Extras/Working with ADWC

Tutorial: Working with Oracle Autonomous Data Warehouse Cloud

This tutorial was built for BDC version 17.4.6 as part of the New Data Lake User Journey. Questions and feedback about the tutorial: david.bayard@oracle.com (mailto:david.bayard@oracle.com)

The Autonomous Data Warehouse Cloud provides an easy-to-use, fully autonomous database that scales elastically, delivers fast query performance and requires no database administration. It is a fully-managed data warehouse designed to support all standard SQL and business intelligence (BI) tools and deliver scalable analytic query performance. Find out more at the ADWC website (https://cloud.oracle.com/en_US/datawarehouse) or check out the product documentation. (https://docs.oracle.com/en/cloud/paas/autonomous-data-warehouse-cloud/index.html)

Note: This tutorial replaces the Working with Oracle Database tutorial, in that this tutorial goes the extra-step of setting up the 12.2 JDBC-OCI connectivity. You can use this same connectivity to other non-ADWC Oracle databases. In other words, if you follow this tutorial, don't follow the steps in the Working with Oracle Database tutorial as that tutorial assumes you want to use the 12.1 thin JDBC driver (not the 12.2 JDBC-OCI driver).

About connecting to ADWC

Connections to the Autonomous Data Warehouse Cloud are made over the public Internet, and all applications must use a secure connection to the Autonomous Data Warehouse Cloud. If you are familiar with using an Oracle Database within your own data center, you may not have previously used these kind of secure SQL*Net connections, so some of the connection details may be new to you.

Connections to Autonomous Data Warehouse Cloud use certificate authentication and Secure Sockets Layer (SSL). This ensures that there is no unauthorized access to the Autonomous Data Warehouse Cloud and that communications between the client and server are fully encrypted and cannot be intercepted or altered. To enable these features, it requires that the client machine has a copy of a special set of files (known as the "Client Credentials" or the "wallet") that are configured for your ADWC instance.

ADWC connections can be made with all types of SQL*Net connections: Oracle Client Interface (OCI), JDBC Thin, and JDBC Thick (JDBC-OCI). For this tutorial, we will use the JDBC Thick (JDBC-OCI) type of connection as JDBC-OCI makes working with wallets easier than using JDBC Thin.

To use JDBC-OCI with BDC, here are the high-level steps we need to follow:

- Copy the Client Credentials zip file from your ADWC instance to BDC
- Install the Oracle Instant Client (the 12.2 version is pre-installed, so no need to do anything here)
- Configure the YARN and Spark environments to work with Oracle Instant Client libraries
- Configure the Zeppelin environment to use the 12.2 Oracle JDBC driver for the JDBC interpreter and Spark interpreter

Learn more about ADWC Connectivity in the product documentation (https://docs.oracle.com/en/cloud/paas/autonomous-data-warehouse-cloud/user/connect-data-warehouse.html).

Copy the Client Credentials zip file from your ADWC instance to BDC

Download the Client Credentials zip file from your ADWC instance

To download client credentials, do the following:

- Navigate to the Service Console for Autonomous Data Warehouse Cloud.
- Choose Administration.
- On the Administration page Choose Download Client Credentials.
- On the Client Credentials dialog, enter a wallet password and confirm the password.
- Click Download to save the client security credentials zip file.

Upload the Client Credentials zip file to your BDC instance

For simplicity, we will upload the Client Credentials using the BDC Console. Follow these steps:

- Navigate to Data Stores in the Big Data Cloud console
- Click on HDFS
- Click on the tmp folder
- Click on the Upload button
- Select the Client Credentials zip file (it will likely be named something like wallet_UNIQUEID_DBNAME.zip) and upload it

At this point, you should see a wallet*.zip file in the HDFS /tmp directory.

Move the Client Credentials zip file from HDFS to the linux file system

Finally, we need to move the client credentials zip file to the linux file system and unzip it.

- Run the following paragraph to move and unzip the client credentials zip file

Script to move the Client Credentials zip to the linux file system

```
%sh
sudo mkdir /opt/oracle/dbconnector/tns_admin
sudo chown zeppelin /opt/oracle/dbconnector/tns_admin
cd /opt/oracle/dbconnector/tns_admin
hadoop fs -ls /tmp
hadoop fs -get "/tmp/wallet*.zip" wallet.zip
unzip -u wallet.zip

#update the wallet directory path in the sqlnet.ora
cp sqlnet.ora sqlnet.ora.save
cat <<EOF > sqlnet.ora
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA = (DIRECTORY="/opt/oracle/dbconnector/tns_admin")))
SSL_SERVER_DN_MATCH=yes
EOF
```

Install the Oracle Instant Client (already installed, just need to test it)

With BDC, the Oracle Instant Client should already be installed. You should see it at /opt/oracle/dbconnector/instantclient. As of 17.4.6, the Oracle Instant Client version is 12.2.0.1.

You can run the next paragraph to test connectivity with SQL*Plus.

Script to test ADWC Connectivity with SQL*Plus (you will need to edit the username/password and tns alias)

```
%sh
export TNS_ADMIN=/opt/oracle/dbconnector/tns_admin
export LD_LIBRARY_PATH=/opt/oracle/dbconnector/instantclient:$LD_LIBRARY_PATH
export PATH=/opt/oracle/dbconnector/instantclient:$PATH


sqlplus 'sh/Welcome1!@FA539F1A563D4_DBAYARDDW_medium.dwcs.oracle.com'
```

```
SQL*Plus: Release 12.2.0.1.0 Production on Fri Jan 5 15:42:35 2018
Copyright (c) 1982, 2016, Oracle.  All rights reserved.
Last Successful login time: Thu Jan 04 2018 21:30:30 +00:00
Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
```

Configure the YARN and Spark environments to work with the Oracle Instant Client

First, connect to Ambari (see the Extra tutorial on Connecting to Ambari if you need help). Then while in Ambari, do the following:

Setup the YARN environment

- In Ambari, click on YARN.
- Then click on Configs
- Then click on Advanced
- Then expand Advanced yarn-env
- Then add these 2 lines to the top of the yarn-env template:

  export TNS_ADMIN=/opt/oracle/dbconnector/tns_admin
  export LD_LIBRARY_PATH=/opt/oracle/dbconnector/instantclient:$LD_LIBRARY_PATH

- Then click Save
- Name the changes “oracle instant client” and click Save again

- If you get a warning message, click Proceed Anyways
- NOTE: you do not need to restart now

Setup the SPARK environment

- In Ambari, click on Spark2
- Click on Configs
- Expand Advanced spark2-defaults
- Edit the spark.driver.extraLibraryPath field and append this value:
  :/opt/oracle/dbconnector/instantclient/
- Then click Save
- Name the changes “oracle instant client” and click Save again
- (Optional) Expand Advanced spark2-env and edit spark_thrift_cmd_opts
  search for ojdbc. change to /u01/bdcsce/opt/oracle/dbconnector/instantclient/ojdbc8.jar
  NOTE ojdbc7 appears 3 times. You need to change all 3

Setup the Zeppelin environment

- In Ambari, click on Zeppelin Notebook
- Click on Configs
- Expand Advanced zeppelin-env
- Then add these 2 lines to the top of the zeppelin_env_content:

  export TNS_ADMIN=/opt/oracle/dbconnector/tns_admin
  export LD_LIBRARY_PATH=/opt/oracle/dbconnector/instantclient:$LD_LIBRARY_PATH
- Then click Save
- Name the changes “oracle instant client” and click Save again

Restart services

- In Ambari, click on the Actions button underneath the list of services
- Choose Restart All Required
- Then click on Confirm Restart All

Configure the Zeppelin JDBC Interpreter

Follow these steps:

- In the BDC console, click on Settings

- Then click on Notebook
- Scroll down to the JDBC interpreter section and click the Edit button
- In the properties section, add the following properties
  - adwc-sh.driver = oracle.jdbc.OracleDriver
  - adwc-sh.user = sh
  - adwc-sh.url = jdbc:oracle:oci8:@FA539F1A563D4_DBAYARDDW_high.dwcs.oracle.com
  - adwc-sh.password = YourPassword
  - Update values like “sh” with your database username. Update the last part of the url with your tns alias. Update password with your password.
- In the dependencies section, overwrite/replace the ojdbc7 dependency with this value:
  /opt/oracle/dbconnector/instantclient/ojdbc8.jar
- Click Save
- Then click OK to restart the JDBC interpreter

Script to patch Zeppelin to fix performance issue (ZEPPELIN-1962)

```
%sh
# There are some performance issues with Zeppelin 0.7.x jdbc interpreter and OracleDB due to a costly (slow) autocompletion feature.
# See https://issues.apache.org/jira/browse/ZEPPELIN-1962  (also being tracked as oracle bug 27037280)
# The workaround for Zeppelin is 0.7.x is to use a patched zeppelin jdbc interpreter jar file which disables autocompletion.

mkdir /tmp/zepjdbc
cd /tmp/zepjdbc
wget https://issues.apache.org/jira/secure/attachment/12874854/zeppelin-jdbc-0.7.2.jar
ls -l zepp*

echo "now putting files into place"
sudo mv /u01/bdcsce/usr/hdp/2.4.2.0-258/zeppelin-spark21/interpreter/jdbc/zeppelin-jdbc-0.7.0.2.6.0.3-8.jar /u01/bdcsce/usr/hdp/2.4.2.0-258/zeppelin-spark21/interpreter/jdbc/zeppelin-jdb
    .6.0.3-8.jar.orig
sudo cp /tmp/zepjdbc/zeppelin-jdbc-0.7.2.jar /u01/bdcsce/usr/hdp/2.4.2.0-258/zeppelin-spark21/interpreter/jdbc/zeppelin-jdbc-0.7.2.jar

hostname -f
echo "done"

# You need to restart the zeppelin jdbc interpreter, but the next paragraph will do that for us
```

Additional step to correct Password for JDBC Interpreter

This step is currently required because of our temporary fix to the ZEPPELIN-1962 issue. Long term, this step will not be needed.

- Copy the hostname of your BDC server. In the above paragraph, the hostname should be printed out on the next-to-last line of the output.
- Click on Settings
- Click on Notebook
- Scroll down to the JDBC Interpreter section
- Click on Edit
- Find the property adwc-sh.jceks.file
- Edit the value of the property so that instead of “hdfs” it now reads "hdfs@hostname" (mailto:hdfs@hostname"). For instance, here is an example after editing:

jceks://hdfs@dcbdec13-bdcsce-1.compute-gse00010212.oraclecloud.internal/user/zeppelin/interpreter-store.jks (jceks://hdfs@dcbdec13-bdcsce-1.compute-gse00010212.oraclecloud.internal/user/zeppelin/interpreter-store.jks)

- Click Save
- Click OK to restart

Script to test Zeppelin JDBC interpreter

```
%jdbc(adwc-sh)
select user from dual
```

Configure the Zeppelin Spark Interpeter

- In the BDC console, click on Settings
- Then click on Notebook
- Scroll down to the spark2 interpreter section and click the Edit button
- In the dependencies section, overwrite/replace the ojdbc7 dependency with this value: /opt/oracle/dbconnector/instantclient/ojdbc8.jar
- Click Save
- Then click OK to restart the JDBC interpreter

Script to ensure that 12.2 jdbc driver is used by Spark executors

```
%sh
sudo mv /opt/oracle/bdcsce/current/lib/ojdbc7.jar ~zeppelin/ojdbc7.jar.dontuse

#make a small permission fix so that spark can query the local file system for later in this tutorial
chmod a+rx /var/lib/zeppelin

echo "done"
```

Script to test Spark interpreter

```
%spark

// BEFORE RUNNING THIS, YOU WILL NEED TO EDIT THIS
//  1.Insert your database Connect String
//  2.Insert your database user name
//  3.Insert your database password


//define URL for the Oracle JDBC driver
println(">>>>>>>Defining url for Oracle JDBC")

val url="jdbc:oracle:oci8:@FA539F1A563D4_DBAYARDDW_high.dwcs.oracle.com"

//define the username and password as properties
println(">>>>>>>Defining Oracle JDBC username and password")
println(url)
val prop = new java.util.Properties
```

SparkSQL to test connection

```
%sql
select table_name, num_rows, last_analyzed from utables_sparksql
where num_rows > 0
order by num_rows desc
```

Using Spark to write to the Oracle Database

Now we will show a working example of writing back to the Oracle Database from Spark.

For this example, we will use Spark to read in some Citibike data and write that data into an Oracle table.

Spark Scala to read CSV and register as Spark SQL temporary table

```
%spark

//a previous tutorial placed the csv file into /var/lib/zeppelin/citibike/201612-citibike-tripdata.csv

val df = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").load("file:/var/lib/zeppelin/citibike/201612-citibike-tripdata.csv")

//cache the data frame for performance
df.cache()


println("Here is the schema detected from the CSV")
df.printSchema()
println("..")
```

Spark to write a DataFrame to an Oracle table

```
%spark



// To make sure we have Oracle friendly column names, lets select against our Spark SQL temp table and rename columns
println("Renaming column names via bike_query...")
val bike_query = sqlContext.sql("""select `Trip Duration` TRIPDURATION,
`Start Time` STARTTIME,
`Stop Time` STOPTIME,
`Start Station ID` STARTSTATIONID,
`Start Station Name` STARTSTATIONNAME,
`Start Station Latitude` STARTSTATIONLATITUDE,
`Start Station Longitude` STARTSTATIONLONGITUDE,
`End Station ID` ENDSTATIONID,
`End Station Name` ENDSTATIONNAME,
`End Station Latitude` ENDSTATIONLATITUDE,
`End Station Longitude` ENDSTATIONLONGITUDE,
`Bike ID` BIKEID,
`User Type` USERTYPE,
`Birth Year` BIRTHYEAR,
`Gender` GENDER
 from bike_trips_csvtemp""")

bike_query.show()
bike_query.printSchema()


import org.apache.spark.sql.SaveMode
//possible SaveModes are SaveMode.Append, SaveMode.Overwrite, SaveMode.ErrorIfExists, SaveMode.Ignore


println("Writing Spark DataFrame to Oracle Database.  This may take a few minutes.")
bike_query.write
   .mode(SaveMode.Overwrite)
   .jdbc(url,"CITIBIKE_ORCL",prop)


//your Spark dataframe needs to use valid Oracle column names (i.e. no spaces, no reserved words, etc).  If you need to rename dataframe fields, you can do operations like this
//val newdDF=oldDF.withColumnRenamed("Birth Year","BirthYear")


println("done")
```

JDBC to query Oracle and see the new Oracle table

```
%jdbc(adwc-sh)
select * from CITIBIKE_ORCL
```