**ASSIGNMENT-6.3**

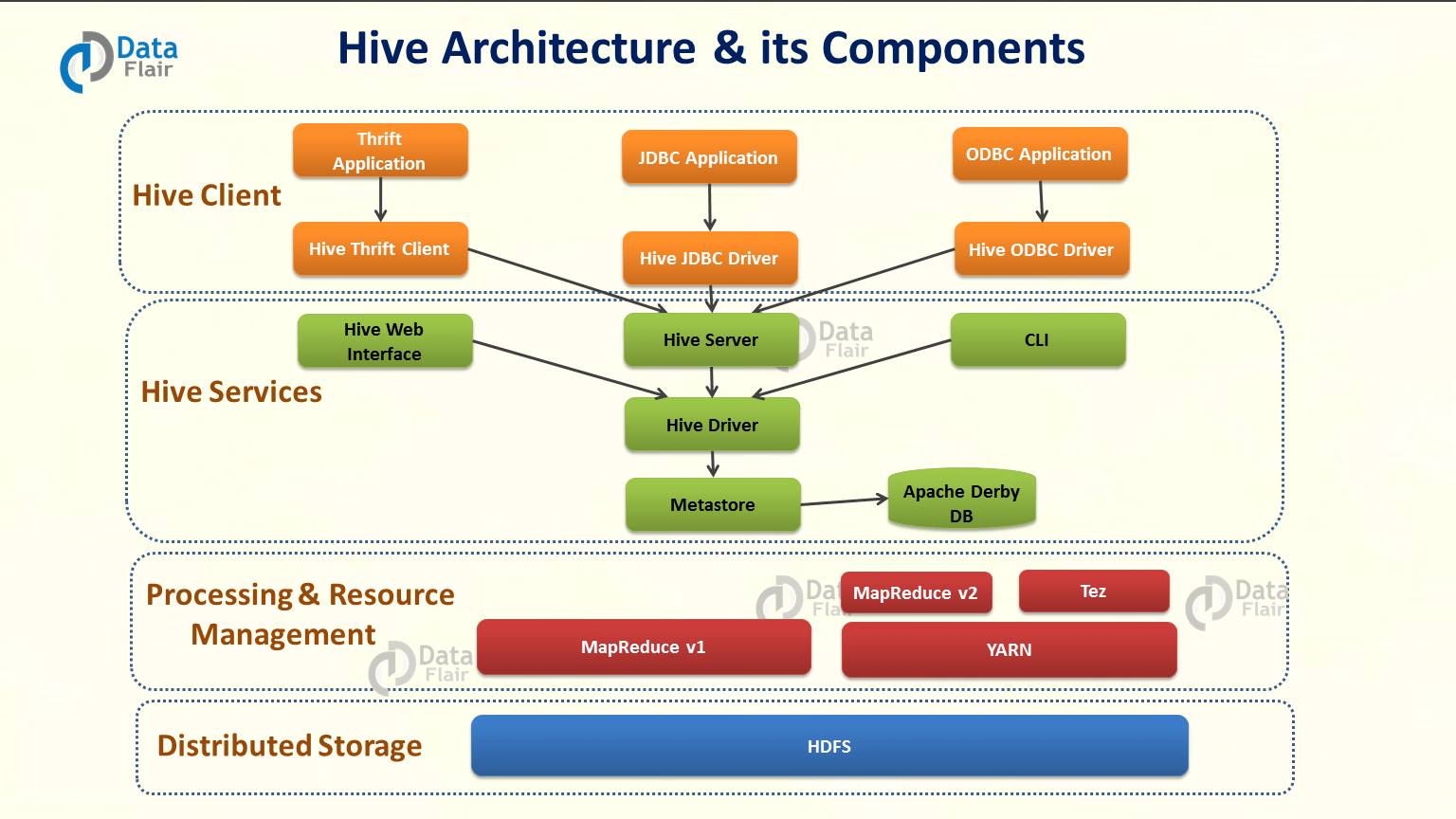
1. Explain Hive Architecture in Brief.

**What is Hive?**

Apache Hive is an ETL and Data warehousing tool built on top of [Hadoop](http://data-flair.training/blogs/apache-hive-tutorial/). It makes job easy for performing operations like:

* Analysis of huge datasets
* Ad-hoc queries
* Data Encapsulation

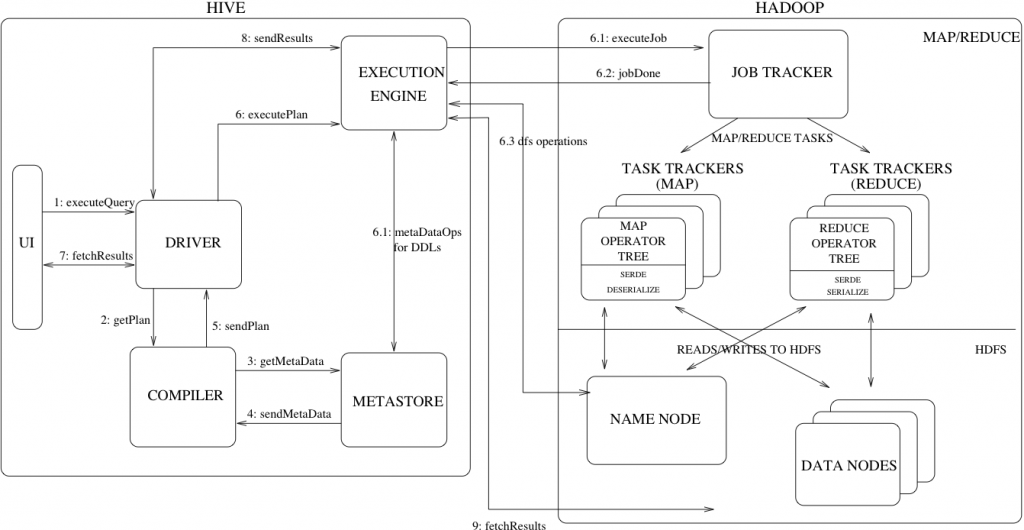
The below diagram describes the Architecture of Hive and Hive components. It also describes the flow in which a query is submitted into Hive and finally processed using the [**MapReduce**](http://data-flair.training/blogs/hadoop-mapreduce-introduction-tutorial-comprehensive-guide/) framework:



This represents when user comes with CLI(Hive Terminal) it directly connected to Hive Drivers,When User comes with JDBC/ODBC(JDBC Program) at that time by using API(Thrift Server) it connected to Hive driver and when the user comes with Web GUI(Ambari server) it directly connected to Hive Driver.

The hive driver receives the tasks(Queries) from user and send to Hadoop architecture.The Hadoop architecture uses name node,data node,job tracker and task tracker for receiving and dividing the work what Hive sends to Hadoop ([**Mapreduce Architecture**](http://www.hadooptpoint.com/hadoop-mapreduce/)) .

The below diagram represents clear internal **Hadoop Hive Architecture :**



The above diagram shows how a typical query flows through the system

**Step 1 :-**The UI calls the execute interface to the Driver

**Step 2 :-**The Driver creates a session handle for the query and sends the query to the compiler to generate an execution plan

**Step 3&4 :-**The compiler needs the metadata so send a request for getMetaData and receives the sendMetaData request from MetaStore.

**Step 5 :-**This metadata is used to typecheck the expressions in the query tree as well as to prune partitions based on query predicates. The plan generated by the compiler  is a DAG of stages with each stage being either a map/reduce job, a metadata operation or an operation on HDFS. For map/reduce stages, the plan contains map operator trees (operator trees that are executed on the mappers) and a reduce operator tree (for operations that need reducers).

**Step 6 :-** The execution engine submits these stages to appropriate components (steps 6, 6.1, 6.2 and 6.3). In each task (mapper/reducer) the deserializer associated with the table or intermediate outputs is used to read the rows from HDFS files and these are passed through the associated operator tree.Once the output generate  it is written to a temporary HDFS file though the serializer. The temporary files are used to provide the to subsequent map/reduce stages of the plan.For DML operations the final temporary file is moved to the table’s location

**Step 7&8&9 :-**For queries, the contents of the temporary file are read by the execution engine directly from HDFS as part of the fetch call from the Driver.

1. Explain Hive Components in Brief.

The major components of Apache Hive-

* **Hive Clients** – Apache Hive supports all application written in languages like C++, Java, Python etc. using JDBC, Thrift and ODBC drivers. Thus, one can easily write Hive client application written in a language of their choice.
* **Hive Services** – Hive provides various services like web Interface, CLI etc. to perform queries.
* **Processing framework and Resource Management** – Hive internally uses Hadoop MapReduce framework to execute the queries.
* **Distributed Storage** – As seen above that Hive is built on the top of Hadoop, so it uses the underlying HDFS for the distributed storage.

Now,let us see them in detail.

**Hive Clients:**

The Hive supports different types of client applications for performing queries. These clients are categorized into 3 types:

* Thrift Clients – As Apache Hive server is based on Thrift, so it can serve the request from all those languages that support Thrift.
* JDBC Clients – Apache Hive allows Java applications to connect to it using JDBC driver. It is defined in the class apache.hadoop.hive.jdbc.HiveDriver.
* ODBC Clients – ODBC Driver allows applications that support ODBC protocol to connect to Hive. For example JDBC driver, ODBC uses Thrift to communicate with the Hive server.

### Hive Services:

Apache Hive provides various services as shown in above diagram. Now, let us look at each in detail:

**a) CLI(Command Line Interface) –** This is the default shell that Hive provides, in which you can execute your Hive queries and command directly.

**b) Web Interface –** Hive also provides web based GUI for executing Hive queries and commands.

**c)** **Hive Server** – It is built on Apache Thrift and thus is also called as Thrift server. It allows different clients to submit requests to Hive and retrieve the final result.

**d) Hive Driver** – Driver is responsible for receiving the queries submitted Thrift, JDBC, ODBC, CLI, Web UL interface by a Hive client.

* **Complier** –After that hive driver passes the query to the compiler. Where parsing, type checking, and semantic analysis takes place with the help of schema present in the metastore.
* **Optimizer** – It generates the optimized logical plan in the form of a DAG (Directed Acyclic Graph) of MapReduce and HDFS tasks.
* **Executor** – Once compilation and optimization complete, execution engine executes these tasks in the order of their dependencies using Hadoop.

**e) Metastore** – [Metastore](http://data-flair.training/blogs/apache-hive-metastore/) is the central repository of Apache Hive metadata in the Hive Architecture. It stores metadata for Hive tables (like their schema and location) and partitions in a relational database. It provides client access to this information by using metastore service API. Hive metastore consists of two fundamental units:

* A service that provides metastore access to other Apache Hive services.
* Disk storage for the Hive metadata which is separate from [HDFS](http://data-flair.training/blogs/comprehensive-hdfs-guide-introduction-architecture-data-read-write-tutorial/) storage.