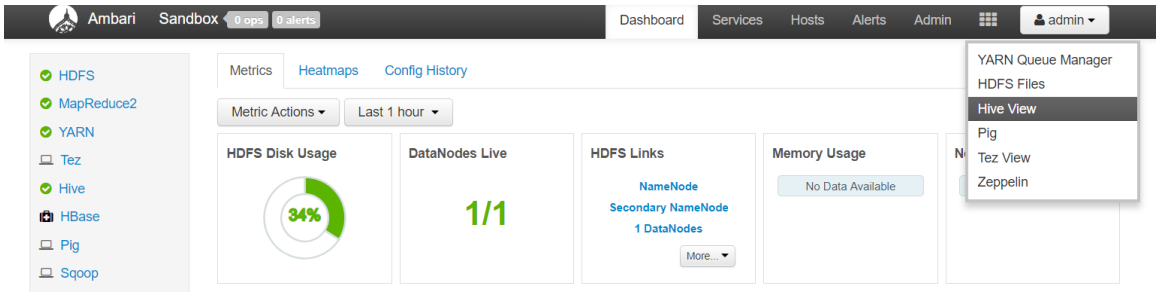
APACHE HIVE Tutorial

Prepared by: MOBEEN AHMED

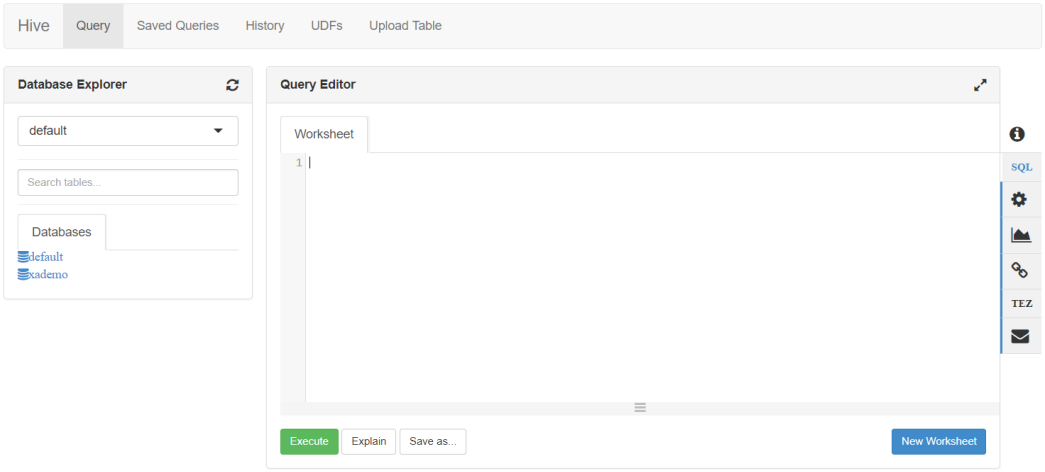
Hive

Hive is a platform used to develop SQL type scripts to do MapReduce operations. Hive gives SQL like interface to query data stored in various files systems and databases integrated with Hadoop. Initially hive was developed by Facebook to provide SQL access to Hadoop data for analysis. Apache Hive is the data warehouse on the top of Hadoop, which enables ad-hoc analysis over structured and semi-structured data.

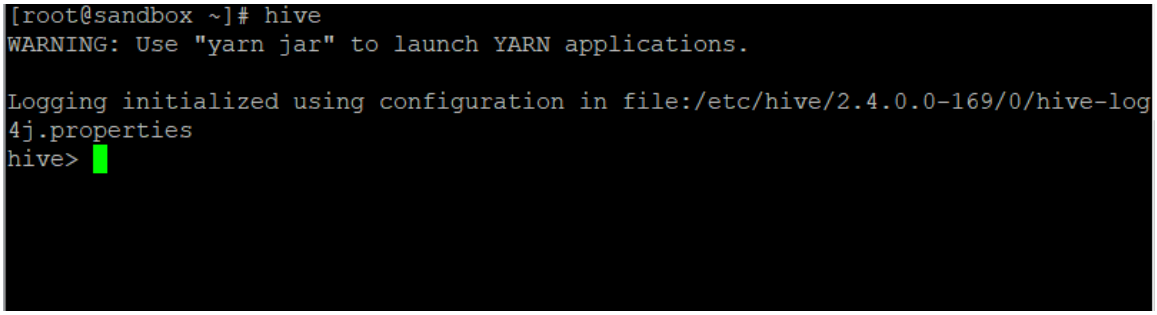
You can access the Hive in Ambari by clicking on this icon and click on Hive View.



By clicking 'Hive View', you will be redirected to hive view page.



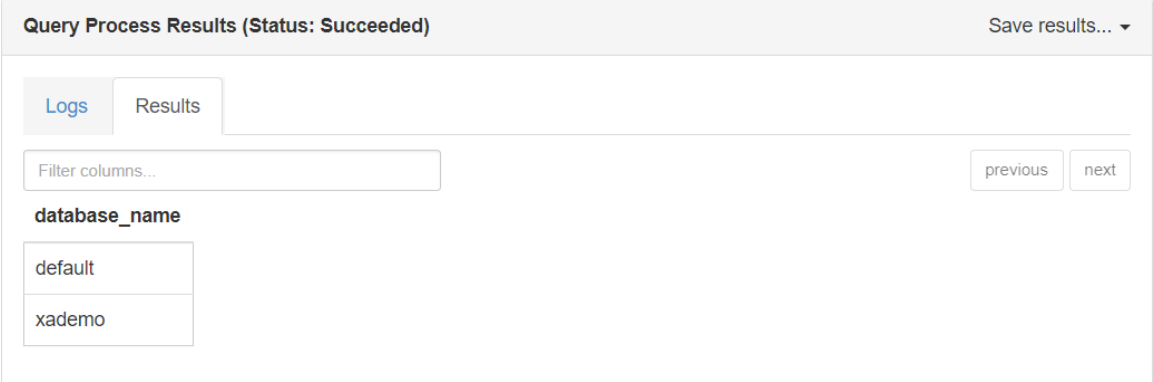
Here you can write your queries and press "Execute" to run the query. Or you can access the using shell using putty. Login to the machine and type hive and press enter. You can use either method to access hive.



The first query we are going to write is

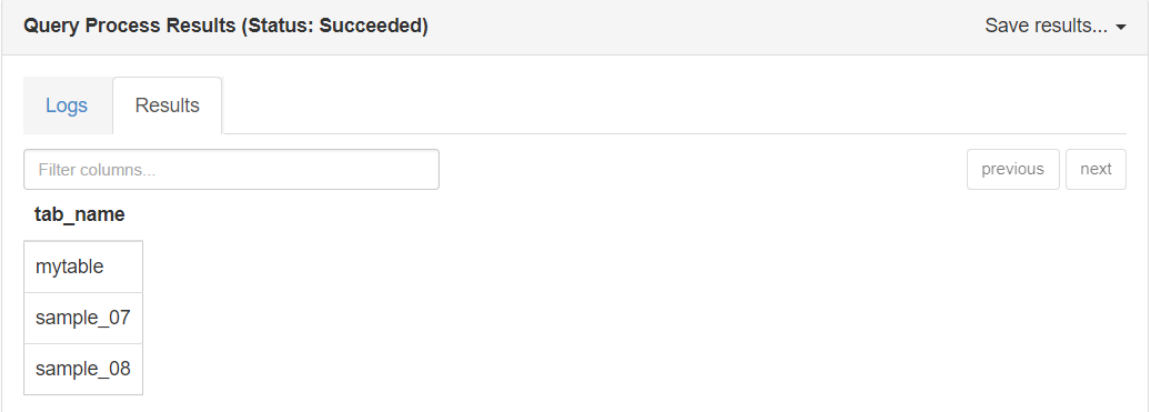
Show databases:

> *show databases;*



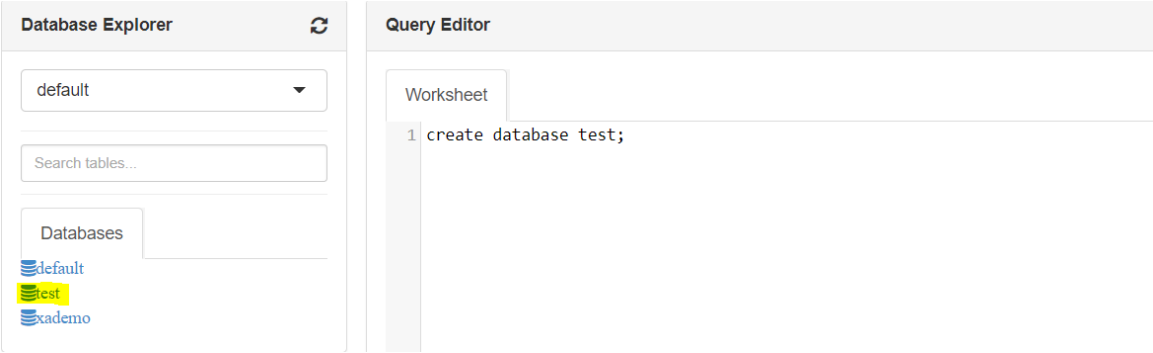
Show tables:

> *show tables;*



Create Database:

> *create database test;*



Note: Please note that you have to press refresh

Delete Database:

> drop database test;

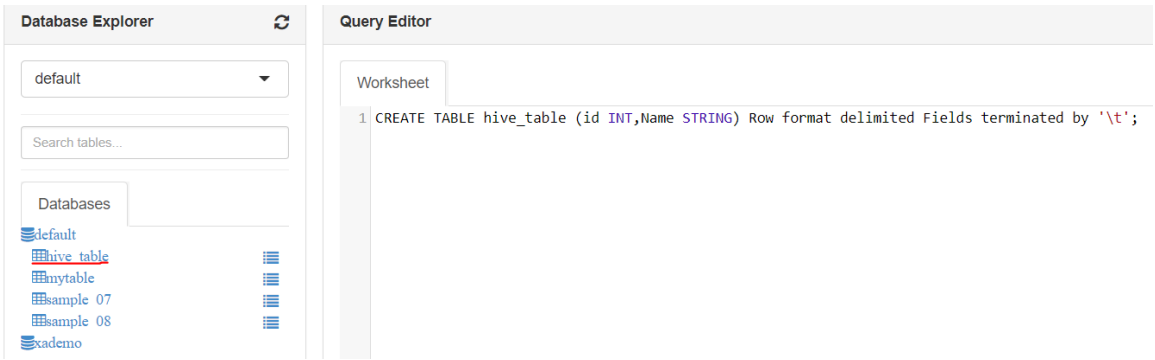
Create Table:

There are two types of table in hive: **Managed** & **External tables**. The differences between these two tables are when you drop a managed table hive deletes both data and metadata, and when a table is external table is dropped hive only deletes its metadata.

1- Create Managed Table:

> CREATE TABLE hive_table (id INT,Name STRING)

> Row format delimited Fields terminated by '\t';



Load data in table:

Create a file hive_table.txt in NDFS using vi command and paste the following data.

1001 Owais

1002 Haider

1003 Fahad

1004 Abdullah

1005 Qaiser

1006 Khalid

1007 Raheel

1008 Ahmed

1009 Faisal

1010 Saad

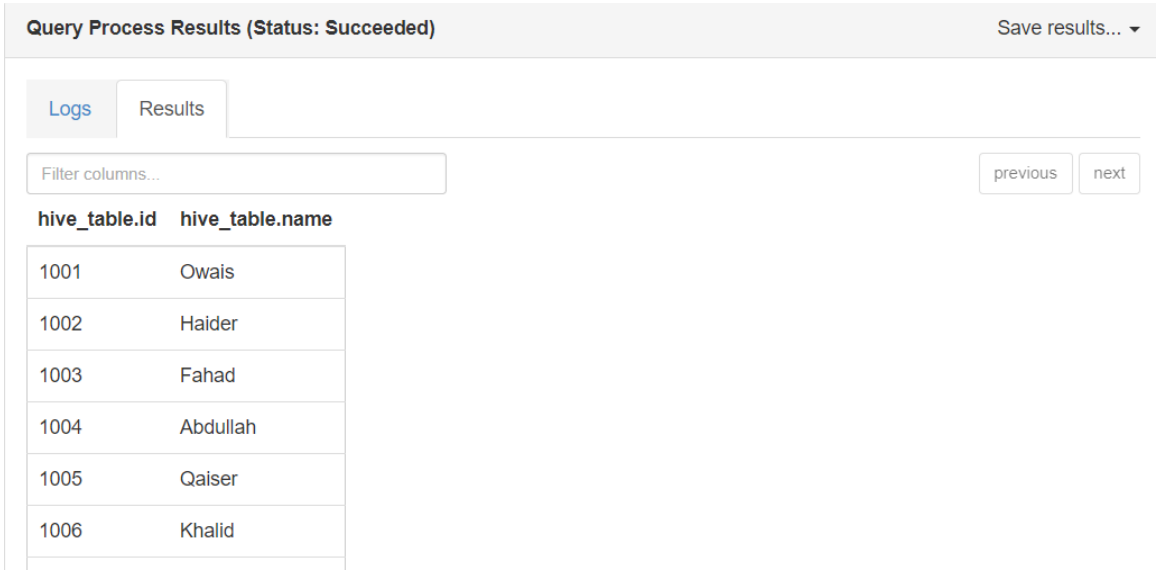
Put the file in NDFS and then move it to the folder in HDFS.

$ hadoop fs -put hive_table.txt /Class-2/hive

Now to insert data into hive table we can use

> LOAD DATA INPATH '/Class-2/hive/hive_table.txt' into table hive_table;

*> select * from hive_table;*



hive_table.id	hive_table.name
1001	Owais
1002	Haider
1003	Fahad
1004	Abdullah
1005	Qaiser
1006	Khalid

Drop Table:

> drop table hive_table;

2- Create External Table:

> CREATE EXTERNAL TABLE hive_table (id INT,Name STRING)

> Row format delimited Fields terminated by '\t'

> LOCATION '/Class-2/hive';

Again load the data into external hive table but this time we have specified a location where table data will be stored. You can use the above **load** command to load data in hive table.

Drop External Table:

> drop table hive_table;

Now check if the underlying directory delete or not?

Partition in Hive:

Hive Partitions is a way to organizes tables into partitions by dividing tables into different parts based on partition keys. It is very useful to cut data during query to reduce query times.

There are two types of partition. **Static** & **Dynamic**

Static Partition:

When loading big files into hive table static partition is preferred. Static partition is more effective in loading data. You "statically" add a partition in table and move the file into the partition of the table. If you want to use the static partition in the hive you should set property:
> *set hive.mapred.mode = strict*

You can find this property in hive-site.xml.

Dynamic Partition:

In dynamic partition we load the data from the non-partitioned table. It takes more time in loading data as compared to static partition. Dynamic partition is more suitable in loading large data stored in a table. To set dynamic

> *set hive.exec.dynamic.partition.mode=nonstrict*

To do dynamic partition we will create an external table.

> create table All_Pakistan (RollNo INT, Name string,Province string)

> row format delimited fields terminated by ',';

Now create a new file "table_partition.txt" just like above and copy the data in file.

1001,Owais,Punjab

1002,Haider,Blouchistan

1003,Fahad,KPK

1004,Abdullah,Punjab

1005,Qaiser,Sindh

1006,Khalid,KPK

1007,Raheel,Blouchistan

1008,Ahmed,KPK

1009,Faisal,Sindh

1010,Saad,Punjab

Now load the data in hive table and perform the following query to see all records.

> *select * from hive_table;*

all_pakistan.rollno	all_pakistan.name	all_pakistan.province
1001	Owais	Punjab
1002	Haider	Blouchistan
1003	Fahad	KPK
1004	Abdullah	Punjab
1005	Qaiser	Sindh
1006	Khalid	KPK
1007	Raheel	Blouchistan
1008	Ahmed	KPK

Order by:

Select * from All_Pakistan order by name;

all_pakistan.rollno	all_pakistan.name	all_pakistan.province
1004	Abdullah	Punjab
1008	Ahmed	KPK
1003	Fahad	KPK
1009	Faisal	Sindh
1002	Haider	Blouchistan
1006	Khalid	KPK
1001	Owais	Punjab
1005	Qaiser	Sindh
1007	Raheel	Blouchistan
1010	Saad	Punjab

Group By:

> select province,count(*) from All_Pakistan group by province;

province	_c1
Blouchistan	2
KPK	3
Punjab	3
Sindh	2

Create Partitioned Table:

To create data partitioning in Hive we will use following command.

> create table Pak_part(RollNo INT, Name string) PARTITIONED BY(Province string);

This query will create new table Pak_part with partition on "Province". To set the property for dynamic partition following command will be used.

> set hive.exec.dynamic.partition.mode=nonstrict

Now we are going to insert the data into partitioned table from "All_Pakistan" table which we have created above.

> INSERT OVERWRITE TABLE Pak_part PARTITION(Province) SELECT RollNo,Name,Province from All_Pakistan;

And now when we take a look at HDFS folder we will see partitions of data on the base of provinces.

Name	Size	Last Modified	Owner	Group
..				
.hive-staging_hive_2018-09-07_20-00-20_164_8426260130706171473-1	-	2018-09-08 01:00	hive	hdfs
province=Blouchistan	-	2018-09-08 01:00	hive	hdfs
province=KPK	-	2018-09-08 01:00	hive	hdfs
province=Punjab	-	2018-09-08 01:00	hive	hdfs
province=Sindh	-	2018-09-08 01:00	hive	hdfs

Bucketing in Hive:

Partitioning in hive offers a way of segregating hive table data into multiple files/directories but partitioning will gives effective results when we have same sized partitioned and limited number of partitions. To overcome this problem hive provides another way of decomposing data into more manageable parts called bucketing. The Bucketing concept is based on Hash function, which depends on the type of the bucketing column. Records which are bucketed by the same column will always be saved in the same bucket. To populate the bucketed table, we need to set the property, so that Hive knows to create the number of buckets declared in the table definition.

> *hive.enforce.bucketing = true*

We will create a new table for bucketing by using this command:

> *create table All_New_Pakistan(RollNo INT, Name string)*

> *clustered by (RollNo) INTO 3 BUCKETS;*

Use the above dynamic partition method to insert data in "All_New_Pakistan" table.

> *INSERT OVERWRITE TABLE All_New_Pakistan SELECT RollNo,Name from All_Pakistan;*

After inserting the data open HDFS view to see the bucketed files.

all_new_pakistan

Name	Size	Last Modified	Owner	Group
..				
.hive-staging_hive_2018-09-07_20-34-23_878_5642257559924916616-1	-	2018-09-08 01:34	hive	hdfs
000000_0	0.1 kB	2018-09-08 01:34	hive	hdfs
000001_0	0.1 kB	2018-09-08 01:34	hive	hdfs
000002_0	0.1 kB	2018-09-08 01:34	hive	hdfs

Hive over XML:

To read XML data in hive lets create a students.xml file and place it in HDFS.

<student> <id>1</id> <name>Milind</name> <age>25</age> </student>

<student> <id>2</id> <name>Ramesh</name> <age>Testing</age> </student>

Create New Hive table:

Now let's create a new hive for xml data.

> create table student_xml(studinfo string) ;

After creation of table insert the data in table.

> load data inpath '/Class-2/hive/students.xml' into table student_xml;

When we query the loaded data we will see something like this

*> select * from student_xml;*

student_xml.studinfo
<student> <id>1</id> <name>Milind</name> <age>25</age> </student>
<student> <id>2</id> <name>Ramesh</name> <age>Testing</age> </student>

This is not what we were expecting this is just a raw data we placed in our file. For better proper results let's create a view over "student_xml" table by using the following command:

> create view student_xml_view as SELECT xpath_int(studinfo ,'student/id'), xpath_string(studinfo ,'student/name'), xpath_string(studinfo ,'student/age')

FROM student_xml;

Now let's query this view to see the output.

*> select * from student_xml_view;*

student_xml_view._c0	student_xml_view._c1	student_xml_view._c2
1	Milind	25
2	Ramesh	Testing

That is our required output. Basically the above code of hive is extracting the tags like id, name and age from the main student data and place in a view which we have created over the "student_xml" table.