How to connect to a PostgresSql database via JDBC toolkit

This document describes a step by step instruction to connect to a PostgresSql database via JDBC toolkit.

1 Download the PostgresSql jdbc driver

https://jdbc.postgresql.org/download/postgresql-42.1.3.jar create a opt directory in your SPL project directory copy the driver jar library **postgresql-42.1.3.jar** in to opt directory

2 - Create a database in your postgresql server

login as root

install postgresql and start service postgresql

```
yum install postgresql*

su - postgres
mkdir -p /var/lib/pgsql/data
initdb -D /var/lib/pgsql/data
```

Add or edit the following line in your /var/lib/pgsql/data/postgresql.conf :

```
listen_addresses = '*'
```

Add the following line as the first line of /var/lib/pgsql/data/pg_hba.conf. It allows access to all databases for all users with an encrypted password:

```
# TYPE DATABASE USER CIDR-ADDRESS  METHOD
host  all  all 0.0.0.0/0 md5
```

Change user o root and start the service postgresql and check the service status.

```
exit
service postgresql restart
service postgresql status -l
```

Create database and a test table.

```
su - postgres
-bash-4.1$ psql

postgres=# CREATE DATABASE STREAMS OWNER postgres;
CREATE DATABASE
postgres=# \c streams
```

```
postgres=# ALTER USER "postgres" WITH PASSWORD 'postpass';
ALTER ROLE
postgres=# CREATE TABLE streamsTable ( personid int, lastname varchar(255), firstname varcha
CREATE TABLE
postgres=# INSERT INTO streamsTable VALUES (1234, 'myName', 'myFistName', 'Berliner Street',
INSERT 0 1
postgres=# INSERT INTO streamsTable VALUES (2345, 'myName2', 'myFistName2', 'Berliner Street
INSERT 0 1
postgres=# select * from streamsTable;
personid | lastname |  firstname  |     address      |  city
----------+----------+-------------+------------------+--------
1234 | myName   | myFistName  | Berliner Street  | Berlin
2345 | myName2  | myFistName2 | Berliner Street2 | Berlin
(2 rows)
postgres=# \q
```

Check the port from streams server Der default port for postgres is 5432 for example

```
telnet 90.300.101.203 5432
Trying 90.300.101.203...
Connected to 90.300.101.203.
Escape character is '^]'.
```

3 - Create a SPL project in your Streams server

```
namespace application;
use com.ibm.streamsx.jdbc::* ;
// ************************************************************************
// JDBCPostgreSQL demonstrates how to connect to a PostgreSQL database and select data from a
// JDBCRun operator.
// Required Streams Version = 4.1.x.x
// Required JDBC Toolkit Version = 1.2.0
// https://github.com/IBMStreams/streamsx.jdbc
// PostgreSQL jdbc driver version 42 or higher
// https://jdbc.postgresql.org/download/postgresql-42.1.3.jar
// To connect to database, the following parameters need to be specified:
// jdbcDriverLib : the jdbc driver libraries (download the jdbc driver file from https://jdbc
// and store it in opt folder, e.g. opt/postgresql-42.1.3.jar )
// jdbcClassName : the class name for PostgreSQL jdbc driver (org.postgresql.Driver)
// jdbcUrl : the database URL. (e.g. jdbc:postgresql://<your postgresql IP address>/<your dat
// dbcUser :  the database user on whose behalf the connection is being made.
// jdbcPassword : the user's password.
// transactionSize : 2
// set the transactionSize > 1 for postgresql database
// In the SPL sample:
// "select" operator demonstrates how to run SQL statement from stream attribute via statemen
// In this sample the JDBCRun operator connect to the database and read all rows from test ta
// write them into data/output.txt
//
// ************************************************************************/
composite JDBCPostgreSQL
{
  param

expression<rstring> $jdbcDriverLib : getSubmissionTimeValue("jdbcDriverLib", "opt/postgresql-
expression<rstring> $jdbcClassName : getSubmissionTimeValue("jdbcClassName", "org.postgresql.
expression<rstring> $jdbcUrl : getSubmissionTimeValue("jdbcUrl", "jdbc:postgresql://192.168.2
expression<rstring> $jdbcUser : getSubmissionTimeValue("jdbcUser", "postgres");
expression<rstring> $jdbcPassword : getSubmissionTimeValue("jdbcPassword", "postpass");
expression<rstring> $statement : getSubmissionTimeValue("statement", "SELECT * from streamsTa

type
            // the postgres database deliver the select results with small capital letters
```

```
            // lastname is correct and not LastName
resultSchema = int32 personid,
rstring lastname,
rstring firstname,
rstring address,
rstring city;

graph

        stream<rstring sql> pulse = Beacon() {
param
iterations : 1u ;
initDelay : 2.0;
output
pulse : sql = "SELECT personid, lastname, firstname, address, city FROM streamsTable";

}

stream<resultSchema> select = JDBCRun(pulse){
param
jdbcDriverLib: $jdbcDriverLib;
jdbcClassName: $jdbcClassName;
jdbcUrl: $jdbcUrl;
jdbcUser: $jdbcUser;
jdbcPassword: $jdbcPassword;
statement:  $statement;
                        //  statementAttr:    sql;
                        // it is possible to get the select statement via input port
                        // or put it direct in statement operator
transactionSize : 2;
}

() as SelectSink = FileSink(select)
{
logic
state :
{
mutable int64 counter = 0;
}

onTuple select :
{
printStringLn((rstring)personid + "," + lastname + "," + firstname + "," + address + "," + ci
}
  
param
      file: "output.txt";
format: csv ;
flush: 1u;// flush the output file after 1 tuple
}

}
```

4 - Make the SPL application

create a Makefile

and run make

```
.PHONY: all distributed clean
JDBC_TOOLKIT_INSTALL = $(STREAMS_INSTALL)/toolkits/com.ibm.streamsx.jdbc
SPLC_FLAGS ?= -a
SPLC = $(STREAMS_INSTALL)/bin/sc
SPL_CMD_ARGS ?= -t $(JDBC_TOOLKIT_INSTALL)
```

```
 SPL_MAIN_COMPOSITE = application::JDBCPostgreSQL

all: distributed

distributed:
rm -rf output
JAVA_HOME=$(STREAMS_INSTALL)/java $(SPLC) $(SPLC_FLAGS) -M $(SPL_MAIN_COMPOSITE) $(SPL_CMD_AR

clean:
$(SPLC) $(SPLC_FLAGS) -C -M $(SPL_MAIN_COMPOSITE)
rm -rf output
```