Cloudera Development Cluster Test Report

Author: Fahad Khan

Designation: Hadoop/BigData Consultant

Date:

Table of Contents

[Introduction 4](#_Toc3217359)

[HDFS Tests 4](#_Toc3217360)

[1.1 HDFS access and Kerberos authentication 4](#_Toc3217361)

[1.1.1 HDFS access without Kerberos 4](#_Toc3217362)

[1.1.2 HDFS access with Kerberos 4](#_Toc3217363)

[1.2 Namenode HA 5](#_Toc3217364)

[1.3 HDFS data at rest encryption 6](#_Toc3217365)

[1.3.1 KMS and KTS Services 6](#_Toc3217366)

[2 YARN, MapReduce and Spark 9](#_Toc3217367)

[2.1 MapReduce Benchmark with Terasort & Teragen 9](#_Toc3217368)

[2.1.1 Teragen Command to Generate 1 TB of Data with HDFS Replication Set to 1. 9](#_Toc3217369)

[2.1.2 Teragen Command to Generate 1 TB of Data with HDFS Default Replication. 10](#_Toc3217370)

[2.1.3 Terasort Command to Sort Data with HDFS Replication Set to 1 11](#_Toc3217371)

[*2.1.4* Terasort Command to Sort Data with HDFS Default Replication 12](#_Toc3217372)

[*2.1.5* Adjusting the data volume to 3 times of cluster RAM (YARN memory in total across all nodes), and run the above 4 steps once more. 12](#_Toc3217373)

[2.2 WordCount with Spark 12](#_Toc3217374)

[3 HIVE and IMPALA Tests 13](#_Toc3217375)

[3.1 Kerberos and LDAP login for HIVE 13](#_Toc3217376)

[3.2 Kerberos and LDAP login for IMPALA 15](#_Toc3217377)

[3.3 Table Authorizations 15](#_Toc3217378)

[3.4 HIVE queries on the basic dataset 16](#_Toc3217379)

[3.5 IMPALA queries with Parquet 18](#_Toc3217380)

[3.6 IMPALA queries with Kudu 19](#_Toc3217381)

[3.7 HIVE/IMPALA queries in HUE 20](#_Toc3217382)

[4 HBase Test 21](#_Toc3217383)

[4.1 HBase access with Kerberos 21](#_Toc3217384)

[4.2 HBase access without Kerberos 21](#_Toc3217385)

[5 CM and Navigator Tests 22](#_Toc3217386)

[5.1 LDAP integration with CM 22](#_Toc3217387)

[5.2 Navigator audit. 23](#_Toc3217388)

[5.3 Navigator data lineage. 23](#_Toc3217389)

[6 Cloudera Enterprise Search (Solr) Test 24](#_Toc3217390)

[6.1 Solr access with Kerberos 24](#_Toc3217391)

[6.2 Solr access without Kerberos 24](#_Toc3217392)

[7 Informatica connectivity Test 24](#_Toc3217393)

[8 Tableau connectivity Test 24](#_Toc3217394)

# Introduction

This report contains the sanity check of Cloudera components and results of DEV/QA Cluster for MoF.

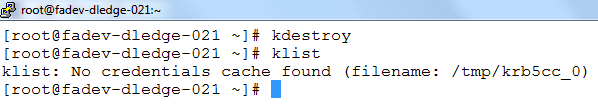
# HDFS Tests

## HDFS access and Kerberos authentication

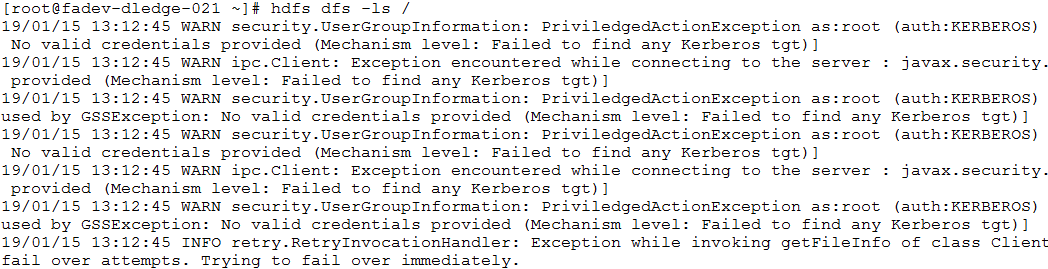
### HDFS access without Kerberos

As a comparison, this test is to validate that users cannot access HDFS without signing against their AD accounts.

1. First, make sure that the user is not signing onto Kerberos.

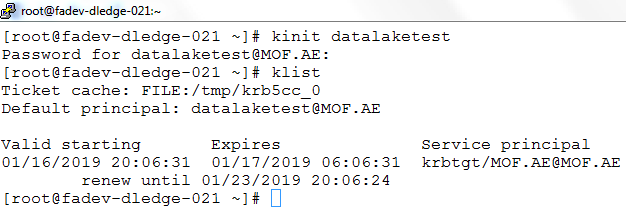


1. Try to access HDFS and expected result is user will get authentication error.

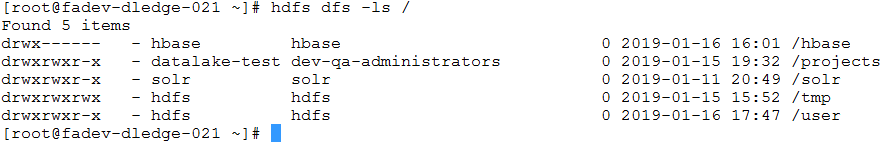


### HDFS access with Kerberos

1. Sign in onto AD user (datalaketest) via Kerberos.

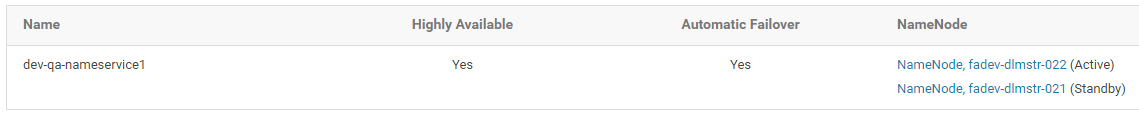


1. Access the HDFS again. You will see change in the output.

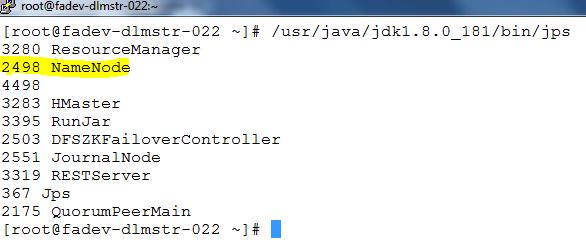


## Namenode HA

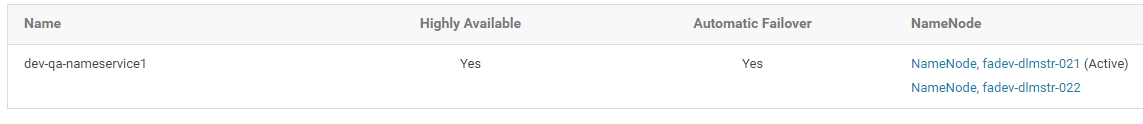
1. You can see in below snippet, HDFS is in HA mode.



1. First, check the active and standby Namenodes in CM, and the process id of the active Namenode in respective VM. As shown in the figure,
   1. The Active Namenode, at that time, is in *dlmstr-022*.
   2. The Standby Namenode is in *dlmstr-0212*.
   3. The process id of the active Namenode is 2498.



1. Use the ‘*kill -9’* command to abruptly kill the active Namenode process. Sometime later, we’ll find in CM that the active Namenode is automatically switched from *dlmstr-022* to *dlmstr-021*, and the previous Namenode in *dlmstr-022* goes down.

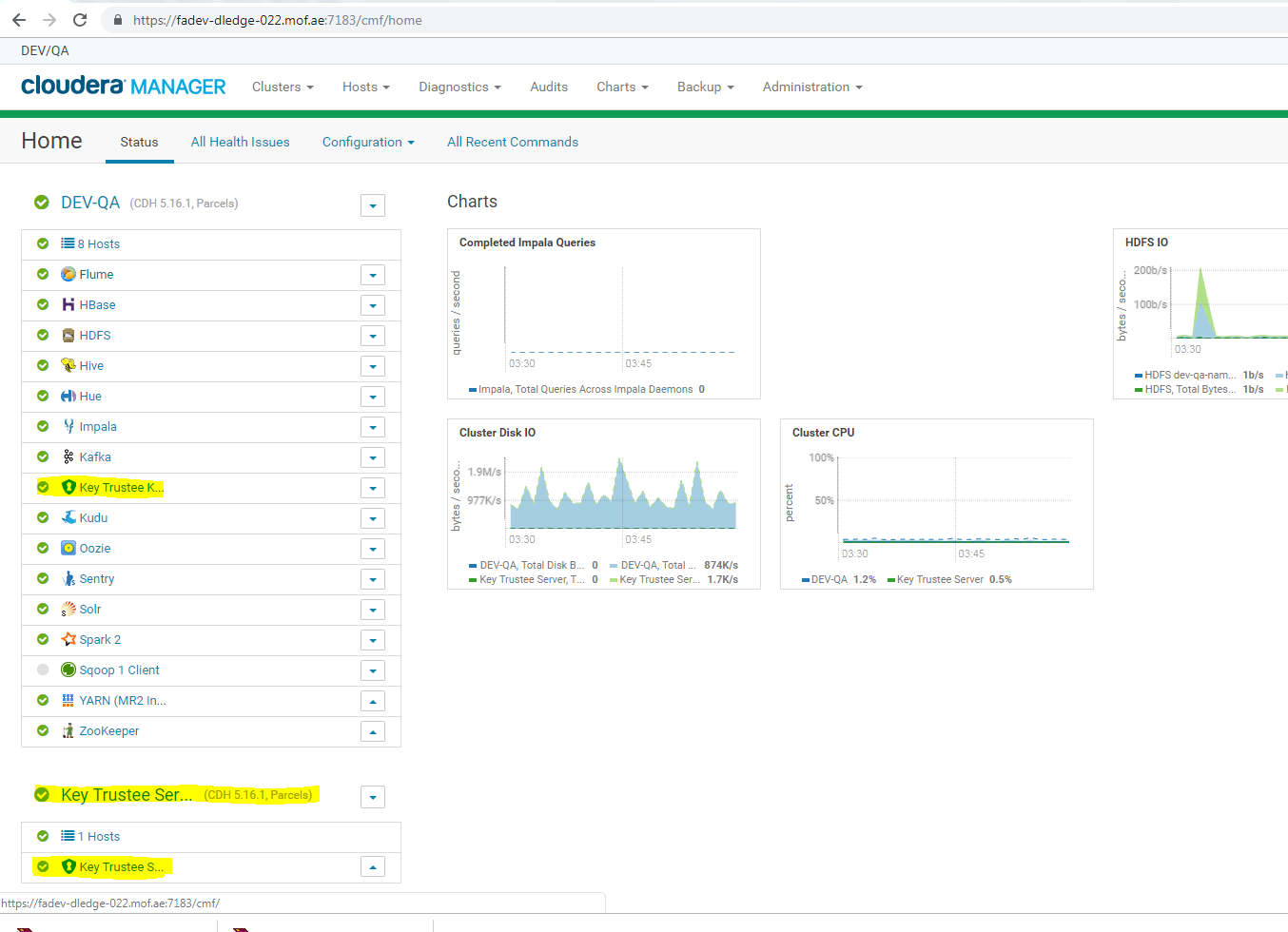


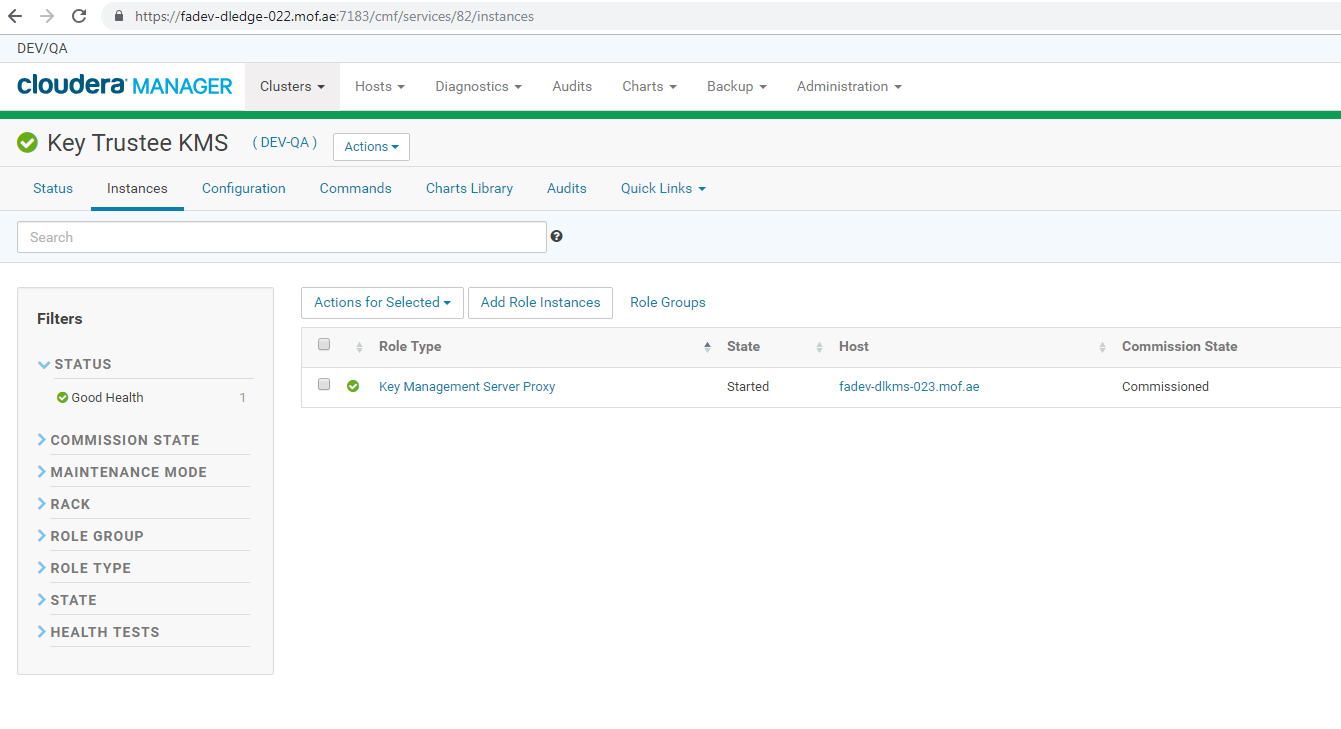
## HDFS data at rest encryption

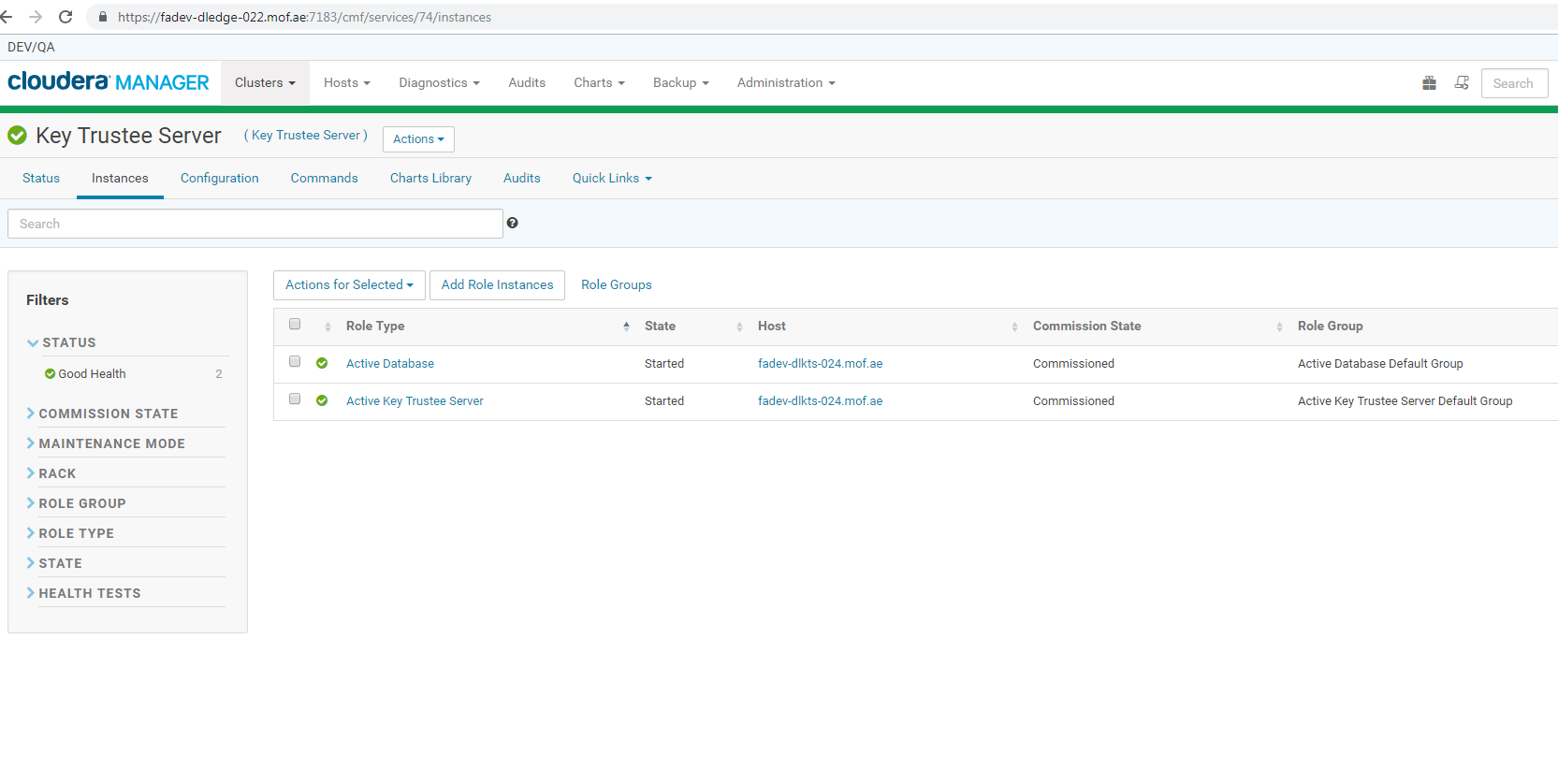
Cloudera clusters can use a combination of data at rest encryption mechanisms, including [HDFS transparent encryption](https://www.cloudera.com/documentation/enterprise/5-16-x/topics/cdh_sg_hdfs_encryption.html#xd_583c10bfdbd326ba--5a52cca-1476e7473cd--7f85) and [Cloudera Navigator Encrypt](https://www.cloudera.com/documentation/enterprise/5-16-x/topics/sg_navigator_encrypt.html#concept_navigator_encrypt). Both of these data at rest encryption mechanisms can be augmented with key management using [Cloudera Navigator Key Trustee Server](https://www.cloudera.com/documentation/enterprise/5-16-x/topics/sg_key_trustee.html#concept_key_trustee).

[Transparent encryption for HDFS](http://blog.cloudera.com/blog/2015/01/new-in-cdh-5-3-transparent-encryption-in-hdfs/) implements transparent end-to-end encryption of data read from and written to HDFS blocks across cluster.

## 1.3.1 KMS and KTS Services

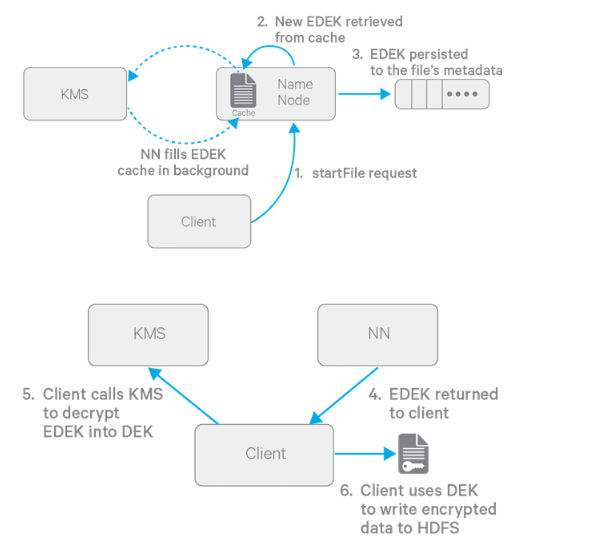






HDFS Transparent Encryption:

HDFS transparent encryption introduces the concept of an encryption zone (EZ), which is a directory in HDFS whose contents will be automatically encrypted on write and decrypted on read. Each encryption zone is associated with a key (EZ Key) specified by the key administrator when the zone is created. Below is the key decrypt, data access mechanism



Sample test for HDFS Transparent Encryption and Data access in encryption zone:

***-- create encryption key***

*[hadmin@fadev-dledge-022 ~]$ hadoop key create <encryption\_key\_name>*

*[hadmin@fadev-dledge-022 ~]$ hadoop key create encrpttest\_key*

*encrpttest\_key has been successfully created with options Options{cipher='AES/CTR/NoPadding', bitLength=128, description='null', attributes=null}.*

*org.apache.hadoop.crypto.key.kms.LoadBalancingKMSClientProvider@6ea20f02 has been updated.*

***-- Create a new empty directory and make it an encryption zone***

*[hadmin@fadev-dledge-022 ~]$ hdfs crypto -createZone -keyName <encryption\_key\_name> -path <encryption\_directory(hdfs) >*

*[hadmin@fadev-dledge-022 ~]$ hdfs crypto -createZone -keyName encrpttest\_key -path /user/encrpttest*

*Added encryption zone /user/encrpttest*

*[hadmin@fadev-dledge-022 ~]$*

*[hadmin@fadev-dledge-022 ~]$ cat test.txt*

*hsadkjasdkjsa*

*sakdjasjd;a*

*sa;kjd;sj*

***--Existing data can be encrypted by coping it copied into the new encryption zones or you can add files to an encryption zone by copying them over to the encryption zone***

*[hadmin@fadev-dledge-022 ~]$ hdfs dfs -put ./test.txt /user/encrpttest*

*[hadmin@fadev-dledge-022 ~]$*

*[hadmin@fadev-dledge-022 ~]$*

*[hadmin@fadev-dledge-022 ~]$ hdfs dfs -cat /user/encrpttest/test.txt*

*hsadkjasdkjsa*

*sakdjasjd;a*

*sa;kjd;sj*

*[hadmin@fadev-dledge-022 ~]$*

*[hadmin@fadev-dledge-022 ~]$ hdfs dfs -cat /user/encrpttest/test.txt*

*hsadkjasdkjsa*

*sakdjasjd;a*

*sa;kjd;sj*

***--Switch to the user who is not having access to this encryption key and test the data access***

*[hadmin@fadev-dledge-022 ~]$ kinit dbabu*

*Password for dbabu@MOF.AE:*

*[hadmin@fadev-dledge-022 ~]$ klist*

*Ticket cache: FILE:/tmp/krb5cc\_1001*

*Default principal: dbabu@MOF.AE*

*Valid starting Expires Service principal*

*03/07/2019 16:10:02 03/08/2019 02:10:02 krbtgt/MOF.AE@MOF.AE*

*renew until 03/14/2019 16:09:52*

*[hadmin@fadev-dledge-022 ~]$ hdfs dfs -cat /user/encrpttest/test.txt*

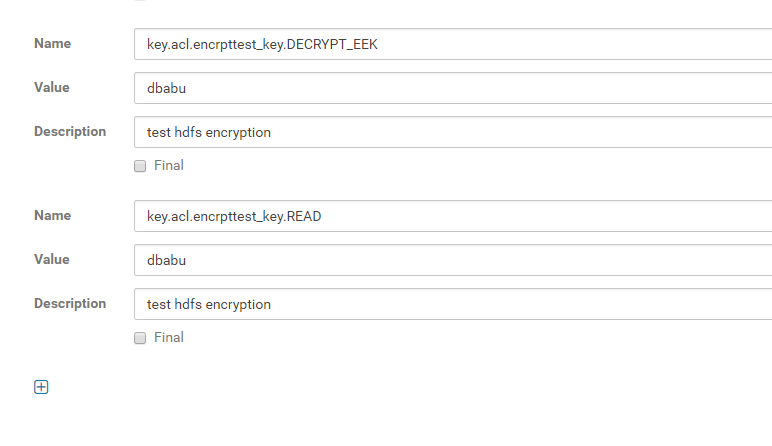
*cat:* ***User [dbabu] is not authorized to perform [DECRYPT\_EEK] on key with ACL name [encrpttest\_key]!!***

*[hadmin@fadev-dledge-022 ~]$*

***--Set Encryption key access to specific users or groups through KMS service configuration***

*Cloudera Manager -> KMS -> Configuration*

*Add below fields with the newly created encryption key and users/groups*



*Note:*

*Value field in above fields should be having comma separated list of users followed by (space) followed by comma separated list of groups. If you have only groups then mention space at the start of the value and then followed by groups.*

***--(After updating KMS config with providing key access to this user, test the encrypted data access***

*[hadmin@fadev-dledge-022 ~]$ hdfs dfs -cat /user/encrpttest/test.txt*

*hsadkjasdkjsa*

*sakdjasjd;a*

*sa;kjd;sj*

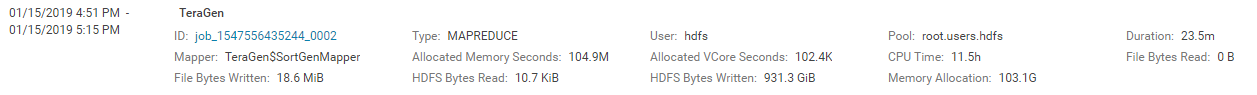
# YARN, MapReduce and Spark

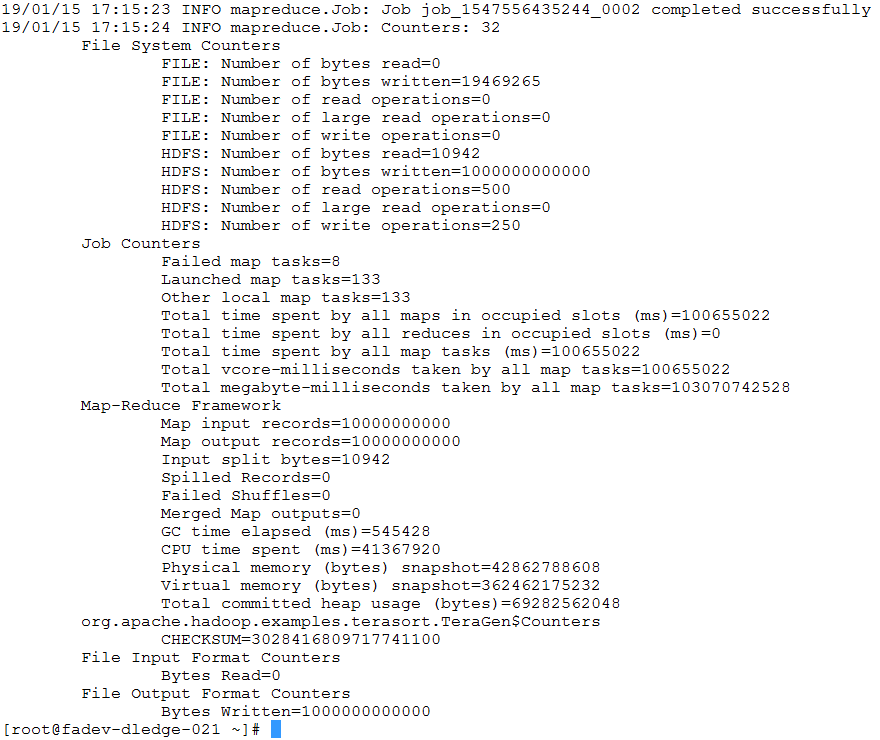
## MapReduce Benchmark with Terasort & Teragen

### Teragen Command to Generate 1 TB of Data with HDFS Replication Set to 1.

**yarn jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar teragen -Ddfs.replication=1 -Dmapreduce.job.maps=125 10000000000**

**/user/datalake-test/TS\_input1**



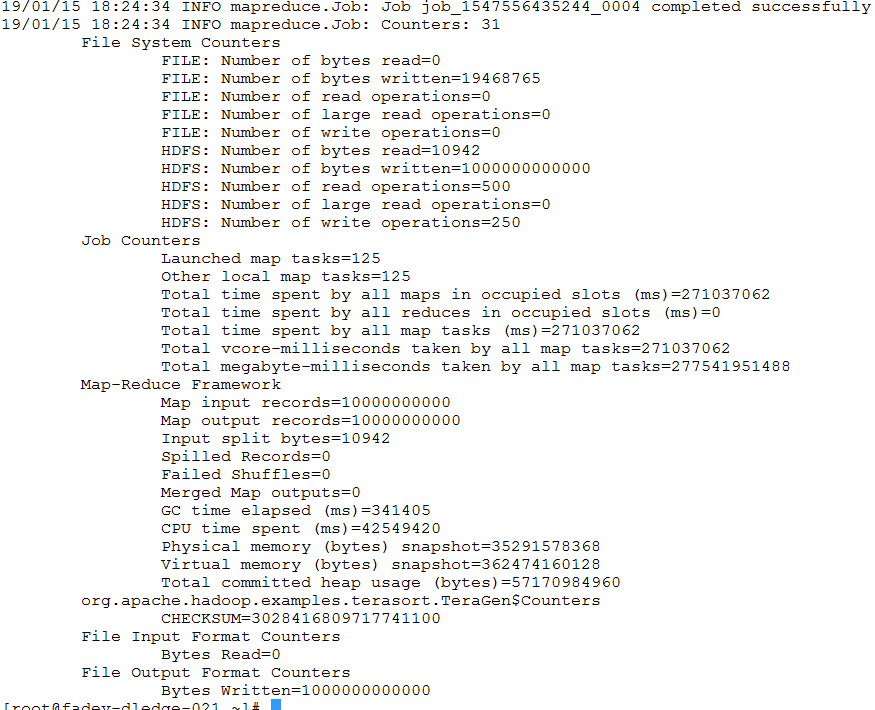


The above command generates 1 TB of data with an HDFS replication factor of 1, using 125 mappers. This command would be appropriate for a cluster with 125 disks.

### Teragen Command to Generate 1 TB of Data with HDFS Default Replication.

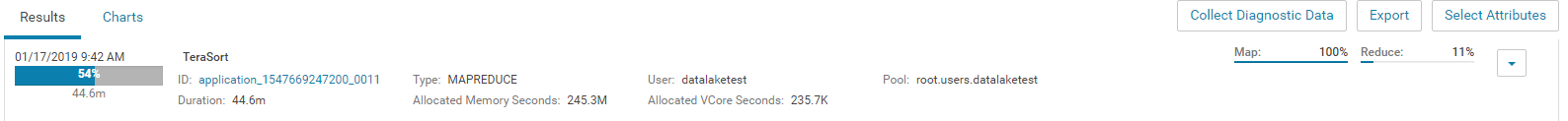
***yarn jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar teragen -Dmapreduce.job.maps=125 10000000000 /user/datalake-test/TS\_input2***





The above command generates 1 TB of data with the default HDFS replication factor (usually 3), using 125 mappers. This command would be appropriate for a cluster with 125 disks.

### Terasort Command to Sort Data with HDFS Replication Set to 1

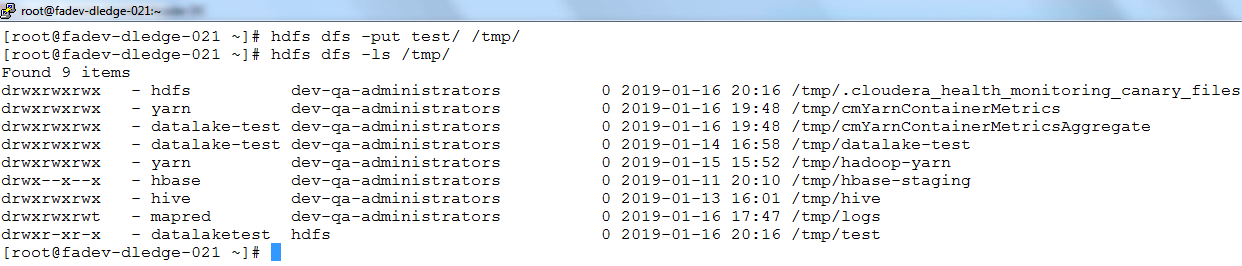


### Terasort Command to Sort Data with HDFS Default Replication

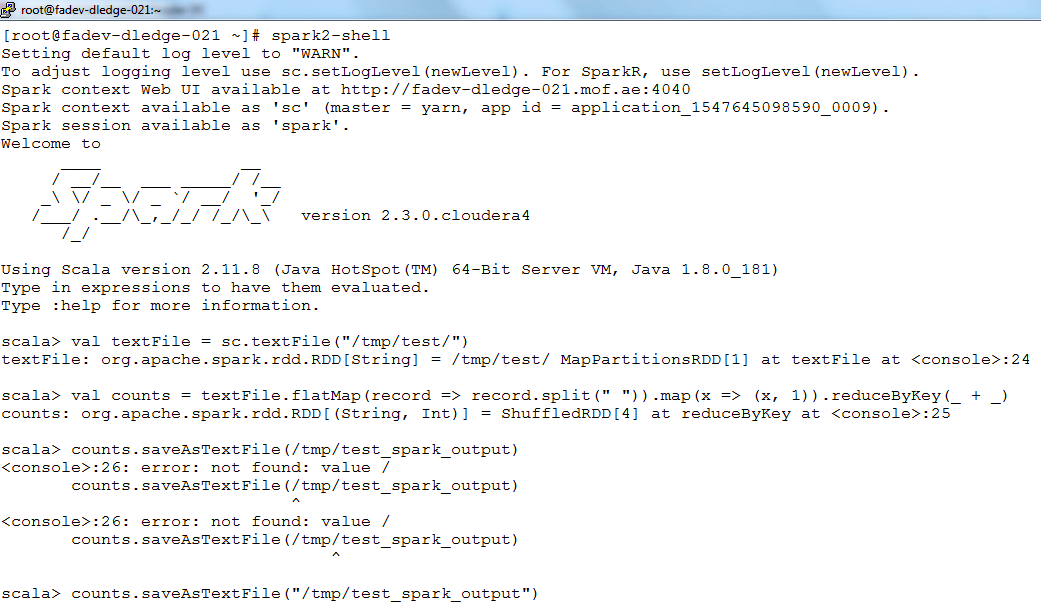
### Adjusting the data volume to 3 times of cluster RAM (YARN memory in total across all nodes), and run the above 4 steps once more.

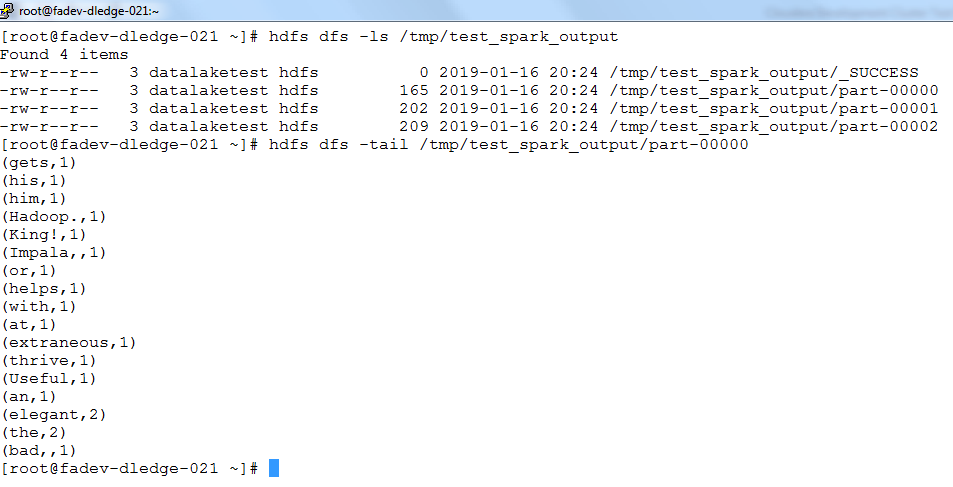
## WordCount with Spark

1. Sign in with Kerberos via *datalaketest*.
2. Run ***'hdfs dfs -put test /tmp'*** to upload some text files onto HDFS.

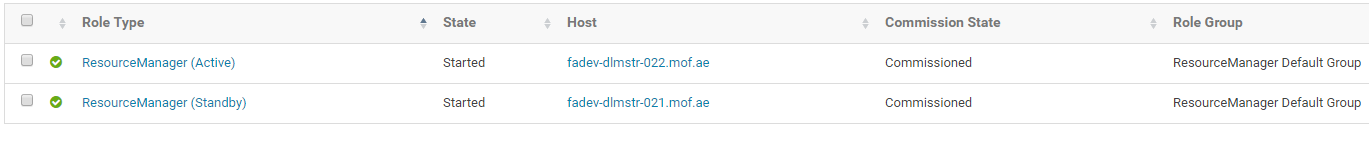


1. Use ***‘spark-shell’*** to enter Spark command line shell, and input the following Spark scripts.





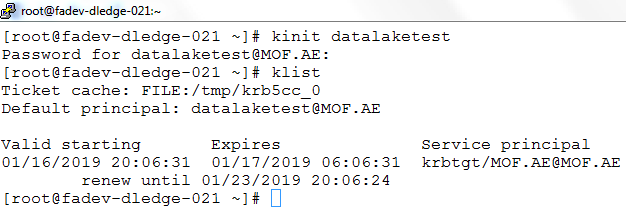
2.3 Resource Manager HA



# HIVE and IMPALA Tests

## Kerberos and LDAP login for HIVE

1. Sign in onto AD user (*datalaketest*) via Kerberos.

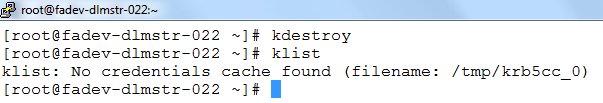


1. Use the following command to login to hive shell via JDBC connection string.

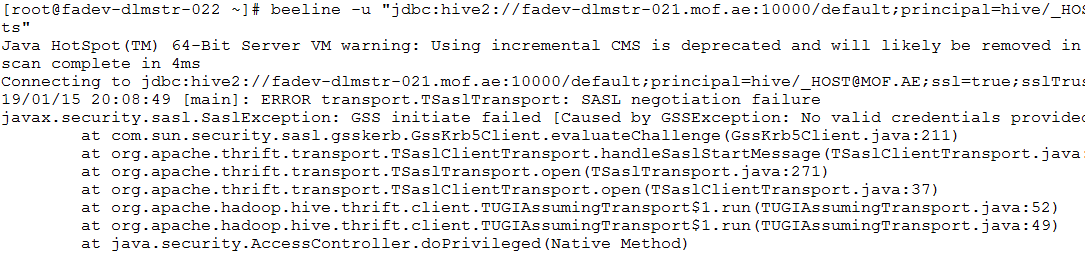
***beeline -u "jdbc:hive2://fadev-dlmstr-021.mof.ae:10000/default;principal=hive/\_HOST@MOF.AE;ssl=true;sslTrustStore=/usr/java/jdk1.8.0\_181/jre/lib/security/jssecacerts"***



1. Use ‘***kdestroy***’ to sign off from Kerberos, and use klist to make sure of it.

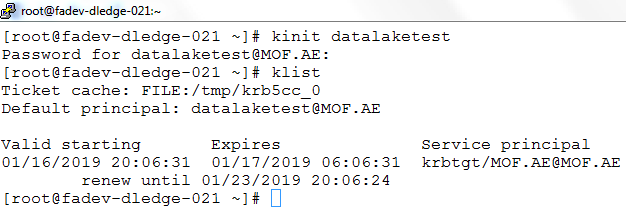


1. Use the following command to enter hive shell via LDAP username and password.



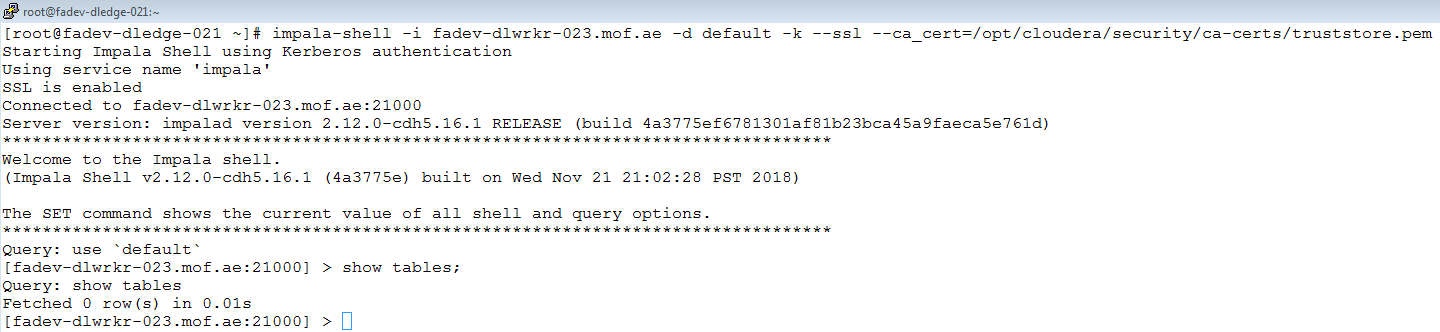
## Kerberos and LDAP login for IMPALA

1. Sign in onto AD user (*datalaketest*) via Kerberos.

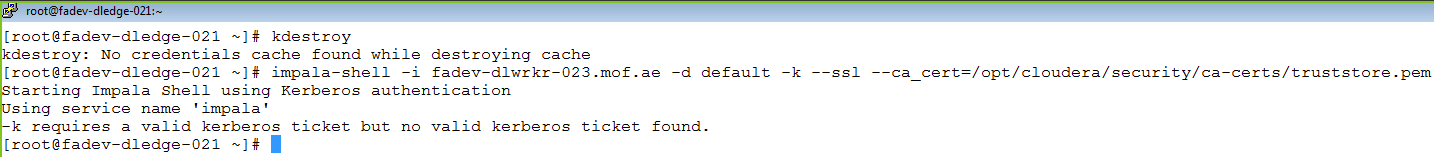


1. Login to ***impala-shell*** using below Impala connection string:

***impala-shell -i fadev-dlwrkr-023.mof.ae -d default -k --ssl --ca\_cert=/opt/cloudera/security/ca-certs/truststore.pem***

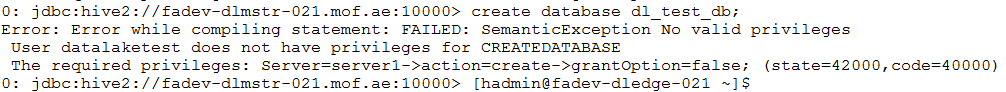


1. Use ‘***kdestroy’*** to sign off from Kerberos, and/or make sure Kerberos is not signed in.



## Table Authorizations

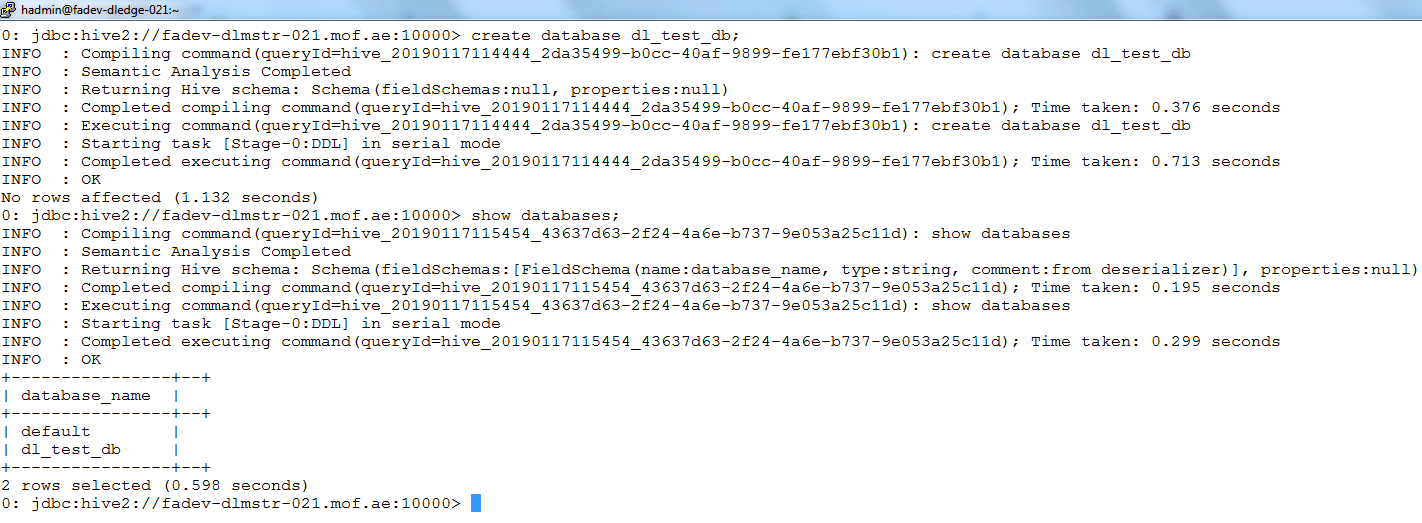
1. Sign in as ***datalaketest***, and make sure that ***datalaketest*** has no privileges on creating tables.



1. Create Role and grant all the administration privileges on groups to ***datalaketest*** via Hue.

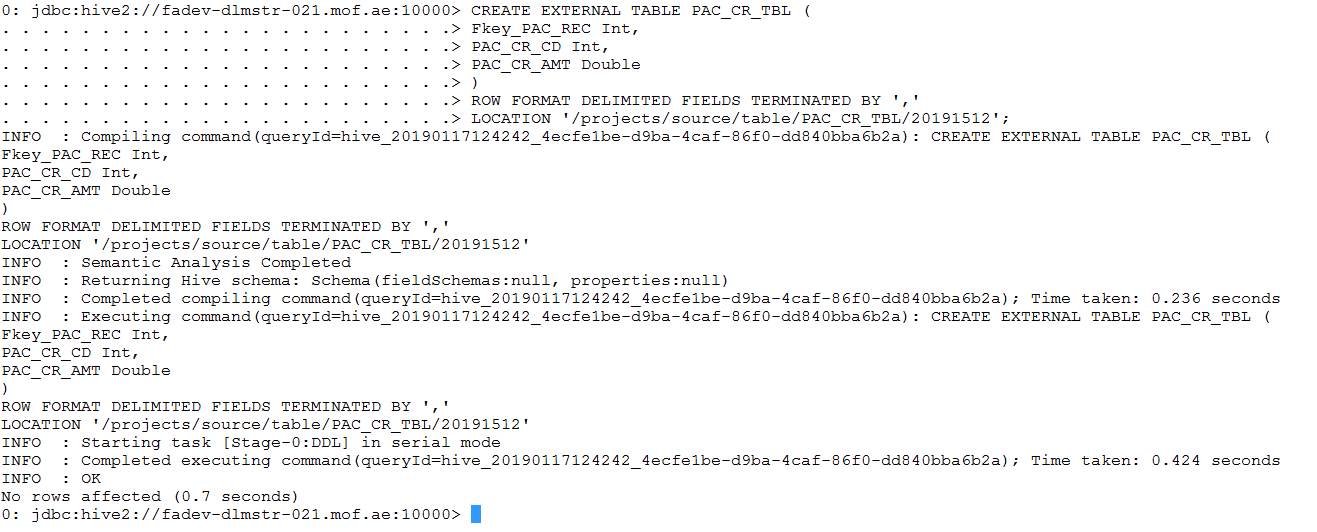


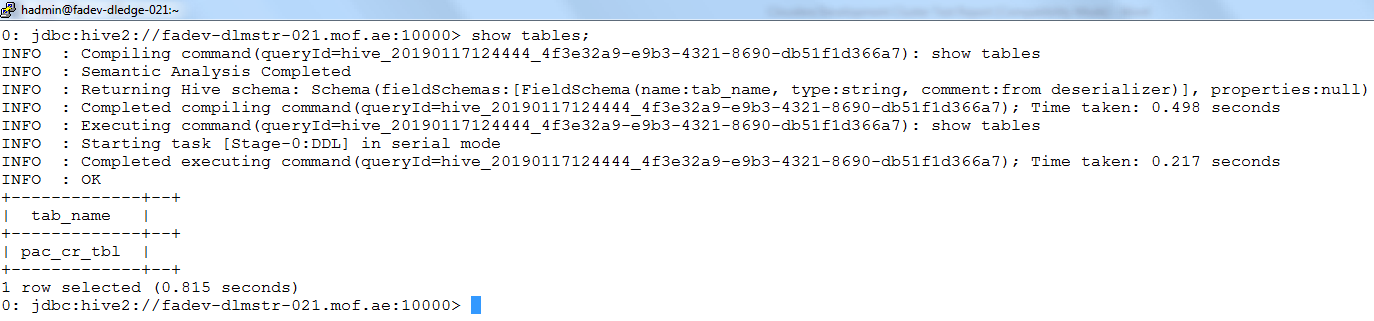
1. Create database named *dl\_test\_db*.



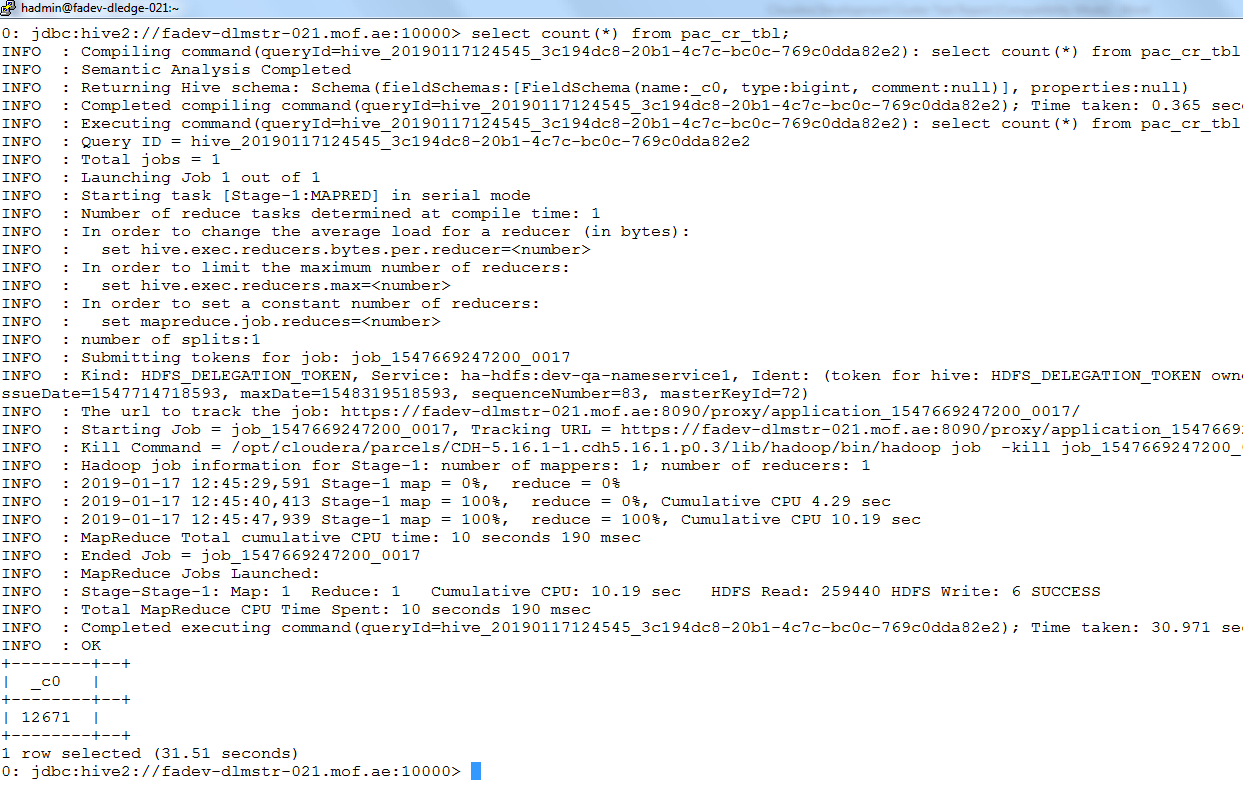
## HIVE queries on the basic dataset

1. Using MoF sample data creating hive table on top of it.





1. Execute a query statement over the dataset.

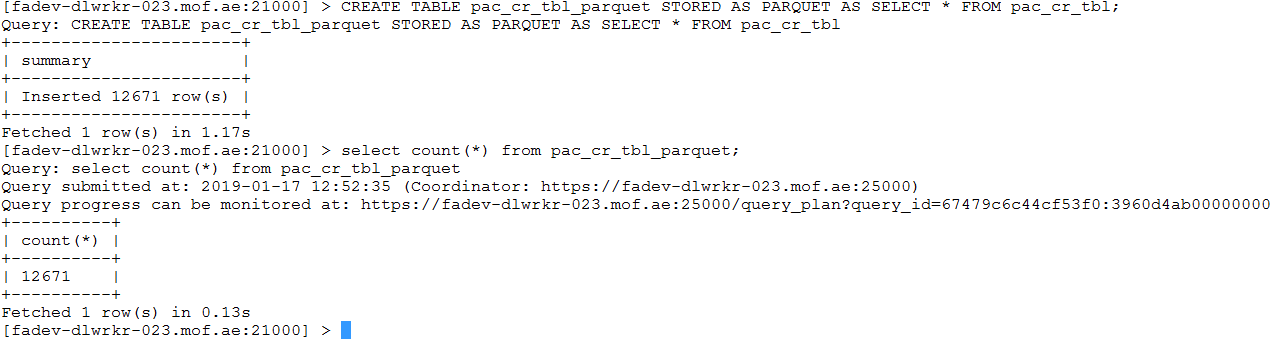


## IMPALA queries with Parquet

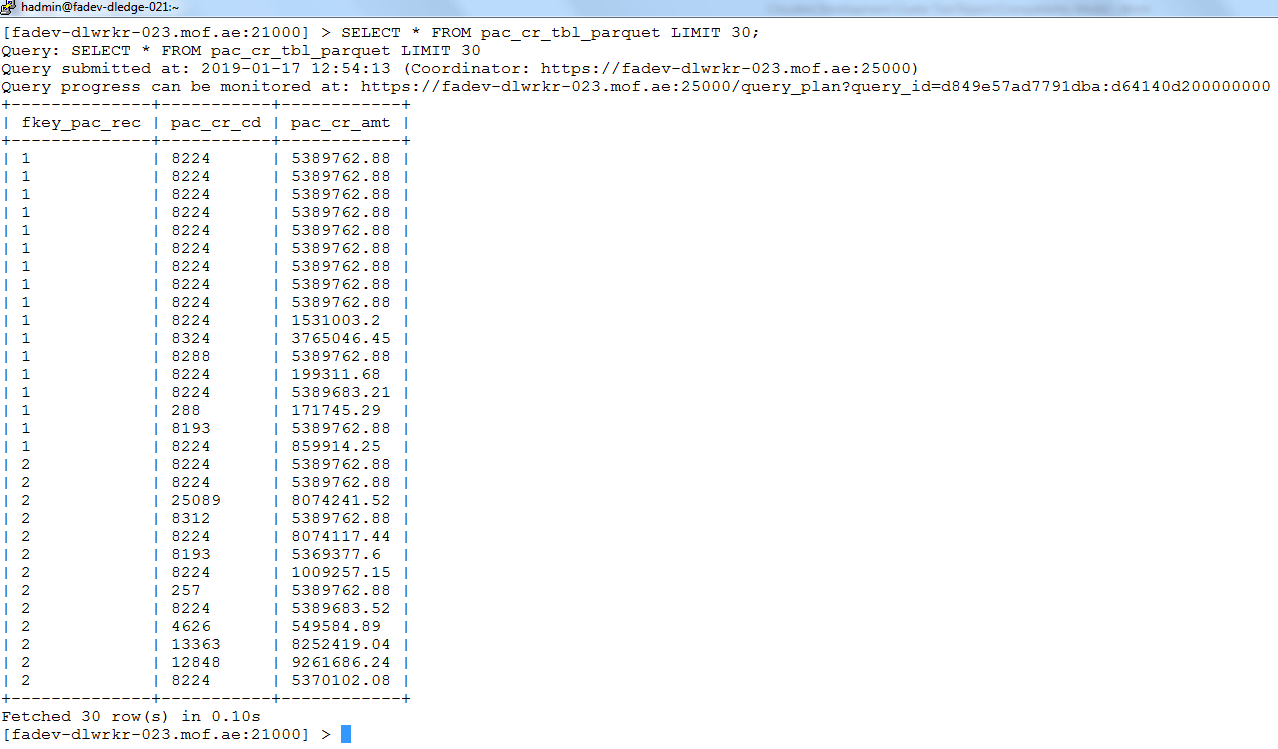
1. Execute invalidate metadata command in impala.



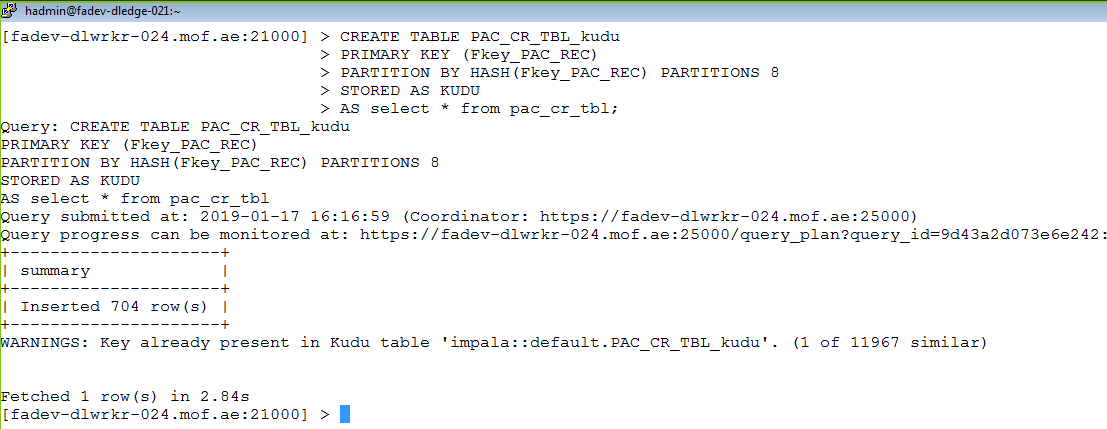
1. Create table with the Parquet format in Impala.

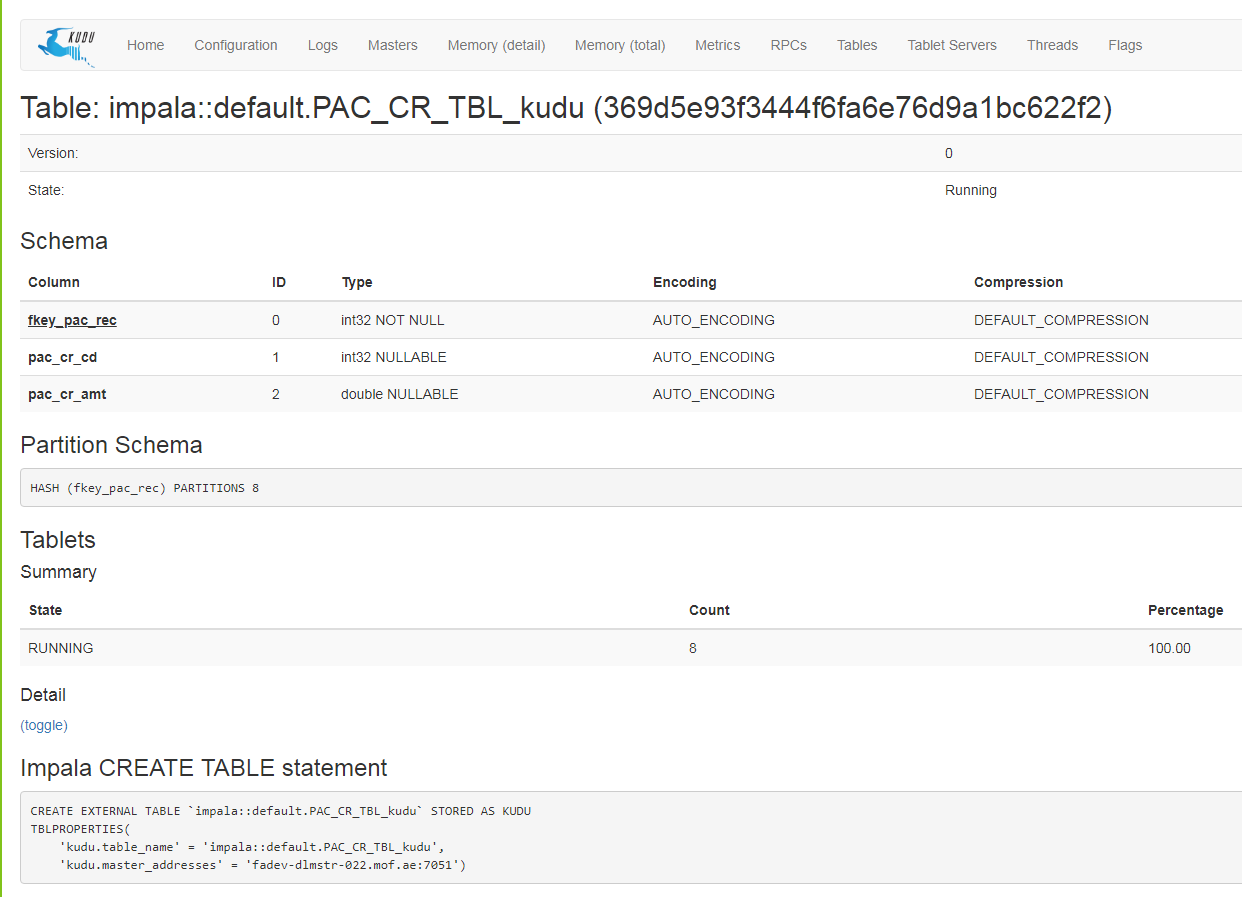


1. Execute select statement over the table



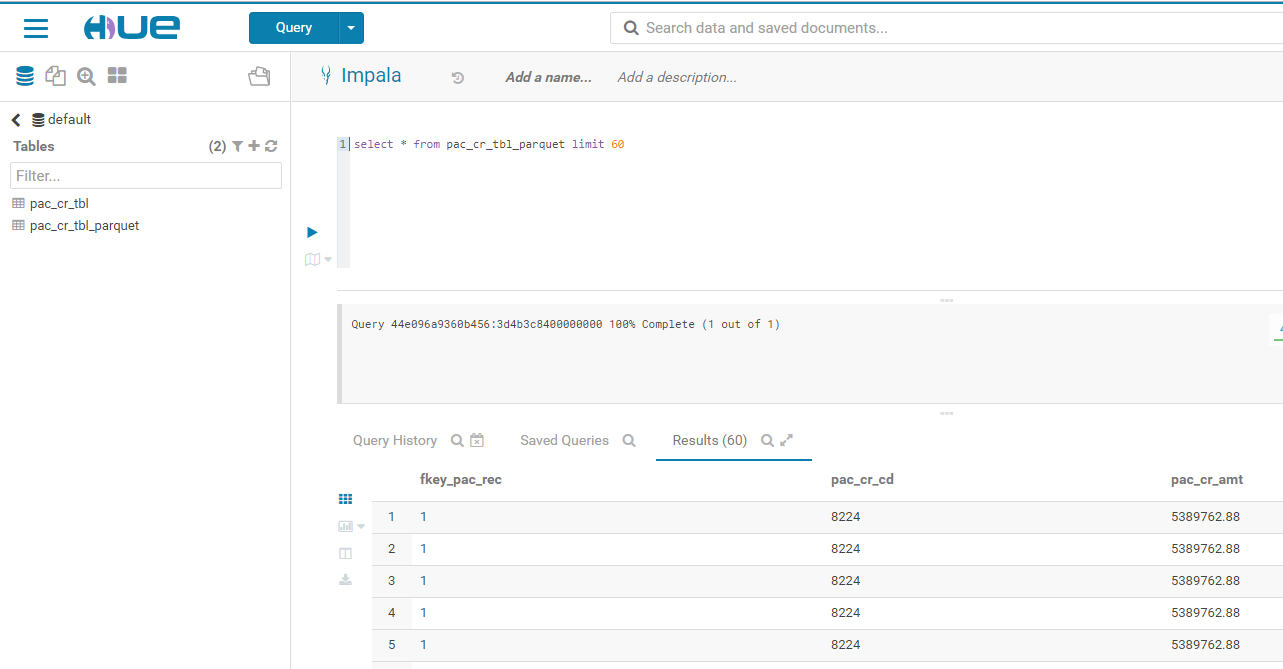
## IMPALA queries with Kudu





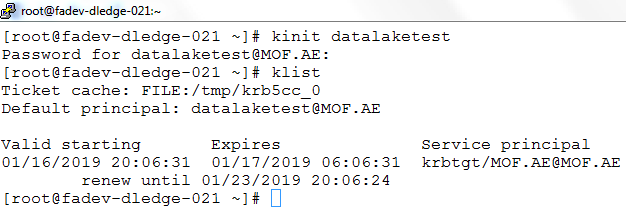
## HIVE/IMPALA queries in HUE

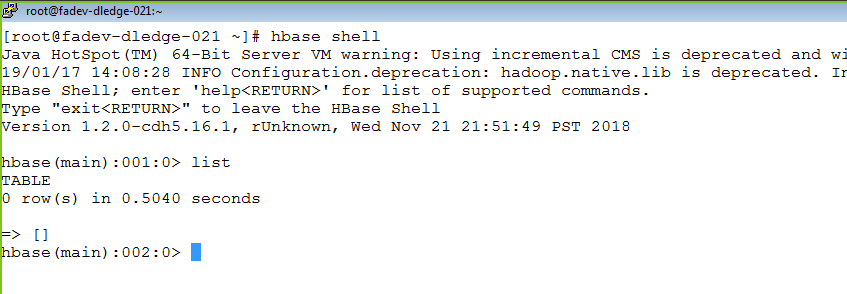
1. Execute same query statement in HUE



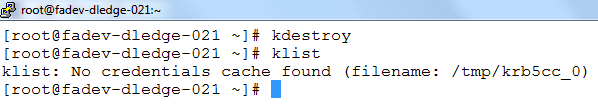
# HBase Test

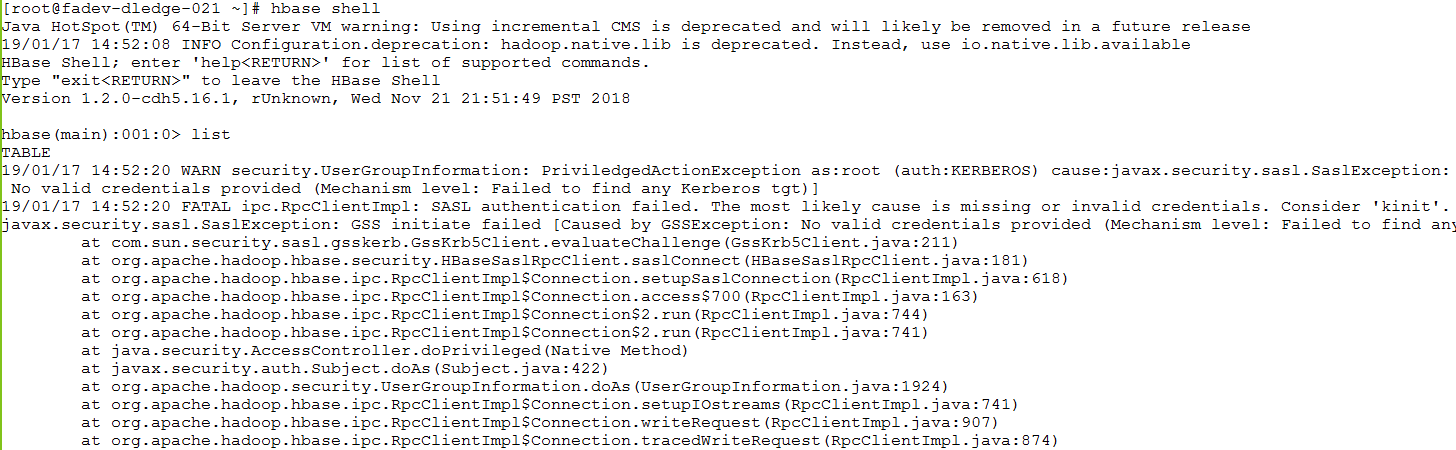
## HBase access with Kerberos





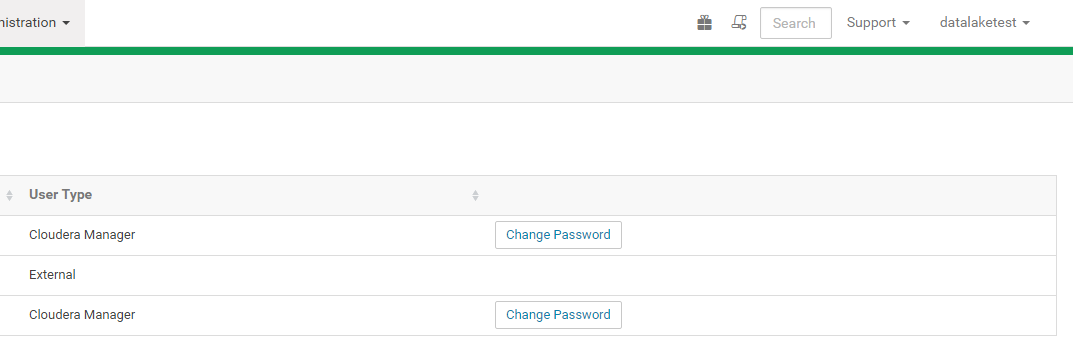
## HBase access without Kerberos

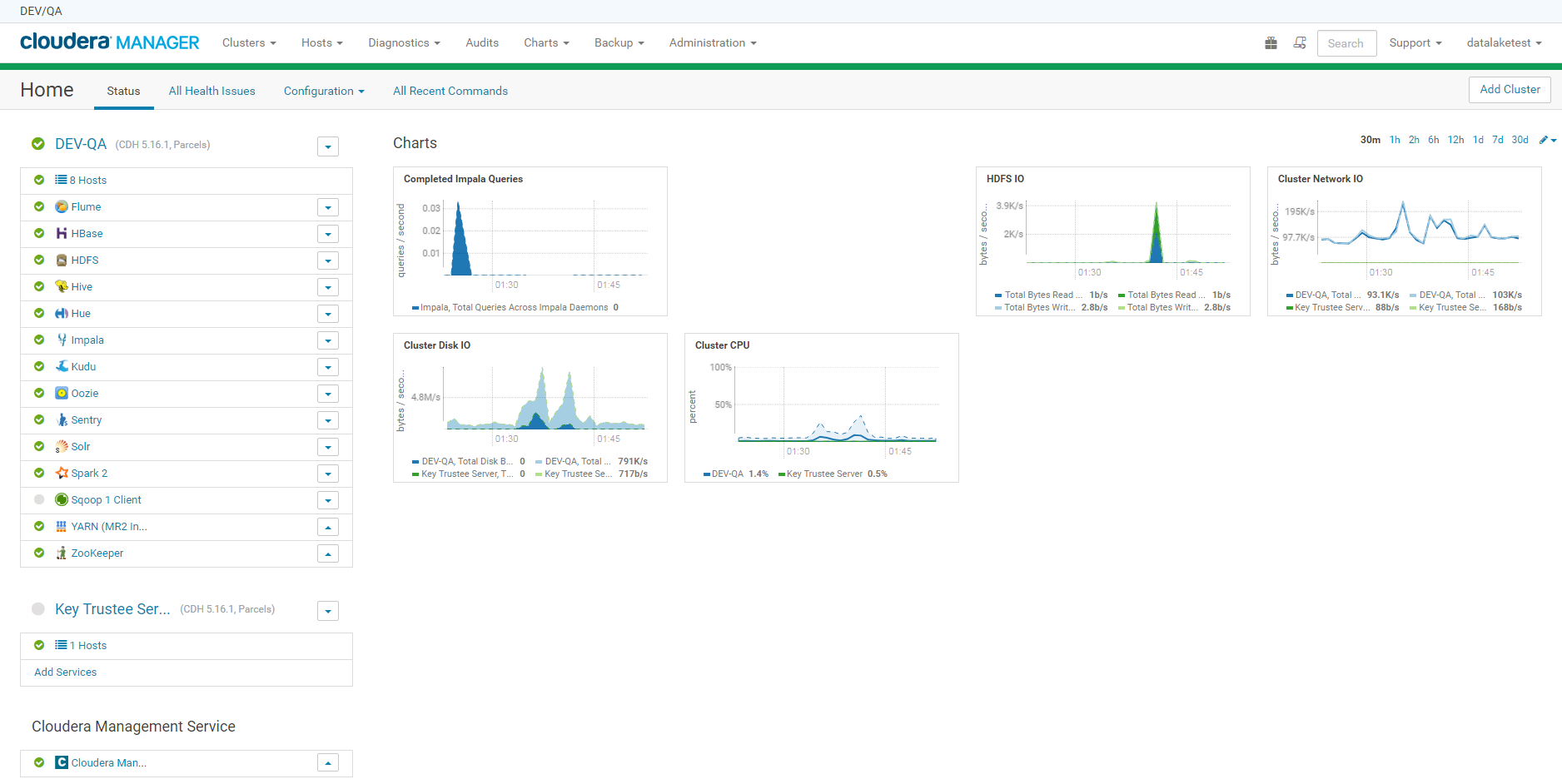




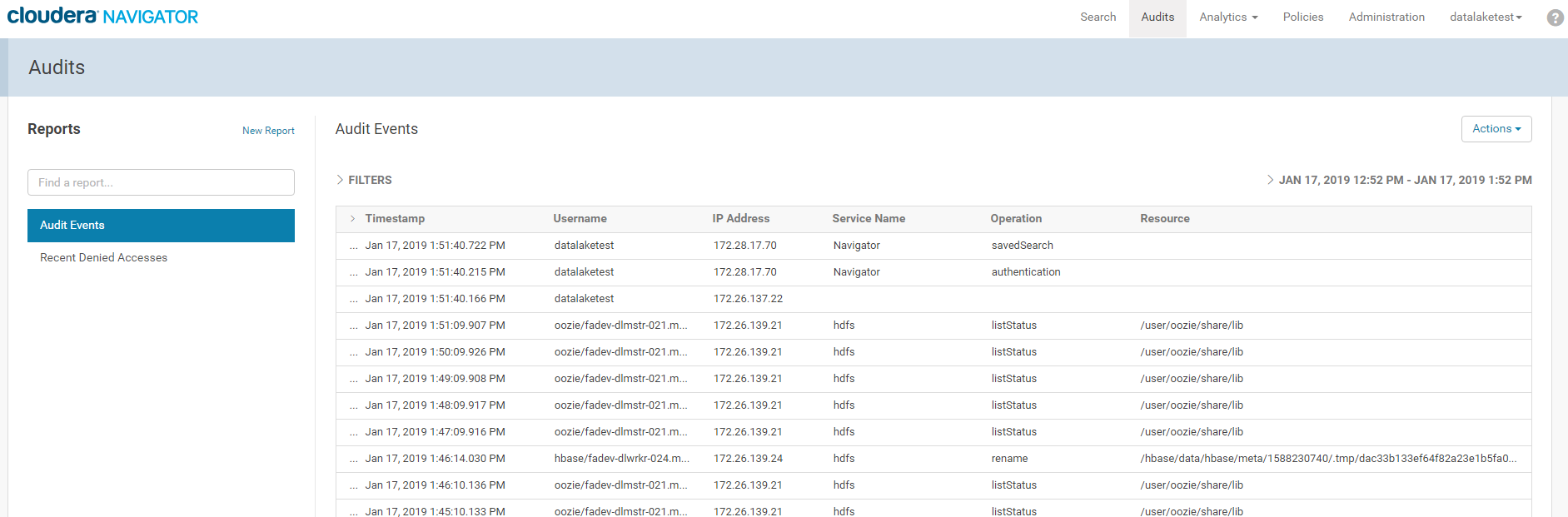
# CM and Navigator Tests

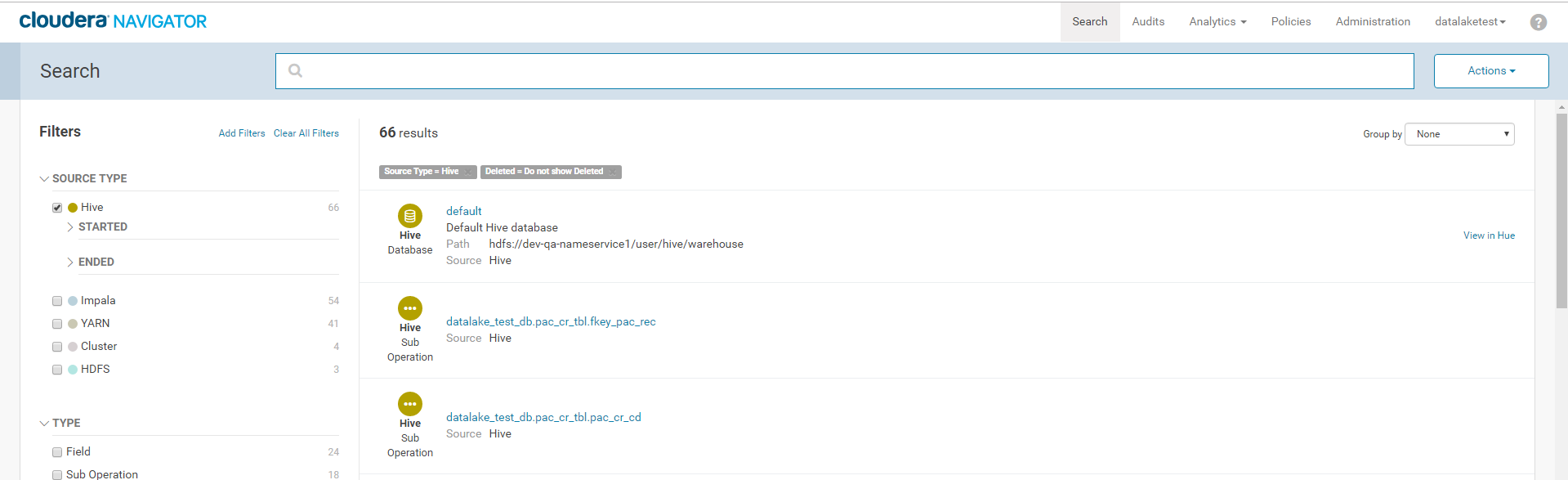
## LDAP integration with CM



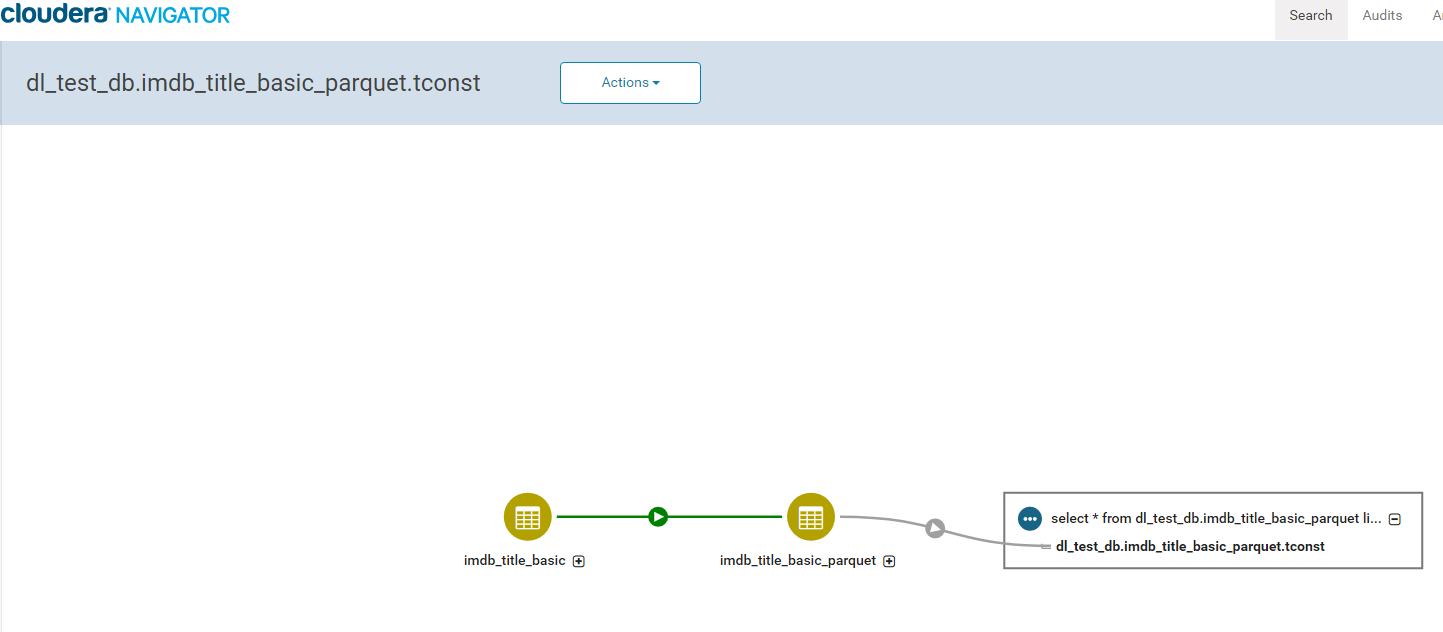


## Navigator audit.





## Navigator data lineage.



# Cloudera Enterprise Search (Solr) Test

## Solr access with Kerberos

## Solr access without Kerberos

# Informatica connectivity Test

<TBD>

# Tableau connectivity Test

<TBD>