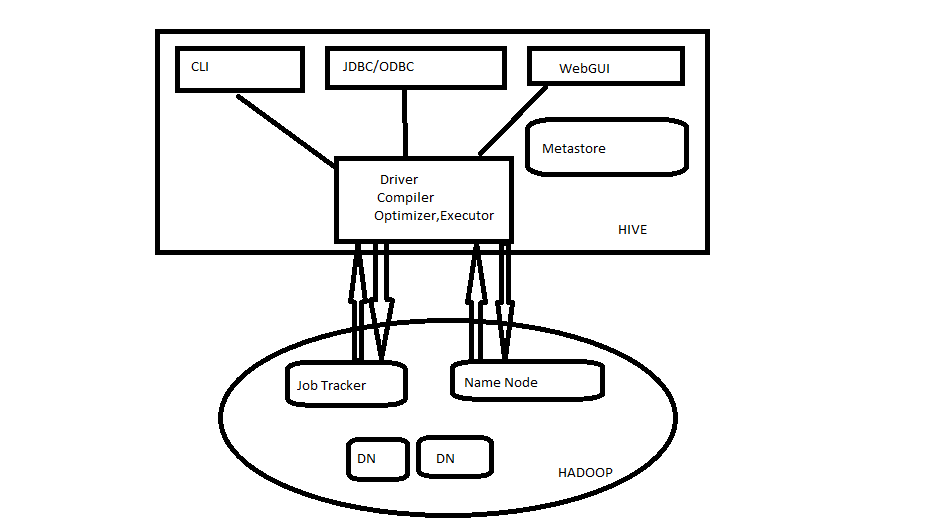
**HIVE Architecture:**

Hive is a Data warehouse solution built on top of Hadoop. It provides an SQL dialect called as Hive Query Language for querying data stored in the Hadoop Distributed File system (HDFS).

Hive stores all the meta information about database and table in metadata DB locally to Hive.



The above diagram shows the basic **Hadoop Hive architecture**. Primarily The diagram represents CLI (Command Line Interface),JDBC/ODBC and Web GUI (Web Graphical User Interface ).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/" \t "_blank)**) .

**Major Components of Hive:**

**UI :-**UI means User Interface, The user interface for users to submit queries and other operations to the system.

**Driver :-**The Driver is used for receives the quires from UI .This component implements the notion of session handles and provides execute and fetch APIs modeled on JDBC/ODBC interfaces.

**Compiler :-**The component that parses the query, does semantic analysis on the different query blocks and query expressions and eventually generates an execution plan with the help of the table and partition metadata looked up from the metastore.

**MetaStore :-**The component that stores all the structure information of the various tables and partitions in the warehouse including column and column type information, the serializers and deserializers necessary to read and write data and the corresponding HDFS files where the data is stored.

**Execution Engine :-**The component which executes the execution plan created by the compiler. The plan is a DAG of stages. The execution engine manages the dependencies between these different stages of the plan and executes these stages on the appropriate system components.