Title: Setting up Hive JDBC connection with Kerberos using Java, Scala

Shubhro Banerjee [Project Manager]
11-April-2019

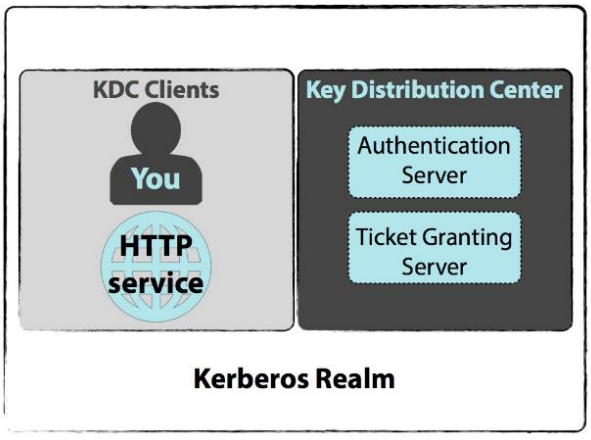
Table of contents

1. Introduction to Kerberos

For the trivia-loving folks, Kerberos' name comes from Greek mythology, the three-headed guard dog of Hades.

Kerberos is a protocol for authentication created by MIT as a solution to these network security problems. Kerberos uses tickets to authenticate instead of storing passwords locally or sending them over the internet. It takes a third-party (a Key Distribution Centre) to authenticate between a client and a service or host machine. In Kerberos, a ticket is a proof of identity for the particular service requested on your local machine (creation of a ticket is described below), which is then used to access the requested service that is within a Kerberos realm.

In Kerberos, Admins create realms – Kerberos realms – that will encompass all that is available to access. A Kerberos Realm will have - the Client as well as the service or host you want to request and the Key Distribution Centre (KDC)



When requesting access to a service or host, three interactions take place between you (client) and:

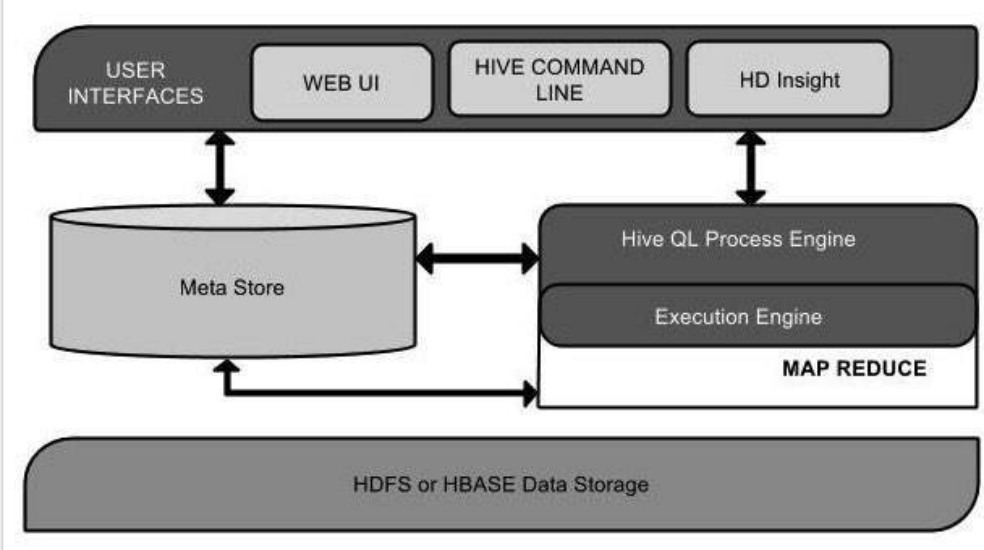
1) The Authentication Server
2) The Ticket Granting Server
3) Service or host machine that you're wanting access to.

Once Authentication passes thru these interactions, client receives the access to the server.

2. Introduction to Hive

Apache Hive is a data warehousing solution for Hadoop which provides data summarization, query, and ad-hoc analysis large datasets stored in Hadoop's HDFS. It is used to process structured and semi-structured data in Hadoop.

A Typical Hive Architecture looks like,



The various Components in the diagram are,

Unit Name	Operation
User Interface	Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).
Meta Store	Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.
Hive QL Process Engine	Hive QL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of

	writing MapReduce program in Java, we can write a query for MapReduce job and process it.
Execution Engine	The conjunction part of Hive QL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavour of MapReduce.
HDFS or HBASE	Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

3. How to determine if Kerberos is configured on Hadoop

To configure Kerberos authentication on HiveServer2, the following properties in hive-site.xml is configured on each node where hiveserver2 is installed:

Property	**Value**
hive.server2.authentication	KERBEROS
hive.server2.authentication.kerberos.principal	HiveServer2 Principle.
hive.server2.authentication.kerberos.keytab	The Keytab file for the HiverServer2 principle

Example,

```
<property>
  <name>hive.server2.authentication</name>
  <value>KERBEROS</value>
  <description>authenticationtype</description>
</property>
<property>
  <name>hive.server2.authentication.kerberos.principal</name>
  <value>HIVE/FQDN@REALM</value>
  <description>HiveServer2 principal. If _HOST is used as the FQDN portion, it will
be replaced with the actual hostname of the running instance.</description>
</property>
<property>
  <name>hive.server2.authentication.kerberos.keytab</name>
  <value>/opt/conf/hive.keytab</value>
  <description>Keytab file for HiveServer2 principal</description>
</property>
```

If the mentioned properties are configured in Hive-site.xml, it implies that Kerberos authentication is enabled on Hive Server and the Hive Databases, Tables and Schemas.

4. Accessing Hive from a remote server thru Hive JDBC Connection with Kerberos

Creating a Keytab file for connecting to Hive or DATALAKE in general:

A keytab is a file containing pairs of Kerberos principals and encrypted keys (which are derived from the Kerberos password). One can use a keytab file to authenticate to various remote systems using Kerberos without entering a password. However, when client change Kerberos password, client will need to recreate all keytabs.

Keytab files are commonly used to allow scripts to automatically authenticate using Kerberos, without requiring human interaction or access to password stored in a plain-text file. The script is then able to use the acquired credentials to access files stored on a remote system.

1) Open putty. (Download Putty http://www.putty.org/)
2) Run the following commands to open the keytab creating interface,
   a) ktutil (run this command from home directory because you will be writing the keytab to it)
   b) Input the following commands,
      (SYNTAX: add_entry {-key|-password} -p principal -k kvno -e enctype)

```
addent -password -p <personal-id>@CORE*******.PROD.COM -k 1 -e aes128-cts-hmac-sha1-96
addent -password -p <personal-id>@CORE*******.PROD.COM -k 1 -e aes256-cts-hmac-sha1-96
addent -password -p <personal-id>@CORE*******.PROD.COM -k 1 -e des3-cbc-sha1
addent -password -p <personal-id>@CORE*******.PROD.COM -k 1 -e arcfour-hmac
addent -password -p <personal-id>@CORE*******.PROD.COM -k 1 -e des-hmac-sha1
addent -password -p <personal-id>@CORE*******.PROD.COM -k 1 -e des-cbc-md5
```

   c) command will create the keytab,

```
wkt <personal-id>.keytab
```

   d) Enter 'exit' to get out of ktutil.

EXAMPLE

```
ktutil:  add_entry -password -p alice@BLEEP.COM -k 1 -e
    aes128-cts-hmac-sha1-96
Password for alice@BLEEP.COM:
ktutil:  add_entry -password -p alice@BLEEP.COM -k 1 -e
    aes256-cts-hmac-sha1-96
Password for alice@BLEEP.COM:
ktutil:  write_kt keytab
ktutil:
```

Creating a JAAS Login Configuration File.

JAAS authentication is performed in a pluggable fashion, so Java applications can remain independent from underlying authentication technologies. Configuration information such as the desired authentication technology is specified at runtime. The source of the configuration information (for example, a file or a database) is up to the **class javax.security.auth.login.Configuration**. It reads configuration information from configuration files.

Sample JAAS file, (gss-jaas.conf)

```
com.sun.security.jgss.initiate {
   com.sun.security.auth.module.Krb5LoginModule required
   useKeyTab=true
   useTicketCache=false
   principal="<principal>" //example: <personal-id>@CORE*******.PROD.COM
   doNotPrompt=true
   keyTab="<directory-path>/<personal-id>.keytab"
   debug=true;
};
```

Creating a Kerberos configuration file:

krb5.conf contains configuration information needed by the Kerberos V5 library. This includes information describing the default Kerberos realm, and the location of the Kerberos key distribution centers for known realms.

Sections are delimited by square braces; within each section, there are relations where tags can be assigned to have specific values. Tags can also contain a subsection, which contains further relations or subsections. A tag can be assigned to multiple values.

krb5.conf Syntax:

```
[section1]

tag1 = value_a
tag1 = value_b
tag2 = value_c

[section 2]

tag3 = {
subtag1 = subtag_value_a
subtag1 = subtag_value_b
subtag2 = subtag_value_c
}
tag4 = {
subtag1 = subtag_value_d
subtag2 = subtag_value_e
}
```

Example (sample kbr5.conf),

```
[logging]
default = FILE:/opt/appkrb5/var/log/krb5libs.log
kdc = FILE:/opt/app/krb5/var/log/krb5kdc.log
admin_server = FILE:/opt/app/krb5/var/log/kadmind.log

[libdefaults]
default_realm = <realm-name>
dns_lookup_realm = false
dns_lookup_kdc = true
ticket_lifetime = 24h
renew_lifetime = 7d
rdns = false
default_tkt_enctypes = aes128-cts des3-cbc-sha1 rc4-hmac des-cbc-md5 des-cbc-crc
default_tgs_enctypes = aes128-cts des3-cbc-sha1 rc4-hmac des-cbc-md5 des-cbc-crc
permitted_enctypes = aes128-cts des3-cbc-sha1 rc4-hmac des-cbc-md5 des-cbc-crc

[realms]
<realm-name> = {
kdc = <server-name>:<port-number>
admin_server = <server-name>:<port-number>
kpasswd_server = <server-name>:<port-number>
}

[domain_realm]
<server-name> = <realm-name>
<server-name> = <realm-name>
```

NOTE: all the fields marked as <> will be provided by System Admin team who have installed and configured Kerberos on servers.

Running HQL (Hive Query Language) from a Shell Script using keytab:

Use "kinit" to Obtain and cache Kerberos ticket-granting ticket from the script,

SYNTAX: kinit [commands] <principal name> [<password>]

Command Option	Description
-A	Do not include addresses.
-f	Issue a forward able ticket.
-p	Issue a proxiable ticket.
-c <cache_name>	The cache name (i.e., FILE:d:\temp\mykrb5cc).
-k	Use keytab
-t <keytab_filename>	The keytab name (i.e,d:\winnt\profiles\duke\krb5.keytab).
<principal>	The principal name (i.e., duke@example.com).
<password>	The principal's Kerberos password. (DO NOT SPECIFY ON COMMAND LINE OR IN A SCRIPT.)

-help	Displays instructions.

Shell Script Example,

```
#!/bin/bash

kinit -kt <directory-path>/sample_file.keytab <personal-id>@CORE*******.PROD.COM

hive -S -e "CREATE DATABASE IF NOT EXISTS sample_database;"

hive -S -e "CREATE TABLE IF NOT EXISTS sample_database.table1(field STRING) ROW
FORMATED DELIMITED;"

hive -S -e "LOAD DATA INPATH 'hive_shubhro/file1.txt' OVERWRITE INTO TABLE
sample_database.table1;"
```

Running HQL (Hive Query Language) from a Java/Scala using hive JDBC and keytab:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class HiveJDBC {

   private static String driverName = "org.apache.hive.jdbc.HiveDriver";
   public static void main(String[] args) throws Exception {
    Connection con = null;
    Statement stmt = null;
    try {
              System.out.println("HiveJDBC Start");
              Class.forName(driverName);
              System.out.println("HiveJDBC driverName accepted");

System.setProperty("java.security.auth.login.config","/home/shubhro/gss-jaas.conf");
              System.setProperty("sun.security.jgss.debug","true");
              System.setProperty("javax.security.auth.useSubjectCredsOnly","false");

       System.setProperty("java.security.krb5.conf","/home/shubhro/krb5.conf");

              System.out.println("getting connection");

       con =DriverManager.getConnection("jdbc:hive2://<cluster server
names>/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2");
              System.out.println("got connection");

              stmt = con.createStatement();
              String sql = "show databases";
              System.out.println("Running: " + sql);
              ResultSet res = stmt.executeQuery(sql);
              while (res.next()) {
                 System.out.println(res.getString(1));
              }

              ResultSet res1 = stmt.executeQuery("select * from
sample_database.table1 limit 10");
              while (res1.next()) {
               System.out.println(res1.getString(1));
```

```
                }
                }
                catch (Exception e) {
                  e.printStackTrace();
                }
                finally {
                  stmt.close();
                  con.close();
                }
  }
}
```

Script to run HiveJDBC class file,

```
#!/bin/bash
HADOOP_HOME=/usr/hdp/current/hadoop-client
HADOOP_HOME_CLIENT=/usr/hdp/current/hadoop-client/client
HIVE_HOME=/usr/hdp/current/hive-client
HADOOP_CORE=`ls /usr/hdp/current/hadoop-client/hadoop-common.jar`
HADOOP_CLIENT=`ls /usr/hdp/2.6.5.4-1/hadoop-mapreduce//hadoop-mapreduce-client-core-
2.7.3.2.6.5.4-1.jar`
COMMON_CONFIG=`ls /usr/hdp/current/hadoop-client/client/commons-configuration.jar`
CLASSPATH=.:$HADOOP_CORE:$HADOOP_CLIENT:$COMMON_CONFIG:$HIVE_HOME/conf

for jar_file_name in ${HIVE_HOME}/lib/*.jar
do
CLASSPATH=$CLASSPATH:$jar_file_name
done

for hdp_jar_file_name in ${HADOOP_HOME_CLIENT}/*.jar
do
CLASSPATH=$CLASSPATH:$hdp_jar_file_name
done

$JAVA8_HOME/bin/java -cp $CLASSPATH HiveJDBC
```

5. References

https://web.mit.edu/kerberos/

https://en.wikipedia.org/wiki/Apache_Hive

https://mapr.com/docs/52/Hive/HiveServer2-KerberosAuth.html

https://docs.oracle.com/javase/8/docs/technotes/guides/security/jgss/tutorials/LoginConfigFile.html

https://kb.iu.edu/d/aumh

https://linux.die.net/man/5/krb5.conf

https://wiki.web.att.com/display/ED/Copy+of+How+to+Create+a+Keytab+for+Connecting+with+SAS+to+KM+Core+Datalake

https://docs.oracle.com/javase/7/docs/technotes/tools/windows/kinit.html

Disclaimer

Tech Mahindra, herein referred to as TechM provide a wide array of presentations and reports, with the contributions of various professionals. These presentations and reports are for informational purposes and private circulation only and do not constitute an offer to buy or sell any securities mentioned therein. They do not purport to be a complete description of the markets conditions or developments referred to in the material. While utmost care has been taken in preparing the above, we claim no responsibility for their accuracy. We shall not be liable for any direct or indirect losses arising from the use thereof and the viewers are requested to use the information contained herein at their own risk. These presentations and reports should not be reproduced, re-circulated, published in any media, website or otherwise, in any form or manner, in part or as a whole, without the express consent in writing of TechM or its subsidiaries. Any unauthorized use, disclosure or public dissemination of information contained herein is prohibited. Unless specifically noted, TechM is not responsible for the content of these presentations and/or the opinions of the presenters. Individual situations and local practices and standards may vary, so viewers and others utilizing information contained within a presentation are free to adopt differing standards and approaches as they see fit. You may not repackage or sell the presentation. Products and names mentioned in materials or presentations are the property of their respective owners and the mention of them does not constitute an endorsement by TechM. Information contained in a presentation hosted or promoted by TechM is provided “as is” without warranty of any kind, either expressed or implied, including any warranty of merchantability or fitness for a particular purpose. TechM assumes no liability or responsibility for the contents of a presentation or the opinions expressed by the presenters. All expressions of opinion are subject to change without notice.

Thank You

Visit us at techmahindra.com