How to connect to the HBASE database of Analytics Engine on IBM Cloud via JDBC toolkit

This document describes a step by step instruction to connect to a HBASE database on IBM cloud

via **JDBC** toolkit.

1 - Download the phoenix jdbc drivers from:

https://github.com/IBM-Cloud/IBM-Analytics-Engine/tree/master/jdbcsamples/TestPhoenix/lib

and copy them in the **opt** directory in your project workspace:

```
JDBCHbase/opt/phoenix-4.9.0-HBase-1.1-client.jar
JDBCHbase/opt/phoenix-queryserver-4.9.0-HBase-1.1.jar
JDBCHbase/opt/phoenix-tracing-webapp-4.9.0-HBase-1.1.jar
JDBCHbase/opt/phoenix-4.9.0-HBase-1.1-queryserver.jar
JDBCHbase/opt/phoenix-queryserver-client-4.9.0-HBase-1.1.jar
```

2 - Create an Analytics Engine service on IBM Cloud and create a service credential

Create a Analytics Engine service on IBM cloud.

https://console.bluemix.net/catalog/?search=Analytics%20Engine

IBM Analytics Engine documentation

https://console.bluemix.net/docs/services/AnalyticsEngine/index.html#introduction

Create a service credential for Analytics Engine service on IBM cloud.

The following sample shows an Analytics Engine service credentials:

```
{
 "apikey": "xyxyxyxyxyxyxyxyxyxyxyxyxyx",
  "cluster": {
  "cluster_id": "20180404-125209-123-VNhbnQRZ",
  "password": "IAEPASSWORD",
  "service_endpoints": {
    "ambari_console": "https://chs-nxr-123-mn001.bi.services.us-south.bluemix.net:9443",
    "hive_jdbc": "jdbc:hive2://chs-nxr-123-mn001.bi.services.us-south.bluemix.net:8443/;ssl=t
    "livy": "https://chs-nxr-123-mn001.bi.services.us-south.bluemix.net:8443/gateway/default/
    "notebook_gateway": "https://chs-nxr-123-mn001.bi.services.us-south.bluemix.net:8443/gate
    "notebook_gateway_websocket": "wss://chs-nxr-123-mn001.bi.services.us-south.bluemix.net:8
    "oozie_rest": "https://chs-nxr-123-mn001.bi.services.us-south.bluemix.net:8443/gateway/de
    "phoenix_jdbc": "jdbc:phoenix:thin:url=https://chs-nxr-123-mn001.bi.services.us-south.blu
```

```
    "spark_history_server": "https://chs-nxr-123-mn001.bi.services.us-south.bluemix.net:8443/
    "spark_sql": "jdbc:hive2://chs-nxr-123-mn001.bi.services.us-south.bluemix.net:8443/;ssl=t
    "ssh": "ssh clsadmin@chs-nxr-123-mn003.bi.services.us-south.bluemix.net",
    "webhdfs": "https://chs-nxr-123-mn001.bi.services.us-south.bluemix.net:8443/gateway/defau
    },

  "user": "clsadmin"
  }
},
```

For a JDBC connection to HBASE we need from Analytics Engine service credentials 3 parameters:

The jdbcurl in this sample DB2 service credentials is:

```
"jdbcurl": "jdbc:phoenix:thin:url=https://chs-nxr-123-mn001.bi.services.us-south.bluemix.net:
```

Copy "jdbcurl" "user" and "password" from IBM Analytics Engine service credential and put it in the following spl file

2 - Create a Streams application

Create a new project **JDBCHbase** in your workspace:

```
~/workspace/JDBCHbase/application/JDBCHbase.spl
~/workspace/JDBCHbase/Makefile
```

JDBCHbase.spl

```
// ****************************************************************************
// * Copyright (C) 2018 International Business Machines Corporation
// * All Rights Reserved
// ****************************************************************************
// JDBCRUN_HBASE.spl demonstrates how to connect to a HBASe database
// on IBM Analytics Engine via streams toolkit com.ibm.streamsx.jdbc
// and Apache Phoenix JDBC
// It demonstrates also how to:
// create a table
// insert data into table
// select data from table
// and drop a table
// via JDBCRun operator.
//
// To connect to database, the following parameters need to be specified:
// jdbcDriverLib : the jdbc driver library (download the jdbc driver and store it in opt fold
// e.g.   opt/db2jcc4.jar)
// jdbcClassName : the class name for jdbc driver (e.g. com.ibm.db2.jcc.DB2Driver)
// jdbcUrl       : This parameter specifies the database url that JDBC driver uses
// to connect to a database and it must have exactly one value of type rstring.
// The syntax of jdbc url is specified by database vendors.
// For example, jdbc:db2://<server>:<port>/<database>the database URL.
// and For Phoenix JDBC "jdbc:phoenix:thin:url=https://<database-host-nama>:<server:port>/<da
// jdbcUser      : the database user on whose behalf the connection is being made.
// jdbcPassword  : the database user's password.
//
// This sample runs with streams JDBC toolkit version 1.3 or higher
// https://github.com/IBMStreams/streamsx.jdbc
// and with phoenix jdbc driver.
// https://github.com/IBM-Cloud/IBM-Analytics-Engine/tree/master/jdbcsamples/TestPhoenix/lib
// or direct from https://phoenix.apache.org/index.html
```

```
// Download and copy the following jar libraries and copy them in opt directory
// phoenix-4.9.0-HBase-1.1-client.jar
// phoenix-queryserver-4.9.0-HBase-1.1.jar
// phoenix-tracing-webapp-4.9.0-HBase-1.1.jar
// phoenix-4.9.0-HBase-1.1-queryserver.jar
// phoenix-queryserver-client-4.9.0-HBase-1.1.jar
//
// ************************************************************************

namespace application;
use com.ibm.streamsx.jdbc::* ;
use com.ibm.streamsx.jdbc.types::* ;

composite JDBCHbase
{
  param
expression<rstring> $jdbcDriverLib: getSubmissionTimeValue("jdbcDriverLib", "opt/phoenix-core
expression<rstring> $jdbcClassName: getSubmissionTimeValue("jdbcClassName", "org.apache.phoen
expression<rstring> $jdbcUrl :getSubmissionTimeValue("jdbcUrl", "jdbc:phoenix:thin:url=https
expression<rstring> $jdbcUser :getSubmissionTimeValue("jdbcUser", "clsadmin");
expression<rstring> $jdbcPassword : getSubmissionTimeValue("jdbcPassword", "IAEPASSWORD");


 expression<rstring> $createSql :"CREATE TABLE IF NOT EXISTS JDBCRUN_HBASE (id bigint ,m.fna
 expression<rstring> $selectSql :"SELECT * FROM JDBCRUN_HBASE";
expression<rstring> $dropSql :"DROP TABLE IF EXISTS JDBCRUN_HBASE";
expression<rstring> $insertSql :"UPSERT INTO JDBCRUN_HBASE VALUES( ";


  type
insertSchema  = int32 ID, rstring FNAME, rstring LNAME;
resultsSchema = int32 ID, rstring FNAME, rstring LNAME;

  graph


   /**
   * The pulse is a Beacon operator that generates counter.
   */
stream<int32 counter> pulse = Beacon() {
        logic
        state: mutable int32 i = 0;
param
initDelay   : 2.0;
iterations  : 11u ;
period:3.0;
output
pulse : counter = i++;
}

   /**
   * genStatement is Custom operator that generates sql statements.
   */
stream<rstring statement>  genStatement = Custom(pulse)
   {
        logic

state: mutable rstring stmt = "UPSERT INTO JDBCRUN_HBASE values( 0, 'FNAME0', 'LNAME0')";

onTuple pulse :
{
if ( counter == 0)
{
stmt = $dropSql;
}
else if ( counter == 1)
{
```

```
stmt = $createSql;
}
else if ( counter == 10)
{
stmt = $selectSql;
}
else
{
stmt = $insertSql + (rstring)counter + ", 'FNAME" + (rstring)counter + "' , 'LNAME"  + (rstri
}

printStringLn( (rstring)counter + "  SQL Statement = " + stmt ) ;
submit({ statement = stmt }, genStatement) ;
}
}


   /**
   * runSql is a JDBCRun operator.
   * It runs a user-defined SQL statement that is based on an input tuple.
   */
stream<insertSchema> runSql = JDBCRun(genStatement)
{
param
jdbcDriverLib: $jdbcDriverLib ;
jdbcClassName: $jdbcClassName ;
jdbcUrl: $jdbcUrl ;
jdbcUser: $jdbcUser ;
jdbcPassword: $jdbcPassword;
statementAttr: statement ;
sqlFailureAction : "log";
}

   /**
   * printResults is a Custom operator that prints the sql results.
   */
() as printResults = Custom(runSql)
{
  logic
   onTuple runSql :
  {
if(ID > 0)
{
printStringLn((rstring) ID + "," + FNAME + "," + LNAME);
}
}
}


}
```

4 - Make the SPL application

To create this SPL application the new version of JDBC toolkit (1.3.0) is required.

Download and copy the latest version of streamsx.jdbc in your workspace.

https://github.com/IBMStreams/streamsx.jdbc

The Makefile makes also the toolkit.

```
#######################################################################
# Copyright (C)2014, 2018 International Business Machines Corporation and
```

```
# others. All Rights Reserved.
#################################################################

.PHONY: all clean

//SPLC_FLAGS = -t $(STREAMS_INSTALL)/toolkits/com.ibm.streamsx.jdbc  --data-directory data
SPLC_FLAGS = -t ../streamsx.jdbc/com.ibm.streamsx.jdbc  --data-directory data

SPLC = $(STREAMS_INSTALL)/bin/sc
SPL_CMD_ARGS ?=
SPL_COMP1NAME=JDBCHbase
SPL_MAIN_COMPOSITE1 = application::$(SPL_COMP1NAME)
BUILD_OUTPUT_DIR = output

all: clean
        $(SPLC) $(SPLC_FLAGS) -M  $(SPL_MAIN_COMPOSITE1) --output-dir ./$(BUILD_OUTPUT_DIR)

clean:
        $(SPLC) $(SPLC_FLAGS) -C -M $(SPL_MAIN_COMPOSITE1) --output-dir output
        -rm -rf toolkit.xml
```

Be aware of tabs in Makefile

5 - Run the SPL application

Change the database credentials in SPL file with your IBM HBASE database credentials and run

```
$> make
```

Start the application with

```
$> output/bin/standalone
```

Or you can submit the job on your local Streams server with:

```
$ streamtool submitjob output/application.JDBCHbase.sab
```

6 - Submit the spl application on IBM Streams Cloud

Create a Streaming Analytics on IBM Cloud

https://console.bluemix.net/catalog/?search=streams

Start the service

Lunch the application

It starts the IBM Streams console.

Now it is possible here to submit a SAB file as job

The SAB file is located in your project output directory:

```
output/application.JDBCHbase.sab
```

7 - Check the result on hbase server

To check the result login on IAE server and check the contain of table with hbase shell.

```
$> ssh clsadmin@<your-IAE-ip-address>
$ > hbase shell

 hbase(main):001:0> list
  JDBCRUN_HBASE
  SYSTEM.CATALOG
  SYSTEM.FUNCTION
  SYSTEM.SEQUENCE
  SYSTEM.STATS
 ...

 hbase(main):001:0> scan 'JDBCRUN_HBASE'
 ROW                                   COLUMN+CELL
 \x80\x00\x00\x00\x00\x00\x00\x02    column=M:FNAME, timestamp=1523608569941, value=FNAME2
 \x80\x00\x00\x00\x00\x00\x00\x02    column=M:LNAME, timestamp=1523608569941, value=LNAME2
 \x80\x00\x00\x00\x00\x00\x00\x02    column=M:_0, timestamp=1523608569941, value=x
 .....
```

For more details about the **hbase shell** check this link:

https://github.com/IBMStreams/streamsx.jdbc