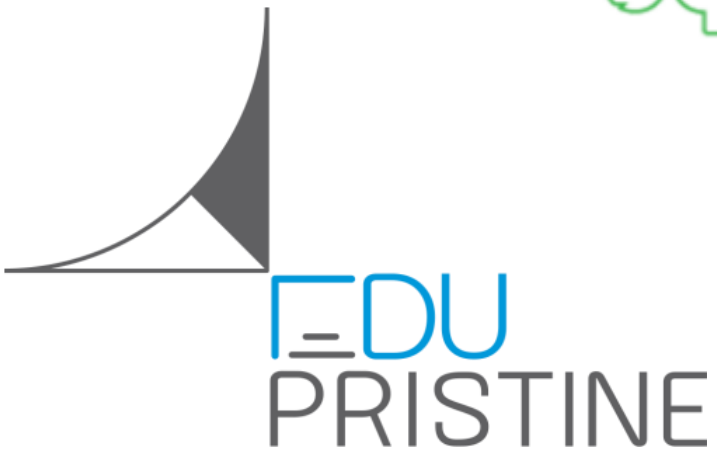

HBase

Shell commands

Data Ingestion

Integration

Type hbase in shell:

```
Commands:
Some commands take arguments. Pass no args or -h for usage.
  shell           Run the HBase shell
  hbck            Run the hbase 'fsck' tool
  snapshot        Create a new snapshot of a table
  snapshotinfo    Tool for dumping snapshot information
  wal             Write-ahead-log analyzer
  hfile           Store file analyzer
  zkcli           Run the ZooKeeper shell
  upgrade         Upgrade hbase
  master          Run an HBase HMaster node
  regionserver    Run an HBase HRegionServer node
  zookeeper       Run a Zookeeper server
  rest            Run an HBase REST server
  thrift          Run the HBase Thrift server
  thrift2         Run the HBase Thrift2 server
  clean           Run the HBase clean up script
  classpath       Dump hbase CLASSPATH
  mapredcp        Dump CLASSPATH entries required by mapreduce
  pe              Run PerformanceEvaluation
  ltt             Run LoadTestTool
  version         Print the version
  CLASSNAME       Run the class named CLASSNAME
```

You see many options!

For developer, it is shell!

Lets deep dive in to shell.

Type hbase shell

```
pankaj@ip-172-31-4-182:~$ hbase shell
17/01/06 04:16:56 INFO Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.8.0, rUnknown, Tue Jul 12 16:02:46 PDT 2016

hbase(main):001:0>
```

Type hbase help

```
COMMAND GROUPS:
  Group name: general
  Commands: status, table_help, version, whoami

  Group name: ddl
  Commands: alter, alter_async, alter_status, create, describe, disable, disable_all, drop, drop_all, enable, enable_all,
exists, get_table, is_disabled, is_enabled, list, locate_region, show_filters

  Group name: namespace
  Commands: alter_namespace, create_namespace, describe_namespace, drop_namespace, list_namespace, list_namespace_tables

  Group name: dml
  Commands: append, count, delete, deleteall, get, get_counter, get_splits, incr, put, scan, truncate, truncate_preserve

  Group name: tools
  Commands: assign, balance_switch, balancer, balancer_enabled, catalogjanitor_enabled, catalogjanitor_run, catalogjanitor
_switch, close_region, compact, compact_mob, compact_rs, flush, major_compact, major_compact_mob, merge_region, move, norm
alize, normalizer_enabled, normalizer_switch, split, trace, unassign, wal_roll, zk_dump

  Group name: replication
  Commands: add_peer, append_peer_tableCFs, disable_peer, disable_table_replication, enable_peer, enable_table_replication
, list_peers, list_replicated_tables, remove_peer, remove_peer_tableCFs, set_peer_tableCFs, show_peer_tableCFs
```

hbase general commands

hbase status

```
hbase(main):002:0> hbase status
1 active master, 0 backup masters, 3 servers, 0 dead, 21.6667 average load
```

hbase version

```
hbase(main):003:0> hbase version
1.2.0-cdh5.8.0, rUnknown, Tue Jul 12 16:02:46 PDT 2016
```

hbase table_help

```
hbase(main):004:0> hbase table_help
Help for table-reference commands.

You can either create a table via 'create' and then manipulate the table via commands like 'put', 'get', etc.
See the standard help information for how to use each of these commands.

However, as of 0.96, you can also get a reference to a table, on which you can invoke commands.
For instance, you can get create a table and keep around a reference to it via:
```

hbase whoami

```
hbase(main):007:0> hbase whoami
pankaj (auth:SIMPLE)
    groups: labusers
```

Hbase DML and DDL command line operations

Following Operations will performed:

1. Put- Data Ingestion
2. Data Retrieval:
    1. count- number of records
    2. get – Retrieve a row
    3. scan – Read entire table
    4. delete - delete a particular row
    5. put – update table
3. Drop table:
    1. disable – Notifying region servers that table will be removed
    2. drop – table will be dropped by master
4. Refer to **commandLineHbase.pdf** file for lab activities.

Sample casestudy

Consider a telecom sector is having 100 million customers. Now the challenge is how to store all possible information of a given customer.

Solution is on NoSQL:

Let us create a table with name, **telecom_username** with following column families. Fill in your name in place of username:

1. Demographics – Database table
2. Plansactive - JSON File
3. Latest3mails – TSV File
4. Paymentsinfo - Database table

*** Data created is only for practice.

Create table on HBase

hbase shell

create 'telco_username',
{NAME=>'Demographics'},{NAME=>'Plansactive'},{NAME=>'Latest3mails'},{NAME=>'Paymentsinfo'}

list 'telco.*'

```
hbase(main):001:0> create 'telco_instructor',{NAME=>'Demographics'},{NAME=>'Plansactive'},{NAME=>'Latest3mails'},{NAME=>'P
aymentsinfo'}
0 row(s) in 2.4130 seconds

=> Hbase::Table - telco_instructor
hbase(main):002:0> list 'telco.*'
TABLE
```

Before ingesting data, create tables on MySQL.

```
--login to mysql:

mysql -h 54.149.41.179 -u username -p
use database;


--create tables:
create table Demographics (custid varchar(10),age int, gender varchar(1));

create table paymentsinfo (custid varchar(10),ontime1 int,delayed1
int,total1 int);
```

```
mysql> create table Demographics (custid varchar(10),age int, gender varchar(1));
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> create table paymentsinfo (custid varchar(10),ontime1 int,delayed1 int,total1 int);
Query OK, 0 rows affected (0.01 sec)
```

Ingestion of Structured Data using Sqoop contd..

```
--Insert into demographics:
insert into Demographics (custid,age,gender) values('9065267058',46,'m');
insert into Demographics (custid,age,gender) values('9314449592',20,'f');
insert into Demographics (custid,age,gender) values('9514131750',21,'f');
insert into Demographics (custid,age,gender) values('9888944896',25,'f');
insert into Demographics (custid,age,gender) values('9566797883',36,'m');
insert into Demographics (custid,age,gender) values('9569949682',47,'f');
insert into Demographics (custid,age,gender) values('9002620762',28,'f');
insert into Demographics (custid,age,gender) values('9266241062',49,'m');
insert into Demographics (custid,age,gender) values('9058015369',20,'m');
insert into Demographics (custid,age,gender) values('9130634376',23,'m');
insert into Demographics (custid,age,gender) values('9725923789',21,'m');
```

Ingestion of Structured Data using Sqoop contd..

```
--Insert into paymentsinfo:
Insert into paymentsinfo(custid ,ontime1 ,delayed1 ,total1 ) values ('9065267058',16,43,59);
Insert into paymentsinfo(custid ,ontime1 ,delayed1 ,total1 ) values ('9314449592',24,47,71);
Insert into paymentsinfo(custid ,ontime1 ,delayed1 ,total1 ) values ('9514131750',32,19,51);
Insert into paymentsinfo(custid ,ontime1 ,delayed1 ,total1 ) values ('9888944896',11,37,48);
Insert into paymentsinfo(custid ,ontime1 ,delayed1 ,total1 ) values ('9566797883',38,38,76);
Insert into paymentsinfo(custid ,ontime1 ,delayed1 ,total1 ) values ('9569949682',18,41,59);
Insert into paymentsinfo(custid ,ontime1 ,delayed1 ,total1 ) values ('9002620762',30,20,50);
Insert into paymentsinfo(custid ,ontime1 ,delayed1 ,total1 ) values ('9266241062',23,36,59);
Insert into paymentsinfo(custid ,ontime1 ,delayed1 ,total1 ) values ('9058015369',36,26,62);
Insert into paymentsinfo(custid ,ontime1 ,delayed1 ,total1 ) values ('9130634376',19,43,62);
Insert into paymentsinfo(custid ,ontime1 ,delayed1 ,total1 ) values ('9725923789',16,38,54);
```

MYSQL <--> SQOOP <--> HBASE

Enter dbname and username, to import Demographics table using sqoop

```
sqoop import \
--connect jdbc:mysql://54.149.41.179/dbname --username username -P \
--table Demographics \
--columns "custid,age,gender" \
--hbase-table telco_username \
--column-family Demographics \
--hbase-row-key custid -m 1
```

Run the following command in HBase:

get 'telco_username', 9065267058

```
hbase(main):002:0> get 'telco_instructor', 9065267058
COLUMN                                CELL
 Demographics:age                     timestamp=1483978028908, value=46
 Demographics:gender                  timestamp=1483978028908, value=m
2 row(s) in 0.0440 seconds
```

MYSQL <--> SQOOP <--> HBASE

Enter dbname and username, to import paymentsinfo table using sqoop

```
sqoop import \
--connect jdbc:mysql://54.149.41.179/dbname --username username -P \
--table paymentsinfo \
--columns "custid,ontime1,delayed1,total1" \
--hbase-table telco_username \
--column-family Paymentsinfo \
--hbase-row-key custid -m 1
```

Run the following command in HBase:

get 'telco_username', 9725923789

```
hbase(main):003:0> get 'telco_instructor', 9725923789
COLUMN                                CELL
 Demographics:age                     timestamp=1483978028908, value=21
 Demographics:gender                  timestamp=1483978028908, value=m
 Paymentsinfo:delayed1                timestamp=1483981728892, value=38
 Paymentsinfo:ontime1                 timestamp=1483981728892, value=16
 Paymentsinfo:total1                  timestamp=1483981728892, value=54
5 row(s) in 0.0080 seconds
```

Ingestion of Raw CSV file using JAVA API

Using winscp, copy email.csv file to cluster

Create a directory on HDFS:
hadoop fs -mkdir hbasepractice

hadoop fs -put email.csv hbasepractice

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=, -Dimporttsv.columns=HBASE_ROW_KEY,Latest3mails:mail3,Latest3mails:mail2,Latest3mails:mail1 telco_username hbasepractice/email.csv

Run the following command in HBase:

get 'telco_username', 9888944896

```
hbase(main):004:0> get 'telco_instructor', 9888944896
COLUMN                          CELL
 Demographics:age               timestamp=1483978028908, value=25
 Demographics:gender            timestamp=1483978028908, value=f
 Latest3mails:mail1             timestamp=1483982058013, value=
 Latest3mails:mail2             timestamp=1483982058013, value=
 Latest3mails:mail3             timestamp=1483982058013, value=xxxx xxx xx
 Paymentsinfo:delayed1          timestamp=1483981728892, value=37
 Paymentsinfo:ontime1           timestamp=1483981728892, value=11
 Paymentsinfo:total1            timestamp=1483981728892, value=48
8 row(s) in 0.0120 seconds
```

Ingestion of semistructured data using Apache Pig

Using WinSCP, copy the plans.json file to cluster.

hadoop fs -put plans.json hbasepractice

Login to Pig

```
Plansactive = LOAD 'hbasepractice/plans.json' USING
JsonLoader('custid:chararray, messageplan:chararray,
                dataplan:chararray,        callplan:chararray,
                comboplan:chararray');

STORE Plansactive INTO 'hbase://telco_username' USING
org.apache.pig.backend.hadoop.hbase.HBaseStorage(
' Plansactive:messageplan
 Plansactive:dataplan
 Plansactive:callplan
 Plansactive:comboplan'
);
```

Run the following command in HBase:

get 'telco_username', 9065267058

```
hbase(main):002:0> get 'telco_instructor', 9065267058
COLUMN                          CELL
 Demographics:age               timestamp=1483978028908, value=46
 Demographics:gender            timestamp=1483978028908, value=m
 Latest3mails:mail1             timestamp=1483982058013, value=
 Latest3mails:mail2             timestamp=1483982058013, value=xxxx xxx xx
 Latest3mails:mail3             timestamp=1483982058013, value=xxxx xxx xx
 Plansactive:callplan           timestamp=1483985217693, value=yes
 Plansactive:dataplan           timestamp=1483985217693, value=
 Plansactive:key                timestamp=1483985217693, value=yes
 Plansactive:messageplan        timestamp=1483985217693, value=
9 row(s) in 0.0550 seconds
```

Access table with Hive

Now on this table, access can be provided to specific information to different departments.

All helpdesk should know is, your demographics information. Lets enable it:

```
create database hbase_hive_username;
use hbase_hive_username;

create external table if not exists helpdesk(custid string, age int,
gender string)
stored by'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES("hbase.columns.mapping" = ":key, Demographics:age,
Demographics:gender")
TBLPROPERTIES("hbase.table.name"="telco_username");

select count(*) from helpdesk where age>30;

select * from helpdesk where custid= 9065267058;
```

Access table with Hive – Marketing Department

```
create external table if not exists marketing (custid int, age int,
gender string,
messageplan string, dataplan string, callplan string, comboplan string)
stored by'org.apache.hadoop.hive.hbase.HBaseStorageHandler‘
 WITH SERDEPROPERTIES("hbase.columns.mapping" = ":key,
Demographics:age,
Demographics:gender,
Plansactive:messageplan,
Plansactive:dataplan,
Plansactive:callplan,
Plansactive:comboplan ")
TBLPROPERTIES("hbase.table.name"="telco_username");

Select * from marketing limit 10;
-- Number of customer who hold messageplan

Select count(*) from marketing where messageplan is not null;
```

Conclusion

In this lab, you have learnt the following:

1. HBase shell commands
2. Sqoop <--> HBase integration
3. Pig <--> HBase integration
4. Hive <--> HBase integration
5. Raw file load to HBase