SQL Client Example using KnoxShell in Apache Knox

The KnoxShell release artifact provides a small footprint client environment that removes all unnecessary server dependencies, configuration, binary scripts, etc. It is comprised of a couple different things that empower different sorts of users.

- A set of SDK type classes for providing access to Hadoop resources over HTTP
- A Groovy based DSL for scripting access to Hadoop resources based on the underlying SDK classes
- A KnoxShell Token based Sessions to provide a CLI SSO session for executing multiple script

While testing the KnoxShell examples for the 0.14.0 Apache Knox release, I realized that using the KnoxShell for access to HiveServer2 was not easily done.

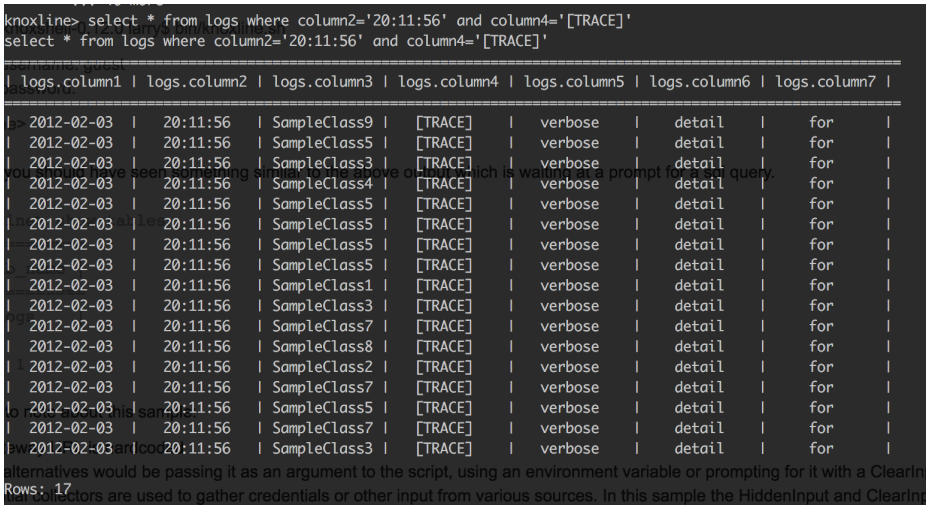
This is due to the fact that we are leveraging the knoxshell executable jar which makes is difficult to add additional classes and jars to the classpath for the executing script.

I needed to create a launch script that called the main class of the executable jar while also being able to set the classpath with additional jars for Apache Hive clients.

This article will go over the creation of a simple SQL client that we will call "knoxline" by using the KnoxShell Groovy based DSL.

This particular article should work using the 0.14.0 knoxshell download and with previous gateway server releases as well.

We will show how to use a simple groovy script to write a SQL client that can do something like the following:



Download

In the 0.14.0 release, you may get to the knoxshell download through the Apache Knox site.

From this above page click the Gateway client binary archive link or just use the one here.

Unzip this file into your preferred location which will result in a knoxshell-0.14.0 directory and we will refer to that location as the {GATEWAY_HOME}.

CD {GATEWAY_HOME}

You should see something similar to the following:

> *bash-3.2$ ls -l*
> *total 160*
> *-r--r--r--@ 1 larry staff 71714 Dec 6 18:32 LICENSE*
> *-r--r--r--@ 1 larry staff 164 Dec 6 18:32 NOTICE*
> *-rw-r--r--@ 1 larry staff 1452 Dec 6 18:32 README*
> *drwxr-xr-x@ 6 larry staff 204 Dec 14 18:06 bin*
> *drwxr--r--@ 3 larry staff 102 Dec 14 18:06 conf*
> *drwxr-xr-x@ 19 larry staff 646 Dec 14 18:06 samples*

Directory	Description
bin	contains the main knoxshell jar and related shell scripts

conf	only contains log4j config
logs	contains the knoxshell.log file
samples	has numerous examples to help you get started

Setup Truststore for Client

Get/setup truststore for the target Knox instance or fronting load balancer

- if you have access to the server you may use the command
    - `knoxcli.sh export-cert --type JKS`
- copy the resulting gateway-client-identity.jks to your user home directory
- you may also ask your Knox administrator to provide you with the public cert for the gateway and create your own truststore within your user home directory

NOTE: if you see errors related to SSL and PKIX your truststore is not properly setup

Add Hive Client Libraries

In order to add the client libraries that provide the HiveDriver and others, we will add an additional directory to the above structure.

Directory	Description
lib	To contain external jars to add to the classpath for things like HiveDriver

Next we will download the hive standalone client jar which will contain nearly everything we need.

For this article, we will download Hive 1.2.1 standalone jar and copy it to the newly created lib directory.

You can use whatever version client jar is appropriate for your Hive deployment.

Add Commons Logging Jar

Download commons logging jar and copy to the libs directory as well.

Add Launch Script

As I mentioned earlier, we need to add a launch script that will execute the main class of the knoxshell executable jar while allowing us to set additional jars on the classpath.
Save the following to a file named knoxline.sh within the bin directory:

```
java -Dlog4j.configuration=conf/knoxshell-log4j.properties -cp bin/knoxshell.jar:lib/* org.apache.hadoop.gateway.shell.Shell bin/hive2.
groovy "$@"
```

and save the following to another script in the bin directory called hive2.groovy:

```
import java.sql.DriverManager

import java.sql.SQLException
import org.apache.hadoop.gateway.shell.Credentials

gatewayHost = "localhost";
gatewayPort = 8443;
trustStore = System.getProperty('user.home') + "/gateway-client-trust.jks";
trustStorePassword = "changeit";
contextPath = "gateway/sandbox/hive";
sql = ""
```

```
if (args.length == 0) {
  // accept defaults
  System.out.println(String.format("\nDefault connection args: %s, %d, %s, %s, %s", gatewayHost, gatewayPort, trustStore,
trustStorePassword, contextPath))
} else if (args[0] == "?" || args[0] == "help") {
  System.out.println("\nExpected arguments: {host, port, truststore, truststore-pass, context-path}\n")
  System.exit(0);
} else if (args.length == 5) {
  gatewayHost = args[0];
  gatewayPort = args[1].toInteger();
  trustStore = args[2];
  trustStorePassword = args[3];
  contextPath = args[4];
  System.out.println(String.format("\nProvided connection args: %s, %d, %s, %s, %s", gatewayHost, gatewayPort, trustStore,
trustStorePassword, contextPath))
} else if (args.length > 0) {
  System.out.println("\nERROR: Expected arguments: NONE for defaults or {host, port, truststore, truststore-pass, context-path}\n")
  System.exit(1);
}

connectionString = String.format( "jdbc:hive2://%s:%d/;ssl=true;sslTrustStore=%s;trustStorePassword=%s?hive.server2.transport.
mode=http;hive.server2.thrift.http.path=/%s", gatewayHost, gatewayPort, trustStore, trustStorePassword, contextPath );

System.out.println("<======KnoxLine======>");
System.out.println("powered by Apache Knox");
System.out.println("");

credentials = new Credentials()
credentials.add("ClearInput", "Enter username: ", "user")
  .add("HiddenInput", "Enter pas" + "sword: ", "pass")
credentials.collect()

user = credentials.get("user").string()
pass = credentials.get("pass").string()

// Load Hive JDBC Driver
Class.forName( "org.apache.hive.jdbc.HiveDriver" );

// Configure JDBC connection
connection = DriverManager.getConnection( connectionString, user, pass );

while(1) {
  sql = System.console().readLine 'knoxline> '
  if (!sql.equals("")) {
    System.out.println(sql)

    rs = true;
    statement = connection.createStatement();
    try {
      if (statement.execute( sql )) {
        resultSet = statement.getResultSet()
        int colcount = 0
        colcount = resultSet.getMetaData().getColumnCount();
        row = 0
        header = "| "
        while ( resultSet.next() ) {
          line = "| "
          for (int i = 1; i <= colcount; i++) {
            colvalue = resultSet.getString( i )
            if (colvalue == null) colvalue = ""
            colsize = colvalue.length()
            headerSize = resultSet.getMetaData().getColumnLabel( i ).length()
            if (headerSize > colsize) colsize = headerSize
            if (row == 0) {
              header += resultSet.getMetaData().getColumnLabel( i ).center(colsize) + " | ";
            }
            line += colvalue.center(colsize) + " | ";
          }
          if (row == 0) {

            System.out.println("".padLeft(header.length()-1, "="))

            System.out.println(header);
```

```
            System.out.println("".padLeft(header.length()-1, "="))
          }
          System.out.println(line);
          row++
          }
          System.out.println("\nRows: " + row + "\n");
          resultSet.close();
        }
      }
      catch(SQLException e) {
        //e.printStackTrace()
        System.out.println("SQL Exception encountered... " + e.getMessage())

        if (e.getMessage().contains("org.apache.thrift.transport.TTransportException")) {
          System.out.println("reconnecting... ")
          connection = DriverManager.getConnection( connectionString, user, pass );
        }
      }
      statement.close();
    }
  }
  connection.close();
```

Execute a SQL Commands using KnoxLine

Enter the knoxline sql client at the command line:

I will use the defaults for the arguments in the script:

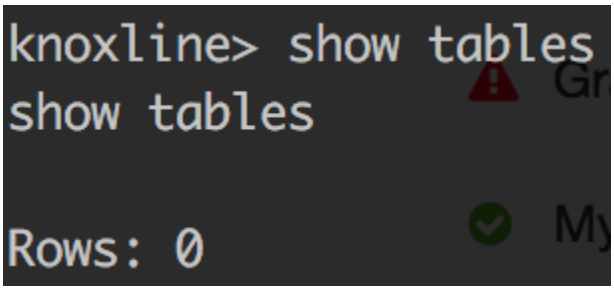
```
Default connection args: localhost, 8443, /Users/larry/gateway-client-trust.jks, changeit, gateway/sandbox/hive
```

Depending on your deployment, you may want to set the above arguments on the CLI below:

```
knoxshell-0.14.0 larry$ bin/knoxline.sh
```

Let's check for existing tables:

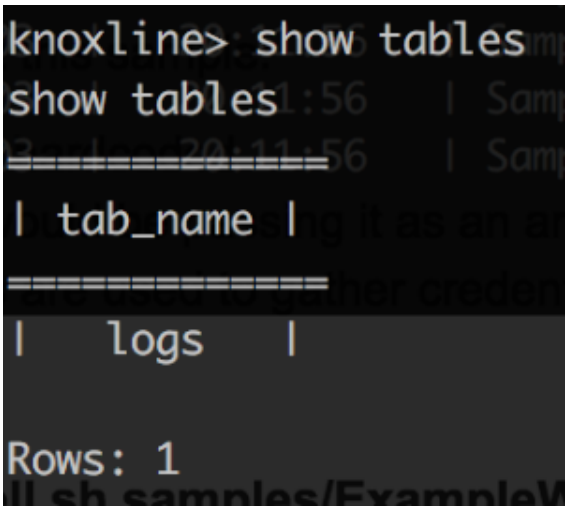
```
knoxline> show tables
```



Let's create a table by loading file from the local disk of the cluster machine:

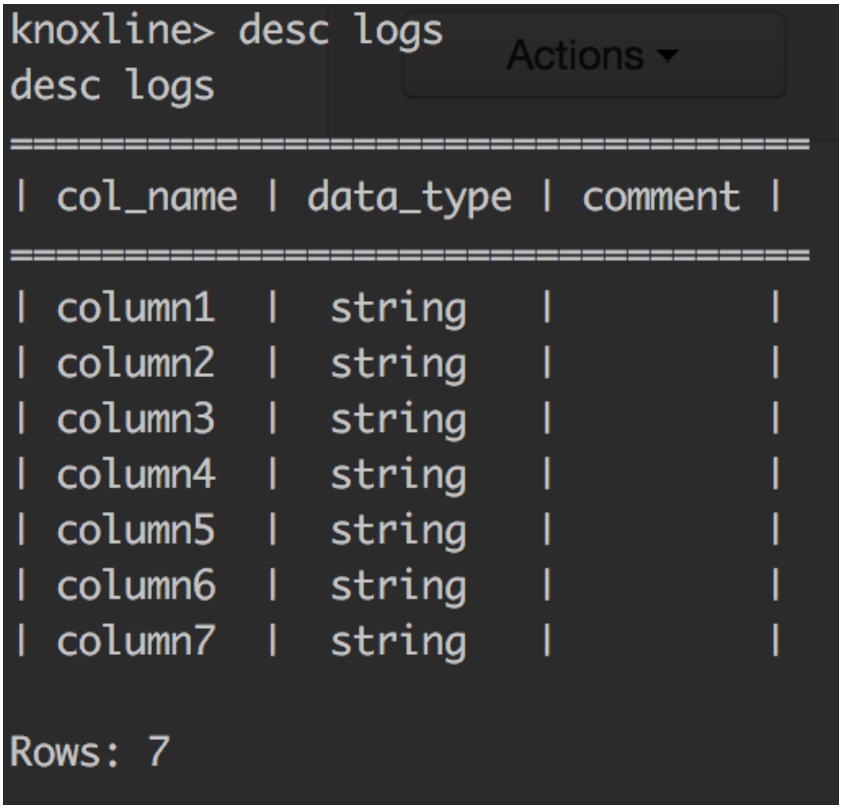
```
knoxline> CREATE TABLE logs(column1 string, column2 string, column3 string, column4 string, column5 string, column6 string,
column7 string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
```

Show the created table:



Show the table description:

> *knoxline> desc logs*

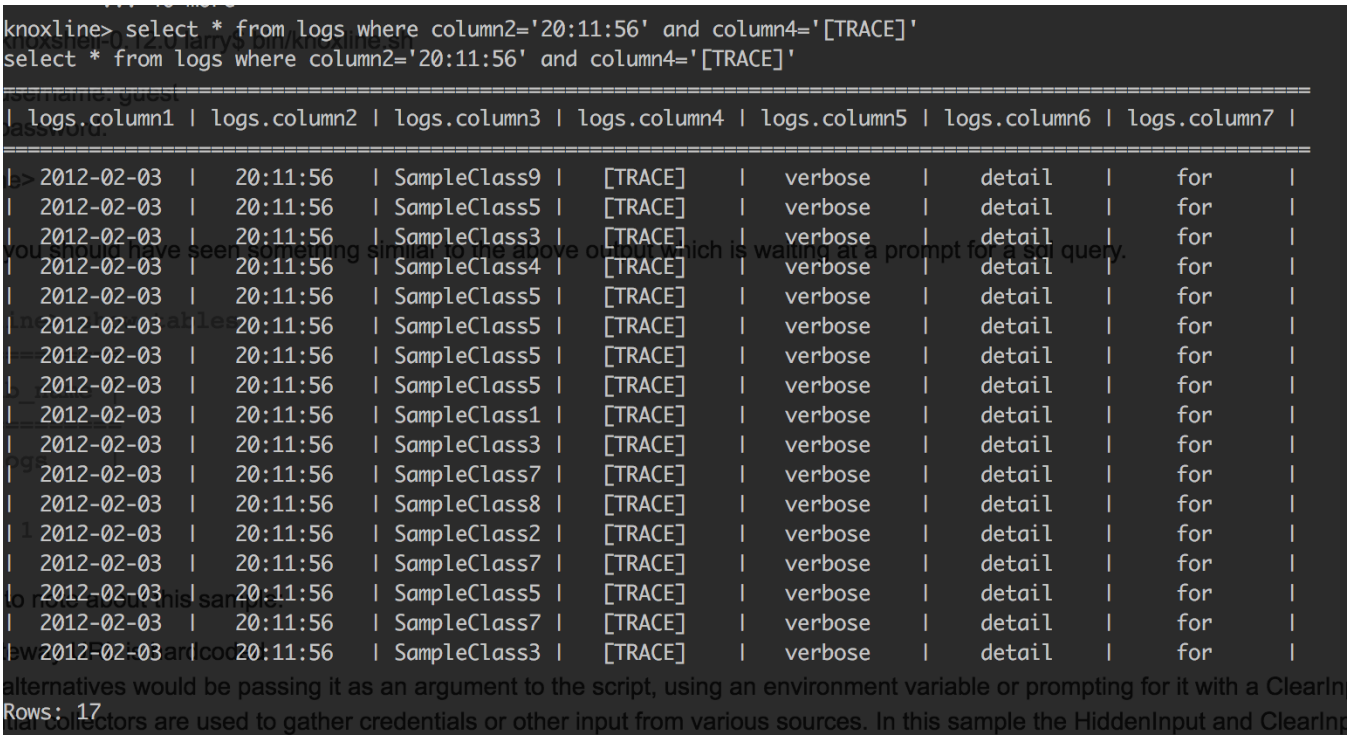


Load the data from the samples.log file in /tmp (copy the sample.log file from ./samples/hive/sample.log to the /tmp directory on your hiveserver2 host):

> *knoxline> LOAD DATA INPATH '/tmp/sample.log' OVERWRITE INTO TABLE logs*

Do a select from the table:

> *knoxline> select * from logs where column2='20:11:56' and column4='[TRACE]'*



```
knoxline> select * from logs where column2='20:11:56' and column4='[TRACE]'
select * from logs where column2='20:11:56' and column4='[TRACE]'
=============================================================================================================
| logs.column1 | logs.column2 | logs.column3 | logs.column4 | logs.column5 | logs.column6 | logs.column7 |
=============================================================================================================
| 2012-02-03   |   20:11:56   | SampleClass9 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass5 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass3 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass4 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass5 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass5 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass5 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass5 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass1 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass3 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass7 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass8 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass2 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass7 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass5 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass7 |   [TRACE]    |   verbose    |    detail    |     for      |
| 2012-02-03   |   20:11:56   | SampleClass3 |   [TRACE]    |   verbose    |    detail    |     for      |
Rows: 17
```

Some things to note about this sample:

1. The gateway URL defaults to the sandbox topology
    - alternatives would be passing it as an argument to the script
2. Credential collectors are used to gather credentials or other input from various sources. In this sample the HiddenInput and ClearInput collectors prompt the user for the input with the provided prompt text and the values are acquired by a subsequent get call with the provided name value.
3. The standard Java classes for JDBC are used rather than the Hadoop session object used for access to the pure REST APIs
4. The resultSet is rendered in the familiar table format of other command line interfaces but shows how to access it for doing whatever scripting needs you have
5. Error handling is more or less non-existent in this example

I hope to bring "knoxline" to the KnoxShell module in a future release just as a simple way to do some quick queries from your KnoxShell environment.