AWS DMS Setup

How to setup AWS DMS:

This is assuming the PostgreSQL DB is already created

SETUP SCHEMA:

A. Get the sqlserver2pgsql.pl script from

https://github.com/dalibo/sqlserver2pgsql/blob/master/INSTALL.md

- We will not be using Kettle

B. Install perl

- http://strawberryperl.com/

C. In MSSQ Studio, right click on the database required to migrate and select Task -> Generate Scripts

- Under SQL Server Management Studio, Right click on the database you want to export
- Select Tasks/Generate Scripts
- Click "Next" on the welcome screen (if it hasn't already been deactivated)
- Select your database
- In the list of things you can export, just change "Script Indexes" from False to True, then "Next"
- Select Tables then "Next"
- Select the tables you want to export (or select all), then "Next"
- Script to file, choose a filename, then "Next"
- Select unicode encoding (who knows..., maybe someone has put accents in objects names, or in comments)
- You will receive a SQL script (in SQL Server syntax) with Db Structure (tables, indexes, constraints, stored procedures, etc)

D. In the terminal, go to where the sqlserver2pgsql.pl script is kept and run the following:
`~/sqlserver2pgsql.pl -f <THE NAME OF THE FILE DUMP THAT CAME FROM PART C> -b before.sql -a after.sql -u unsure.sql -keep_identifier_case -stringtype_unspecified`

- The before script is what you should run before data migration

    - There may be some errors that come up when the perl script reads the MSSQ Db SQL dump. Mostly it's just syntax that can be manipulated inside of the dump.
    - If this comes up as an error it can be removed: 'ALTER TABLE [dbo].[DbNodeClosures] SET (LOCK_ESCALATION = DISABLE)'
    - You will probably have errors with the create views, this is just a whitespace issue. Remove that and you'll be golden.
    - It will probably ignore functions and procedures... Keep that in mind, it may not be important for procedures, and important for functions (although if you've already enabled cdc on tables, you can ignore those ignored functions)
- The after script is what you should run after data migration
    - keep in mind that postgres has a max constraint name size of 63 characters, so currently these will need to be renamed manually. View the alterations.sql file to see what I did (I had 48 alterations to adjust)
- The unsure script is what you should run after data migration but needs review and will more than likely need quite a few edits
    - Often it doesn't have views, functions, and other things
    - View the 'updated-unsure.sql' file as an example as it is what was used for the test migration

E. Connect to the PostgreSQL DB and run the before script (this will build the tables, types, and columns)

- I recommend using pgAdmin4
    - https://www.pgadmin.org/download/pgadmin-4-windows/
    - If you get an error having it run (eternal loading screen) then go into the Registry Editor -> \HKEY_CLASSES_ROOT.js -> Change 'Content Type' to text/javascript

F. Migrate your data over as listed in the 'SETUP AND RUN AWS DMS' section

G. Make sure that everything is migrated over

H. Connect to the PostgreSQL DB and run the after and unsure scripts
- Do so after you've reviewed them to make sure they're accurate and the constraint names are 63 characters or less

I. Go play some Starcraft, you've earned it

SETUP AND RUN AWS DMS:

1. Create your Replication Instance on AWS DMS
    - Not needed anymore, can just use the one that's already built
    - Make sure that the Replication Instance's IP address is allowed to enter the VPC security group of both the Source enpoint's DB and the Target endpoint's DB, if not, add it to both
2. Create your endpoints (source and target)

- Test the connection with the Replication Instance before creating the endpoint (it will create even if the Replication instance can't connect to it)

3. Do everything here on the MSSQ DB
   - https://aws.amazon.com/premiumsupport/knowledge-center/dms-migrate-rds-sqlserver/
   - I know that it sucks to need to enable CDC for each table sooo just run the AutoWriter program if the tables are different then what's there, make sure you replace the strings in the file
   - I've noticed better results (in that the task would run without errors) by just keeping 'exec sys.sp_cdc_stop_job;' and not turning it back on, though keep in mind that you probably won't be able to get your logs, so turn it back on if you need those
   - You may get a table that fails, that's fine as it's probably a table that MSSQ made and isn't in the DB
     - Run these to get the tables if needed

```
For the dbo SCHEMA:
□□SELECT TABLE_NAME
□□FROM INFORMATION_SCHEMA.TABLES
□□WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_SCHEMA='dbo'
□□□
□For the workflow SCHEMA:
□□SELECT TABLE_NAME
□□FROM INFORMATION_SCHEMA.TABLES
□□WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_SCHEMA='workflow'
```

4. Create a DB migration task
   - Task Identifier: Something unique
   - Descriptive Amazon Resource Name (ARN): Skip this
   - Replication instance: Choose the Replication Instance from step 1
   - Select your Endpoints
   - Migration Type - I recommend 'Migrate existing data and replicate ongoing changes'
   - Editing mode: Wizard
   - Target table preparation mode: Do Nothing
   - Include LOB columns in replication: Full LOB mode
   - Maximum LOB size (KB): I set it to 10,000 but you can set it to whatever is needed
   - Enable validation: true
   - Enable CloudWatch logs: true
   - Advanced task settings
     - Create control table in target using schema: dms
     - Enable control table - enable all
   - Table mappings
     - Create two new selection rules and include SCHEMAs 'dbo' and 'workflow'
     - Create one new transformation rule with Target -> Schema, Schema name -> 'dbo', Action -> 'Rename to', Value -> 'public'

- Migration task startup configuration: Automatically start
- Save n start it

Revision #2
Created 18 July 2022 23:24:53 by Bryce Holloway
Updated 9 February 2023 20:20:49 by Bryce Holloway