SRMS Cloud Migration

Last Updated: April 25th 2022

Table of Contents

[Introduction 1](#_Toc100639976)

[Tools 1](#_Toc100639977)

[AWS Schema Conversion Tool 2](#_Toc100639978)

[DBeaver Import / Export Tool 6](#_Toc100639979)

[Amazon RDS and DB Snapshot 8](#_Toc100639980)

[AWS Glue 10](#_Toc100639981)

[Planning 13](#_Toc100639982)

[Database Objects to Migrate: 13](#_Toc100639983)

# Introduction

This document outlines the plan for the Cloud Migration of the SEC Registrant Mapping Service, introducing the tools, procedures and challenges of bringing the application to AWS. Please note that this is a work in progress and may undergo significant changes.

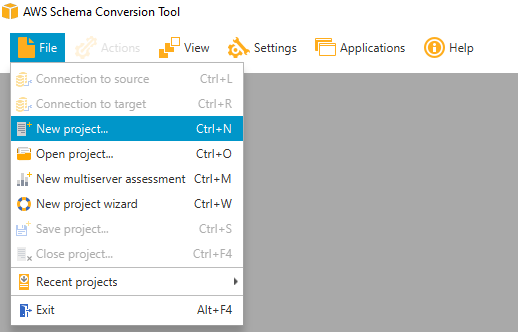
# Tools

This section will outline all the tools to aid in the migration process, they are:

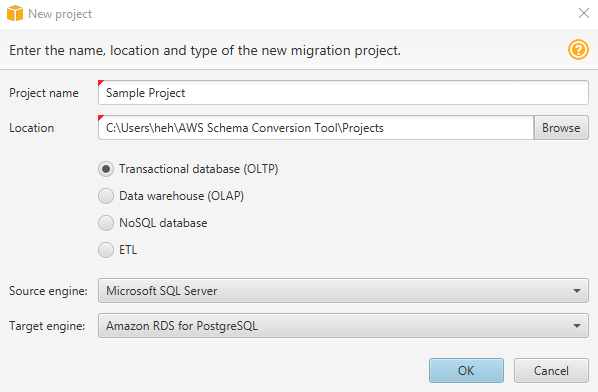
1. AWS Schema Conversion Tool
2. DBeaver Import / Export tool
3. AWS DB Snapshot
4. Amazon Glue

## AWS Schema Conversion Tool

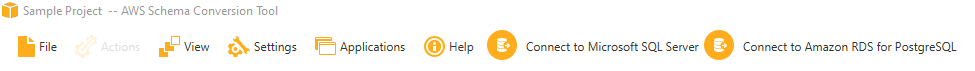
Currently, the data which SRMS relies on is stored in on premise MSSQL servers and plans are being made to transition to PostgreSQL server on the cloud. AWS offers its Schema Conversion Tool (SCT) for the purpose of heterogeneous database migration. Its installation and documentation can be found [here](https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Welcome.html). After the initial setup, a new project can be created as shown below:



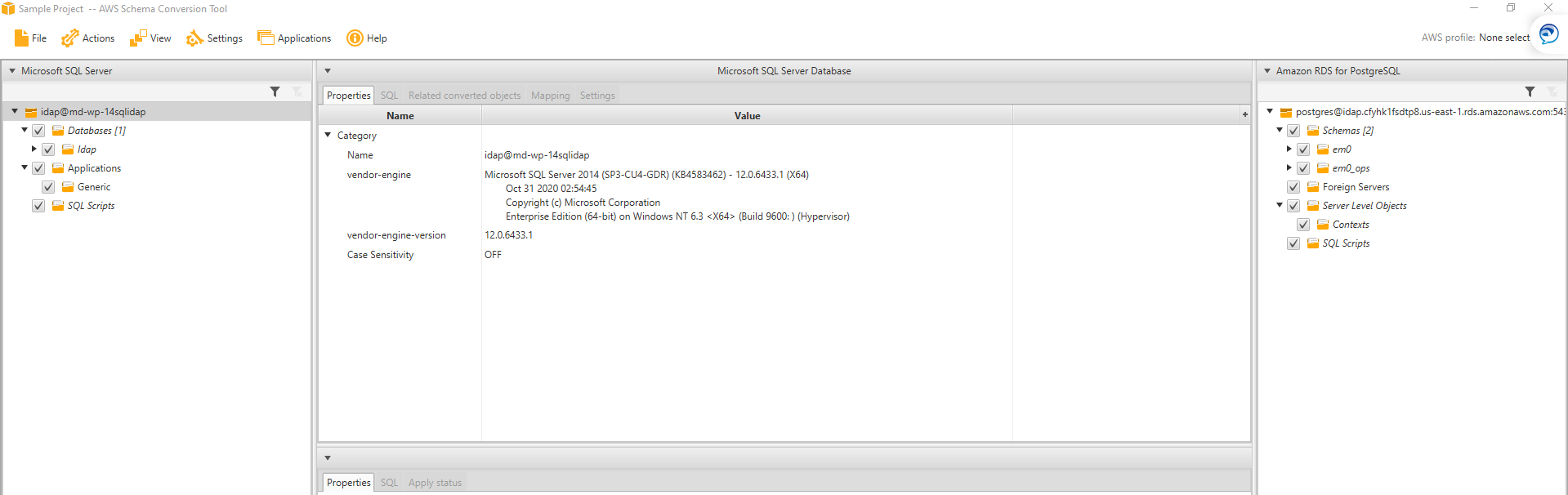
A window will appear – in the case of SRMS, the source engine is MSSQL while the target should be a RDS instance using PostgreSQL.



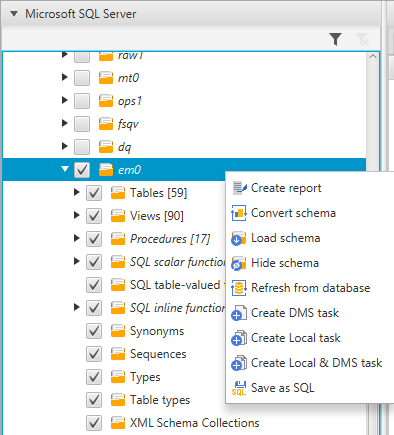
It will be necessary to enter the connection details of the source and target engines, which can be found on the far right of the tool bar.



After a successful connection, the SCT should be populated with more data / options.



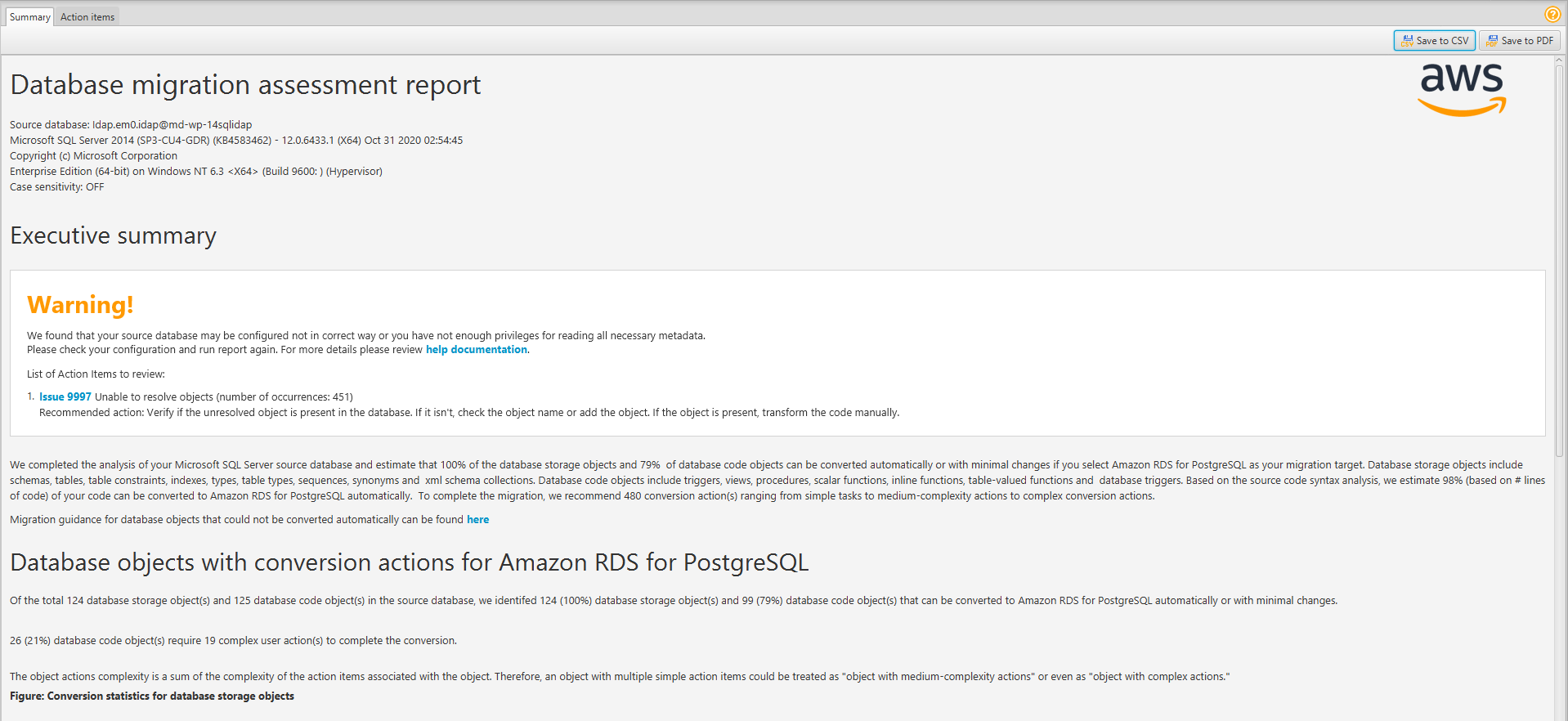
SRMS relies tables from the em0 and em0\_ops schema, so those will be loaded instead of the entire database for the sake of simplicity.

As shown in the screenshot on the left, it is possible to load only specific tables, views, and other database objects by checking them off.

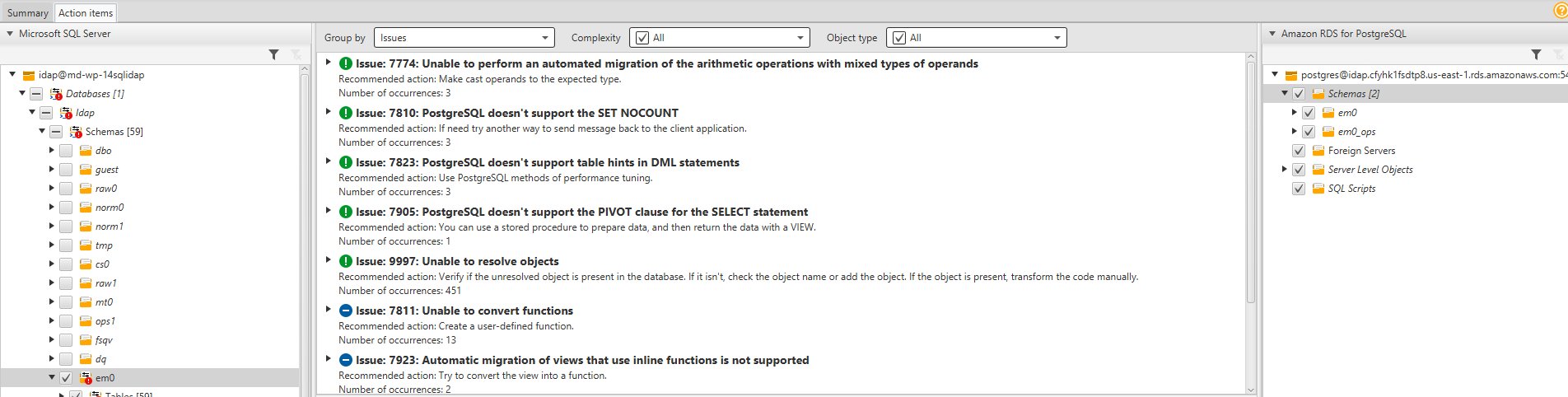
Right clicking on the em0 schema shows various options, but the two most important actions would be to Create report and to Convert Schema.

Before converting the schema, it’s worth creating a report to understand how the conversion would proceed and what issues is present. Fortunately, the report provides suggestions for the issues that are found.

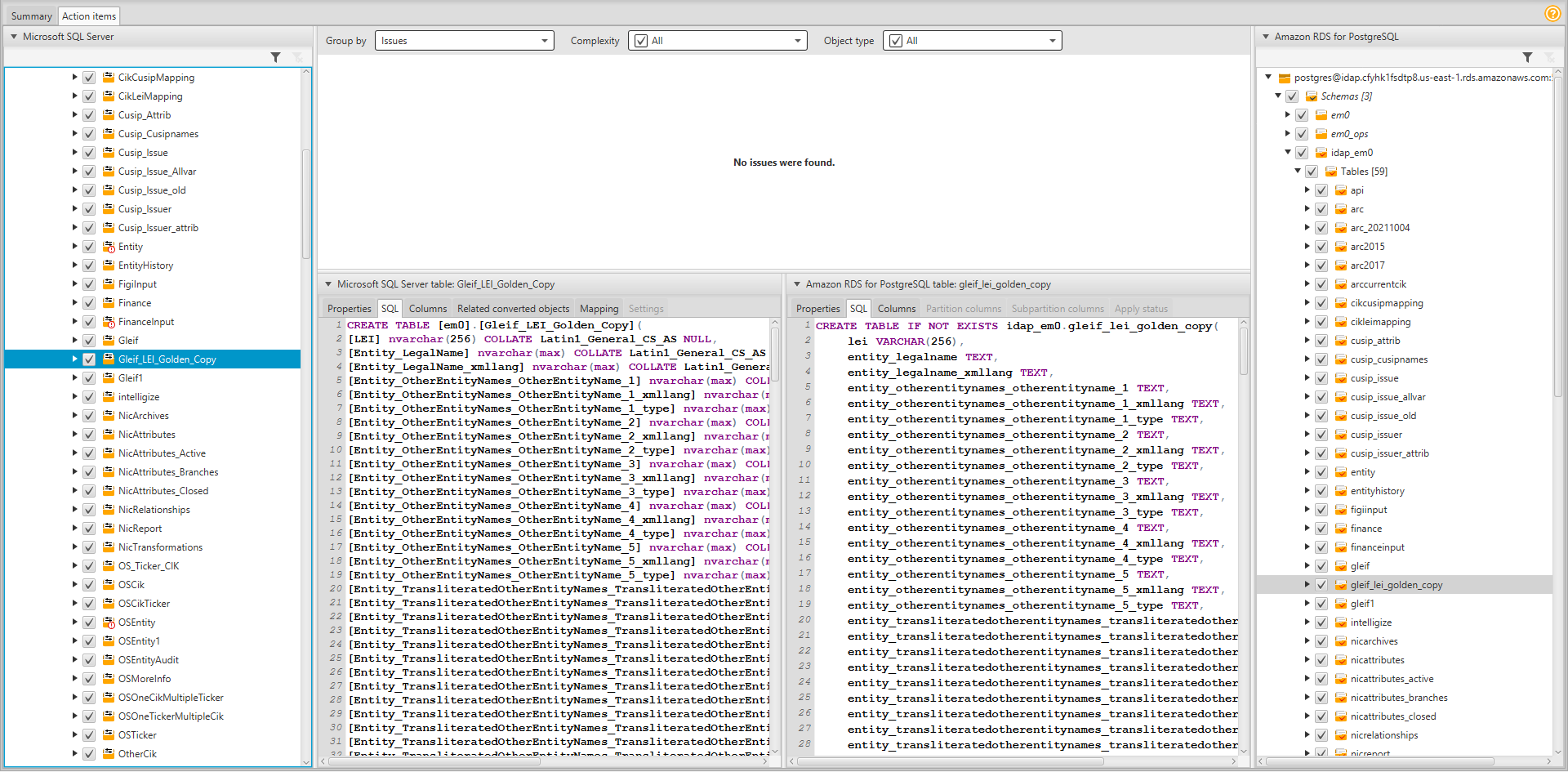
These reports can be exported as a csv or pdf for others to view.



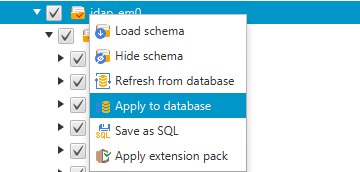
Navigating to the action items, a user can see the issues and their suggested actions.



Generally, the table can be converted with minimal issues but the SCT tool does struggle with some views, functions and procedures. After converting the schema, the tool will be populated with the SQL code for the target engine and issues (if any).



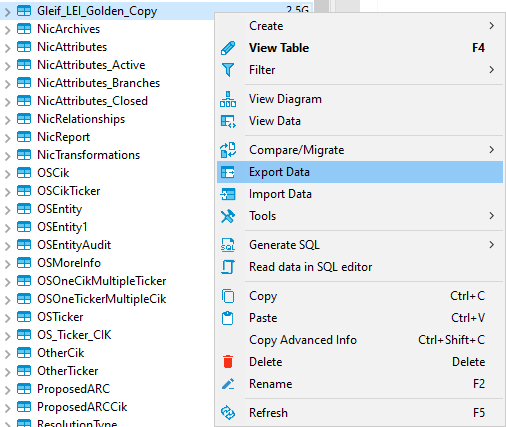
The converted schemas can be applied to the database after right-clicking on the database object on the right-side of the window.

While applying the entire schema to the database is the most straightforward approach, it may not be the best for SRMS. Applying only the objects used by SRMS might be a better approach.

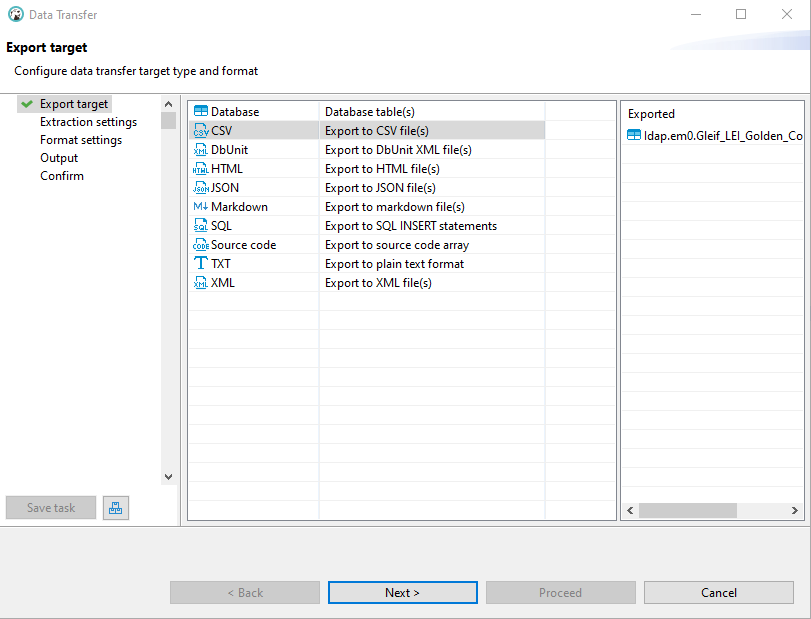
Through the SCT, it is possible to extract the SQL code to recreate a table from the on premise DB on a PostgreSQL instance…

## DBeaver Import / Export Tool

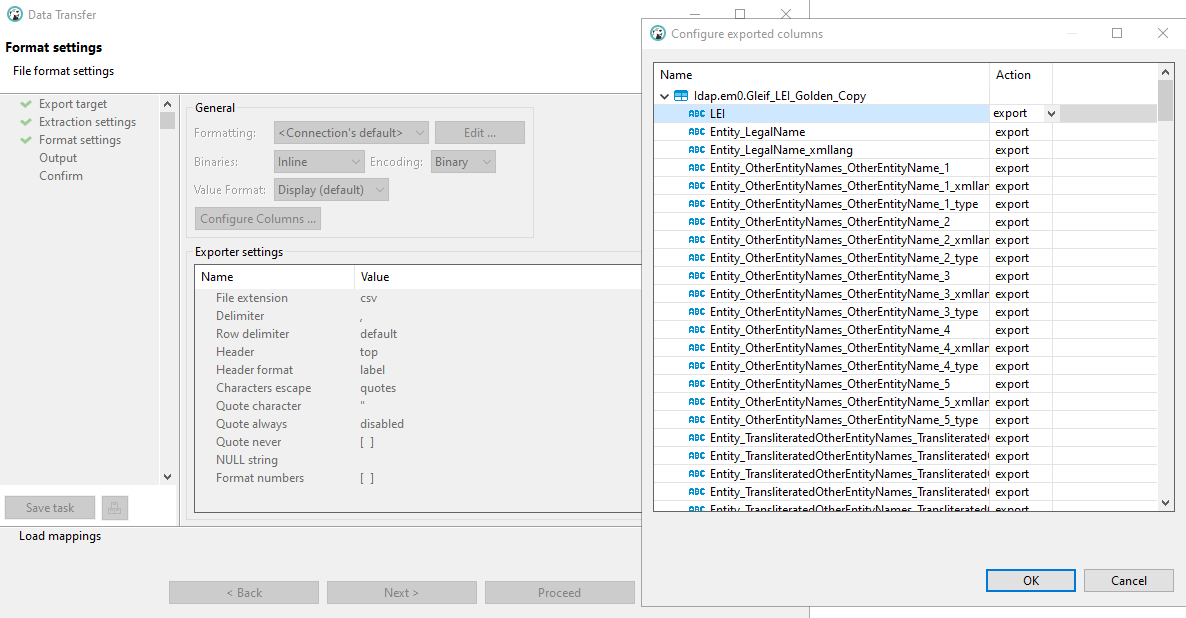
AWS SCT is intended to work in tandem with its Data Migration Service (or DMS), however it requires an internet gateway which is not allowed by the architecture which SRMS is transitioning to. A workaround is to use DBeaver; the software is a db admin tool that allows for connections among different databases. Right-clicking on any table or view allows for the Import Data or Export Data action.



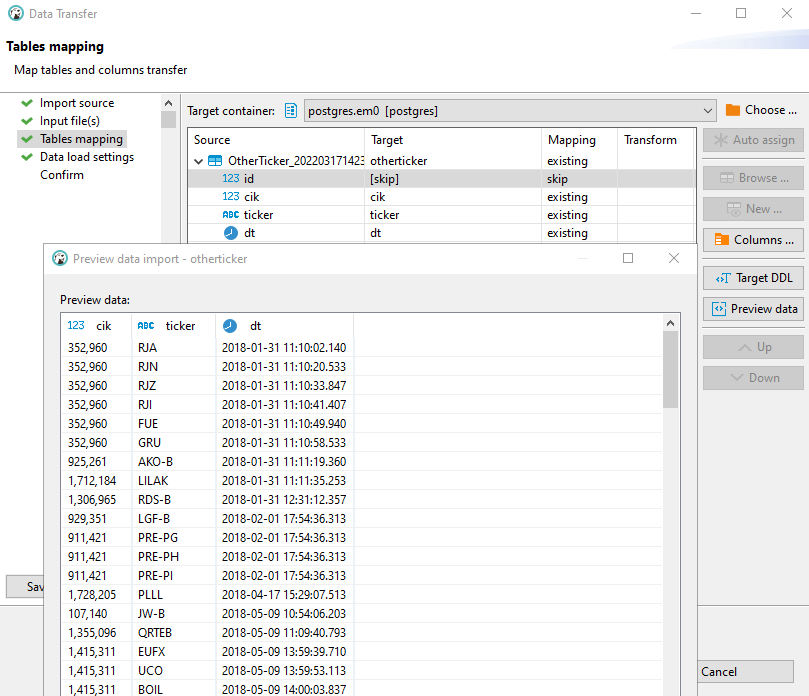
The export target that allows for data to be transferred from one database to another are CSV and SQL, with CSV being the fastest to export / import.



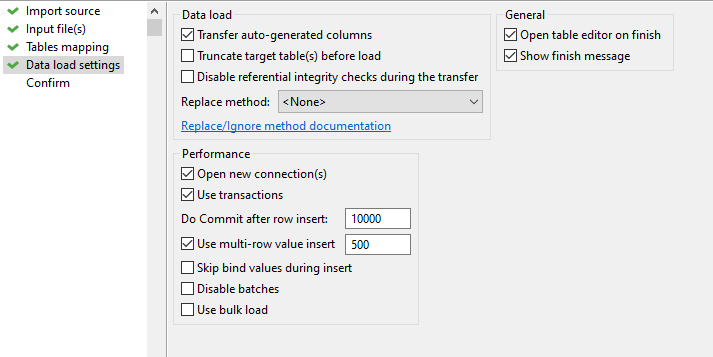
Various options can be configured for data export, but it’s worth noting that columns can be skipped. Some tables may contain autogenerated IDs that should be skipped, as they’ll be regenerated in whatever table they’re migrated to.



When importing data, DBeaver allows for only CSV or another table in the existing database as a source. If data was exported as a SQL script, it should be executed instead of using the import tool. After selecting the input CSV file, it’s possible to modify the mapping between the csv and table column or to skip it. It may be worth previewing the data as well.

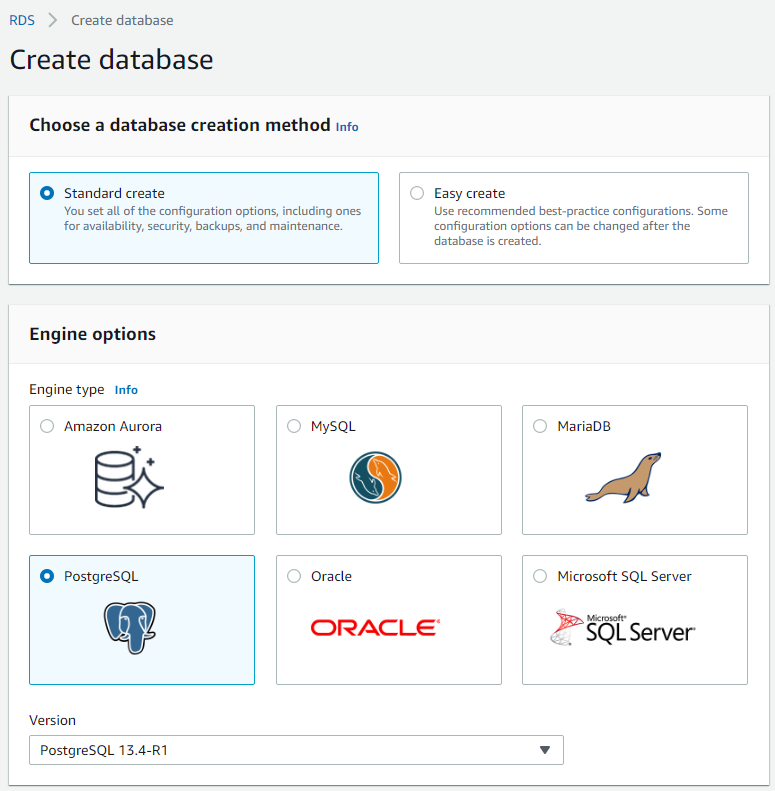


The most important settings would be under data load as its options would affect performance. The default settings is sufficient for most loads, but may require modification on larger data sets.



## Amazon RDS and DB Snapshot

For SRMS’ data to be on the cloud, a new PostgreSQL instance has to be created on Amazon RDS.

**There are certain configurations to use when using the standard create method.

A) Template should be set to Dev/Test

B) A master password should be provided

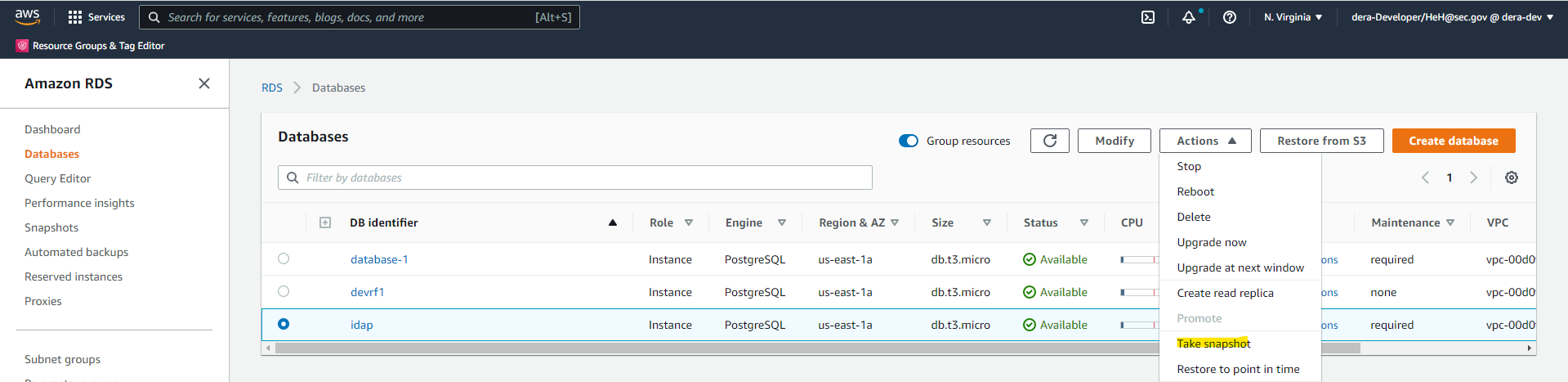
C) DB instance class should be set to Burstable classes, then under the associated dropdown, db.t3.micro should be selected

D) Expand the Additional Configurations tab and disable enhanced monitoring

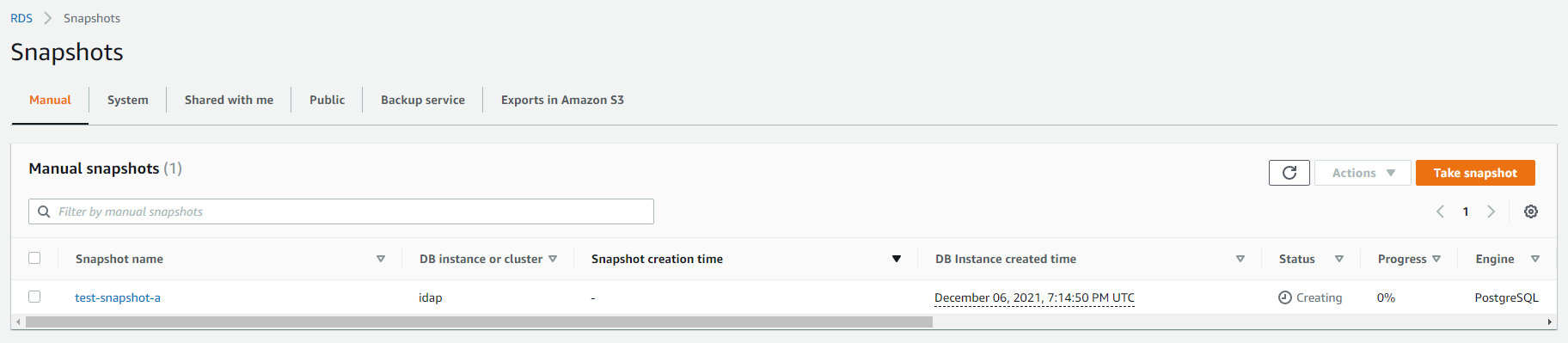
These settings are intended to get a development RDS instance up. In the beginning, existing data can be migrated over to this instance.

A snapshot can be created to capture the storage volume in its state and used to create a production ready instance.

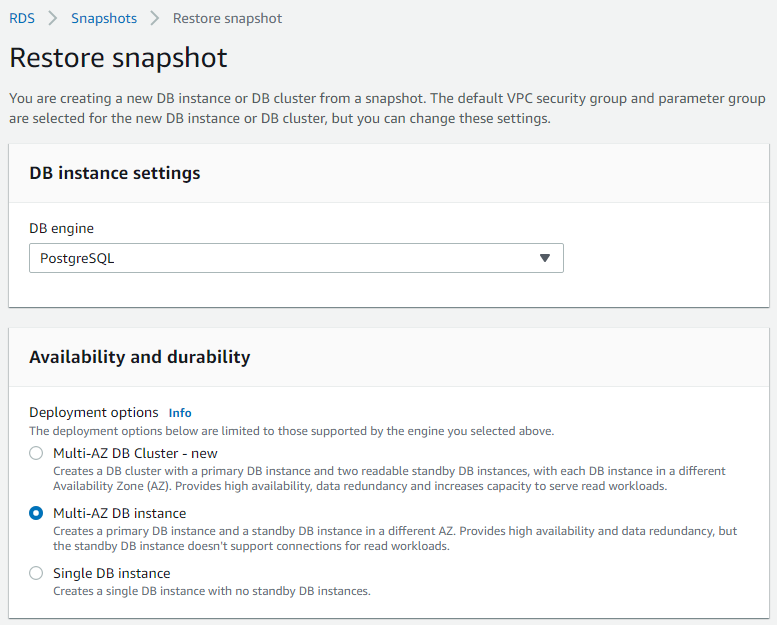
On the Amazon RDS console, one of the action is to Take snapshot after an instance is selected.



From there, the snapshot can be named and created. It’ll be available under the Snapshots page.

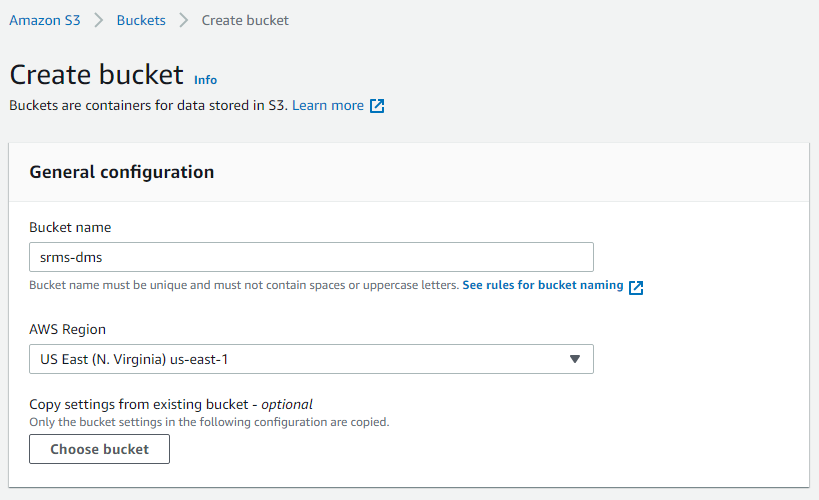


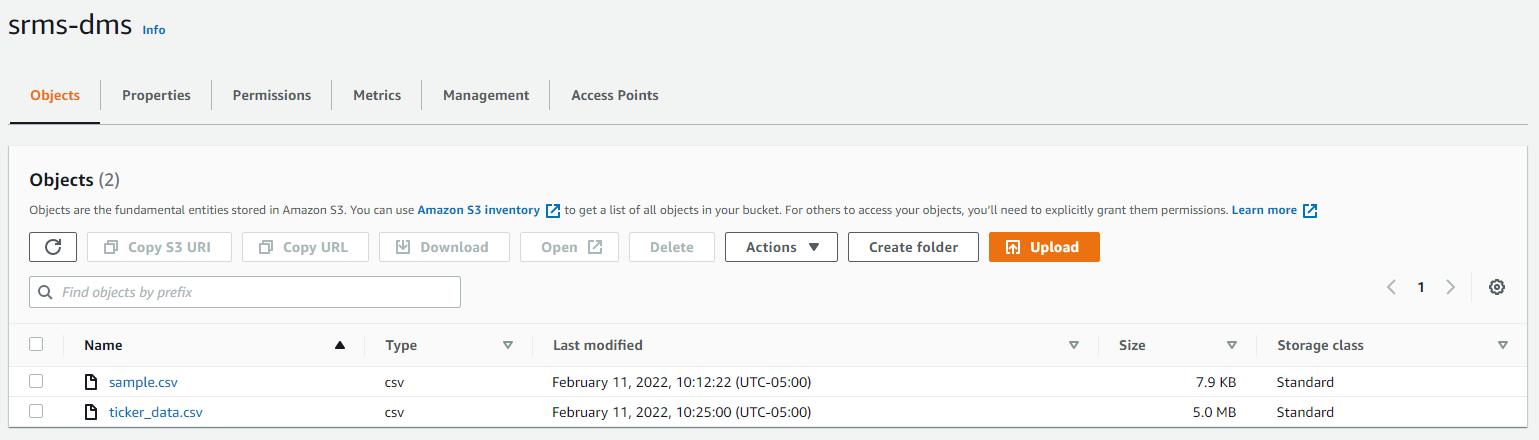
After the snapshot has finished its creation, restoring a snapshot becomes an option under Actions.

 It essentially creates a new DB instance with the state captured by the snapshot. This is an opportunity to reconfigure settings to align with production settings, but can be tweaked again on the RDS console using the Modify button.

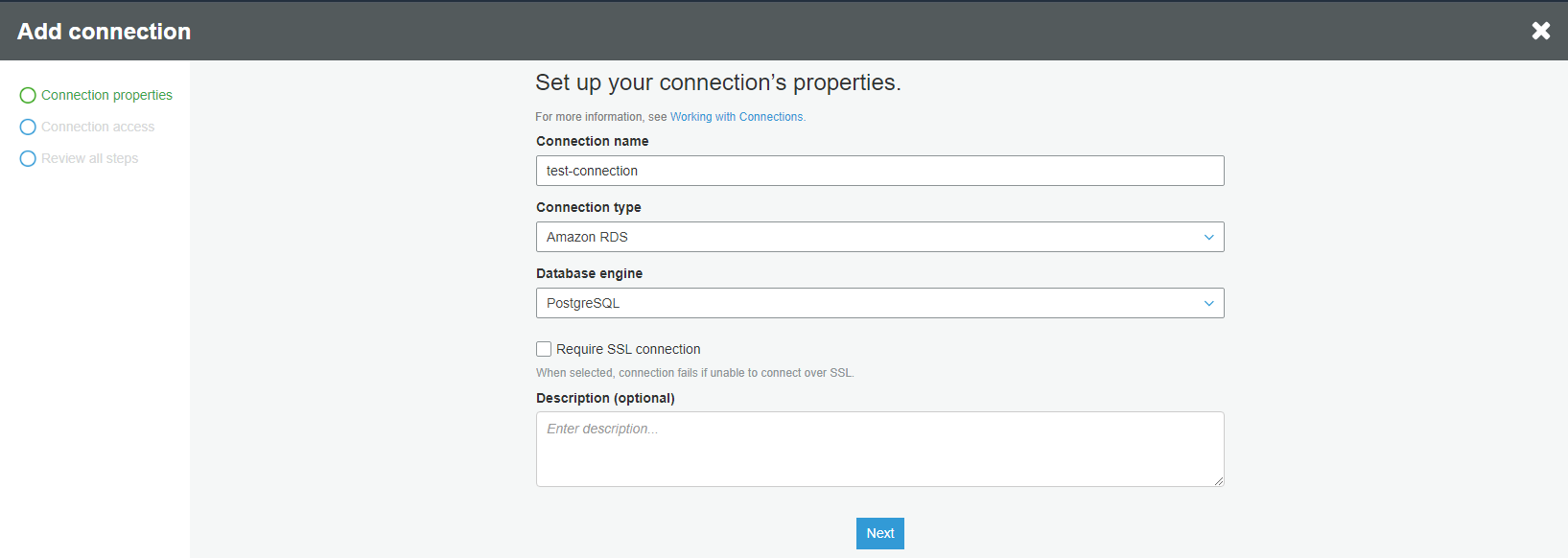
## AWS Glue

A number of tables in SRMS are updated at certain intervals through ETL procedures using the Pentaho client, but it may be necessary to transition to AWS Glue. It holds a number of benefits that could be found in this [url](https://docs.aws.amazon.com/glue/latest/dg/what-is-glue.html), the main advantage being that it’s integrated with many of Amazon’s available services. Since the ETL procedures will occur between a S3 bucket and PostgreSQL instance, some setup work is involved. The bucket will first need to be created to hold the data that will be extracted.

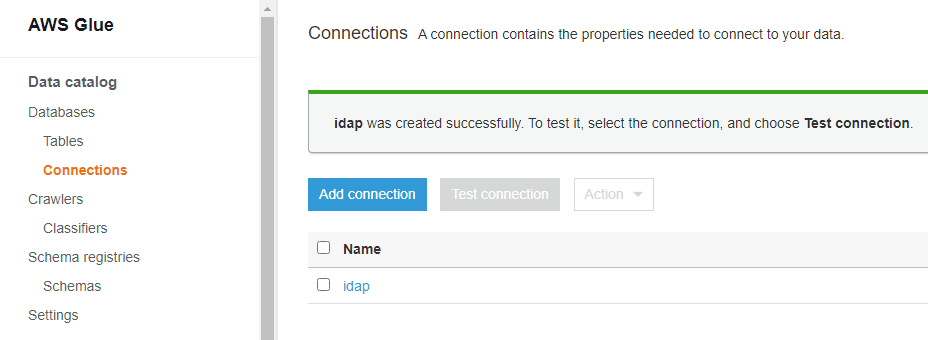




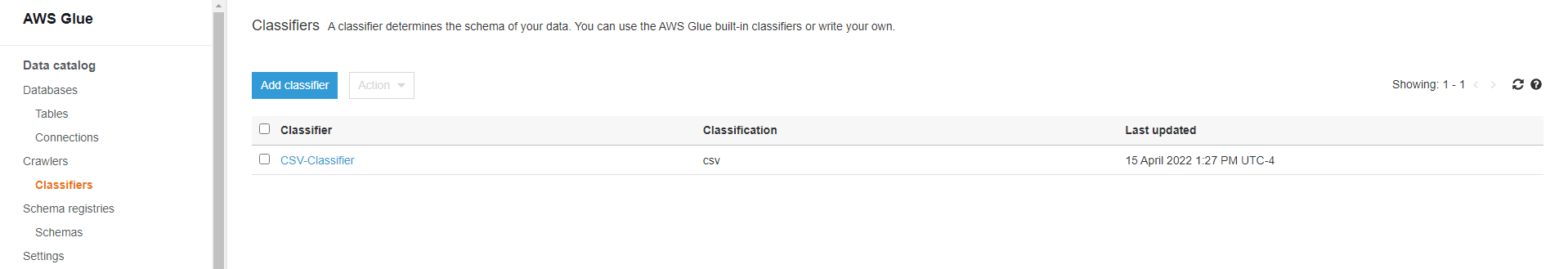
Afterwards, the database instance has to be added as a connection under the AWS Glue console.



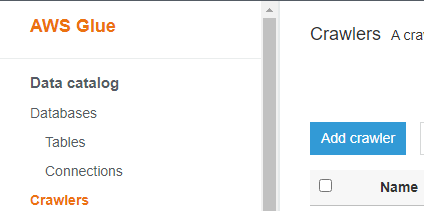
Once the connection is created, it’ll be available for testing.



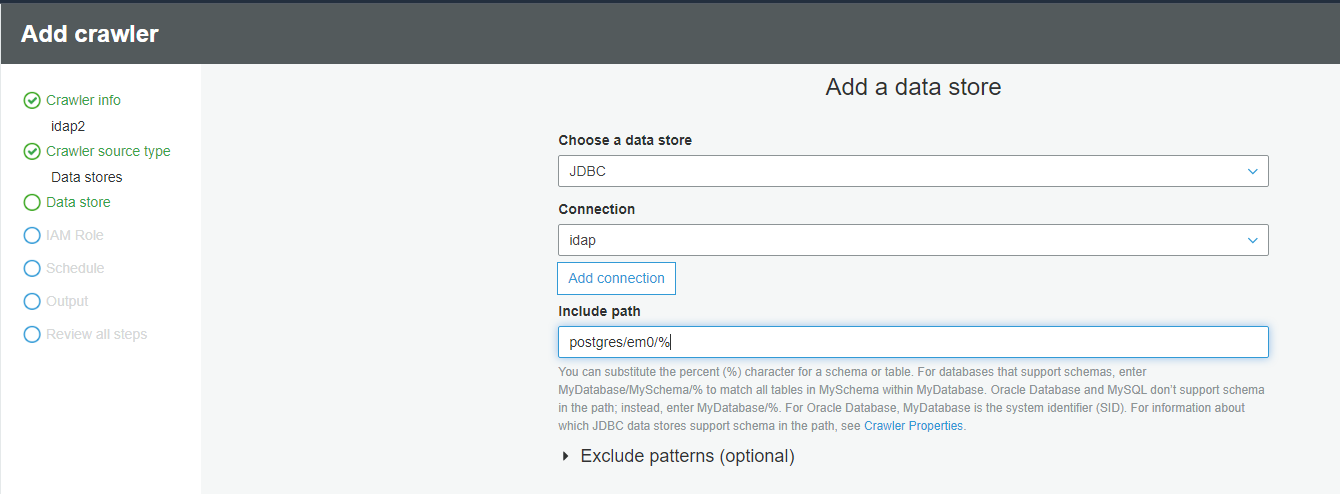
For CSV files, it’s worth creating a classifier to define the schema of our data and ensure that the data parsing is more consistent. In screenshot below, it is called CSV-Classifier.



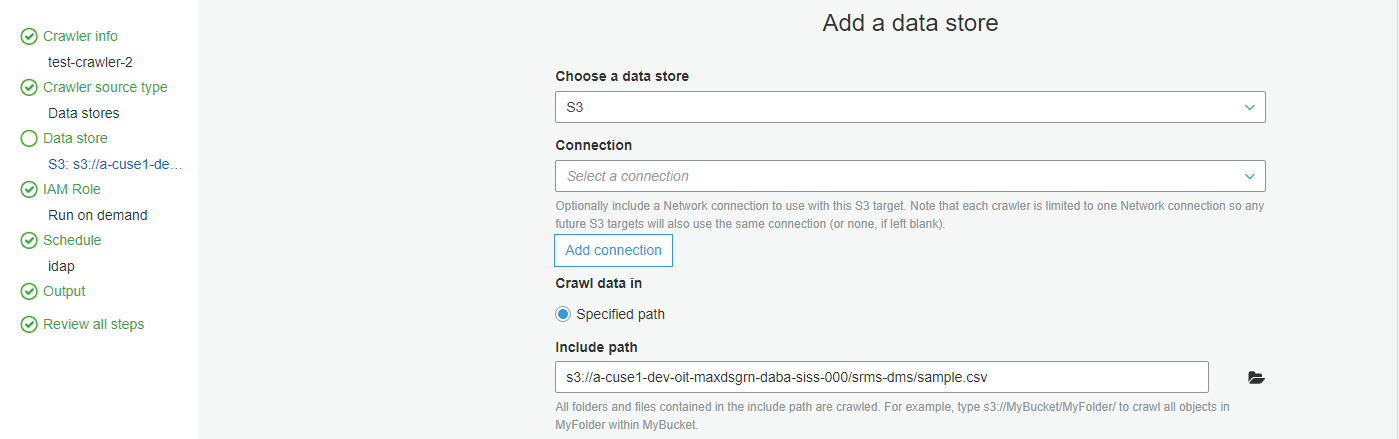
Most of the settings can be left alone, but it’s important to set the classifier type.

It’s necessary to create a crawler that is connected to our data store (PostgreSQL instance in this case) as it will create the schema for the data and then create the metadata tables for AWS Glue Studio. Select Crawlers on the left-hand side and then select Add Crawler.

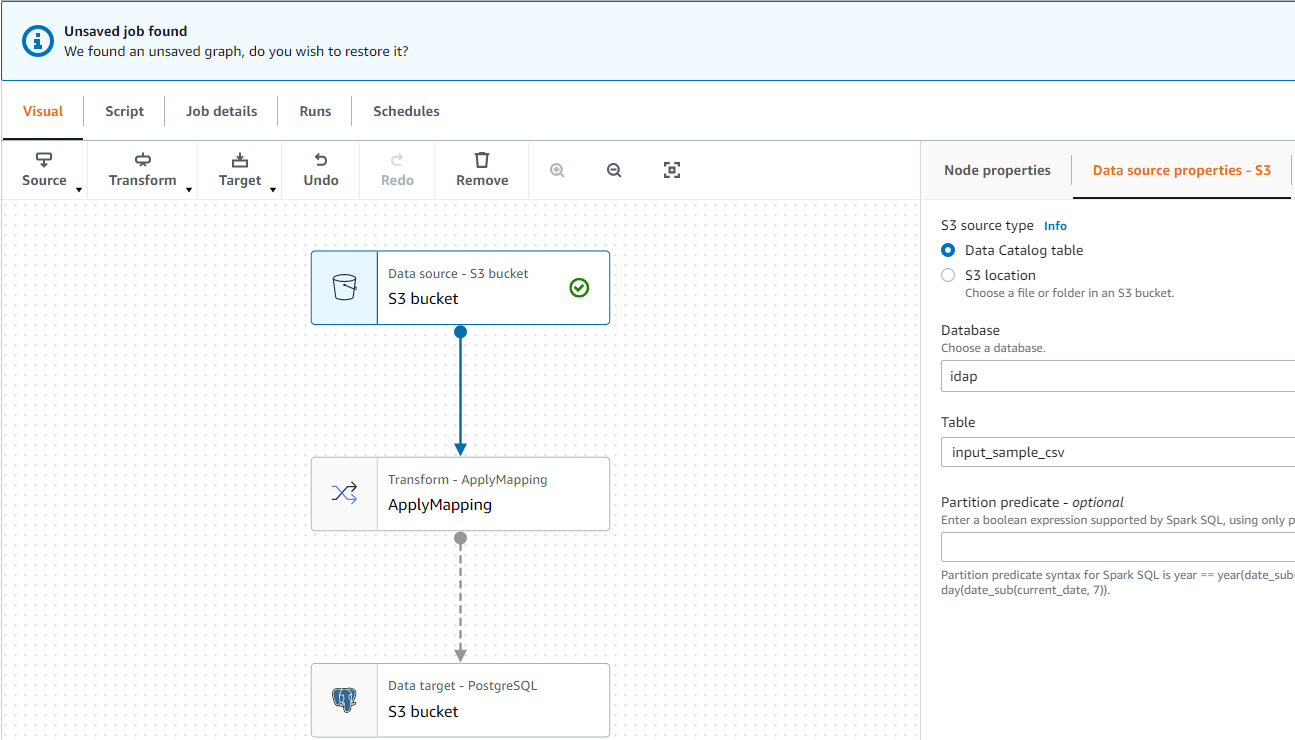
No options need to be changed until the Add Data Store section is reached. The data store should be set to JDBC, with the connection set to what was previously created. The path determines what tables the crawler will go through and the Exclude Patterns tab allows for certain tables to be ignored.



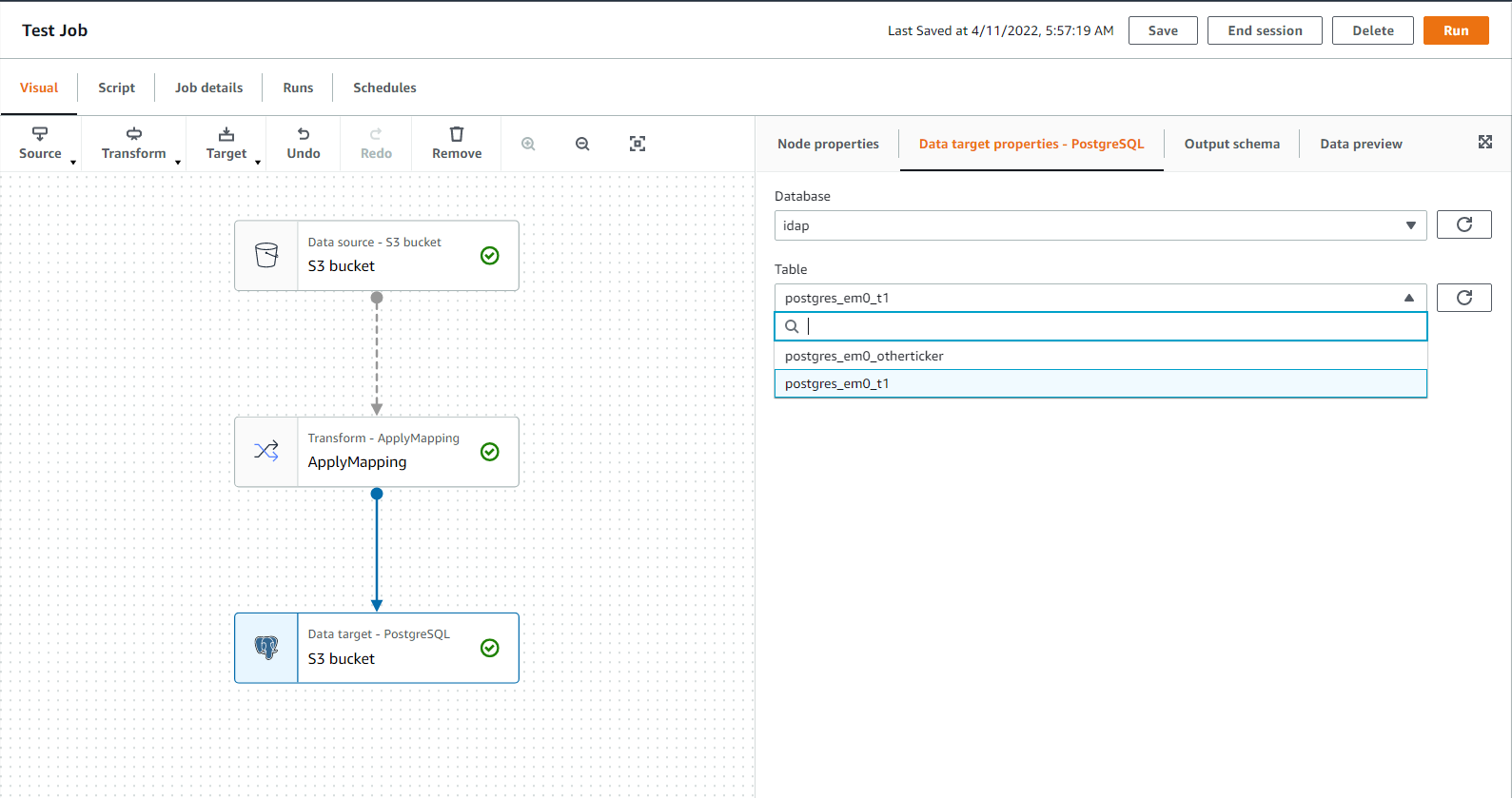
Another crawler has to be created, but this time with a data store that is a s3 bucket. The path should be pointing to the input file.



With the setup work complete, run both crawlers so the tables become available under node properties in AWS Glue Studio. The source will be a S3 bucket, usually containing json or csv files to extract.



Meanwhile the target should be the PostgreSQL instance.



# Planning

With all the tools introduced and available for use, the cloud migration can proceed as such:

1. Begin by converting the schemas from the on premise MSSQL server to the PostgreSQL instance on Amazon RDS
2. Migrate the data to the converted schemas using DBeaver’s import / export tool, transferring the existing data into CSV file or SQL scripts if necessary
3. Create a new instance using Amazon snapshot, and reconfigure its options to conform to production standards
4. For tables that need to have their data updated at a scheduled interval, AWS Glue can be used to run ETL procedures

There are a couple things to note during the migration process:

1. SRMS is reliant on market data supplied by the yahoo finance API, which is updated through a cron job from the idap-adm app. It may be necessary to have SRMS pull the market data from the on premise database, even after it is running on the cloud.
2. It isn’t strictly necessary to use AWS Glue for ETL procedures as the current jobs are being handled by a Pentaho client, which can be easily modified to load data to the PostgreSQL instance instead of the on premise database. Still, transitioning to AWS Glue is better in the long term if its permission issues are resolved.

# Database Objects to Migrate:

em0.saveTicker

em0.updateFpiCountry

em0.updatePEO

em0.updateFPI

em0.updateFYE

em0.updateAccounting

em0.SaveOtherTicker

em0.OtherTicker

em0.ViewCusip

raw0.Filer

em0.viewTickerReolutionMaster1

em0.viewTickerReolutionMaster

em0.ViewEntityHistory

em0.ViewYahooOTCMarketHistory\_tbl

em0\_ops.cik2LatestTicker1

em0.ViewTickerInputHistory

em0\_ops.ViewOpenFigi

em0.ViewTickerCikPublish

em0.ViewTickerCikPublish1

em0.ViewFinanceRefine

em0\_dev.ViewLatestFilerInfo

em0\_ops.ExceptionTickerMultipleCik

dbo.NormalizeSpace(title)

em0.ViewCompanyTickersJson

em0.ViewCompanyCIKTickersJson

em0.ViewMutualFundJson

em0.ViewCompanyCIKTickersJson

em0\_ops.getCountry

em0\_ops.getPEO

em0\_ops.getFYE

em0\_ops.Fpi

em0\_ops.getAccounting

em0\_ops.getFPI

em0\_ops.ViewISIN\_LEI\_Relationship

em0\_ops.BIC\_LEI\_Relationship

em0\_ops.Gleif\_Repex\_Golden\_Copy

em0\_ops.Gleif\_RR\_Golden\_Copy

em0.View\_Gleif\_OtherEntityNames

em0.View\_Gleif\_TransliteratedOtherEntityNames

em0.View\_Gleif\_TransliteratedOtherAddresses

em0.View\_Gleif\_Registration\_Details

em0.View\_Gleif\_Misc\_Details

em0\_ops.ViewBIC\_LEI\_Relationship

/\* sys.dm\_sql\_(referenced,referencing)\_entities replace older sp\_ functions \*/

/\* in this case looking for em\* objects that are not referenced by anything \*/

with u as (

select schemas.name+'.'+objects.name as uname,

objects.\* from sys.objects

inner join sys.schemas on schemas.schema\_id=objects.schema\_id

and type not in ('PK','UQ','D','F')

and schemas.name like 'em%'

)

select u.\* from u outer apply sys.dm\_sql\_referencing\_entities(uname,'OBJECT') r

where 1=1

and r.referencing\_id is null

and uname not in

('em0.saveTicker', 'em0.updateFpiCountry', 'em0.updatePEO', 'em0.updateFPI', 'em0.updateFYE', 'em0.updateAccounting', 'em0.SaveOtherTicker', '', 'em0.OtherTicker', 'em0.ViewCusip', 'raw0.Filer', 'em0.viewTickerReolutionMaster1', 'em0.viewTickerReolutionMaster', 'em0.ViewEntityHistory', 'em0.ViewYahooOTCMarketHistory\_tbl', 'em0\_ops.cik2LatestTicker1', 'em0.ViewTickerInputHistory', 'em0\_ops.ViewOpenFigi', 'em0.ViewTickerCikPublish', 'em0.ViewTickerCikPublish1', '', 'em0.ViewFinanceRefine', 'em0\_dev.ViewLatestFilerInfo', 'em0\_ops.ExceptionTickerMultipleCik', 'dbo.NormalizeSpace(title)', 'em0.ViewCompanyTickersJson', 'em0.ViewCompanyCIKTickersJson', 'em0.ViewMutualFundJson', 'em0.ViewCompanyCIKTickersJson', '', 'em0\_ops.getCountry', 'em0\_ops.getPEO', 'em0\_ops.getFYE', 'em0\_ops.Fpi', 'em0\_ops.getAccounting', 'em0\_ops.getFPI', 'em0\_ops.ViewISIN\_LEI\_Relationship', 'em0\_ops.BIC\_LEI\_Relationship', 'em0\_ops.Gleif\_Repex\_Golden\_Copy', 'em0\_ops.Gleif\_RR\_Golden\_Copy', 'em0.View\_Gleif\_OtherEntityNames', 'em0.View\_Gleif\_TransliteratedOtherEntityNames', 'em0.View\_Gleif\_TransliteratedOtherAddresses', 'em0.View\_Gleif\_Registration\_Details', 'em0.View\_Gleif\_Misc\_Details', 'em0\_ops.ViewBIC\_LEI\_Relationship')

order by modify\_date asc,create\_date asc

go