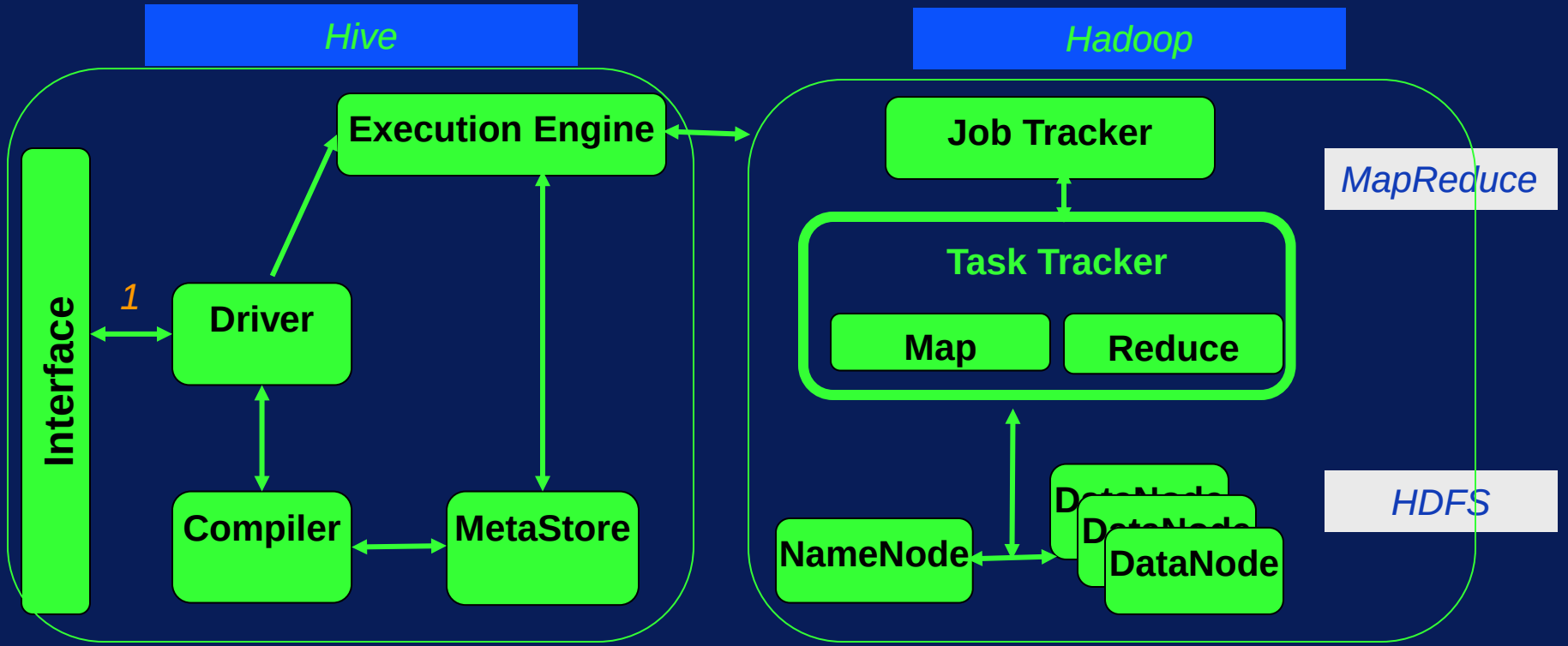
Client Components

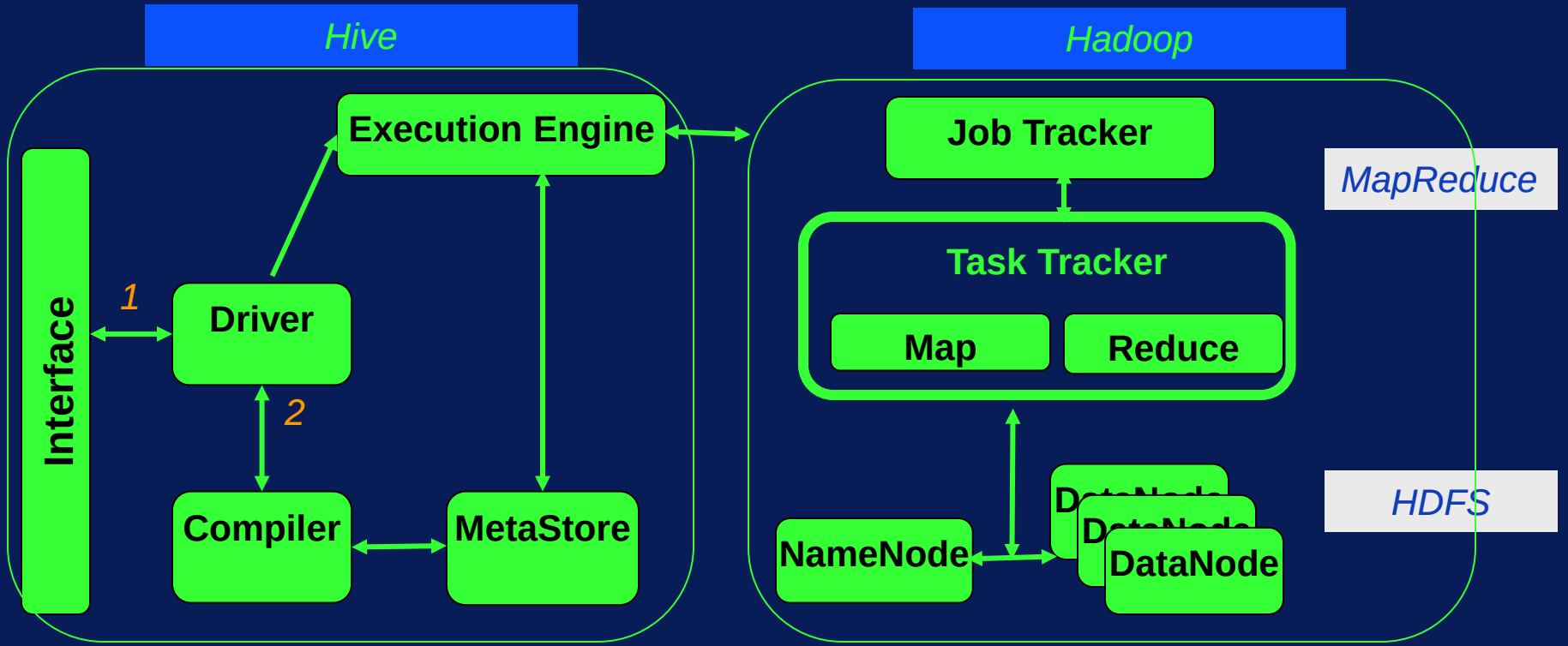
- Command Line Interface(CLI)
- Web UI
- JDBC/ODBC driver

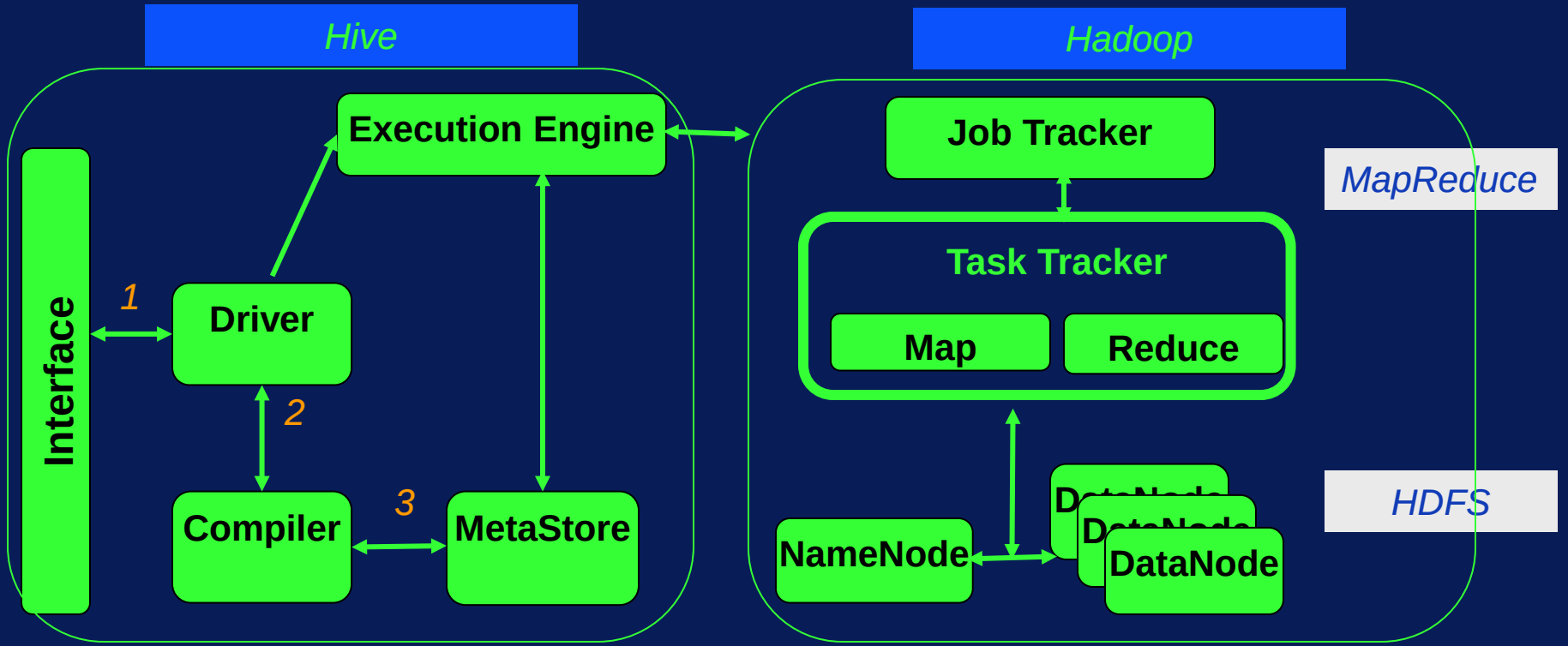

Hive QL
Hive
Execute on Hadoop
Monitor and report

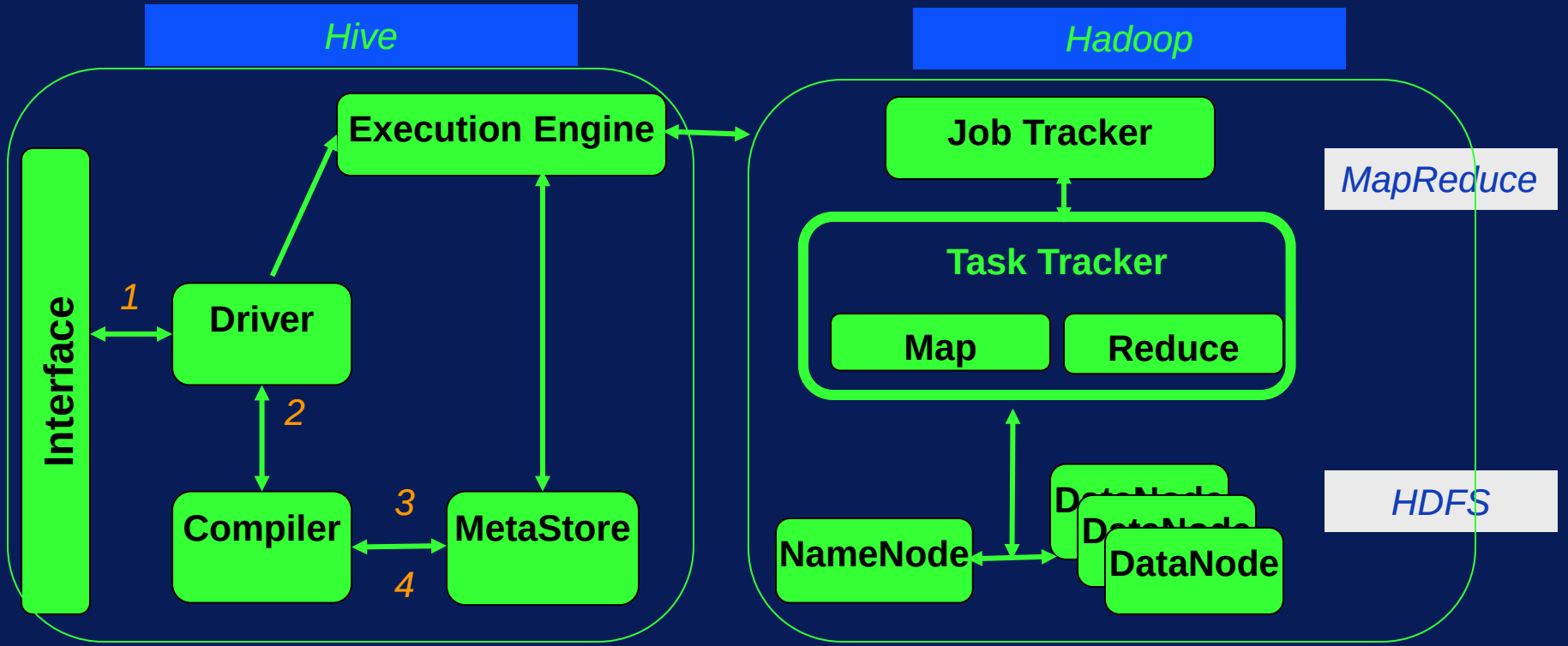
How Hive Works

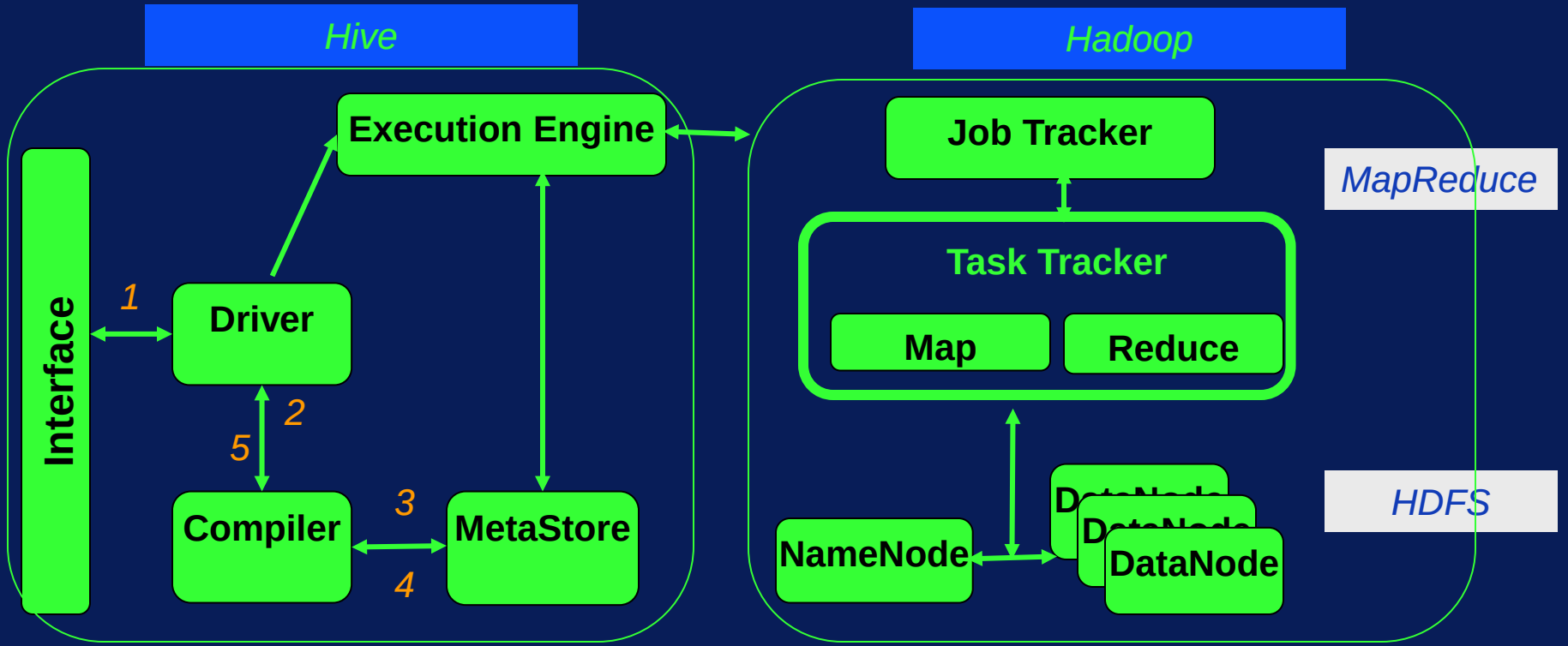


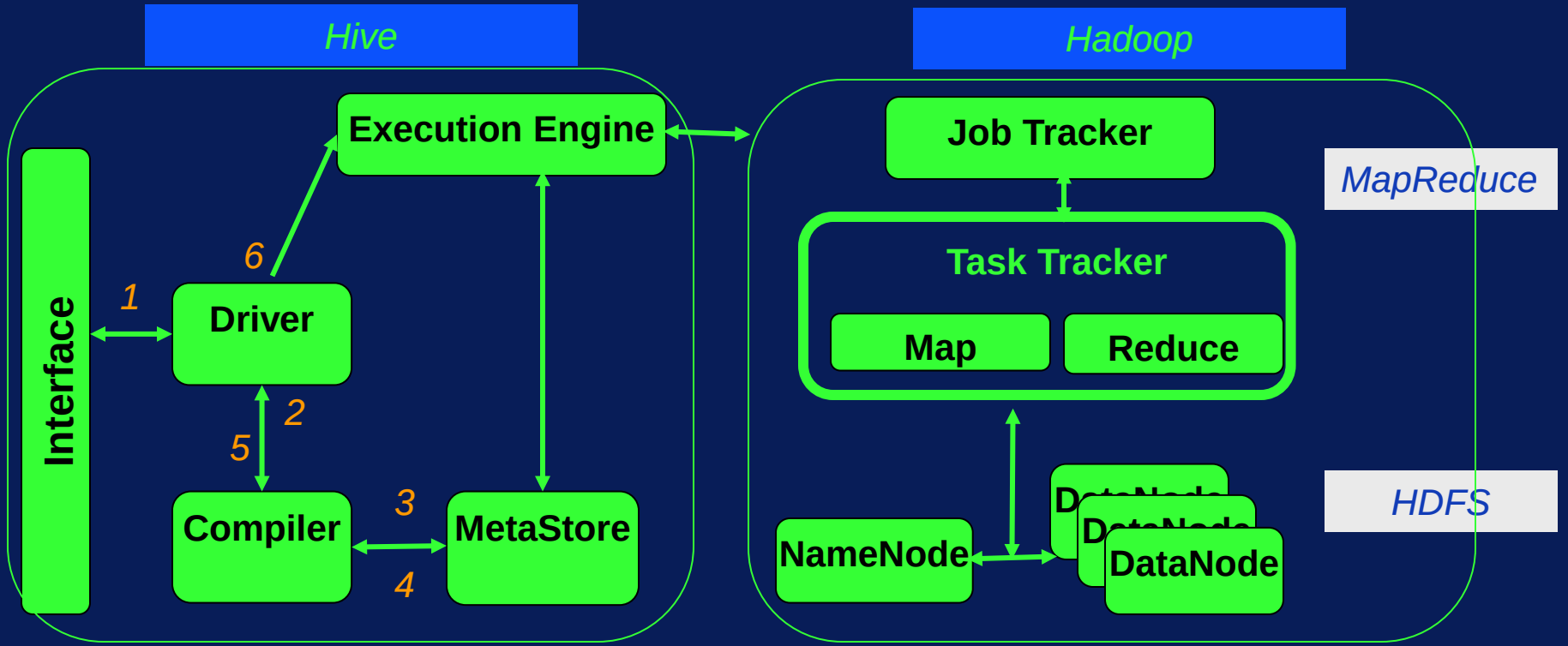

Hive
Hadoop
Execution Engine
Job Tracker
MapReduce
Interface
1
Driver
Task Tracker
Map
Reduce
Compiler
MetaStore
HDFS
NameNode
DataNode

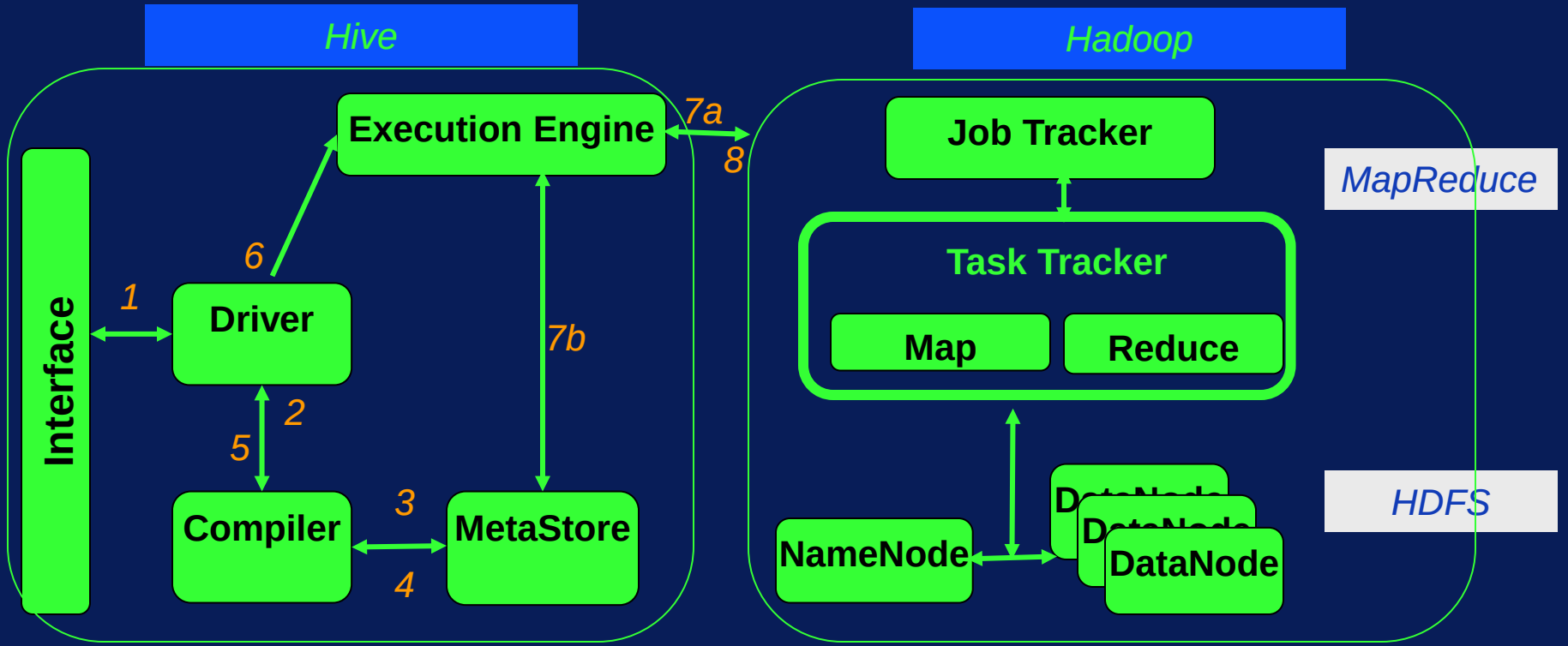

Hive
Hadoop
Execution Engine
Job Tracker
MapReduce
Interface
1
Driver
Task Tracker
Map
Reduce
2
Compiler
MetaStore
HDFS
NameNode
DataNode

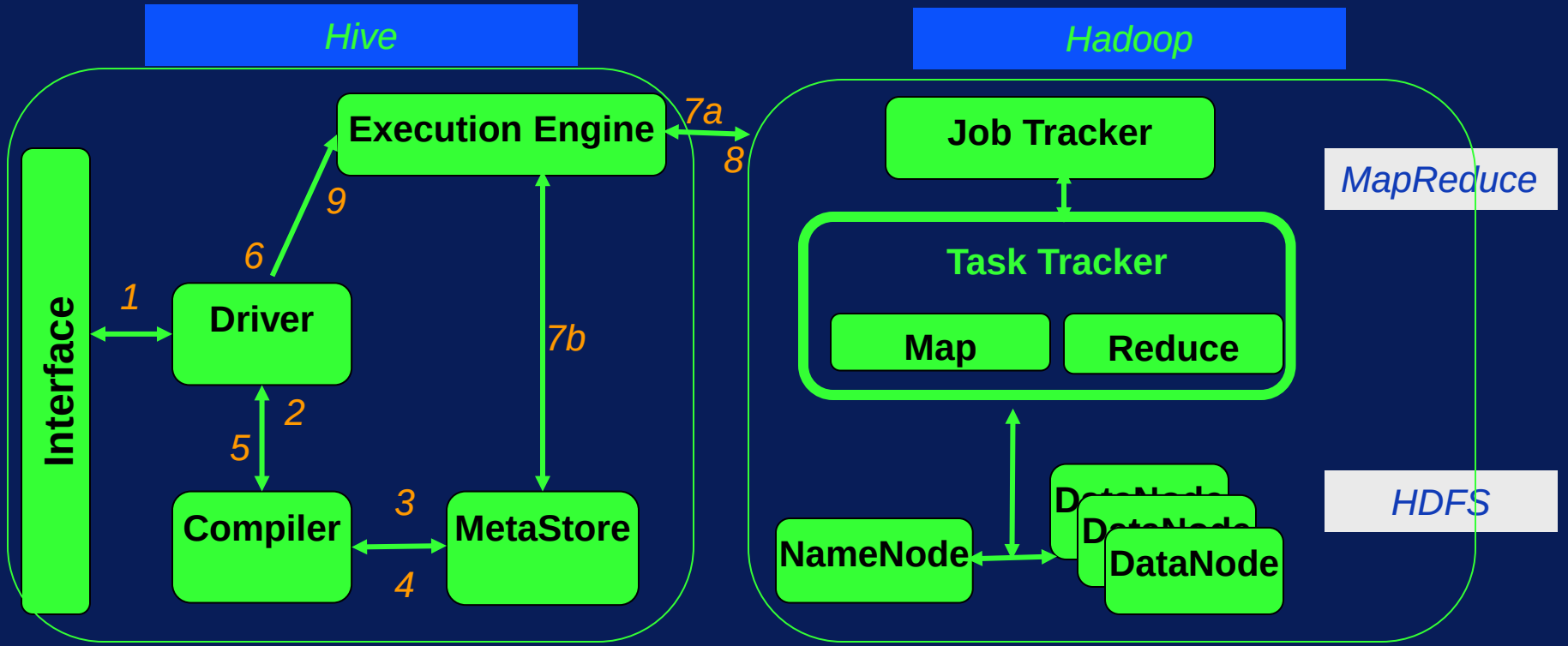

Hive
Hadoop
Execution Engine
Job Tracker
MapReduce
Interface
1
Driver
Task Tracker
Map
Reduce
2
3
Compiler
MetaStore
HDFS
NameNode
DataNode

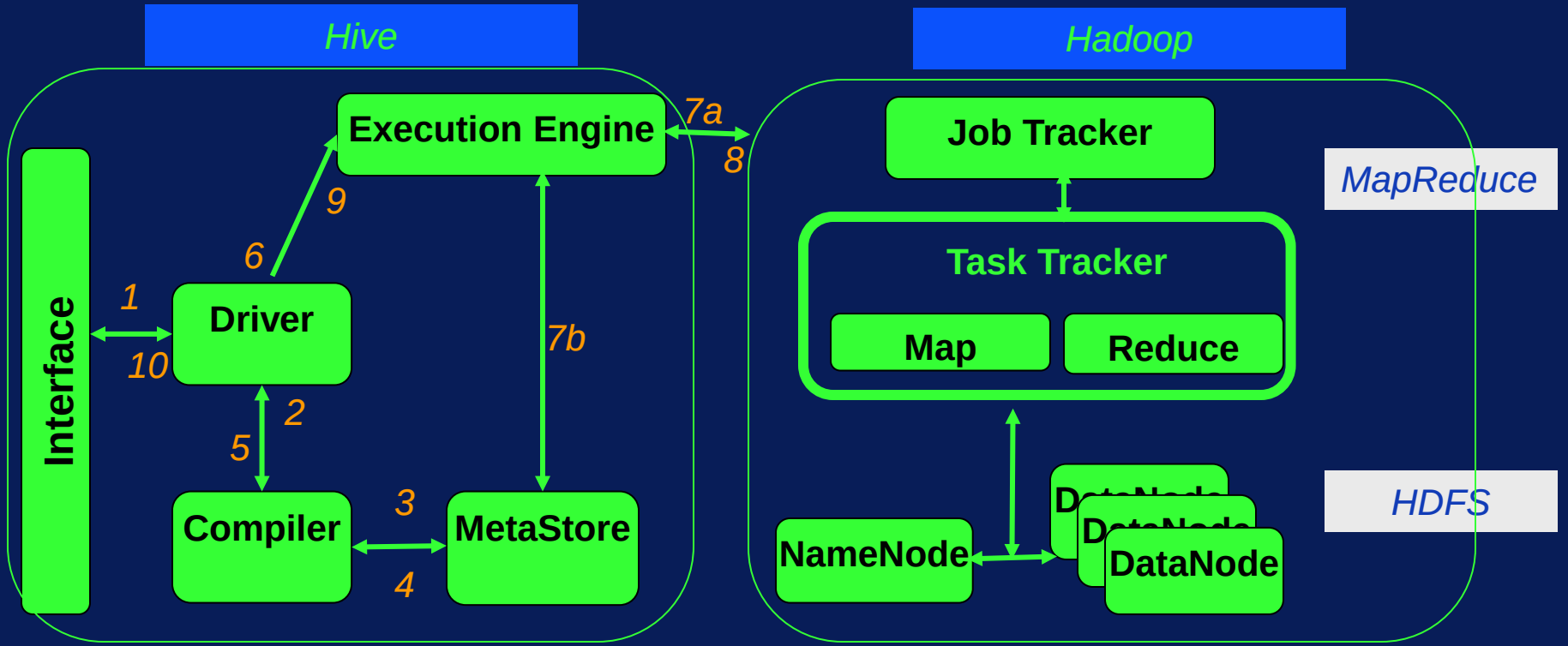

Hive
Hadoop
Execution Engine
Job Tracker
MapReduce
Interface
1
Driver
Task Tracker
Map
Reduce
2
3
Compiler
MetaStore
4
HDFS
NameNode
DataNode


Hive
Hadoop
Execution Engine
Job Tracker
MapReduce
Interface
1
Driver
Task Tracker
Map
Reduce
2
5
3
Compiler
MetaStore
4
HDFS
NameNode
DataNode

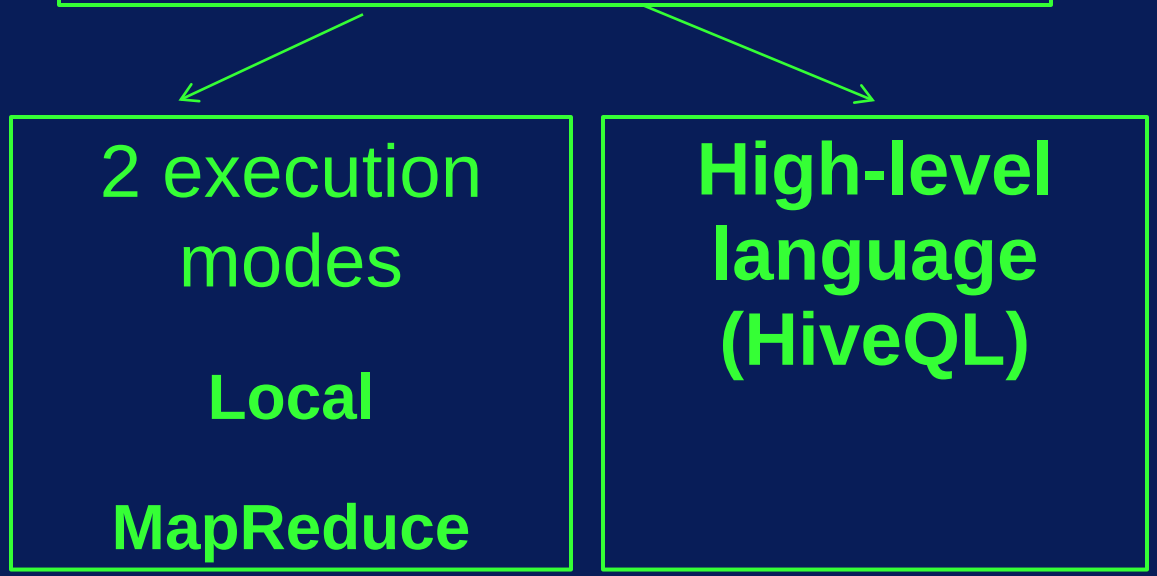

Hive
Hadoop
Execution Engine
Job Tracker
MapReduce
Interface
6
1
Driver
Task Tracker
Map
Reduce
2
5
3
Compiler
MetaStore
4
HDFS
NameNode
DataNode


Hive
Hadoop
Execution Engine
7a
8
Interface
6
1
Driver
7b
2
5
Compiler
3
4
MetaStore
Job Tracker
MapReduce
Task Tracker
Map
Reduce
NameNode
DataNode
HDFS


Hive
Hadoop
Execution Engine
Job Tracker
MapReduce
Task Tracker
Map
Reduce
Interface
Driver
Compiler
MetaStore
NameNode
DataNode
HDFS
1
2
3
4
5
6
7a
7b
8
9


Hive
Hadoop
Execution Engine
Interface
Driver
Compiler
MetaStore
Job Tracker
Task Tracker
Map
Reduce
MapReduce
HDFS
NameNode
DataNode
1
2
3
4
5
6
7a
7b
8
9
10

Ways to run Hive jobs



Two Hive Main Modes Of Operations

Interactive mode

***Via* Console**

Batch mode

By submitting a script

Hive’s Data Units

Databases
Tables
Partitions
Buckets (or clusters)

Very similar to SQL and Relational DBs

3-Levels: Tables → Partitions → Buckets

Data Model

Table maps to a HDFS directory

Partition maps to sub-directories under the table

Bucket maps to files under each partition

Tables

Similar to tables in relational DBs

Each table has corresponding directory in HDFS

Partitions

- Analogous to dense indexes on partition columns
- Nested sub-directories in HDFS for each combination of partition column values
- Allows users to efficiently retrieve rows

Hive Data Structures

Traditional Database concepts

Supports primitive types

Additional types and structures

Hive Data Structures

Traditional database concepts

- Tables
- Rows
- Columns
- Partitions

Hive Data Structures

Basic types

- Integers
- Floats
- Doubles
- Strings

Hive Data Structures

Additionally supports

Associative arrays

map<key-type, value-type>

Lists

list<element type>

Structs

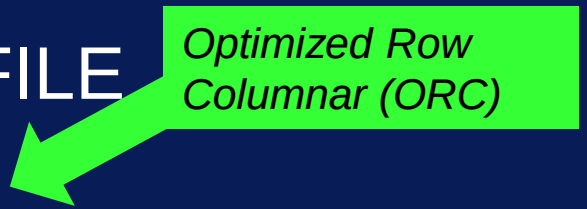
struct<file name: file type…>

Hive File Formats

Hive enables users store different file formats

Performance improvements

- TEXTFILE
- SEQUENCEFILE
- ORC
- RCFILE



Hive Commands

Hive Interface

Command Line interface

Web interface or Hue

Java Database connectivity

Hive Commands

Database

Set of Tables - name conflicts resolution

Table

Set of Rows - have the same columns

Row

A single record - a set of columns

Column

Value and type for a single value

Tables Commands

- SHOW TABLES
- CREATE TABLE
- ALTER TABLE
- DROP TABLE

Hive Commands

CREATE TABLE mytable (myint INT, bar STRING)
PARTITIONED BY (ds STRING);

SHOW TABLES '.*my';

ALTER TABLE mytable **ADD COLUMNS** (new_col INT);

DROP TABLE mytable;

A table in Hive is an HDFS directory in Hadoop

Hive Commands

Schema is known at creation time (like DB schema)

Partitioned tables have “sub-directories”, one for each partition

```
CREATE TABLE mypeople (
id              int,
name            string
)
partitioned by (date string)
```

Hive Query Language

JOIN

```
SELECT t1.a1 as c1, t2.b1 as c2
FROM t1 JOIN t2 ON (t1.a2 = t2.b2);
```

INSERTION

```
INSERT OVERWRITE TABLE t1
SELECT *  FROM t2;
```

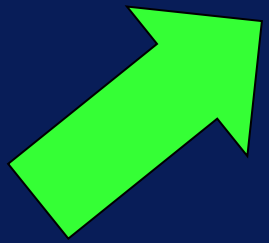
Format Rows

CREATE TABLE mypeople (id INT, name STRING)

ROW FORMAT

DELIMETED FIELDS TERMINATED BY <output format>

LINES TERMINATED BY ‘\n’;



Loading Data into HIVE

HDFS

```
LOAD DATA INPATH ‘mybigdata’
        [OVERWRITE] INTO TABLE mypeople;
```

Local file system

```
LOAD DATA LOCAL INPATH ‘mybigdata’
        INTO TABLE mypeople;
```

Partitions

LOAD DATA INPATH ‘myweblogs’ INTO TABLE mypeople PARTITION (dt=12-12-2020);

BUCKETS

Set hive.enforce.bucketing property to true

```
CREATE TABLE mycustomers(id INT, purchases DOUBLE, name STRING)
CLUSTERED BY id into 32 BUCKETS;
```

BUCKETS

```
SELECT min(cost) FROM mysales
TABLESAMPLE (BUCKET 10 OUT OF 32 ON
rand());
```

VIEWS

Similar to SQL Views

- Virtual table in Metastore
- SHOW TABLES

JOINS

LEFT OUTER JOIN

RIGHT OUTER JOIN

FULL OUTER JOIN

```
hive> SELECT c.ID, c.NAME, c.AGE, o.AMOUNT FROM
CUSTOMERS c JOIN ORDERS o ON
(c.ID = o.CUSTOMER_ID);
```

DROP TABLE

- DROP TABLE MyCustomers;

DELETE PARTITION

```
ALTER TABLE MyCustomers DROP
PARTITION (col2=100);
```

Hive Command Example

Steps:

- **Create a table**
- **Load data into table**
- **Query loaded data**
- **Delete the table**

Using Hive

- **Open the Shell**
- **Open a terminal and type**

  $hive

  - Prompt should appear:

    hive>

Hands-on Example

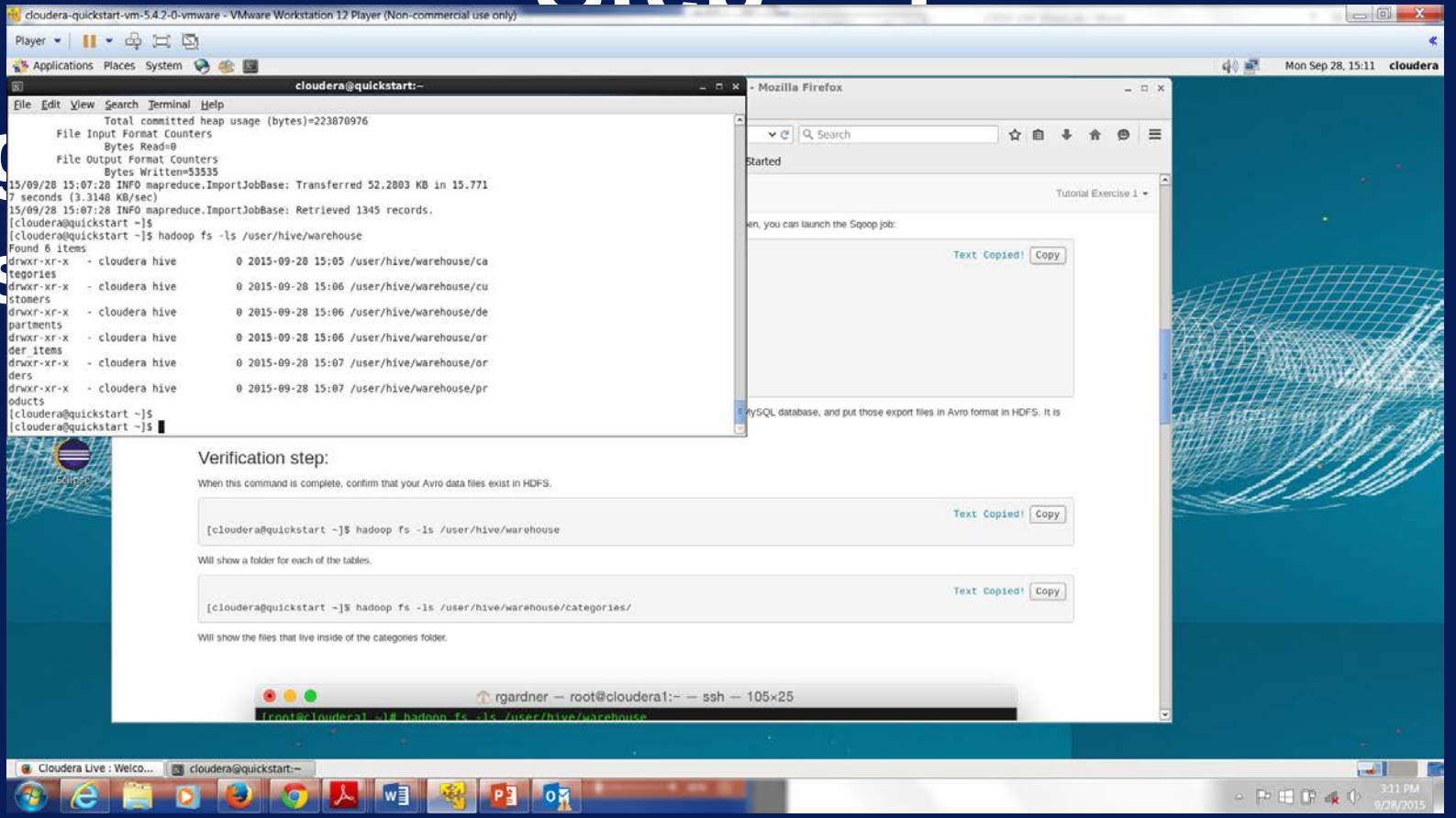
Start the QuickStart VM

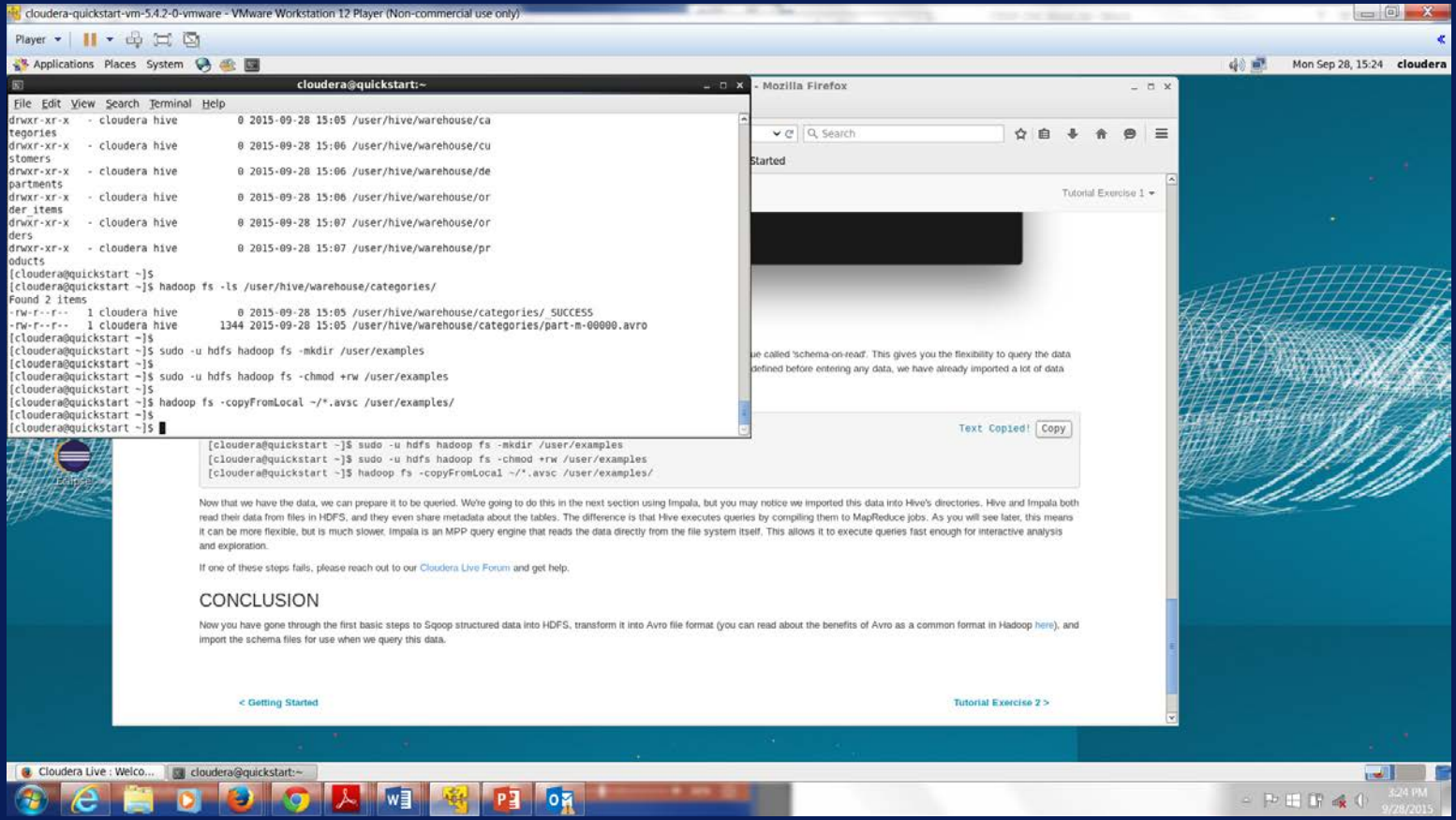
Data Co. Data set

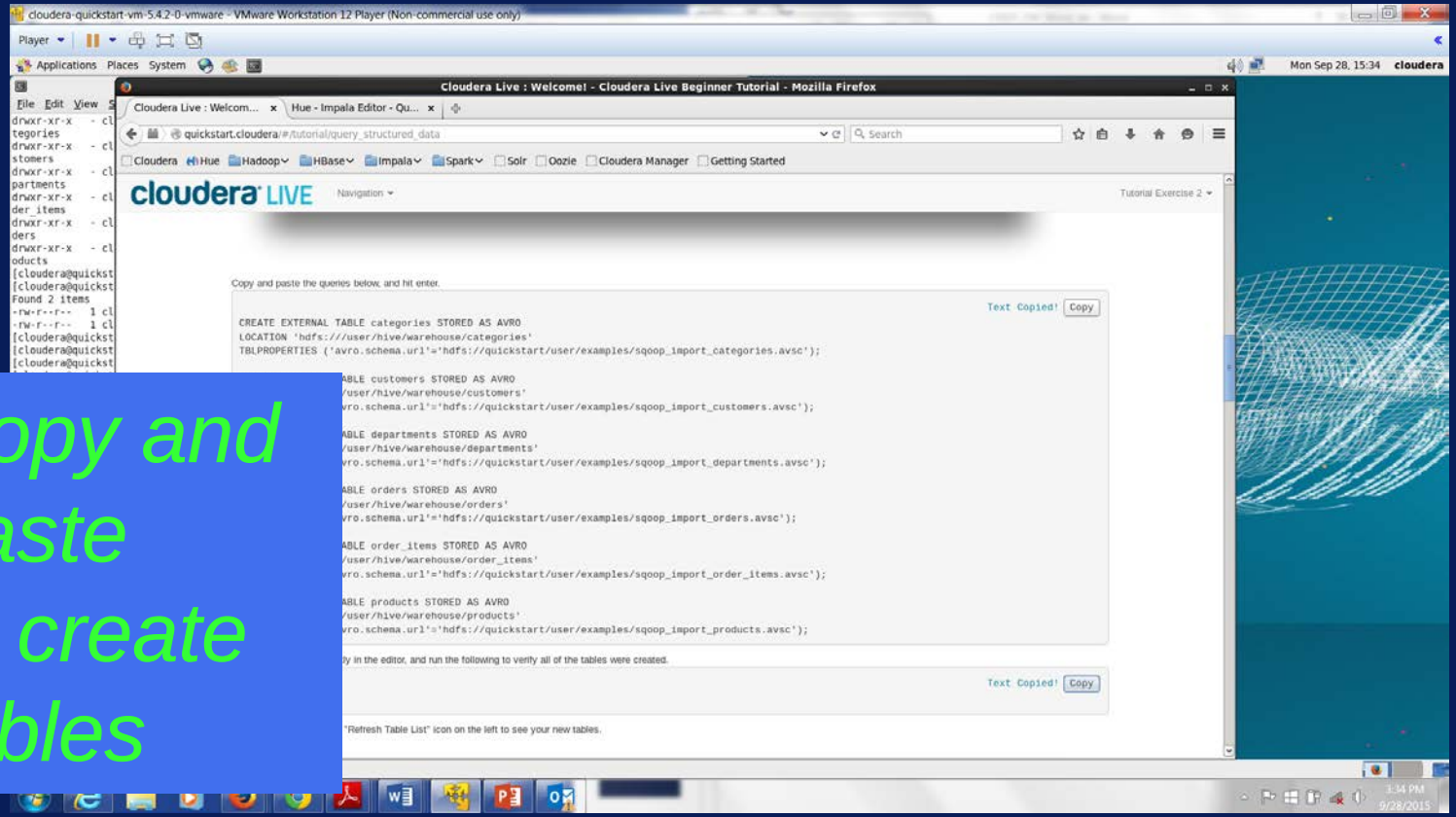
Open Hue

Step 1

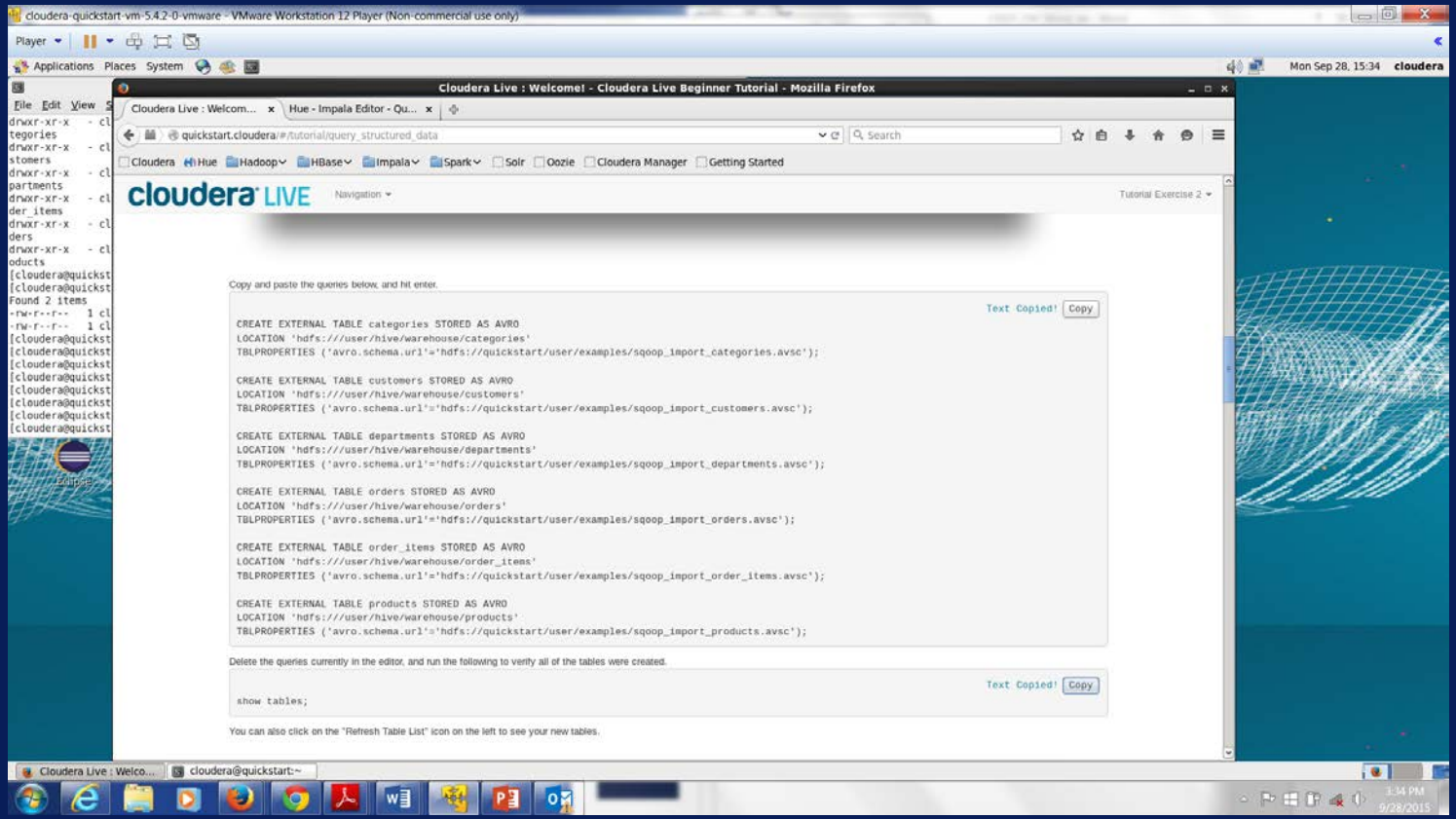
Ing
us


cloudera-quickstart-vm-5.4.2-0-vmware - VMware Workstation 12 Player (Non-commercial use only)
Player
Applications Places System
Mon Sep 28, 15:11 cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help
Total committed heap usage (bytes)=223870976
File Input Format Counters
Bytes Read=0
File Output Format Counters
Bytes Written=53535
15/09/28 15:07:28 INFO mapreduce.ImportJobBase: Transferred 52.2803 KB in 15.771
7 seconds (3.3148 KB/sec)
15/09/28 15:07:28 INFO mapreduce.ImportJobBase: Retrieved 1345 records.
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse
Found 6 items
drwxr-xr-x - cloudera hive 0 2015-09-28 15:05 /user/hive/warehouse/categories
drwxr-xr-x - cloudera hive 0 2015-09-28 15:06 /user/hive/warehouse/customers
drwxr-xr-x - cloudera hive 0 2015-09-28 15:06 /user/hive/warehouse/departments
drwxr-xr-x - cloudera hive 0 2015-09-28 15:06 /user/hive/warehouse/order_items
drwxr-xr-x - cloudera hive 0 2015-09-28 15:07 /user/hive/warehouse/orders
drwxr-xr-x - cloudera hive 0 2015-09-28 15:07 /user/hive/warehouse/products
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
Mozilla Firefox
Started
Tutorial Exercise 1
Text Copied! Copy
MySQL database, and put those export files in Avro format in HDFS. It is
Verification step:
When this command is complete, confirm that your Avro data files exist in HDFS.
Text Copied! Copy
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse
Will show a folder for each of the tables.
Text Copied! Copy
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/categories/
Will show the files that live inside of the categories folder.
rgardner — root@cloudera1:~ — ssh — 105×25
Cloudera Live : Welco...
cloudera@quickstart:~
3:11 PM
9/28/2015


cloudera-quickstart-vm-5.4.2-0-vmware - VMware Workstation 12 Player (Non-commercial use only)
Player
Applications Places System
Mon Sep 28, 15:24 cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help
drwxr-xr-x - cloudera hive 0 2015-09-28 15:05 /user/hive/warehouse/categories
drwxr-xr-x - cloudera hive 0 2015-09-28 15:06 /user/hive/warehouse/customers
drwxr-xr-x - cloudera hive 0 2015-09-28 15:06 /user/hive/warehouse/departments
drwxr-xr-x - cloudera hive 0 2015-09-28 15:06 /user/hive/warehouse/order_items
drwxr-xr-x - cloudera hive 0 2015-09-28 15:07 /user/hive/warehouse/orders
drwxr-xr-x - cloudera hive 0 2015-09-28 15:07 /user/hive/warehouse/products
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/categories/
Found 2 items
-rw-r--r-- 1 cloudera hive 0 2015-09-28 15:05 /user/hive/warehouse/categories/_SUCCESS
-rw-r--r-- 1 cloudera hive 1344 2015-09-28 15:05 /user/hive/warehouse/categories/part-m-00000.avro
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -mkdir /user/examples
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -chmod +rw /user/examples
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hadoop fs -copyFromLocal ~/*.avsc /user/examples/
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$
Mozilla Firefox
Search
Started
Tutorial Exercise 1
called 'schema-on-read'. This gives you the flexibility to query the data
defined before entering any data, we have already imported a lot of data
Text Copied! Copy
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -mkdir /user/examples
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -chmod +rw /user/examples
[cloudera@quickstart ~]$ hadoop fs -copyFromLocal ~/*.avsc /user/examples/
Now that we have the data, we can prepare it to be queried. We're going to do this in the next section using Impala, but you may notice we imported this data into Hive's directories. Hive and Impala both read their data from files in HDFS, and they even share metadata about the tables. The difference is that Hive executes queries by compiling them to MapReduce jobs. As you will see later, this means it can be more flexible, but is much slower. Impala is an MPP query engine that reads the data directly from the file system itself. This allows it to execute queries fast enough for interactive analysis and exploration.
If one of these steps fails, please reach out to our Cloudera Live Forum and get help.
CONCLUSION
Now you have gone through the first basic steps to Sqoop structured data into HDFS, transform it into Avro file format (you can read about the benefits of Avro as a common format in Hadoop here), and import the schema files for use when we query this data.
< Getting Started
Tutorial Exercise 2 >
Cloudera Live : Welco...
cloudera@quickstart:~
3:24 PM
9/28/2015

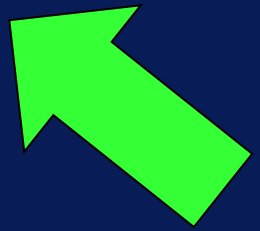


Copy and paste to create tables


cloudera-quickstart-vm-5.4.2-0-vmware - VMware Workstation 12 Player (Non-commercial use only)
Player
Applications Places System
Mon Sep 28, 15:34 cloudera
Cloudera Live : Welcome! - Cloudera Live Beginner Tutorial - Mozilla Firefox
Cloudera Live : Welcom...
Hue - Impala Editor - Qu...
quickstart.cloudera/#/tutorial/query_structured_data
Search
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
cloudera LIVE
Navigation
Tutorial Exercise 2
Copy and paste the queries below, and hit enter.
Text Copied!
Copy
CREATE EXTERNAL TABLE categories STORED AS AVRO
LOCATION 'hdfs:///user/hive/warehouse/categories'
TBLPROPERTIES ('avro.schema.url'='hdfs://quickstart/user/examples/sqoop_import_categories.avsc');
CREATE EXTERNAL TABLE customers STORED AS AVRO
LOCATION 'hdfs:///user/hive/warehouse/customers'
TBLPROPERTIES ('avro.schema.url'='hdfs://quickstart/user/examples/sqoop_import_customers.avsc');
CREATE EXTERNAL TABLE departments STORED AS AVRO
LOCATION 'hdfs:///user/hive/warehouse/departments'
TBLPROPERTIES ('avro.schema.url'='hdfs://quickstart/user/examples/sqoop_import_departments.avsc');
CREATE EXTERNAL TABLE orders STORED AS AVRO
LOCATION 'hdfs:///user/hive/warehouse/orders'
TBLPROPERTIES ('avro.schema.url'='hdfs://quickstart/user/examples/sqoop_import_orders.avsc');
CREATE EXTERNAL TABLE order_items STORED AS AVRO
LOCATION 'hdfs:///user/hive/warehouse/order_items'
TBLPROPERTIES ('avro.schema.url'='hdfs://quickstart/user/examples/sqoop_import_order_items.avsc');
CREATE EXTERNAL TABLE products STORED AS AVRO
LOCATION 'hdfs:///user/hive/warehouse/products'
TBLPROPERTIES ('avro.schema.url'='hdfs://quickstart/user/examples/sqoop_import_products.avsc');
Delete the queries currently in the editor, and run the following to verify all of the tables were created.
Text Copied!
Copy
show tables;
You can also click on the "Refresh Table List" icon on the left to see your new tables.
Cloudera Live : Welco...
cloudera@quickstart:~
3:34 PM
9/28/2015

CREATE TABLE

```
CREATE EXTERNAL TABLE categories STORED
AS AVRO
```



```
LOCATION
'hdfs:///user/hive/warehouse/categories'


TBLPROPERTIES
('avro.schema.url'='hdfs://quickstart/user/examples
/sqoop_import_categories.avsc');
```

CREATE TABLE

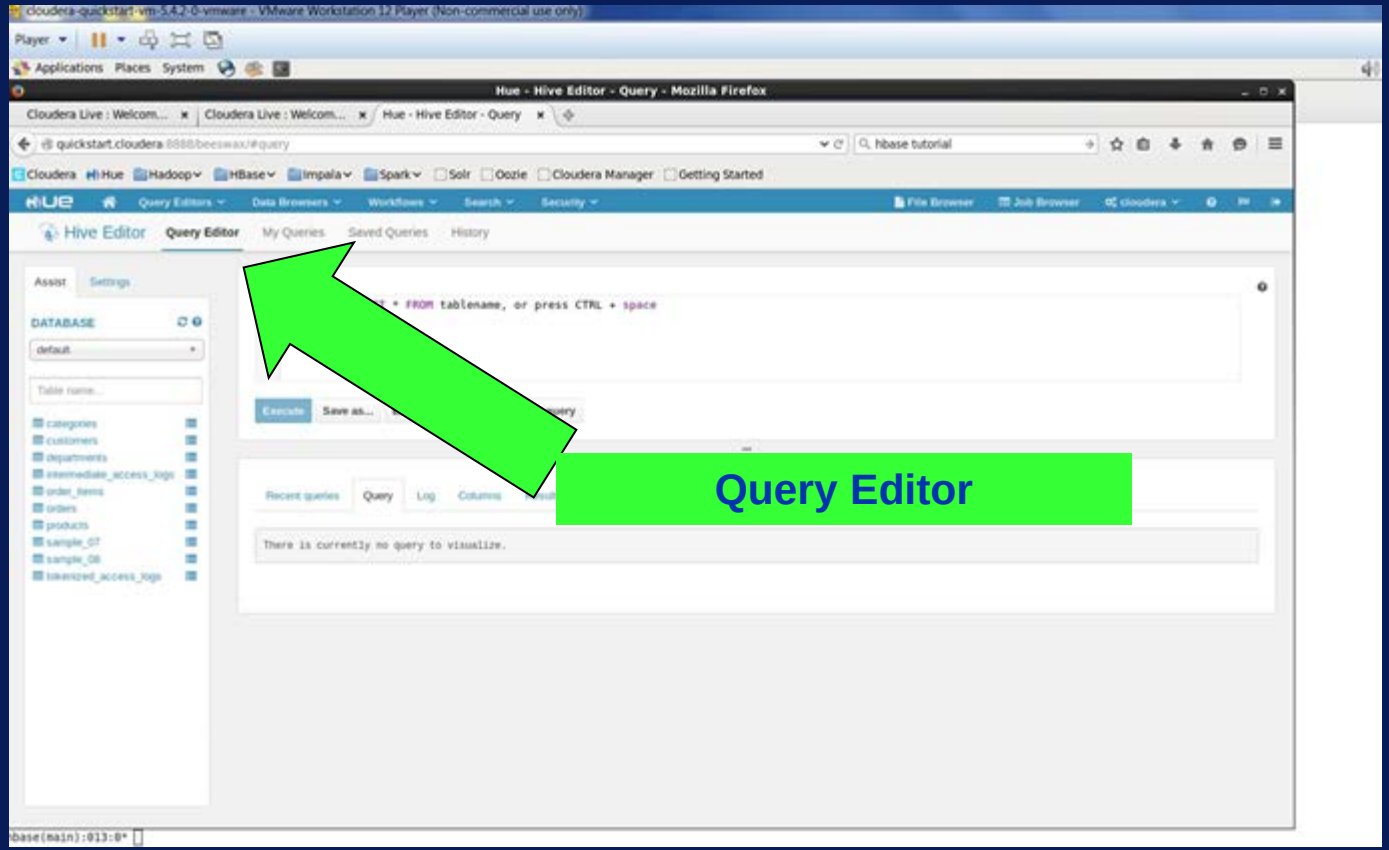
```
CREATE EXTERNAL TABLE customers STORED AS AVRO
LOCATION
'hdfs:///user/hive/warehouse/customers'

TBLPROPERTIES
('avro.schema.url'='hdfs://quickstart/user/examples/sqoop_import_customers.avsc');
```

Create tables for each category

- Categories
- Customers
- Departments
- Order items
- Orders
- Products

HUE the Hadoop UI



Hive Query Examples on DataCo. Data in the VM

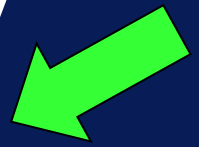
```
SHOW TABLES;
SELECT * FROM CUSTOMERS;
```

Find the top salaries above 80K in 2007

```
SELECT sample_07.description, sample_07.salary
FROM
  sample_07
WHERE
( sample_07.salary > 80000)
ORDER BY sample_07.salary DESC
LIMIT 1000
```

Salary growth sorted from 2007-08

```
SELECT s07.description, s07.salary, s08.salary,
  s08.salary - s07.salary
FROM
  sample_07 s07 JOIN sample_08 s08
ON ( s07.code = s08.code)
WHERE
s07.salary < s08.salary
ORDER BY s08.salary-s07.salary DESC
LIMIT 1000
```



Job loss among the top earners in 2007

```
SELECT s07.description, s07.total_emp, s08.total_emp, s07.salary
FROM
  sample_07 s07 JOIN
  sample_08 s08
ON ( s07.code = s08.code )
WHERE
( s07.total_emp > s08.total_emp
AND s07.salary > 100000 )
ORDER BY s07.salary DESC
LIMIT 1000
```