# AWS - Data Migration Service – Practical approach

### Introduction

I'm excited to write this article on AWS data migration with a practical approach. Before I delve into the project, I would like to explain a little bit about the need for the AWS DMS and its primary benefits followed by some tools required, dependencies and finally the project.

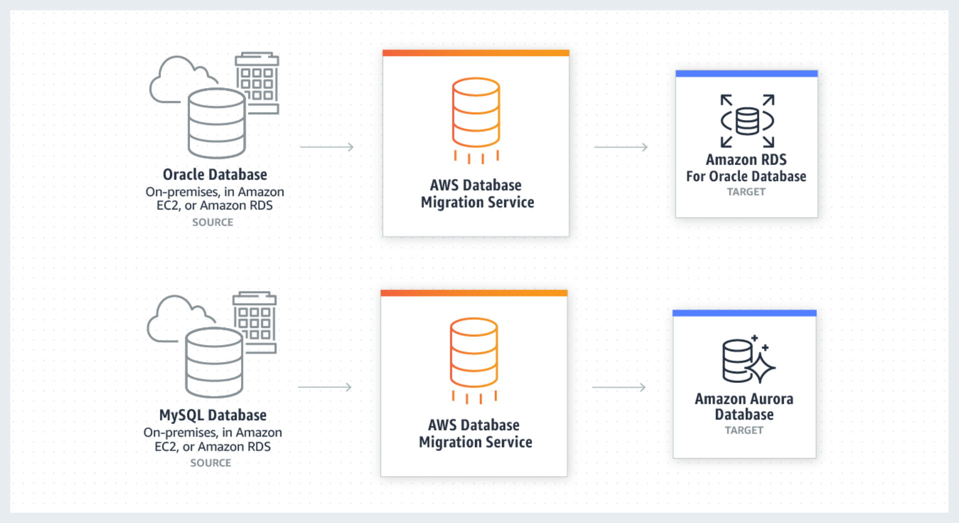
AWS Database Migration Service helps us migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. The AWS Database Migration Service can migrate the data to and from most widely used commercial and open-source databases.

AWS Database Migration Service supports homogenous migrations such as Oracle to Oracle, as well as heterogeneous migrations between different database platforms, such as Oracle or Microsoft SQL Server to Amazon Aurora. With AWS Database Migration Service, we can continuously replicate the data with high availability and consolidate databases into a petabyte-scale data warehouse by streaming data to Amazon Redshift and Amazon S3.

## Use Cases

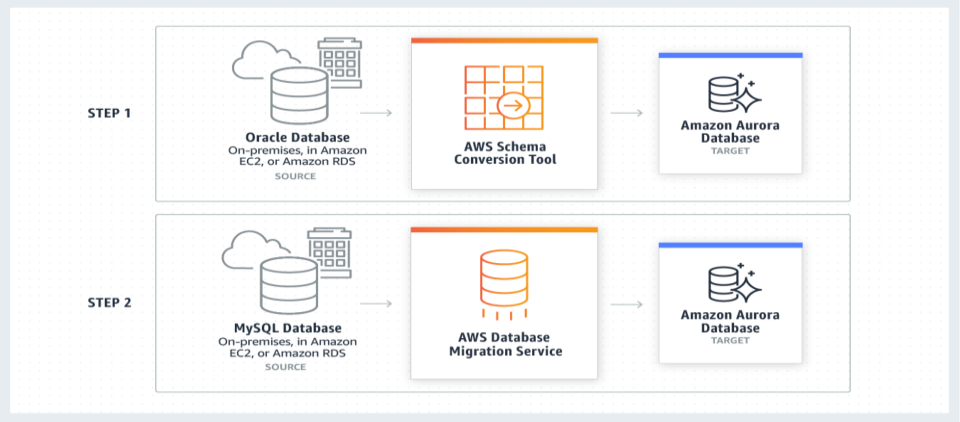
### HOMOGENEOUS DATABASE MIGRATIONS

In homogeneous database migrations, the source and target database engines are the same or are compatible like Oracle to Amazon RDS for Oracle, MySQL to Amazon Aurora, MySQL to Amazon RDS for MySQL, or Microsoft SQL Server to Amazon RDS for SQL Server. Since the schema structure, data types, and database code are compatible between the source and target databases, this kind of migration is a one step process. We create a migration task with connections to the source and target databases, then start the migration with the click of a button. AWS Database Migration Service takes care of the rest. The source database can be located in our own premises outside of AWS, running on an Amazon EC2 instance, or it can be an Amazon RDS database. The target can be a database in Amazon EC2 or Amazon RDS.


Oracle Database
On-premises, in Amazon EC2, or Amazon RDS
SOURCE
AWS Database Migration Service
Amazon RDS For Oracle Database
TARGET
MySQL Database
On-premises, in Amazon EC2, or Amazon RDS
SOURCE
AWS Database Migration Service
Amazon Aurora Database
TARGET

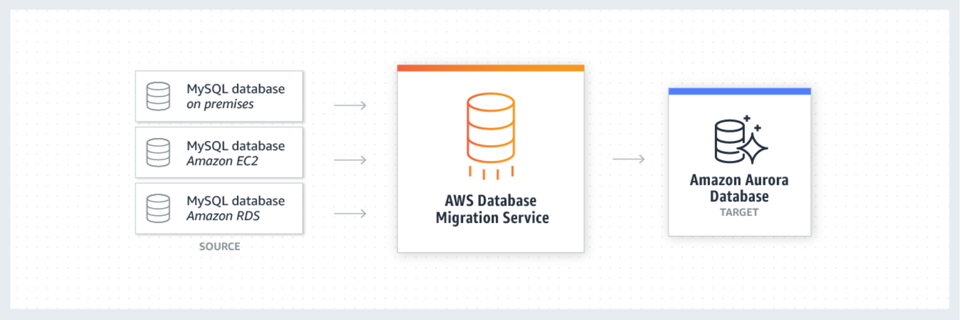
# HETEROGENOUS DATABASE MIGRATIONS

In heterogeneous database migrations the source and target databases engines are different, like in the case of Oracle to Amazon Aurora, Oracle to PostgreSQL, or Microsoft SQL Server to MySQL migrations. In this case, the schema structure, data types, and database code of source and target databases can be quite different, requiring a schema and code transformation before the data migration starts. That makes heterogeneous migrations a two step process. First use the AWS Schema Conversion Tool to convert the source schema and code to match that of the target database, and then use the AWS Database Migration Service to migrate data from the source database to the target database. All the required data type conversions will automatically be done by the AWS Database Migration Service during the migration. The source database can be located in your own premises outside of AWS, running on an Amazon EC2 instance, or it can be an Amazon RDS database. The target can be a database in Amazon EC2 or Amazon RDS.


STEP 1
Oracle Database
On-premises, in Amazon EC2, or Amazon RDS
SOURCE
AWS Schema Conversion Tool
Amazon Aurora Database
TARGET
STEP 2
MySQL Database
On-premises, in Amazon EC2, or Amazon RDS
SOURCE
AWS Database Migration Service
Amazon Aurora Database
TARGET

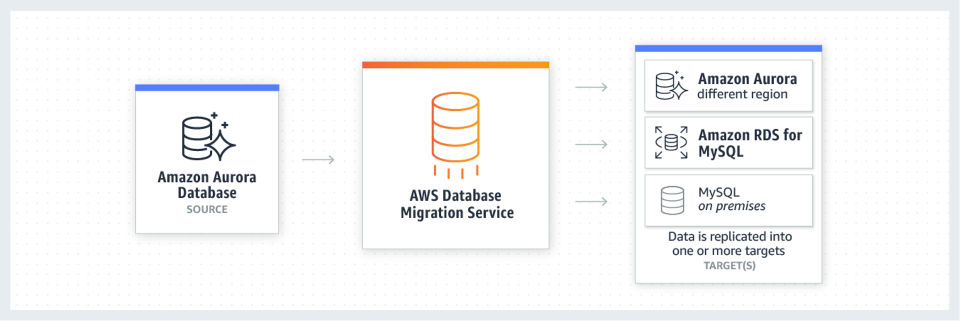
# DATABASE CONSOLIDATION

We can use AWS Database Migration Service to consolidate multiple source databases into a single target database. This can be done for homogeneous and heterogeneous migrations, and you can use this feature with all supported database engines. The source databases can be located in our own premises outside of AWS, running on an Amazon EC2 instance, or it can be an Amazon RDS database. The sources databases can also be spread across different locations. For example, one of the source databases can be in our own premises outside of AWS, while the second one in Amazon EC2, and the third one is an Amazon RDS database. The target can be a database in Amazon EC2 or Amazon RDS.



# CONTINUOUS DATA REPLICATION

We can use AWS Database Migration Service to perform continuous data replication. Continuous data replication has a multitude of use cases including Disaster Recovery instance synchronization, geographic database distribution and Dev/Test environment synchronization. We can use DMS for both homogeneous and heterogeneous data replications for all supported database engines. The source or destination databases can be located in our own premises outside of AWS, running on an Amazon EC2 instance, or it can be an Amazon RDS database. We can replicate data from a single database to one or more target databases or data from multiple source databases can be consolidated and replicated to one or more target databases.



# AWS Schema Conversion Tool (for Heterogenous Database Migrations)

We can use the AWS Schema Conversion Tool (AWS SCT) to convert our existing database schema from one database engine to another. We can convert relational OLTP schema, or data warehouse schema. Our converted schema is suitable for an Amazon Relational Database Service (Amazon RDS) MySQL DB instance, an Amazon Aurora DB cluster, an Amazon RDS PostgreSQL DB instance, or an Amazon Redshift cluster. The converted schema can also be used with a database on an Amazon EC2 instance or stored as data on an Amazon S3 bucket.

The AWS Schema Conversion Tool (AWS SCT) is a standalone application that provides a project-based user interface. AWS SCT is available for Fedora Linux, macOS, Microsoft Windows, and Ubuntu Linux version 15.04. AWS SCT is supported only on 64-bit operating systems. AWS SCT also installs the Java Runtime Environment (JRE) version 8u45.

AWS SCT supports the following OLTP conversions.

| Source Database | Target Database on Amazon RDS |
|---|---|
| Microsoft SQL Server (version 2008 and later) | Amazon Aurora (MySQL or PostgreSQL), Microsoft SQL Server, MySQL, PostgreSQL |
| MySQL (version 5.5 and later) | Amazon Aurora (PostgreSQL), MySQL, PostgreSQL<br>You can migrate schema and data from MySQL to an Amazon Aurora (MySQL) DB cluster without using AWS SCT. For more information, see [Migrating Data to an Amazon Aurora DB Cluster](). |
| Oracle (version 10.2 and later) | Amazon Aurora (MySQL or PostgreSQL), MySQL, Oracle, PostgreSQL |
| PostgreSQL (version 9.1 and later) | Amazon Aurora (MySQL), MySQL, PostgreSQL |
| IBM Db2 LUW (versions 9.1, 9.5, 9.7, 10.5, and 11.1) | Amazon Aurora (MySQL), MySQL, PostgreSQL, Amazon Aurora (PostgreSQL) |
| Apache Cassandra (versions 2.0 and 3.0) | Amazon DynamoDB |
| Sybase | PostgreSQL, Amazon Aurora (PostgreSQL) |

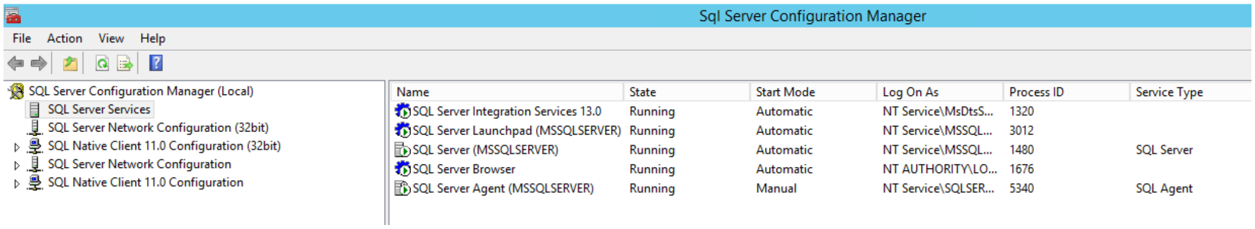
## For this demo, I would like to migrate data from Microsoft SQL Server database installed on AWS EC2 to MySQL (Amazon RDS).

### Steps along with Prerequisites (Not all the steps or objects are mandatory, however I have used them just to make it easier and exploration purposes)

- EC2 Windows_Server-2012-R2_RTM-English-64Bit-Base-2019.02.09 (ami-0d99daa98c8711fe1)

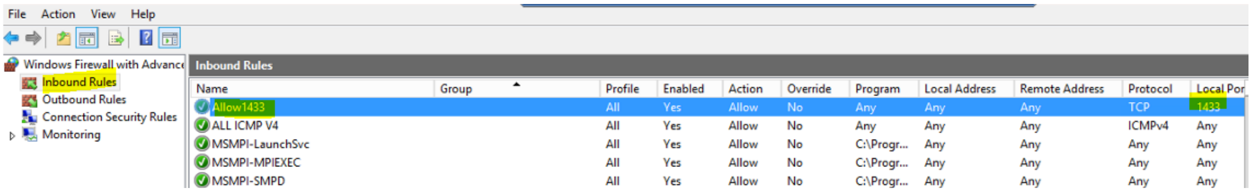
- Microsoft SQL Server Management Studio.  Version-14.0.17289.0

- Created the sample database-Northwind using the scripts from the following GIT repository https://github.com/Microsoft/sql-server-samples/tree/master/samples/databases/northwind-pubs
- Ensure that required SQL Server services are running


Sql Server Configuration Manager
File
Action
View
Help
SQL Server Configuration Manager (Local)
SQL Server Services
SQL Server Network Configuration (32bit)
SQL Native Client 11.0 Configuration (32bit)
SQL Server Network Configuration
SQL Native Client 11.0 Configuration

| Name | State | Start Mode | Log On As | Process ID | Service Type |
|---|---|---|---|---|---|
| SQL Server Integration Services 13.0 | Running | Automatic | NT Service\MsDtsS... | 1320 | |
| SQL Server Launchpad (MSSQLSERVER) | Running | Automatic | NT Service\MSSQL... | 3012 | |
| SQL Server (MSSQLSERVER) | Running | Automatic | NT Service\MSSQL... | 1480 | SQL Server |
| SQL Server Browser | Running | Automatic | NT AUTHORITY\LO... | 1676 | |
| SQL Server Agent (MSSQLSERVER) | Running | Manual | NT Service\SQLSER... | 5340 | SQL Agent |

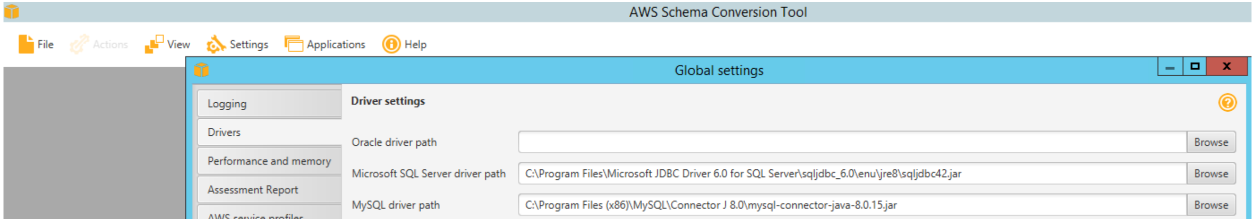
- Firewall may block the port - SQL Server:**1433**, if it is configured that way. For the Data Migration Task to access the DB and perform the migration, this rule (TCP:1433) needs to be enabled



| Name | Group | Profile | Enabled | Action | Override | Program | Local Address | Remote Address | Protocol | Local Por |
|---|---|---|---|---|---|---|---|---|---|---|
| Allow1433 | | All | Yes | Allow | No | Any | Any | Any | TCP | 1433 |
| ALL ICMP V4 | | All | Yes | Allow | No | Any | Any | Any | ICMPv4 | Any |
| MSMPI-LaunchSvc | | All | Yes | Allow | No | C:\Progr... | Any | Any | Any | Any |
| MSMPI-MPIEXEC | | All | Yes | Allow | No | C:\Progr... | Any | Any | Any | Any |
| MSMPI-SMPD | | All | Yes | Allow | No | C:\Progr... | Any | Any | Any | Any |

- Install the SQL drivers and the AWS Schema Conversion Tool (AWS SCT) on the local computer.  I have installed it in my EC2 instance.
[https://docs.aws.amazon.com/dms/latest/sbs/CHAP_SQLServer2Aurora.Steps.InstallSCT.html](https://docs.aws.amazon.com/dms/latest/sbs/CHAP_SQLServer2Aurora.Steps.InstallSCT.html)

- Install and map the respective drivers in the Global Settings page of the AWS SCT



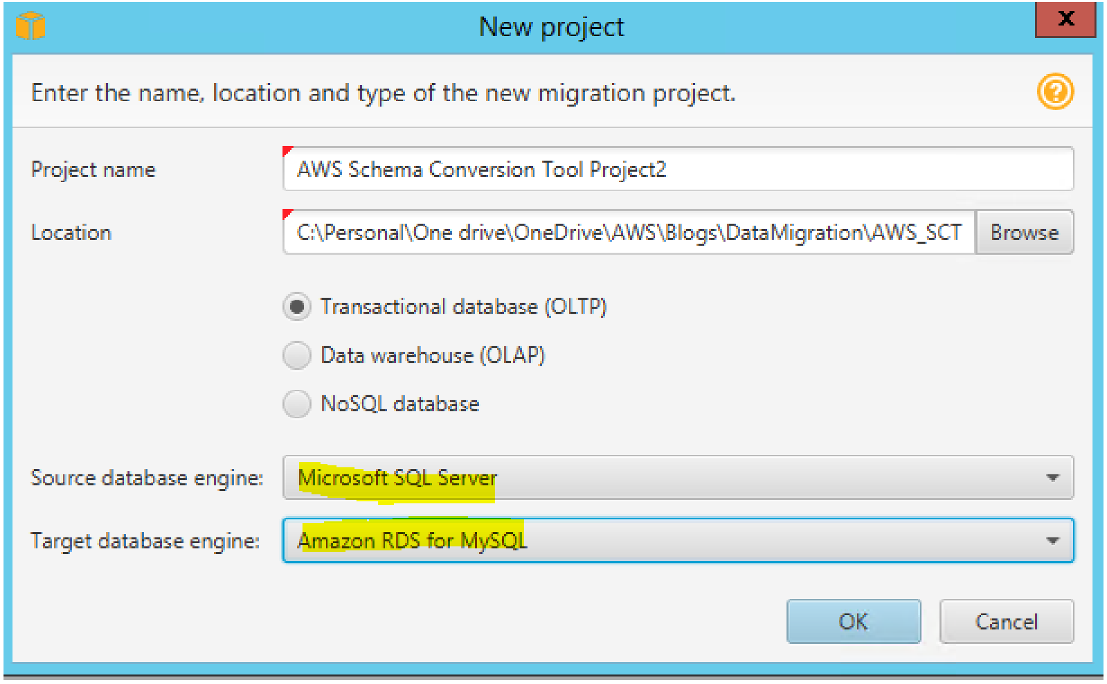
Driver settings

Oracle driver path

Microsoft SQL Server driver path: C:\Program Files\Microsoft JDBC Driver 6.0 for SQL Server\sqljdbc_6.0\enu\jre8\sqljdbc42.jar

MySQL driver path: C:\Program Files (x86)\MySQL\Connector J 8.0\mysql-connector-java-8.0.15.jar

- Create a project and select the Source & Target database engines with the type (OLTP/OLAP/No SQL DB)



Enter the name, location and type of the new migration project.

Project name: AWS Schema Conversion Tool Project2

Location: C:\Personal\One drive\OneDrive\AWS\Blogs\DataMigration\AWS_SCT

● Transactional database (OLTP)

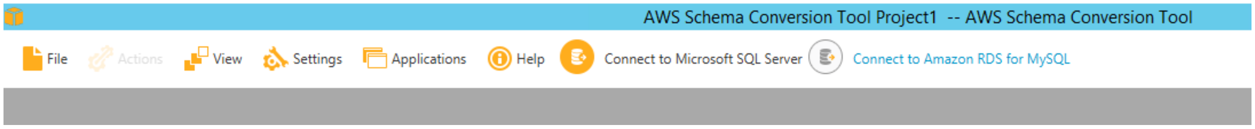
○ Data warehouse (OLAP)

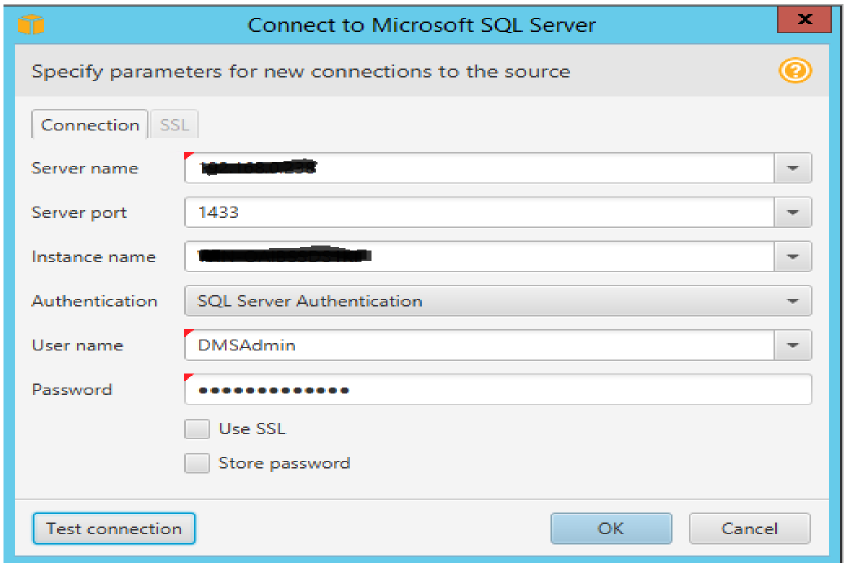
○ NoSQL database

Source database engine: Microsoft SQL Server

Target database engine: Amazon RDS for MySQL

- Setup the connections to the DB Servers


AWS Schema Conversion Tool Project1 -- AWS Schema Conversion Tool
File
Actions
View
Settings
Applications
Help
Connect to Microsoft SQL Server
Connect to Amazon RDS for MySQL

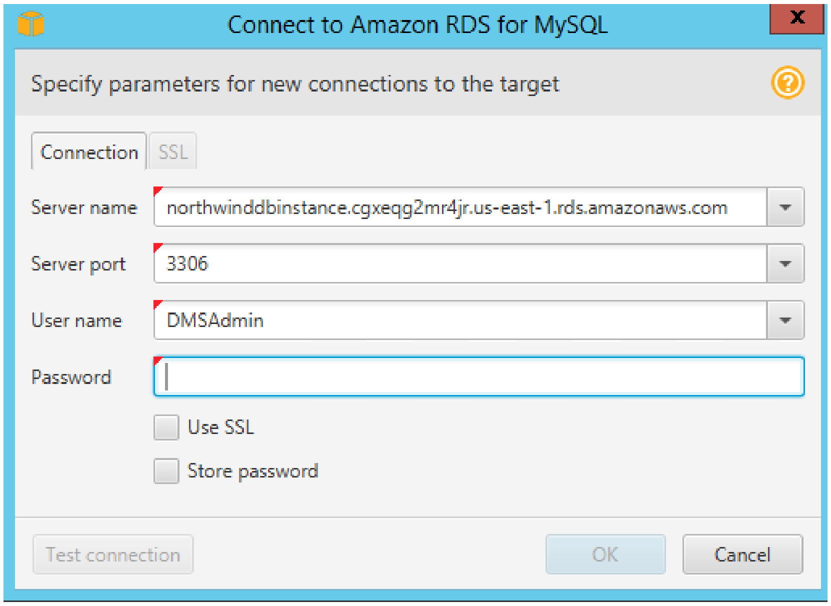


Specify parameters for new connections to the source

| | |
|---|---|
| Server name | [redacted] |
| Server port | 1433 |
| Instance name | [redacted] |
| Authentication | SQL Server Authentication |
| User name | DMSAdmin |
| Password | ••••••••••••• |

□ Use SSL

□ Store password



Specify parameters for new connections to the target

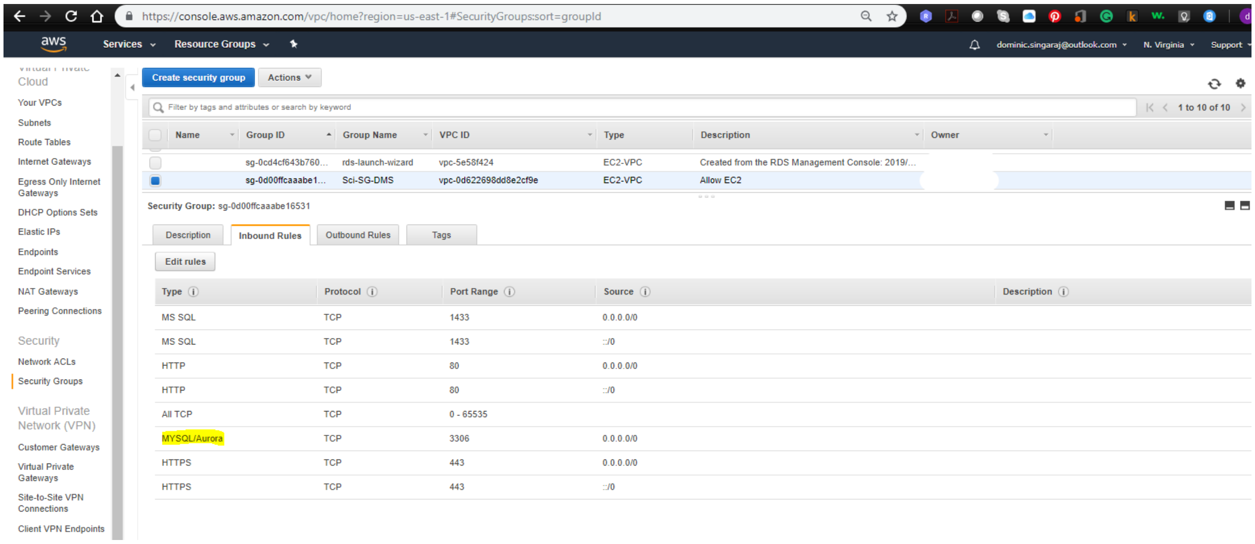
| | |
|---|---|
| Server name | northwinddbinstance.cgxeqg2mr4jr.us-east-1.rds.amazonaws.com |
| Server port | 3306 |
| User name | DMSAdmin |
| Password | |

□ Use SSL

□ Store password

- For the project, I've setup the VPC along with 2 public subnets, 2 private subnets and associated the Internet gateway to the VPC. Configured the Route tables, NACL along with the security groups.

  Please note that I've configured all the elements to be part of the same VPC which includes the following
  - EC2 instance with SQL Server
  - RDS (MySQL Engine)

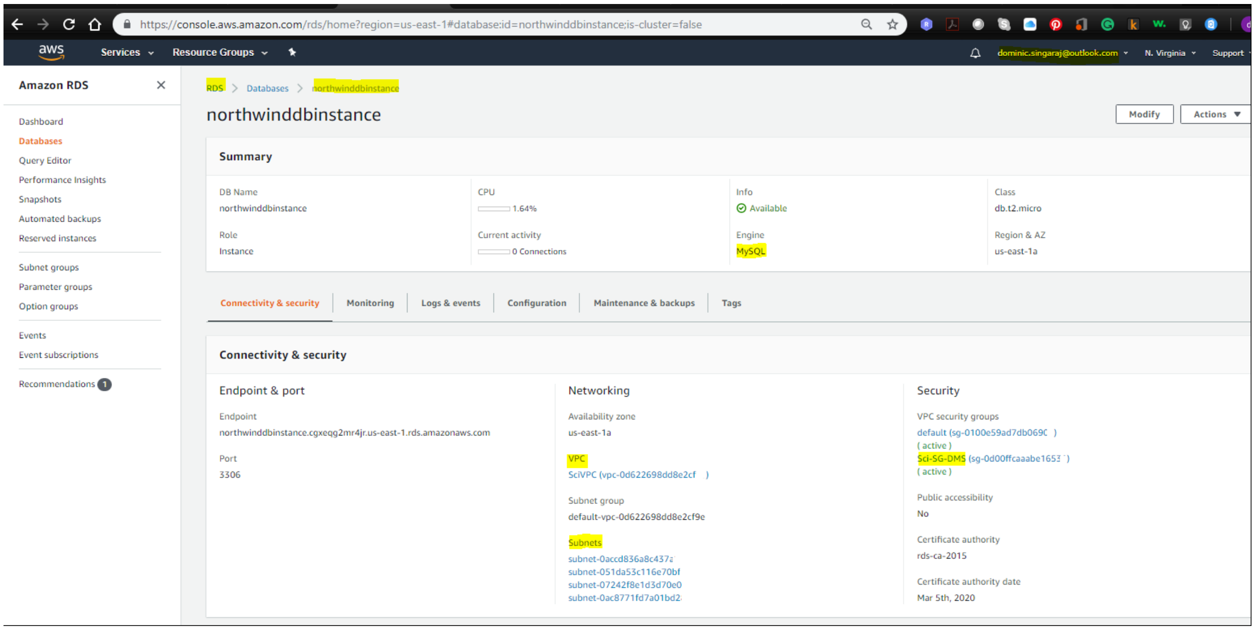


| Name | Group ID | Group Name | VPC ID | Type | Description | Owner |
|---|---|---|---|---|---|---|
| | sg-0cd4cf643b760... | rds-launch-wizard | vpc-5e58f424 | EC2-VPC | Created from the RDS Management Console: 2019/... | |
| | sg-0d00ffcaaabe1... | Sci-SG-DMS | vpc-0d622698dd8e2cf9e | EC2-VPC | Allow EC2 | |

Security Group: sg-0d00ffcaaabe16531

| Type | Protocol | Port Range | Source | Description |
|---|---|---|---|---|
| MS SQL | TCP | 1433 | 0.0.0.0/0 | |
| MS SQL | TCP | 1433 | ::/0 | |
| HTTP | TCP | 80 | 0.0.0.0/0 | |
| HTTP | TCP | 80 | ::/0 | |
| All TCP | TCP | 0 - 65535 | | |
| MYSQL/Aurora | TCP | 3306 | 0.0.0.0/0 | |
| HTTPS | TCP | 443 | 0.0.0.0/0 | |
| HTTPS | TCP | 443 | ::/0 | |

## Created the Amazon RDS DB - NorthwindDBInstance using the console



northwinddbinstance

Summary

DB Name: northwinddbinstance
CPU: 1.64%
Info: Available
Class: db.t2.micro
Role: Instance
Current activity: 0 Connections
Engine: MySQL
Region & AZ: us-east-1a

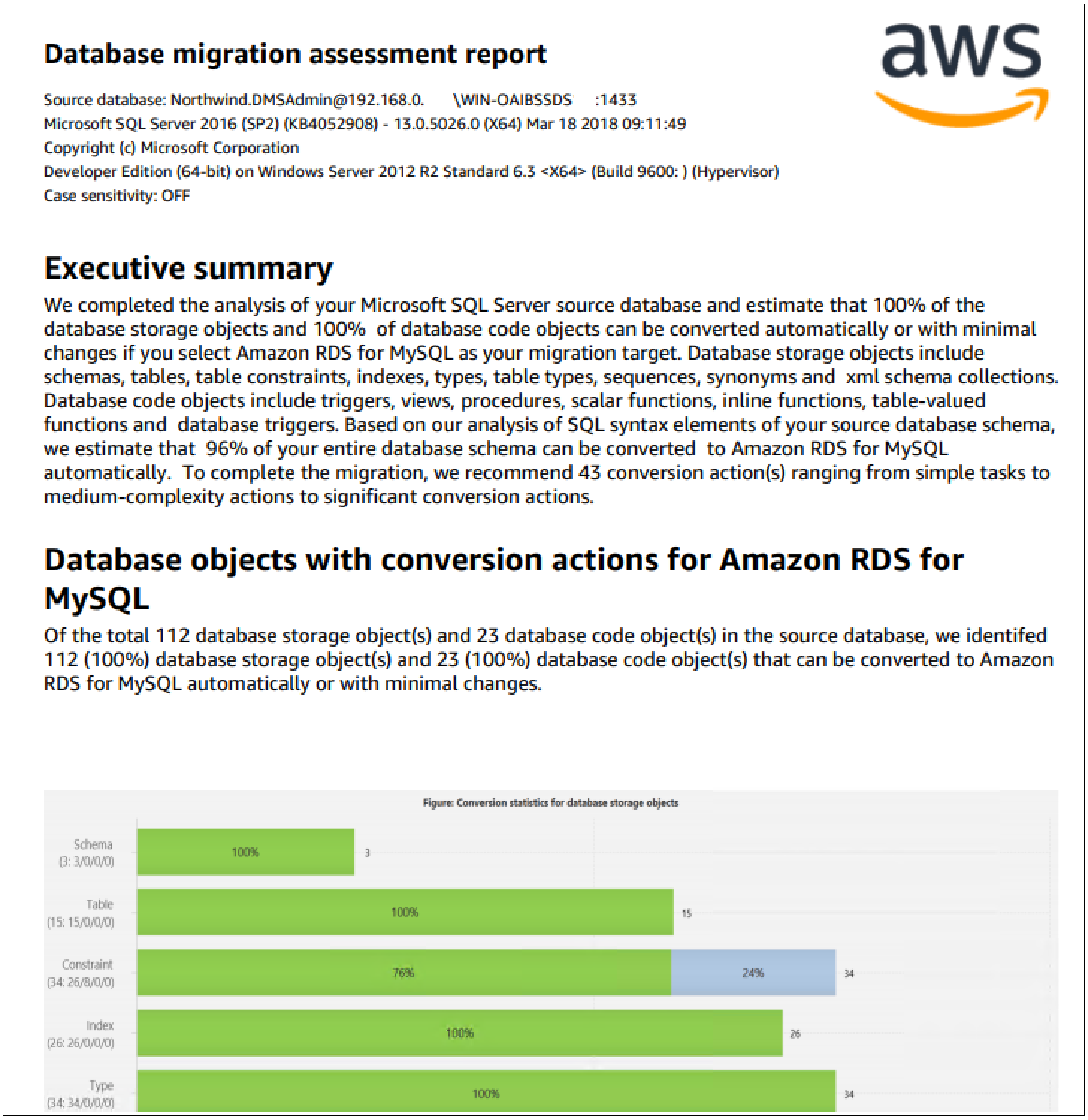
Connectivity & security

Endpoint & port
Endpoint: northwinddbinstance.cgxeqg2mr4jr.us-east-1.rds.amazonaws.com
Port: 3306

Networking
Availability zone: us-east-1a
VPC: SciVPC (vpc-0d622698dd8e2cf )
Subnet group: default-vpc-0d622698dd8e2cf9e
Subnets: subnet-0accd836a8c437a<br>subnet-051da53c116e70bf<br>subnet-07242f8e1d3d70e0<br>subnet-0ac8771fd7a01bd2

Security
VPC security groups: default (sg-0100e59ad7db069C ) ( active ) Sci-SG-DMS (sg-0d00ffcaaabe1653 ) ( active )
Public accessibility: No
Certificate authority: rds-ca-2015
Certificate authority date: Mar 5th, 2020

- In the AWS SCT, Right click on the source database (Northwind) and generate the assessment report. If there are any issues which needs to be handled prior to Schema preparation/DB migration, it would be highlighted in the assessment report and few sample issues can be seen from the below screen which could be fixed manually in the source database, if required.



## Database migration assessment report

Source database: Northwind.DMSAdmin@192.168.0.      \WIN-OAIBSSDS     :1433
Microsoft SQL Server 2016 (SP2) (KB4052908) - 13.0.5026.0 (X64) Mar 18 2018 09:11:49
Copyright (c) Microsoft Corporation
Developer Edition (64-bit) on Windows Server 2012 R2 Standard 6.3 <X64> (Build 9600: ) (Hypervisor)
Case sensitivity: OFF

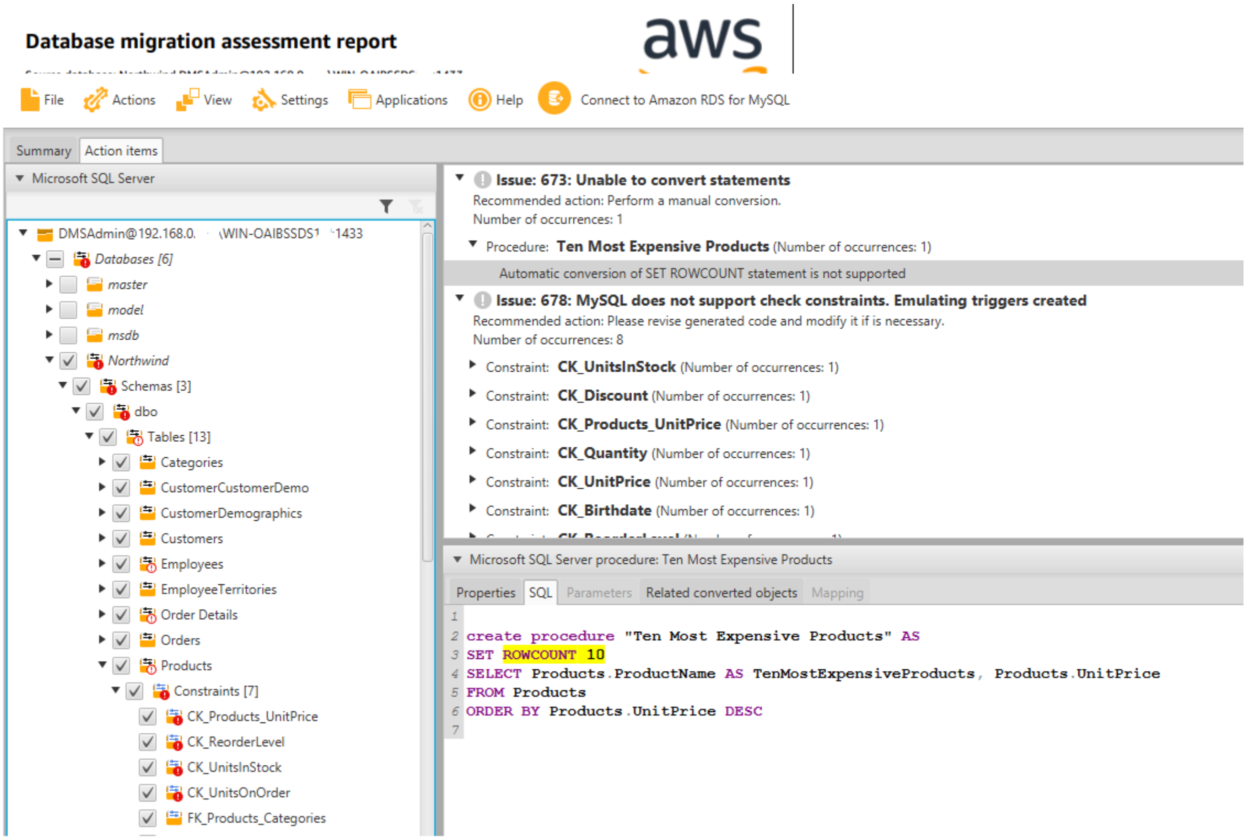
## Executive summary

We completed the analysis of your Microsoft SQL Server source database and estimate that 100% of the database storage objects and 100%  of database code objects can be converted automatically or with minimal changes if you select Amazon RDS for MySQL as your migration target. Database storage objects include schemas, tables, table constraints, indexes, types, table types, sequences, synonyms and  xml schema collections. Database code objects include triggers, views, procedures, scalar functions, inline functions, table-valued functions and  database triggers. Based on our analysis of SQL syntax elements of your source database schema, we estimate that  96% of your entire database schema can be converted  to Amazon RDS for MySQL automatically.  To complete the migration, we recommend 43 conversion action(s) ranging from simple tasks to medium-complexity actions to significant conversion actions.

## Database objects with conversion actions for Amazon RDS for MySQL

Of the total 112 database storage object(s) and 23 database code object(s) in the source database, we identifed 112 (100%) database storage object(s) and 23 (100%) database code object(s) that can be converted to Amazon RDS for MySQL automatically or with minimal changes.

Figure: Conversion statistics for database storage objects



**Issue: 673: Unable to convert statements**
Recommended action: Perform a manual conversion.
Number of occurrences: 1

Procedure: **Ten Most Expensive Products** (Number of occurrences: 1)

Automatic conversion of SET ROWCOUNT statement is not supported

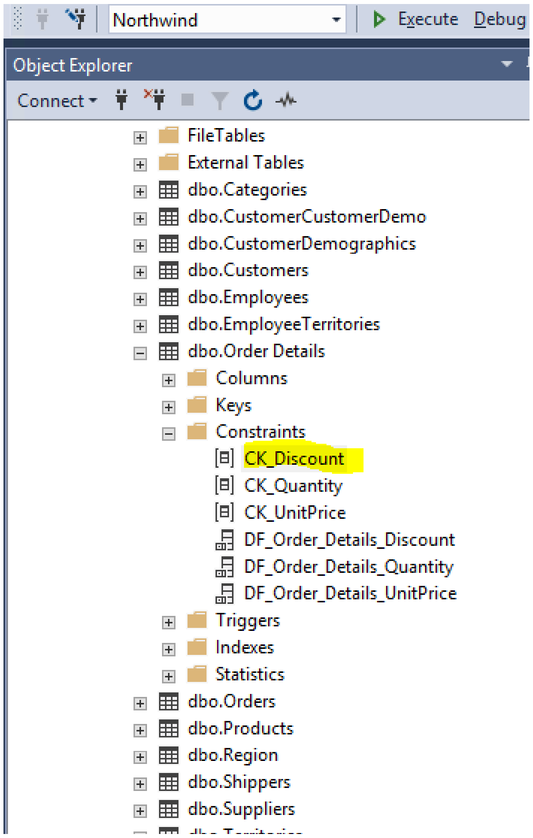
**Issue: 678: MySQL does not support check constraints. Emulating triggers created**
Recommended action: Please revise generated code and modify it if is necessary.
Number of occurrences: 8

Constraint: **CK_UnitsInStock** (Number of occurrences: 1)
Constraint: **CK_Discount** (Number of occurrences: 1)
Constraint: **CK_Products_UnitPrice** (Number of occurrences: 1)
Constraint: **CK_Quantity** (Number of occurrences: 1)
Constraint: **CK_UnitPrice** (Number of occurrences: 1)
Constraint: **CK_Birthdate** (Number of occurrences: 1)

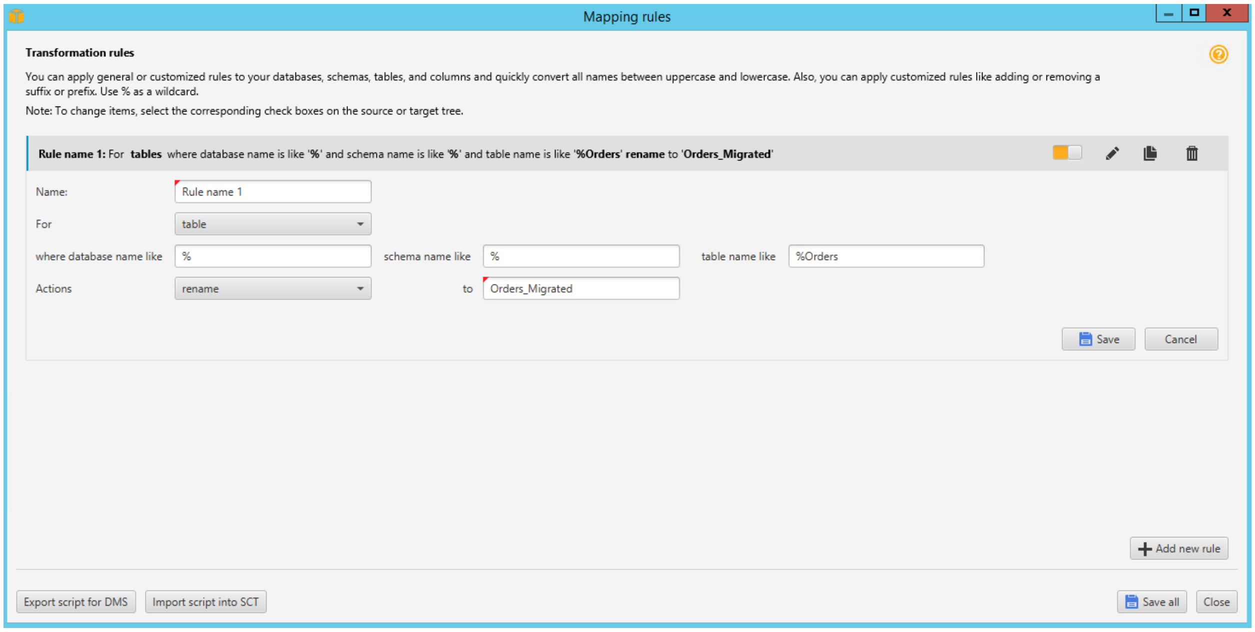
```

create procedure "Ten Most Expensive Products" AS
SET ROWCOUNT 10
SELECT Products.ProductName AS TenMostExpensiveProducts, Products.UnitPrice
FROM Products
ORDER BY Products.UnitPrice DESC

```



Once the issues have been fixed, right click on the source DB and select create schema, so that it can be applied to the target DB which is RDS-MySQL in our case.  Along the way, transformation rules could be applied like shown below



**Transformation rules**

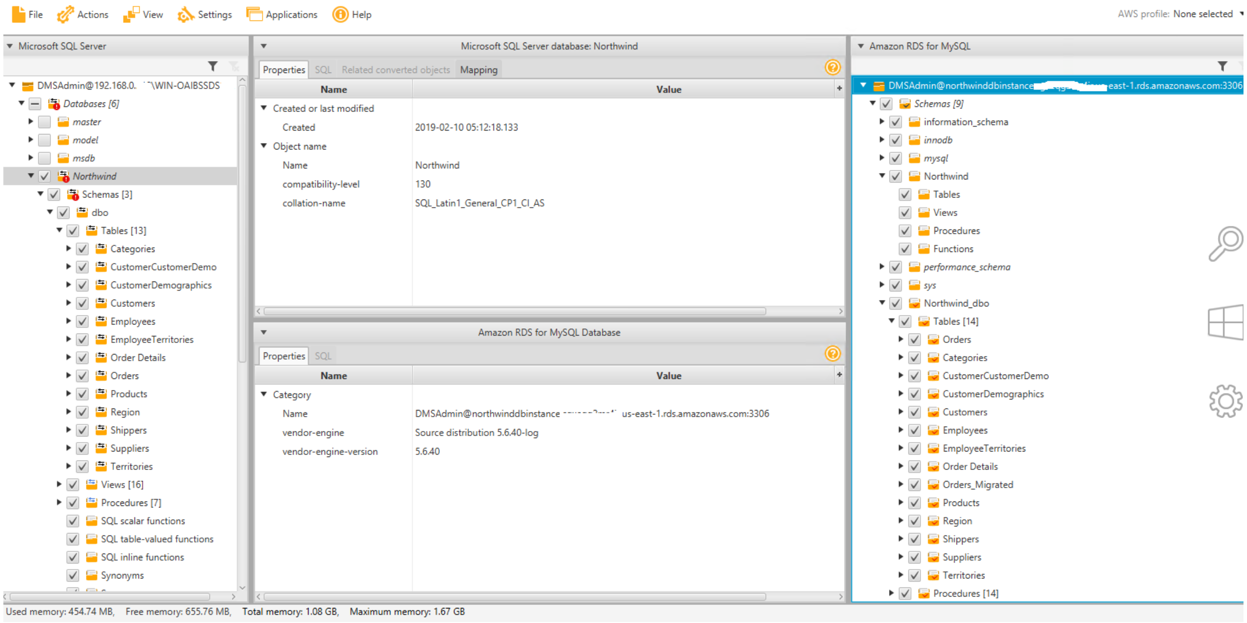
You can apply general or customized rules to your databases, schemas, tables, and columns and quickly convert all names between uppercase and lowercase. Also, you can apply customized rules like adding or removing a suffix or prefix. Use % as a wildcard.

Note: To change items, select the corresponding check boxes on the source or target tree.

**Rule name 1:** For **tables** where database name is like '%' and schema name is like '%' and table name is like '**%Orders**' **rename** to '**Orders_Migrated**'

| | | | | | |
|---|---|---|---|---|---|
| Name: | Rule name 1 | | | | |
| For | table | | | | |
| where database name like | % | schema name like | % | table name like | %Orders |
| Actions | rename | to | Orders_Migrated | | |

## Once the schema has been applied on the target DB, following screen could be seen



AWS profile: None selected

Microsoft SQL Server

DMSAdmin@192.168.0.[redacted]\WIN-OAIBSSDS
- Databases [6]
  - master
  - model
  - msdb
  - Northwind
    - Schemas [3]
      - dbo
        - Tables [13]
          - Categories
          - CustomerCustomerDemo
          - CustomerDemographics
          - Customers
          - Employees
          - EmployeeTerritories
          - Order Details
          - Orders
          - Products
          - Region
          - Shippers
          - Suppliers
          - Territories
        - Views [16]
        - Procedures [7]
        - SQL scalar functions
        - SQL table-valued functions
        - SQL inline functions
        - Synonyms

Microsoft SQL Server database: Northwind

| Name | Value |
|---|---|
| Created or last modified | |
| Created | 2019-02-10 05:12:18.133 |
| Object name | |
| Name | Northwind |
| compatibility-level | 130 |
| collation-name | SQL_Latin1_General_CP1_CI_AS |

Amazon RDS for MySQL Database

| Name | Value |
|---|---|
| Category | |
| Name | DMSAdmin@northwinddbinstance[redacted] us-east-1.rds.amazonaws.com:3306 |
| vendor-engine | Source distribution 5.6.40-log |
| vendor-engine-version | 5.6.40 |

Amazon RDS for MySQL

DMSAdmin@northwinddbinstance[redacted]east-1.rds.amazonaws.com:3306
- Schemas [9]
  - information_schema
  - innodb
  - mysql
  - Northwind
    - Tables
    - Views
    - Procedures
    - Functions
  - performance_schema
  - sys
  - Northwind_dbo
    - Tables [14]
      - Orders
      - Categories
      - CustomerCustomerDemo
      - CustomerDemographics
      - Customers
      - Employees
      - EmployeeTerritories
      - Order Details
      - Orders_Migrated
      - Products
      - Region
      - Shippers
      - Suppliers
      - Territories
    - Procedures [14]

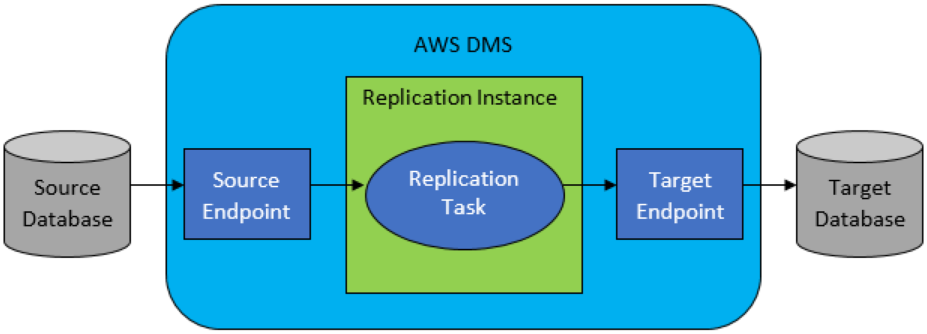
Used memory: 454.74 MB,    Free memory: 655.76 MB,    Total memory: 1.08 GB,    Maximum memory: 1.67 GB

This concludes the activities related to AWS Schema Tool and schema preparation on the target DB. The next step is to start utilizing the AWS Data Migration Service for migrating data from source to target

## AWS – Data Migration Service

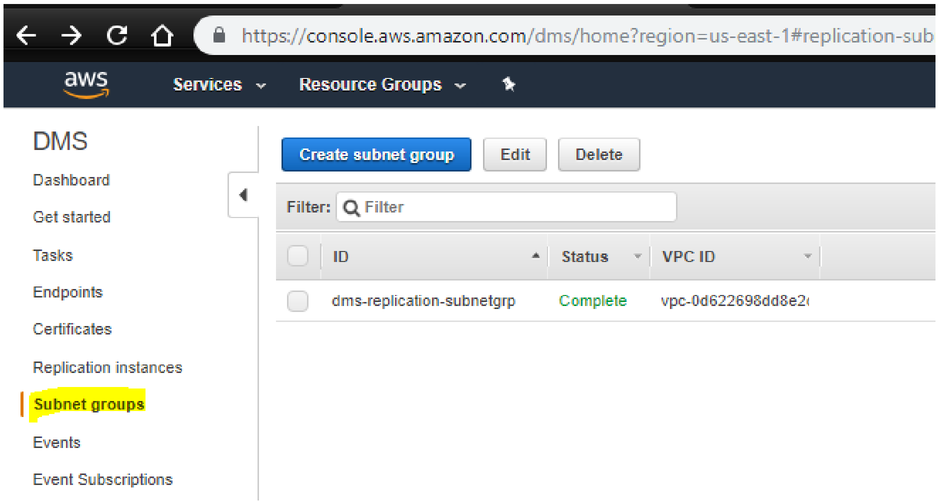
DMS requires the following items to be setup

- Subnet group
- Replication Instance
- Task (Migration)

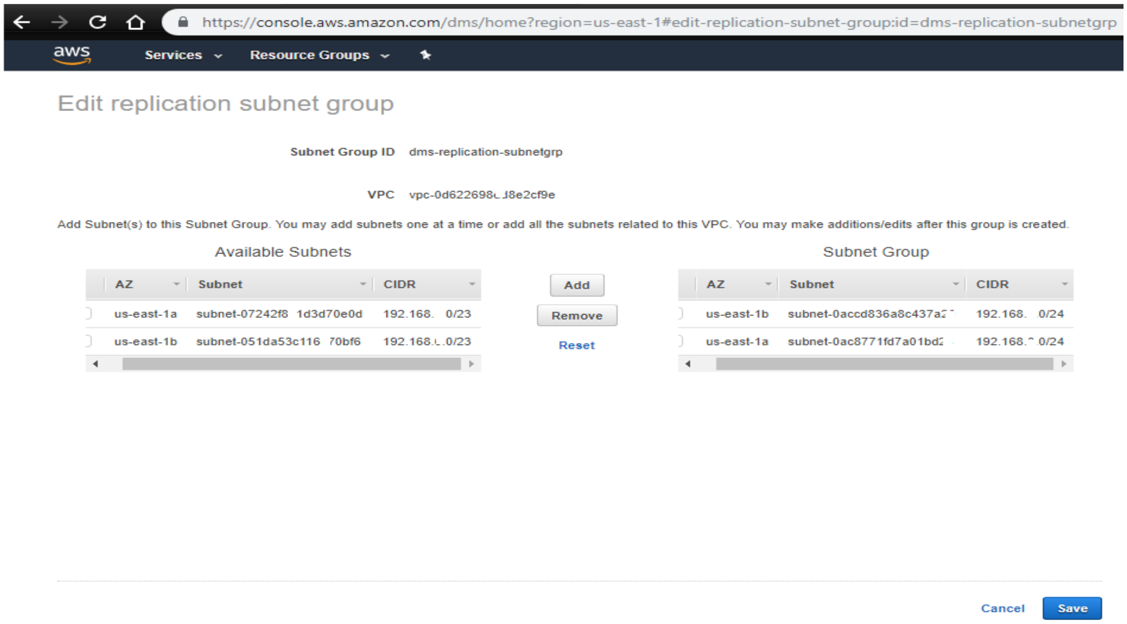


### Subnet group

Create a subnet group which requires subnets (multi-AZ). In my opinion, please select the public subnets as part of subnet group, so that the replication instance may be able to access the EC2 which has the source DB (SQL Server in our case).



| ID | Status | VPC ID |
|---|---|---|
| dms-replication-subnetgrp | Complete | vpc-0d622698dd8e2( |


https://console.aws.amazon.com/dms/home?region=us-east-1#edit-replication-subnet-group:id=dms-replication-subnetgrp
aws
Services
Resource Groups

Edit replication subnet group

Subnet Group ID dms-replication-subnetgrp

VPC vpc-0d622698c.l8e2cf9e

Add Subnet(s) to this Subnet Group. You may add subnets one at a time or add all the subnets related to this VPC. You may make additions/edits after this group is created.

Available Subnets

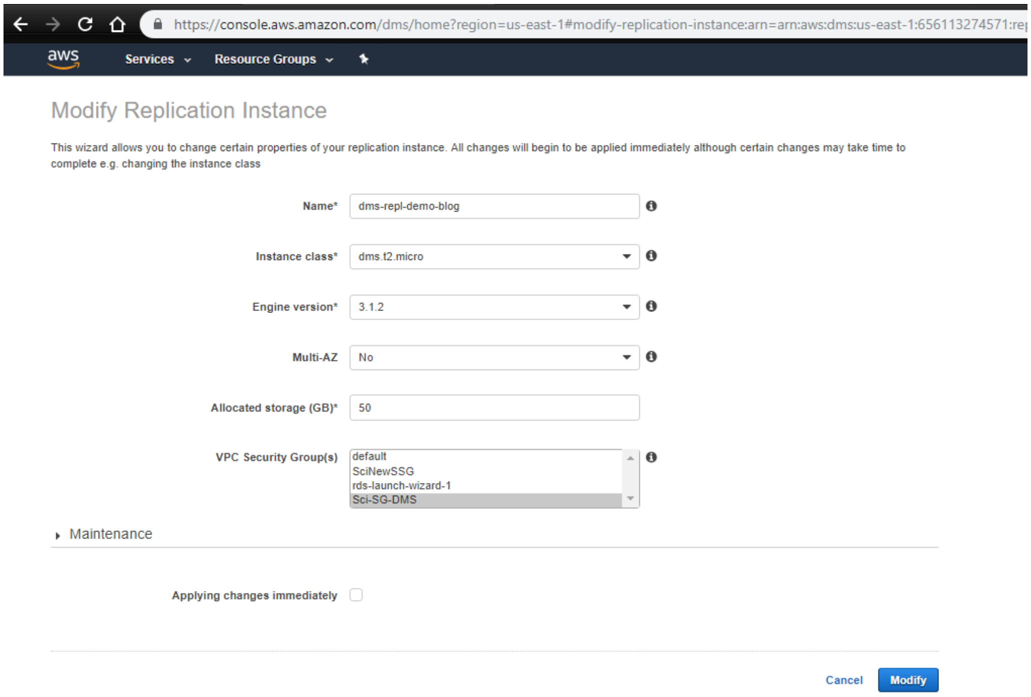
| AZ | Subnet | CIDR |
|---|---|---|
| us-east-1a | subnet-07242f8 1d3d70e0d | 192.168. 0/23 |
| us-east-1b | subnet-051da53c116 70bf6 | 192.168.ι.0/23 |

Subnet Group

| AZ | Subnet | CIDR |
|---|---|---|
| us-east-1b | subnet-0accd836a8c437a2 | 192.168. 0/24 |
| us-east-1a | subnet-0ac8771fd7a01bd2 | 192.168.ˆ 0/24 |

## Replication Instance

AWS DMS uses a replication instance to connect to the source data store, read the source data, and format the data for consumption by the target data store. A replication instance also loads the data into the target data store. Most of this processing happens in memory. However, large transactions might require some buffering on disk. Cached transactions and log files are also written to disk.


https://console.aws.amazon.com/dms/home?region=us-east-1#modify-replication-instance:arn=arn:aws:dms:us-east-1:656113274571:re
aws
Services
Resource Groups

Modify Replication Instance

This wizard allows you to change certain properties of your replication instance. All changes will begin to be applied immediately although certain changes may take time to complete e.g. changing the instance class

Name* dms-repl-demo-blog

Instance class* dms.t2.micro

Engine version* 3.1.2

Multi-AZ No

Allocated storage (GB)* 50

VPC Security Group(s) default, SciNewSSG, rds-launch-wizard-1, Sci-SG-DMS

▸ Maintenance

Applying changes immediately □

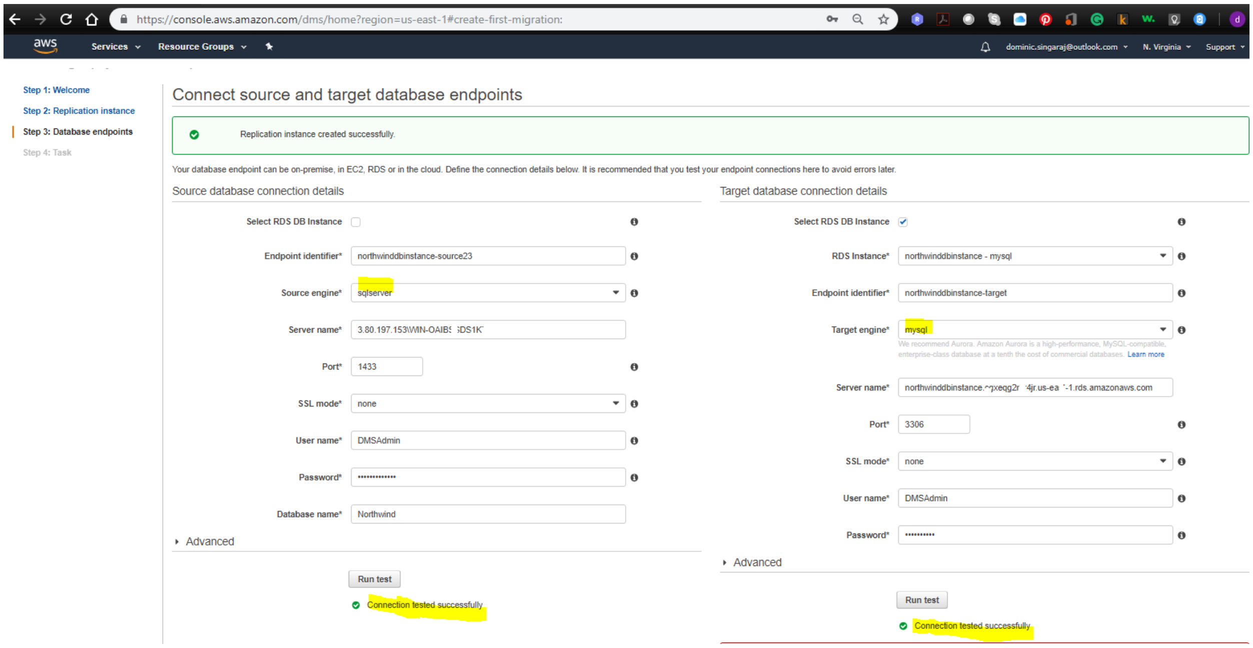
## AWS DMS Task

An AWS Database Migration Service (AWS DMS) task is where all the work happens. You may specify what tables and schemas to use for your migration and any special processing, such as logging requirements, control table data, and error handling.

When creating a migration task, you need to know several things:

- Before you can create a task, you must create a source endpoint, a target endpoint, and a replication instance.
- You can specify many task settings to tailor your migration task. You can set these by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS DMS API. These settings include specifying how migration errors are handled, error logging, and control table information.
- After you create a task, you can run it immediately. The target tables with the necessary metadata definitions are automatically created and loaded, and you can specify ongoing replication.
- By default, AWS DMS starts your task as soon as you create it. However, in some situations, you might want to postpone the start of the task. For example, when using the AWS CLI, you might have a process that creates a task and a different process that starts the task based on some triggering event. As needed, you can postpone your task's start.
- You can monitor, stop, or restart tasks using the AWS DMS console, AWS CLI, or AWS DMS API.

Connect to source and target DB endpoints and test the connection prior to task creation activity. This step will help us to make sure the task executes successfully


https://console.aws.amazon.com/dms/home?region=us-east-1#create-first-migration:
aws
Services
Resource Groups
dominic.singaraj@outlook.com
N. Virginia
Support
Step 1: Welcome
Step 2: Replication instance
Step 3: Database endpoints
Step 4: Task

Connect source and target database endpoints

Replication instance created successfully.

Your database endpoint can be on-premise, in EC2, RDS or in the cloud. Define the connection details below. It is recommended that you test your endpoint connections here to avoid errors later.

Source database connection details

| | |
|---|---|
| Select RDS DB Instance | ☐ |
| Endpoint identifier* | northwinddbinstance-source23 |
| Source engine* | sqlserver |
| Server name* | 3.80.197.153\WIN-OAIB: ;DS1K' |
| Port* | 1433 |
| SSL mode* | none |
| User name* | DMSAdmin |
| Password* | ••••••••••••• |
| Database name* | Northwind |

Advanced

Run test

Connection tested successfully

Target database connection details

| | |
|---|---|
| Select RDS DB Instance | ☑ |
| RDS Instance* | northwinddbinstance - mysql |
| Endpoint identifier* | northwinddbinstance-target |
| Target engine* | mysql<br>We recommend Aurora. Amazon Aurora is a high-performance, MySQL-compatible, enterprise-class database at a tenth the cost of commercial databases. Learn more |
| Server name* | northwinddbinstance.~gxeqg2r 4jr.us-ea '-1.rds.amazonaws.com |
| Port* | 3306 |
| SSL mode* | none |
| User name* | DMSAdmin |
| Password* | •••••••••• |

Advanced

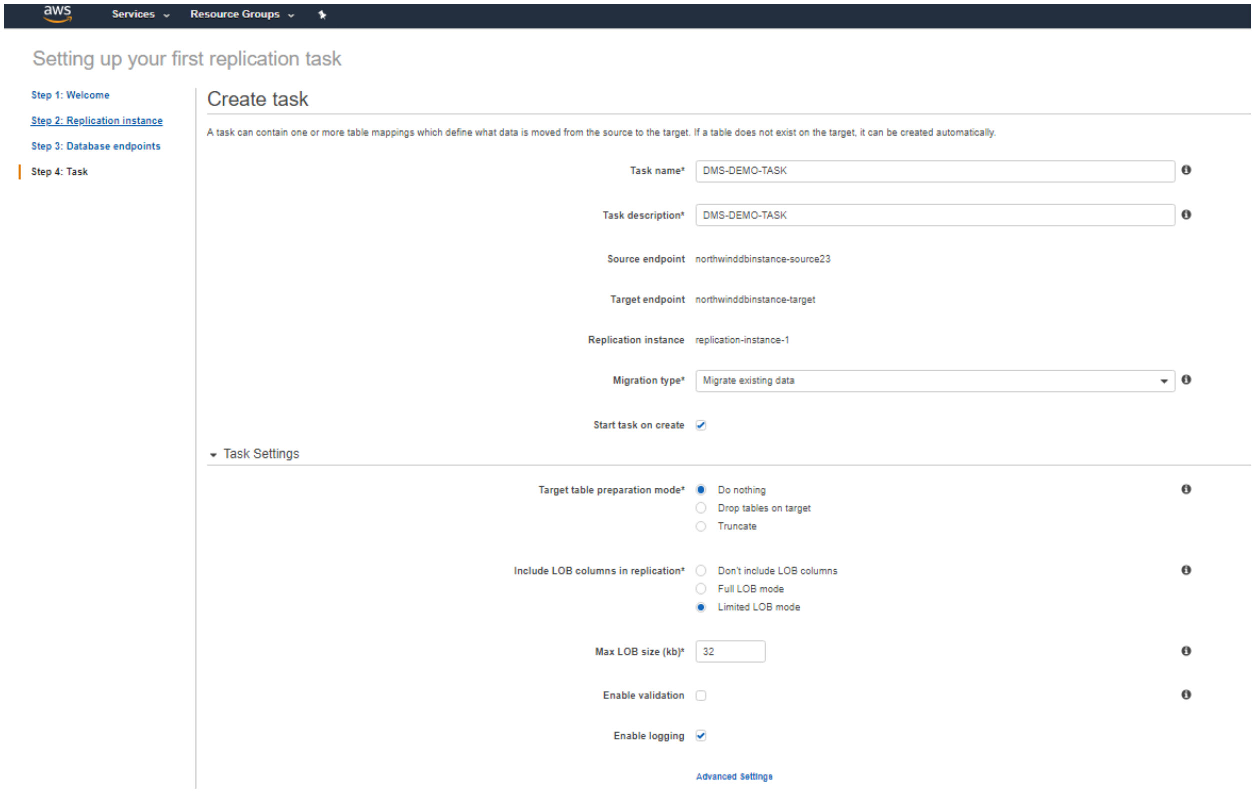
Run test

Connection tested successfully

It is possible to see connection issues like the one highlighted in this URL
https://christierney.com/2018/09/13/aws-database-migration-service-endpoint-connection-issue/

The next step in the wizard is to provide the details related to the task like endpoints, migration type, target table preparation mode, table mapping and logging.

I have enabled the logging option, so that the migration status could be logged and stored in cloud watch for our verification purposes.



Setting up your first replication task

Create task

A task can contain one or more table mappings which define what data is moved from the source to the target. If a table does not exist on the target, it can be created automatically.

Task name* DMS-DEMO-TASK

Task description* DMS-DEMO-TASK

Source endpoint northwinddbinstance-source23

Target endpoint northwinddbinstance-target

Replication instance replication-instance-1

Migration type* Migrate existing data

Start task on create ☑

Task Settings

Target table preparation mode* ● Do nothing ○ Drop tables on target ○ Truncate

Include LOB columns in replication* ○ Don't include LOB columns ○ Full LOB mode ● Limited LOB mode

Max LOB size (kb)* 32

Enable validation □

Enable logging ☑

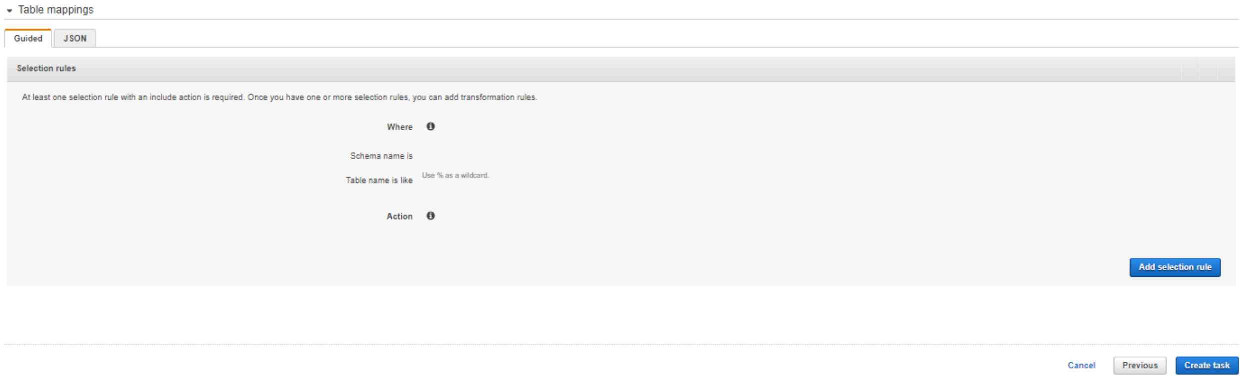


Table mappings

Selection rules

At least one selection rule with an include action is required. Once you have one or more selection rules, you can add transformation rules.

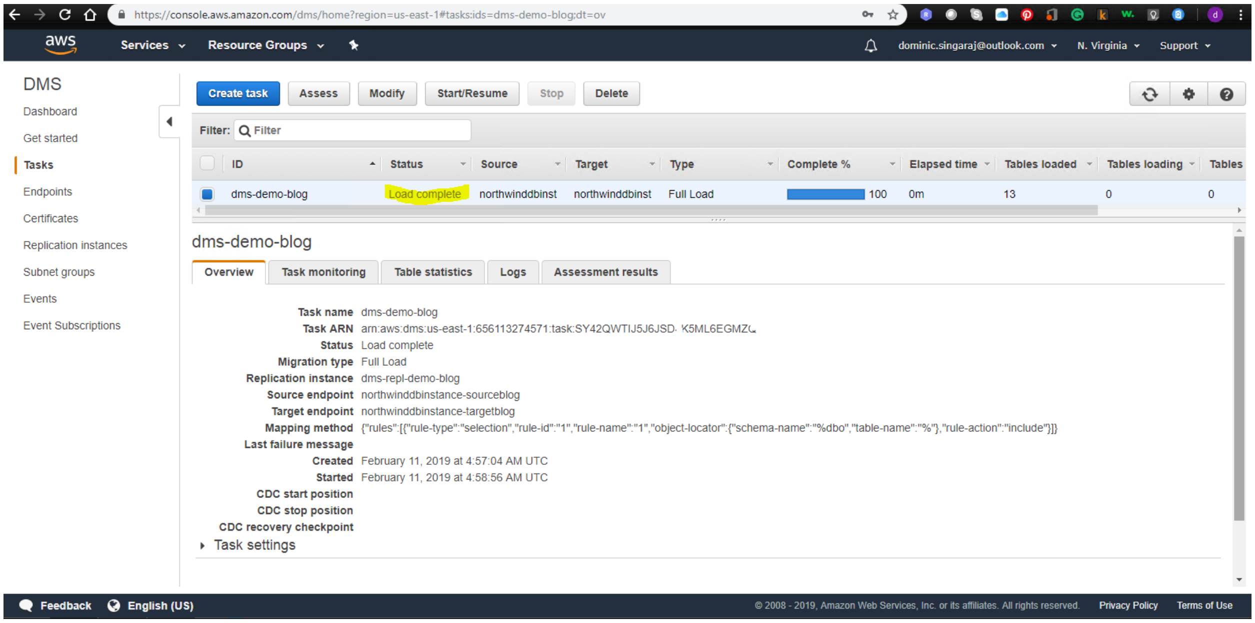
Where

Schema name is

Table name is like Use % as a wildcard.

Action

Once the task has been executed, load status (overview tab) and record counts (table statistics tab) would be displayed like below



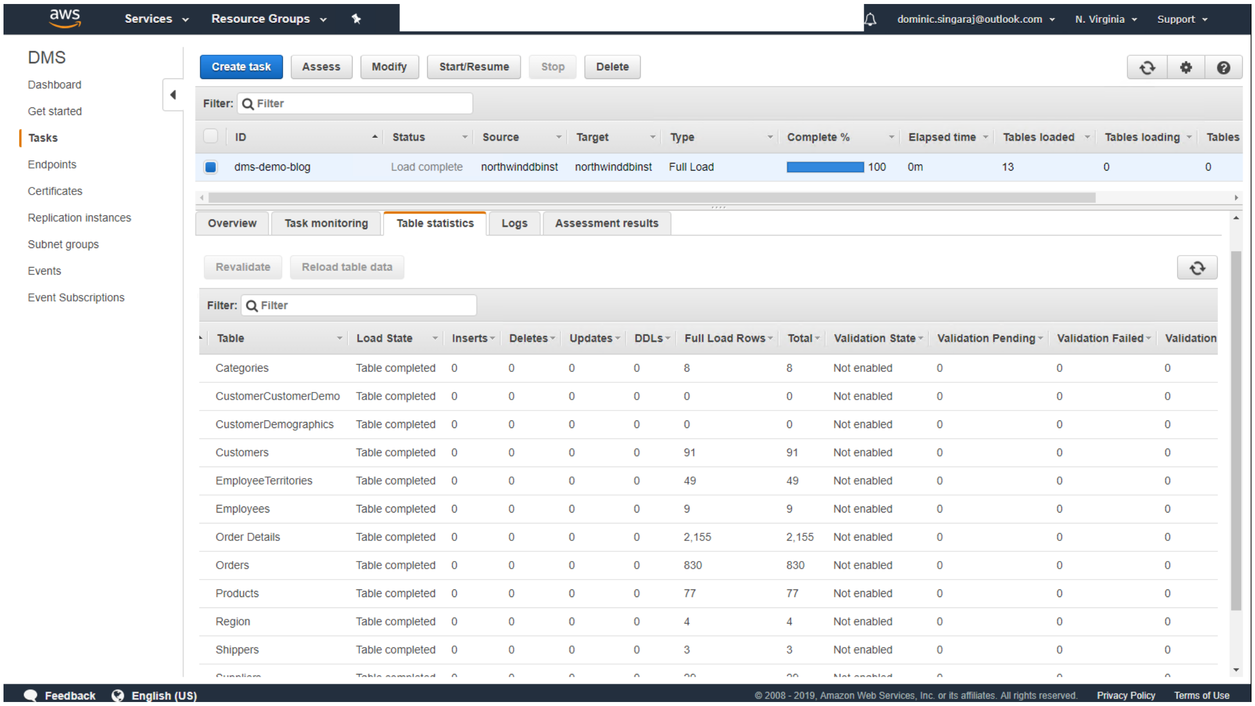
| ID | Status | Source | Target | Type | Complete % | Elapsed time | Tables loaded | Tables loading | Tables |
|---|---|---|---|---|---|---|---|---|---|
| dms-demo-blog | Load complete | northwinddbinst | northwinddbinst | Full Load | 100 | 0m | 13 | 0 | 0 |

dms-demo-blog

Overview | Task monitoring | Table statistics | Logs | Assessment results

| | |
|---|---|
| Task name | dms-demo-blog |
| Task ARN | arn:aws:dms:us-east-1:656113274571:task:SY42QWTIJ5J6JSD· K5ML6EGMZ |
| Status | Load complete |
| Migration type | Full Load |
| Replication instance | dms-repl-demo-blog |
| Source endpoint | northwinddbinstance-sourceblog |
| Target endpoint | northwinddbinstance-targetblog |
| Mapping method | {"rules":[{"rule-type":"selection","rule-id":"1","rule-name":"1","object-locator":{"schema-name":"%dbo","table-name":"%"},"rule-action":"include"}]} |
| Last failure message | |
| Created | February 11, 2019 at 4:57:04 AM UTC |
| Started | February 11, 2019 at 4:58:56 AM UTC |
| CDC start position | |
| CDC stop position | |
| CDC recovery checkpoint | |

Task settings

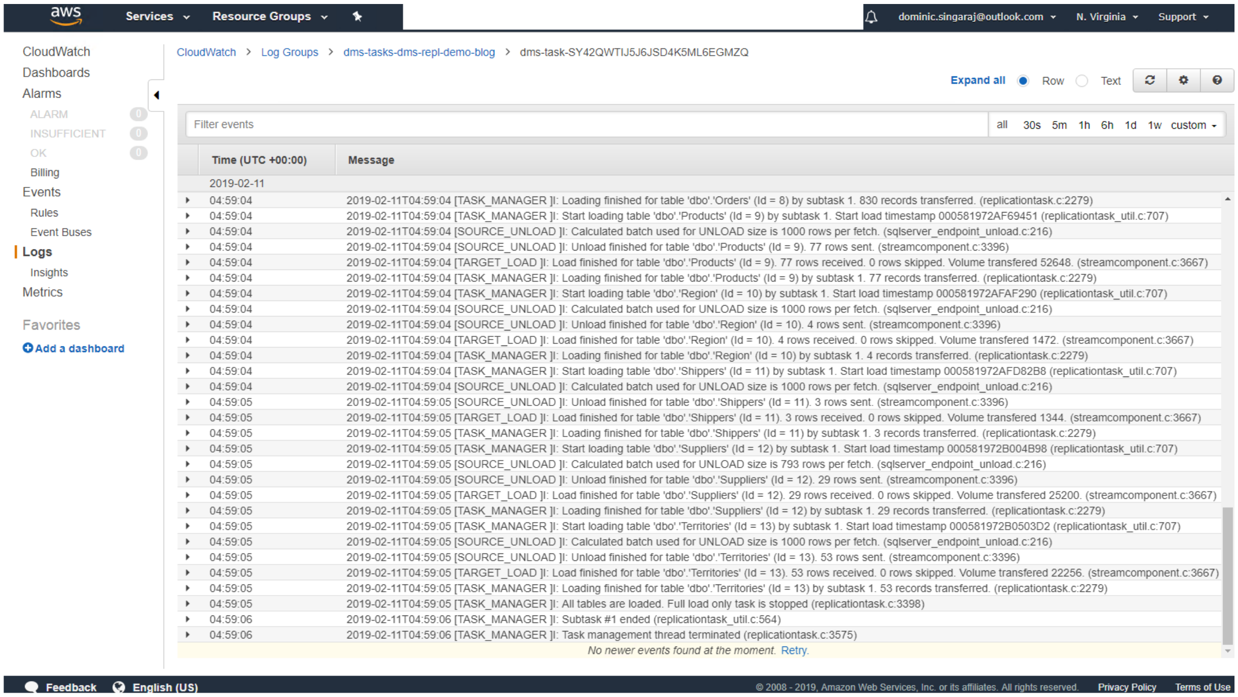


| ID | Status | Source | Target | Type | Complete % | Elapsed time | Tables loaded | Tables loading | Tables |
|---|---|---|---|---|---|---|---|---|---|
| dms-demo-blog | Load complete | northwinddbinst | northwinddbinst | Full Load | 100 | 0m | 13 | 0 | 0 |

Overview | Task monitoring | Table statistics | Logs | Assessment results

| Table | Load State | Inserts | Deletes | Updates | DDLs | Full Load Rows | Total | Validation State | Validation Pending | Validation Failed | Validation |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Categories | Table completed | 0 | 0 | 0 | 0 | 8 | 8 | Not enabled | 0 | 0 | 0 |
| CustomerCustomerDemo | Table completed | 0 | 0 | 0 | 0 | 0 | 0 | Not enabled | 0 | 0 | 0 |
| CustomerDemographics | Table completed | 0 | 0 | 0 | 0 | 0 | 0 | Not enabled | 0 | 0 | 0 |
| Customers | Table completed | 0 | 0 | 0 | 0 | 91 | 91 | Not enabled | 0 | 0 | 0 |
| EmployeeTerritories | Table completed | 0 | 0 | 0 | 0 | 49 | 49 | Not enabled | 0 | 0 | 0 |
| Employees | Table completed | 0 | 0 | 0 | 0 | 9 | 9 | Not enabled | 0 | 0 | 0 |
| Order Details | Table completed | 0 | 0 | 0 | 0 | 2,155 | 2,155 | Not enabled | 0 | 0 | 0 |
| Orders | Table completed | 0 | 0 | 0 | 0 | 830 | 830 | Not enabled | 0 | 0 | 0 |
| Products | Table completed | 0 | 0 | 0 | 0 | 77 | 77 | Not enabled | 0 | 0 | 0 |
| Region | Table completed | 0 | 0 | 0 | 0 | 4 | 4 | Not enabled | 0 | 0 | 0 |
| Shippers | Table completed | 0 | 0 | 0 | 0 | 3 | 3 | Not enabled | 0 | 0 | 0 |
| Suppliers | Table completed | 0 | 0 | 0 | 0 | 29 | 29 | Not enabled | 0 | 0 | 0 |

## Cloud watch log & data migration status



```
Time (UTC +00:00)  Message
2019-02-11
04:59:04  2019-02-11T04:59:04 [TASK_MANAGER ]I: Loading finished for table 'dbo'.'Orders' (Id = 8) by subtask 1. 830 records transferred. (replicationtask.c:2279)
04:59:04  2019-02-11T04:59:04 [TASK_MANAGER ]I: Start loading table 'dbo'.'Products' (Id = 9) by subtask 1. Start load timestamp 000581972AF69451 (replicationtask_util.c:707)
04:59:04  2019-02-11T04:59:04 [SOURCE_UNLOAD ]I: Calculated batch used for UNLOAD size is 1000 rows per fetch. (sqlserver_endpoint_unload.c:216)
04:59:04  2019-02-11T04:59:04 [SOURCE_UNLOAD ]I: Unload finished for table 'dbo'.'Products' (Id = 9). 77 rows sent. (streamcomponent.c:3396)
04:59:04  2019-02-11T04:59:04 [TARGET_LOAD ]I: Load finished for table 'dbo'.'Products' (Id = 9). 77 rows received. 0 rows skipped. Volume transfered 52648. (streamcomponent.c:3667)
04:59:04  2019-02-11T04:59:04 [TASK_MANAGER ]I: Loading finished for table 'dbo'.'Products' (Id = 9) by subtask 1. 77 records transferred. (replicationtask.c:2279)
04:59:04  2019-02-11T04:59:04 [TASK_MANAGER ]I: Start loading table 'dbo'.'Region' (Id = 10) by subtask 1. Start load timestamp 000581972AFAF290 (replicationtask_util.c:707)
04:59:04  2019-02-11T04:59:04 [SOURCE_UNLOAD ]I: Calculated batch used for UNLOAD size is 1000 rows per fetch. (sqlserver_endpoint_unload.c:216)
04:59:04  2019-02-11T04:59:04 [SOURCE_UNLOAD ]I: Unload finished for table 'dbo'.'Region' (Id = 10). 4 rows sent. (streamcomponent.c:3396)
04:59:04  2019-02-11T04:59:04 [TARGET_LOAD ]I: Load finished for table 'dbo'.'Region' (Id = 10). 4 rows received. 0 rows skipped. Volume transfered 1472. (streamcomponent.c:3667)
04:59:04  2019-02-11T04:59:04 [TASK_MANAGER ]I: Loading finished for table 'dbo'.'Region' (Id = 10) by subtask 1. 4 records transferred. (replicationtask.c:2279)
04:59:04  2019-02-11T04:59:04 [TASK_MANAGER ]I: Start loading table 'dbo'.'Shippers' (Id = 11) by subtask 1. Start load timestamp 000581972AFD82B8 (replicationtask_util.c:707)
04:59:04  2019-02-11T04:59:04 [SOURCE_UNLOAD ]I: Calculated batch used for UNLOAD size is 1000 rows per fetch. (sqlserver_endpoint_unload.c:216)
04:59:05  2019-02-11T04:59:05 [SOURCE_UNLOAD ]I: Unload finished for table 'dbo'.'Shippers' (Id = 11). 3 rows sent. (streamcomponent.c:3396)
04:59:05  2019-02-11T04:59:05 [TARGET_LOAD ]I: Load finished for table 'dbo'.'Shippers' (Id = 11). 3 rows received. 0 rows skipped. Volume transfered 1344. (streamcomponent.c:3667)
04:59:05  2019-02-11T04:59:05 [TASK_MANAGER ]I: Loading finished for table 'dbo'.'Shippers' (Id = 11) by subtask 1. 3 records transferred. (replicationtask.c:2279)
04:59:05  2019-02-11T04:59:05 [TASK_MANAGER ]I: Start loading table 'dbo'.'Suppliers' (Id = 12) by subtask 1. Start load timestamp 000581972B004B98 (replicationtask_util.c:707)
04:59:05  2019-02-11T04:59:05 [SOURCE_UNLOAD ]I: Calculated batch used for UNLOAD size is 793 rows per fetch. (sqlserver_endpoint_unload.c:216)
04:59:05  2019-02-11T04:59:05 [SOURCE_UNLOAD ]I: Unload finished for table 'dbo'.'Suppliers' (Id = 12). 29 rows sent. (streamcomponent.c:3396)
04:59:05  2019-02-11T04:59:05 [TARGET_LOAD ]I: Load finished for table 'dbo'.'Suppliers' (Id = 12). 29 rows received. 0 rows skipped. Volume transfered 25200. (streamcomponent.c:3667)
04:59:05  2019-02-11T04:59:05 [TASK_MANAGER ]I: Loading finished for table 'dbo'.'Suppliers' (Id = 12) by subtask 1. 29 records transferred. (replicationtask.c:2279)
04:59:05  2019-02-11T04:59:05 [TASK_MANAGER ]I: Start loading table 'dbo'.'Territories' (Id = 13) by subtask 1. Start load timestamp 000581972B0503D2 (replicationtask_util.c:707)
04:59:05  2019-02-11T04:59:05 [SOURCE_UNLOAD ]I: Calculated batch used for UNLOAD size is 1000 rows per fetch. (sqlserver_endpoint_unload.c:216)
04:59:05  2019-02-11T04:59:05 [SOURCE_UNLOAD ]I: Unload finished for table 'dbo'.'Territories' (Id = 13). 53 rows sent. (streamcomponent.c:3396)
04:59:05  2019-02-11T04:59:05 [TARGET_LOAD ]I: Load finished for table 'dbo'.'Territories' (Id = 13). 53 rows received. 0 rows skipped. Volume transfered 22256. (streamcomponent.c:3667)
04:59:05  2019-02-11T04:59:05 [TASK_MANAGER ]I: Loading finished for table 'dbo'.'Territories' (Id = 13) by subtask 1. 53 records transferred. (replicationtask.c:2279)
04:59:05  2019-02-11T04:59:05 [TASK_MANAGER ]I: All tables are loaded. Full load only task is stopped (replicationtask.c:3398)
04:59:06  2019-02-11T04:59:06 [TASK_MANAGER ]I: Subtask #1 ended (replicationtask_util.c:564)
04:59:06  2019-02-11T04:59:06 [TASK_MANAGER ]I: Task management thread terminated (replicationtask.c:3575)
No newer events found at the moment. Retry.
```

### Conclusion

To summarize, in this article we have seen a way to migrate data from SQL Server installed in the EC2 instance to the RDS-MySQL DB and similar conversion could be done between On-premise DB and AWS supported databases. This article is written with the intention to help the audience who are new to AWS (cloud) data migration. In my next article, I'll be writing about similar data migration exercise with different source & target probably homogenous data migration with some recommended best practices and some focus on high availability, scalability and so on.

### References

https://aws.amazon.com/dms/
https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Welcome.html
https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Installing.html
https://stackoverflow.com/questions/18841744/jdbc-connection-failed-error-tcp-ip-connection-to-host-failed
https://docs.aws.amazon.com/dms/latest/userguide/CHAP_ReplicationInstance.html
https://docs.aws.amazon.com/dms/latest/userguide/Welcome.html
https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.html
https://stackoverflow.com/questions/18841744/jdbc-connection-failed-error-tcp-ip-connection-to-host-failed
Gained AWS knowledge from external trainings like Simplilearn, acloudguru, linux academy