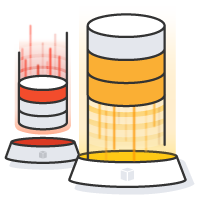
AWS Database Migration Service

AWS Schema Conversion Tool

Oracle to PostgreSQL – Lab Guide



Unni Pillai

Solutions Architect - Big Data

unnik@amazon.com

RUN THIS WORKSHOP IN AP-NORTHEAST-1 (TOKYO)

# Objective

In this lab, you will be performing a migration from Oracle to PostgreSQL using SCT and DMS

High Level Steps

* Create a AWS CloudFormation stack
* Create AWS Database Migration Instances
* Connect to your environment
* Setup AWS Schema Conversion Tool
* Convert the Oracle schema to PostgreSQL
* Create Source Endpoint in AWS DMS
* Create Target Endpoint in AWS DMS
* Create a Migration Task in AWS DMS
* Start the migration
* Generate transactions on Oracle and see the data being migrated to PostgreSQL - CDC



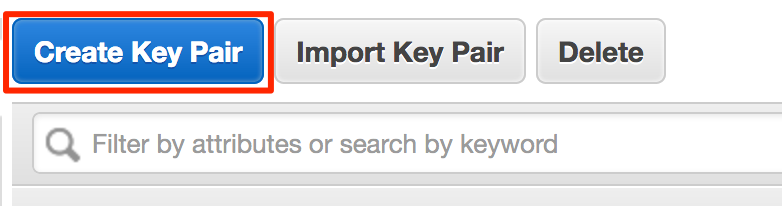
Prerequisites - Generate EC2 key pair

In this step, you will generate an EC2 key pair for use in the Database Migration Workshop labs.

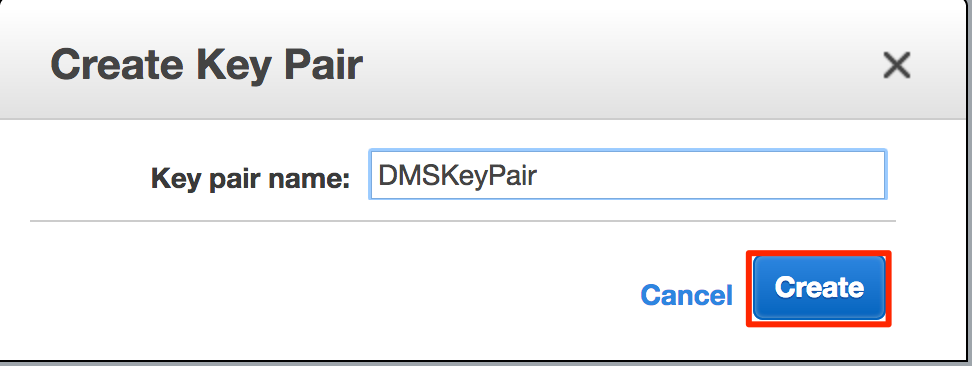
Make sure you are in the right region

|  |
| --- |
| https://ap-northeast-1.console.aws.amazon.com/ec2/v2/home?region=ap-northeast-1#KeyPairs:sort=keyName |

1. Click ‘**Create Key Pair**’



1. Name the key pair ‘**DMSKeyPair**’ and then click ‘**Create**’. At this point, your browser will download a file named ‘**DMSKeyPair.pem**’



1. Download the private key file and store it locally

Prerequisites - Install a SQL Client

* Install a SQL client of your choice; in this lab – we will be using SQLWorkbenchJ screenshots
  + SQL WorkbenchJ: <http://www.sql-workbench.net/downloads.html>
  + DBeaver: <http://dbeaver.jkiss.org/>
  + SQuirrel: <http://squirrel-sql.sourceforge.net/>

Download JDBC Drivers

* Download & keep this file locally
* You will need these to connect to source & target databases using SQL client & AWS Schema Conversation Tool

|  |
| --- |
| https://s3-ap-southeast-1.amazonaws.com/aws-apac-dms-workshop/content/labs/jdbc-drivers/ojdbc7.jar  https://s3-ap-southeast-1.amazonaws.com/aws-apac-dms-workshop/content/labs/jdbc-drivers/postgresql-42.1.1.jar |

Create AWS Cloudformation Stack

In this step, you will launch a AWS Cloudformation template they will setup the following resources needed for this lab.

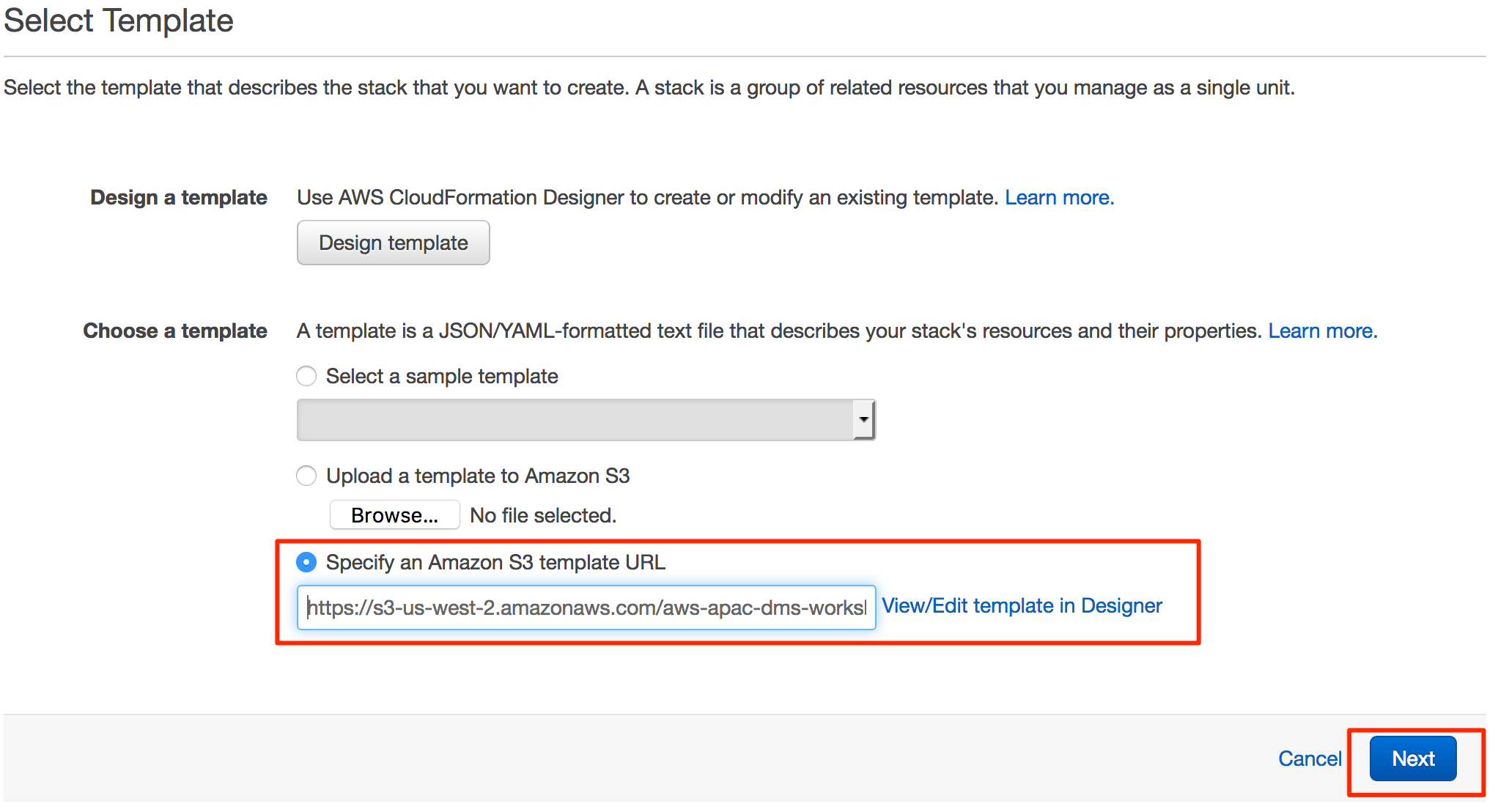
* Source Database: Amazon RDS Oracle (this database will be pre-populated with sample database installed from <https://github.com/awslabs/aws-database-migration-samples>)
* Target Database: Amazon RDS PostgreSQL

## Instructions

* Open the link provided below

|  |
| --- |
| https://ap-northeast-1.console.aws.amazon.com/cloudformation/home?region=ap-northeast-1#/stacks/new?stackName=apac-tech-summit-dms-lab&templateURL=https://s3-ap-southeast-1.amazonaws.com/aws-apac-dms-workshop/content/labs/cloudformation/apac-dms-workshop-cfn.template |

* Click Next



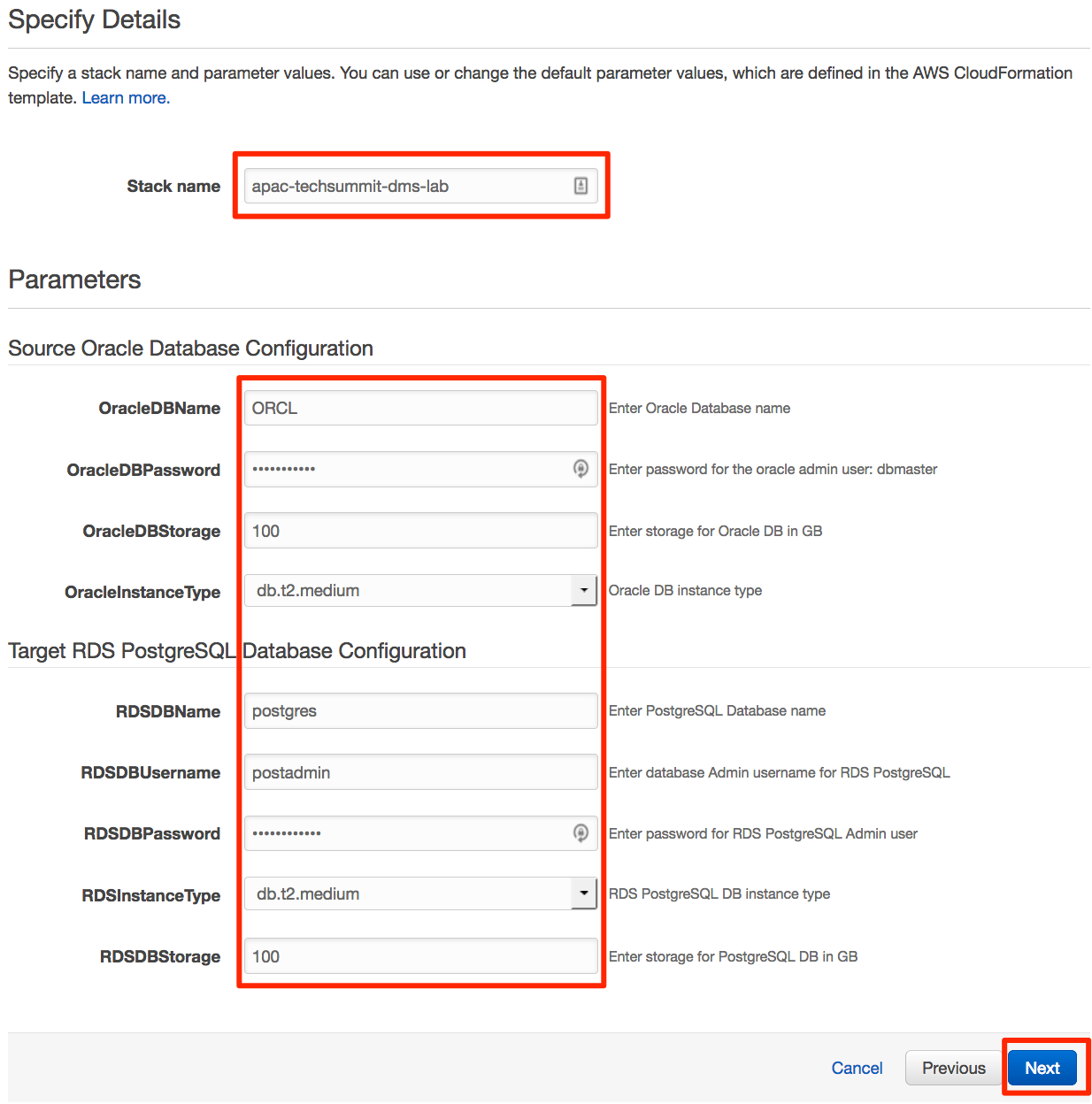
* Enter the following parameters in the ‘Specify Details’ page – *this should be populated by default*
  + Stack name: apac-techsummit-dms-lab
  + Source Oracle Database Configuration

|  |  |
| --- | --- |
| OracleDBName | ORCL |
| OracleDBPassword | oraadmin123 |
| OracleDBStorage | 100 |
| OracleInstanceType | db.t2.medium |

* + Target RDS PostgreSQL Database Configuration

|  |  |
| --- | --- |
| RDSDBName | postgres |
| RDSDBUsername | postadmin |
| RDSDBPassword | postadmin123 |
| RDSInstanceType | db.t2.medium |
| RDSDBStorage | 100 |

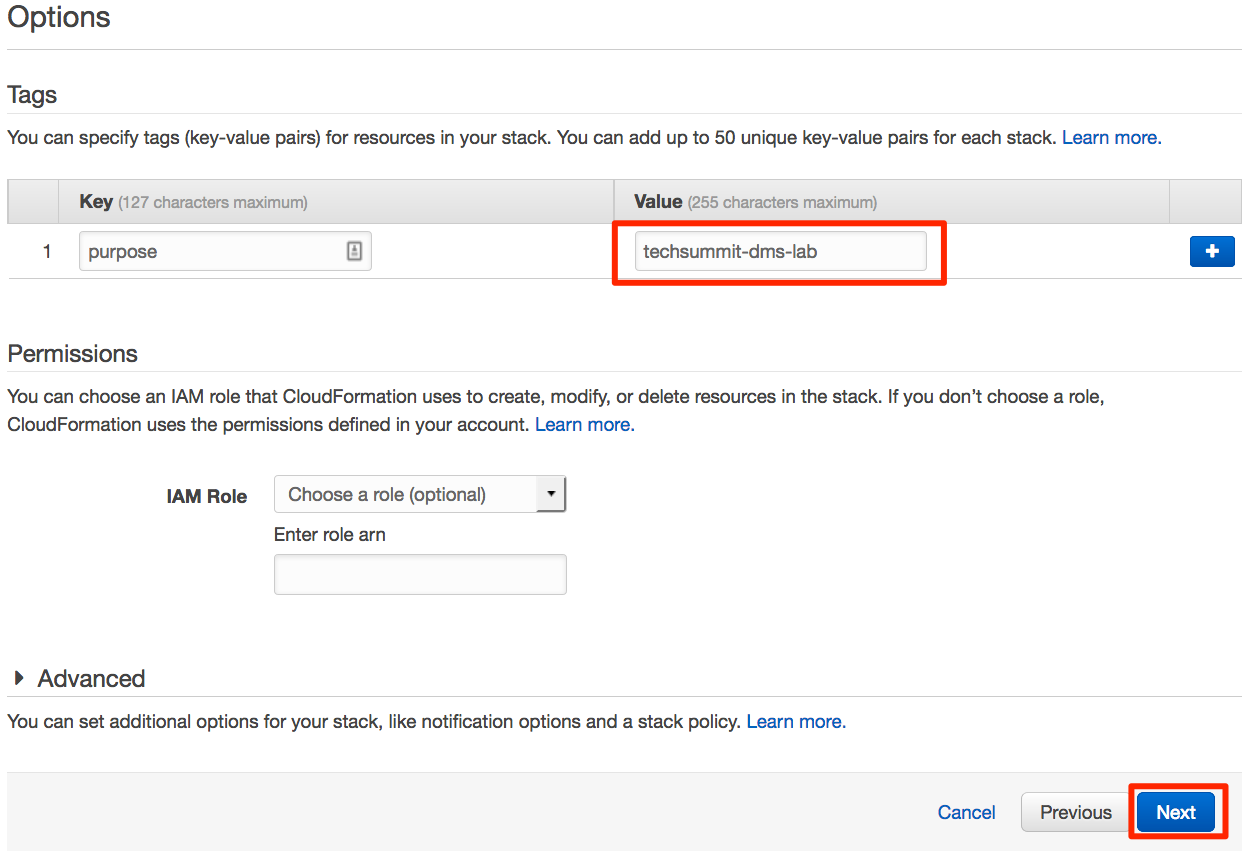
* + Click Next



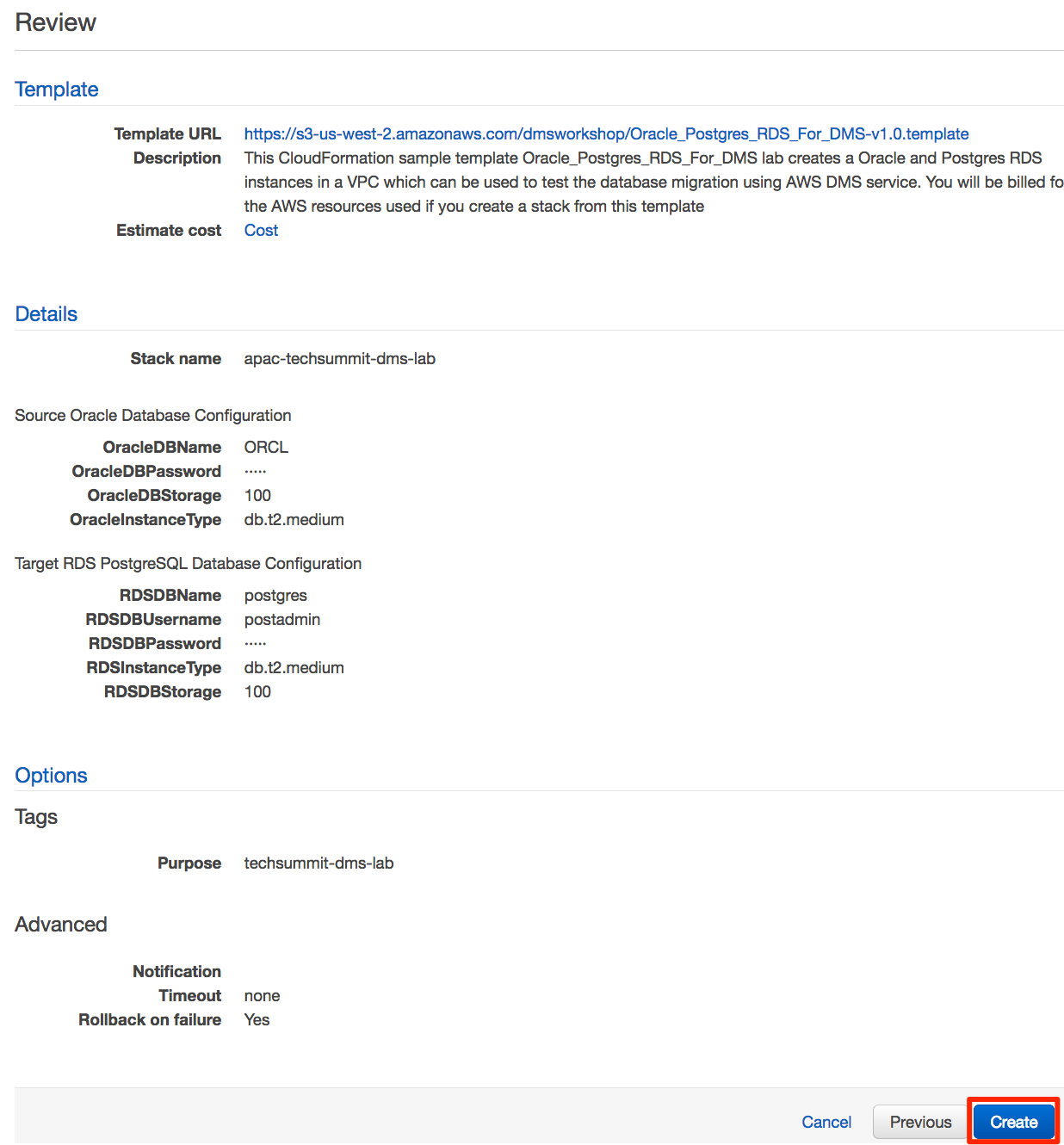
* Tags

|  |  |
| --- | --- |
| Key | purpose |
| Value | techsummit-dms-lab |

* + Click ‘**Next**’



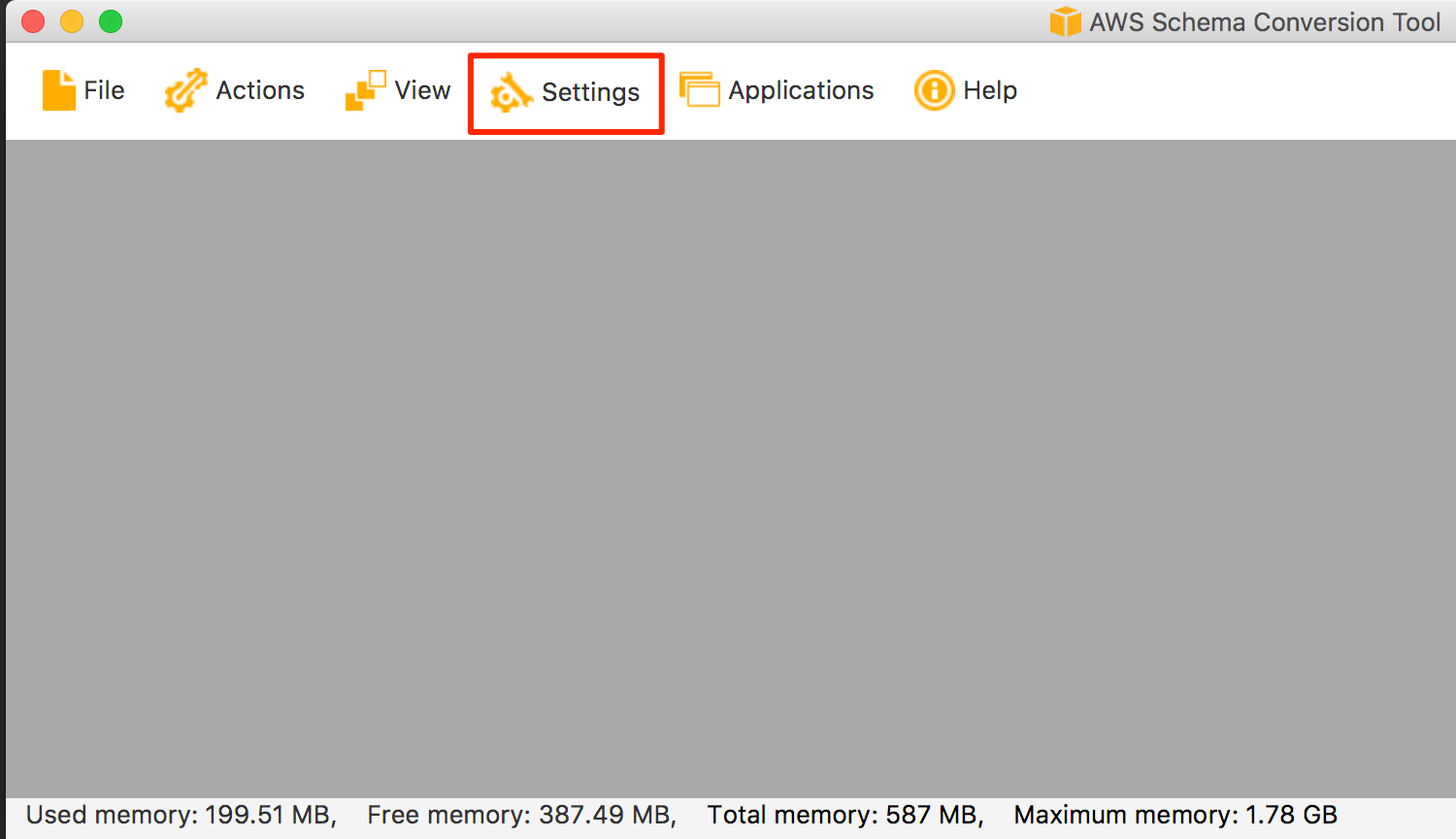
* Review and Click Create



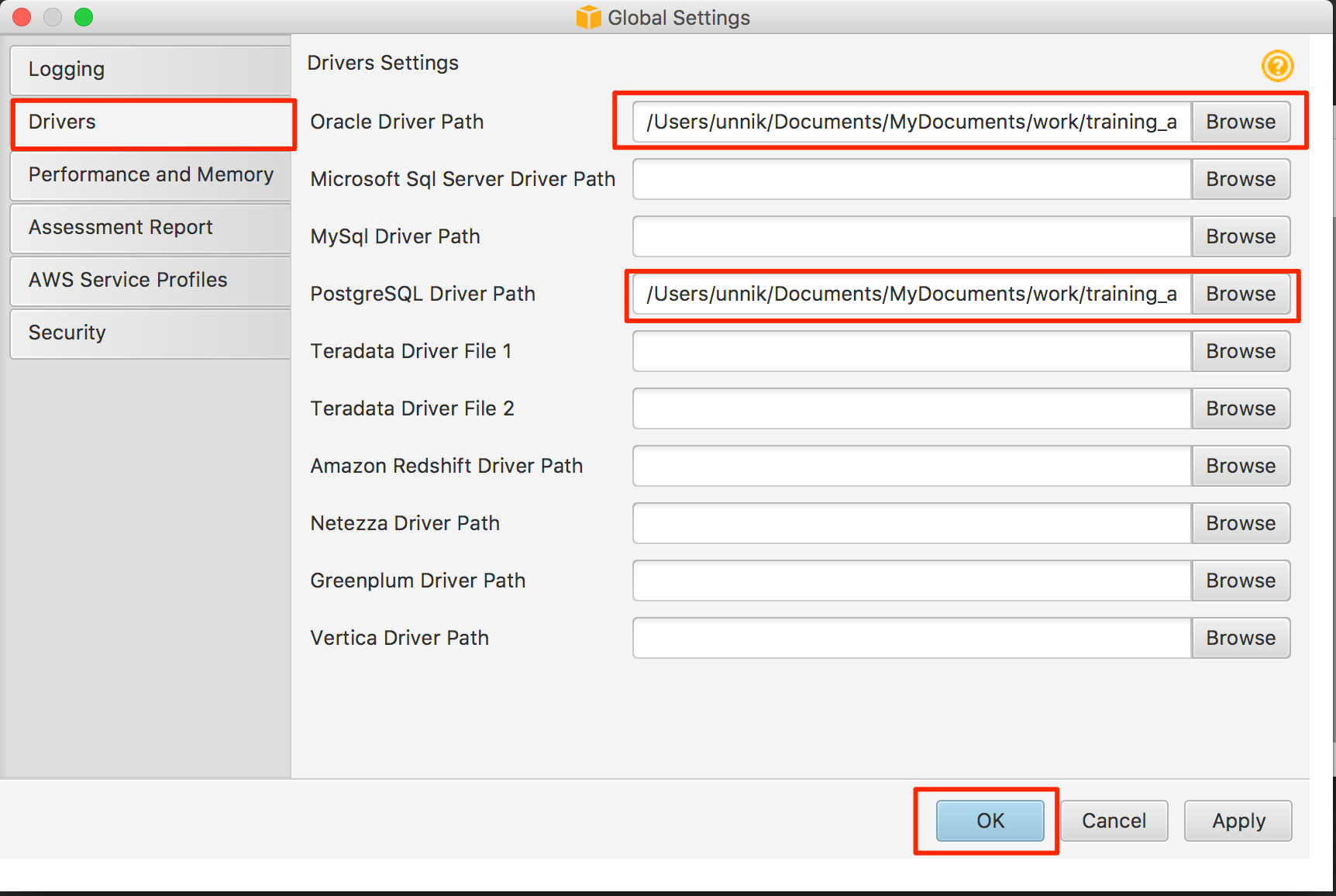
AWS Schema Conversion Tool

# Install AWS Schema Conversion Tool (on your laptop)

* Download the latest version of AWS Schema Conversion Tool (SCT) from this link <http://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_SchemaConversionTool.Installing.html>
* If you already have SCT installed, download the latest version & install it
* JDBC Drivers
  + For connecting to our source (Oracle) vs target (PostgreSQL) you will need the respective JDBC drivers
    - Oracle JDBC driver: <http://bit.ly/2phVpPk>->
    - PostgreSQL JDBC driver: <http://bit.ly/2pt04ZT>->
* Configure SCT with drivers
  + Click on **Settings** > **Global Settings**



* + In the Global Settings window,
    - Goto **Drivers** on left panel
    - Oracle Driver Path: Select the downloaded ojdbc jar file
    - PostgreSQL Driver Path: Select the downloaded postgresql jar file
    - Click **OK** to Proceed



Open Security Groups from Source / Target Database Instances

For you to access Source and Target databases, you will have to add your laptop to Oracle & Postgres Security Groups

|  |
| --- |
| https://ap-northeast-1.console.aws.amazon.com/ec2/v2/home?region=ap-northeast-1#SecurityGroups:sort=groupId |

Add following inbound rule to respective security groups

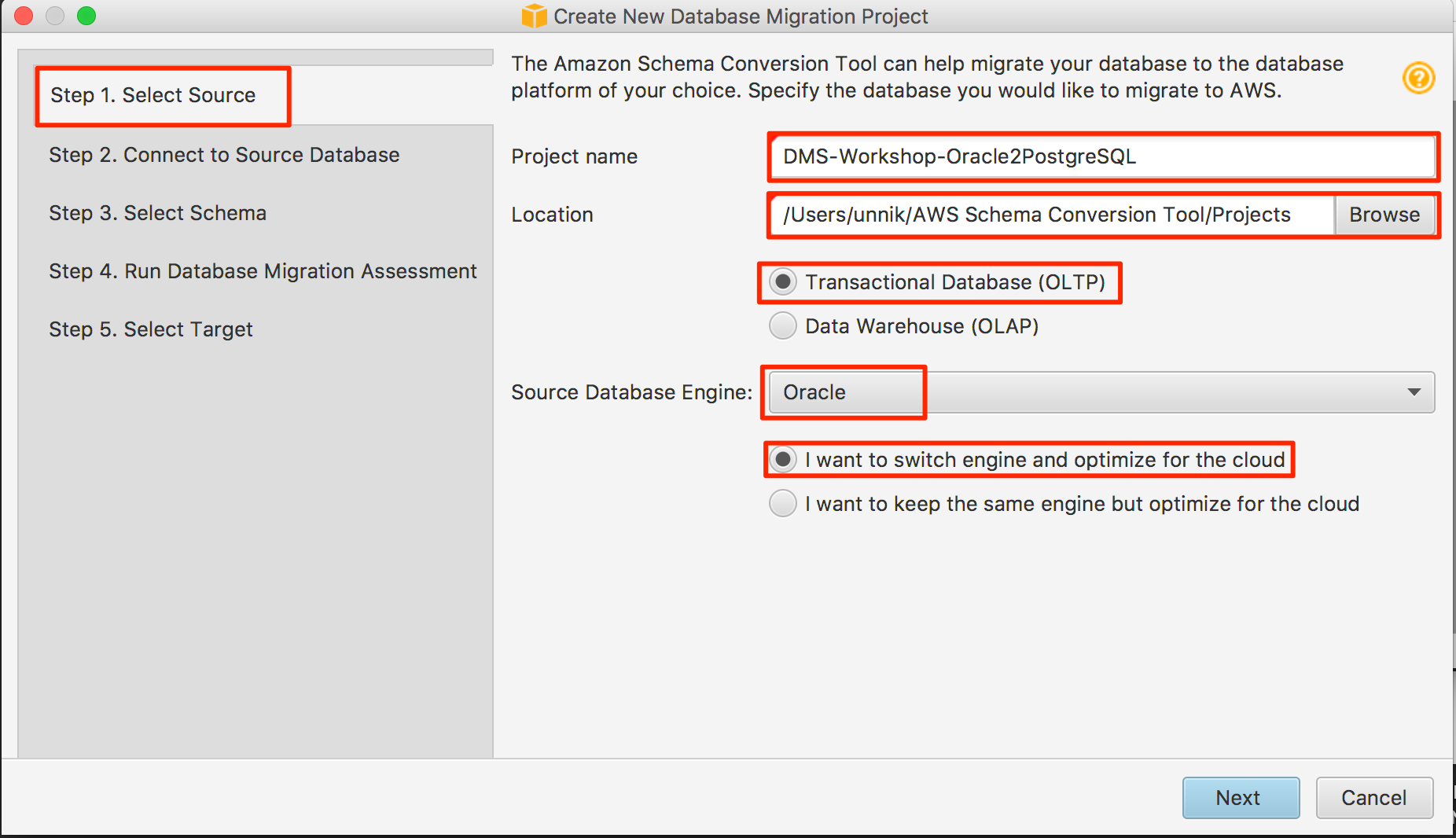
* Oracle Port – 1521 > Open to ‘My IP’
* Postgres Port – 5432 > Open to ‘My IP’

Create a new SCT project

* In AWS SCT, select File > New Project Wizard
* Step 1 – Select Source

|  |  |
| --- | --- |
| Project Name | DMS-Workshop-Oracle2PostgreSQL |
| Location | Leave Default |
| Source Database Engine | Transactional Database (OLTP)  Oracle  I want to switch engine and optimize for the cloud |

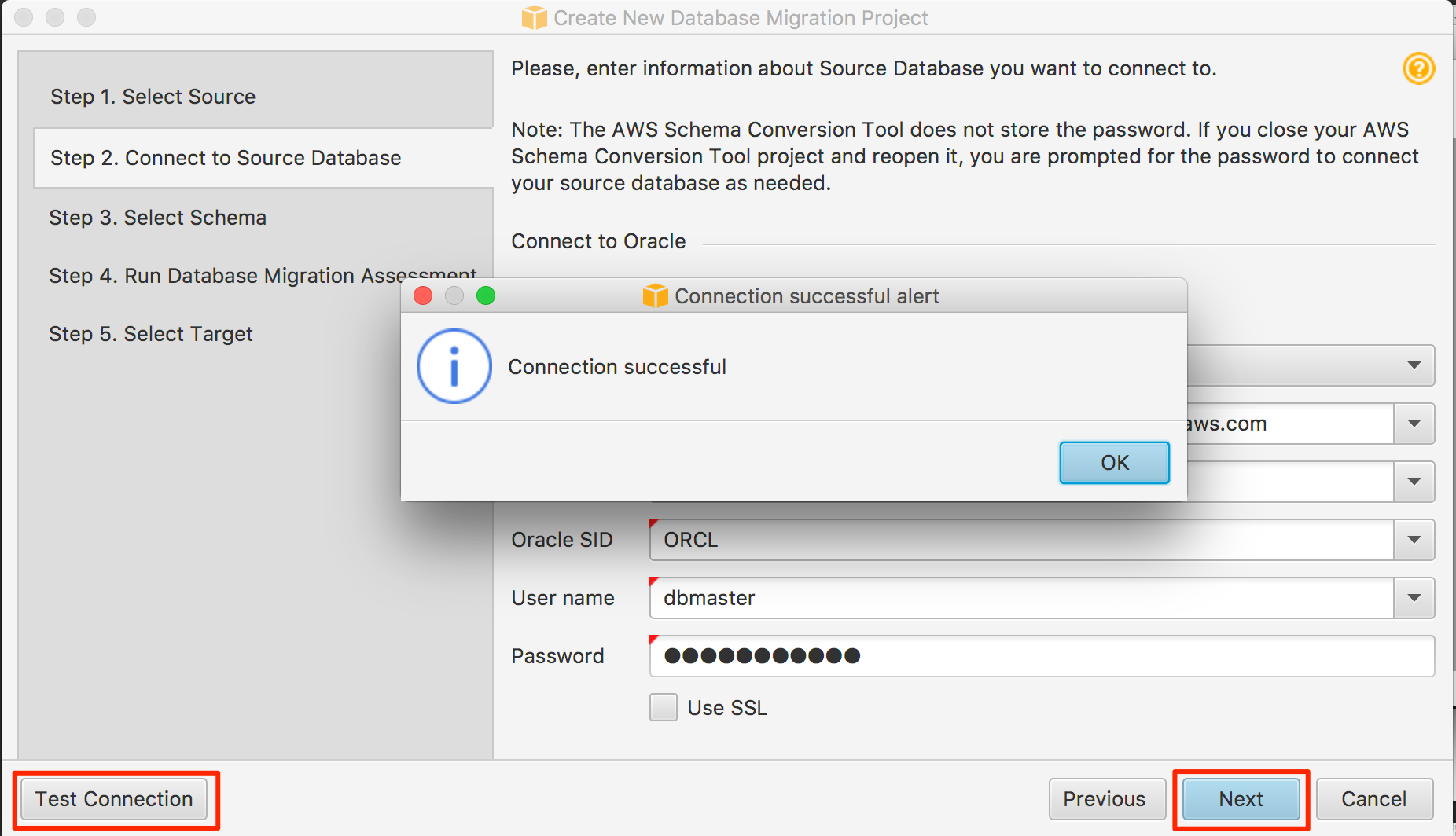
* Click Next to proceed



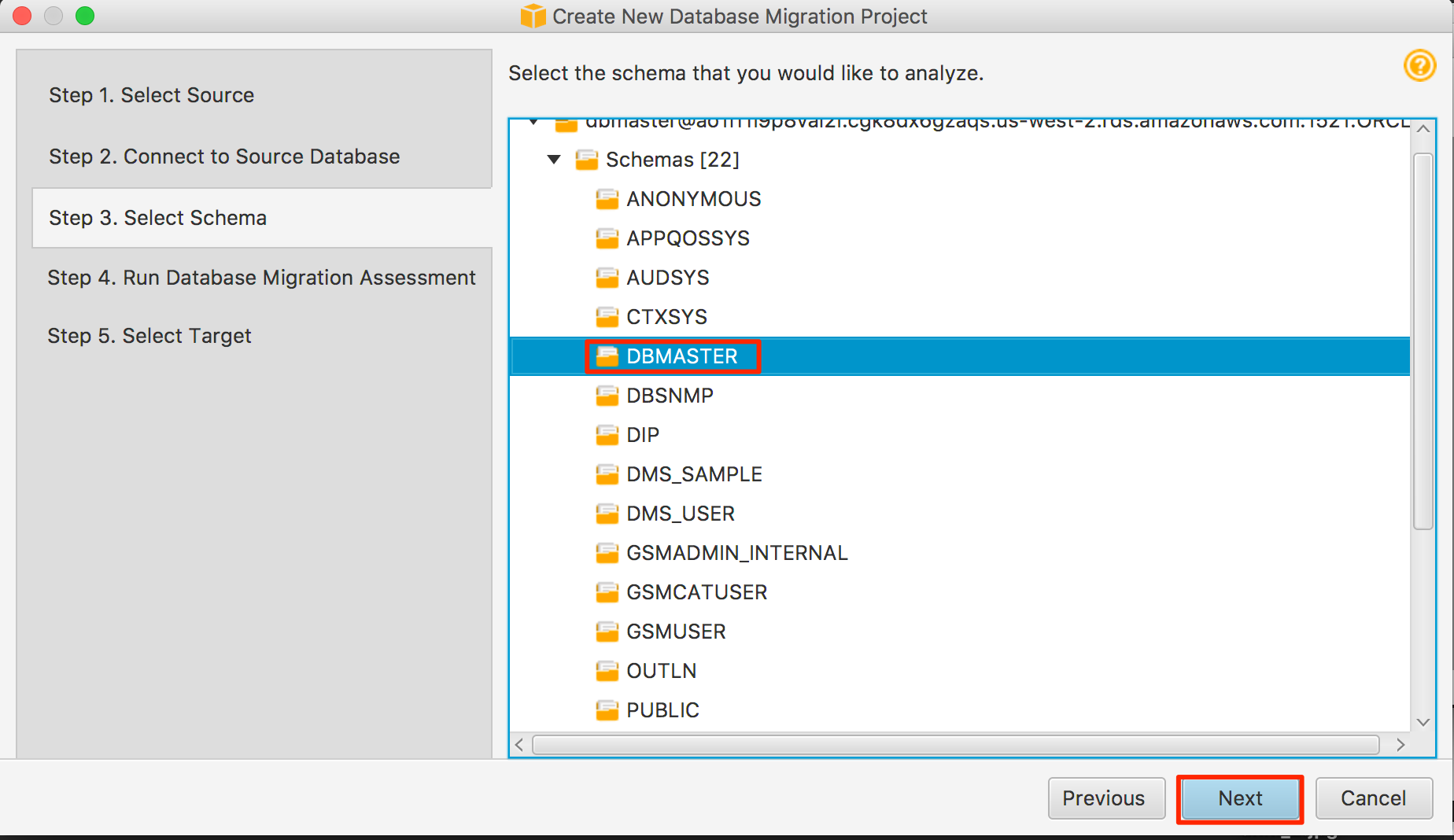
* Step 2 – Connect to Source Database

|  |  |
| --- | --- |
| Type | SID |
| Server Name | DNS name of your Oracle RDS instance |
| Server Port | 1521 |
| Oracle SID | ORCL |
| User name | dbmaster |
| Password | oraadmin123 |

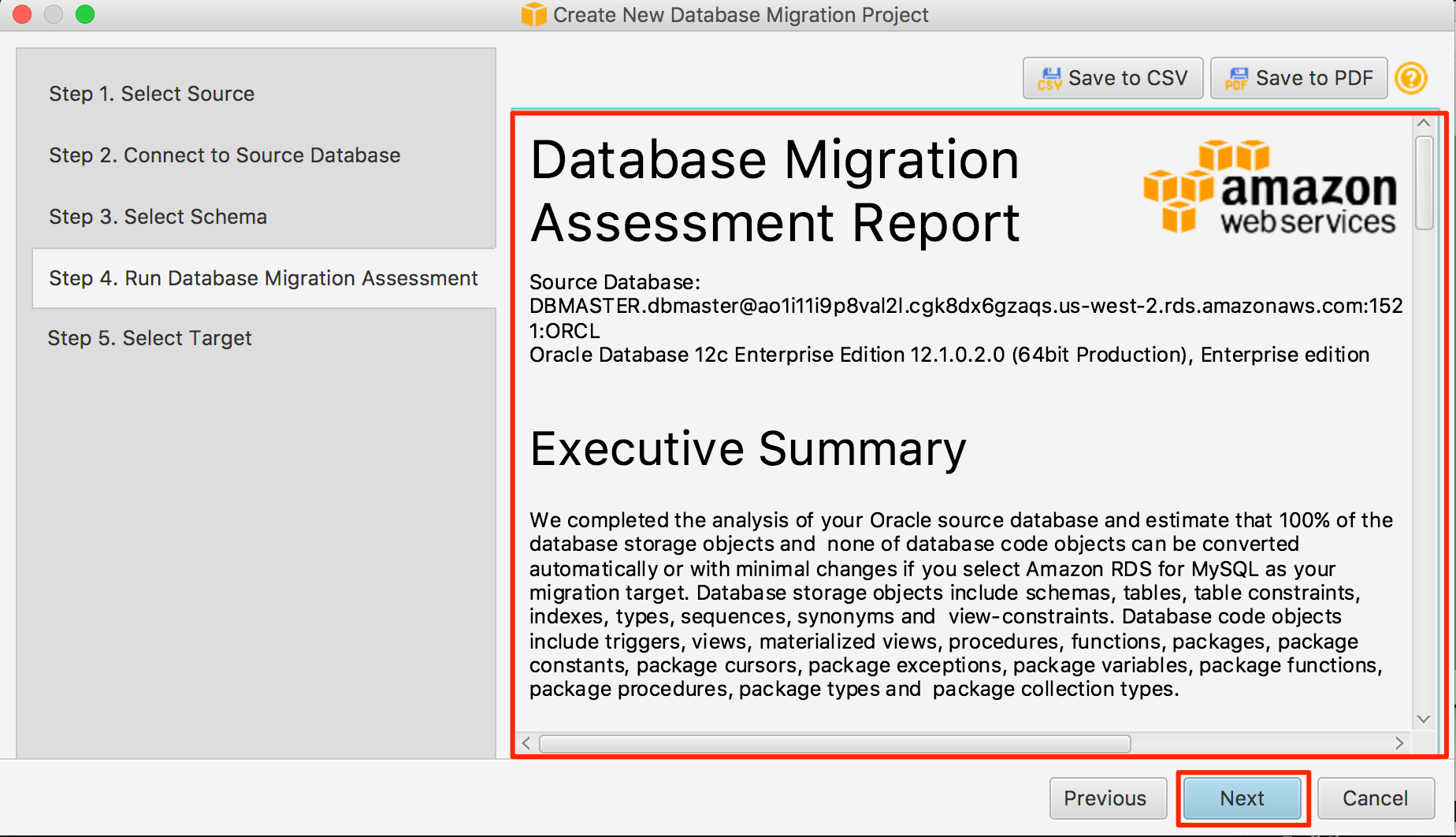
* + Click ‘**Test Connection**’ – Make sure you get a ‘**Connection Successful**’ message
  + Click ‘**Next’** to proceed



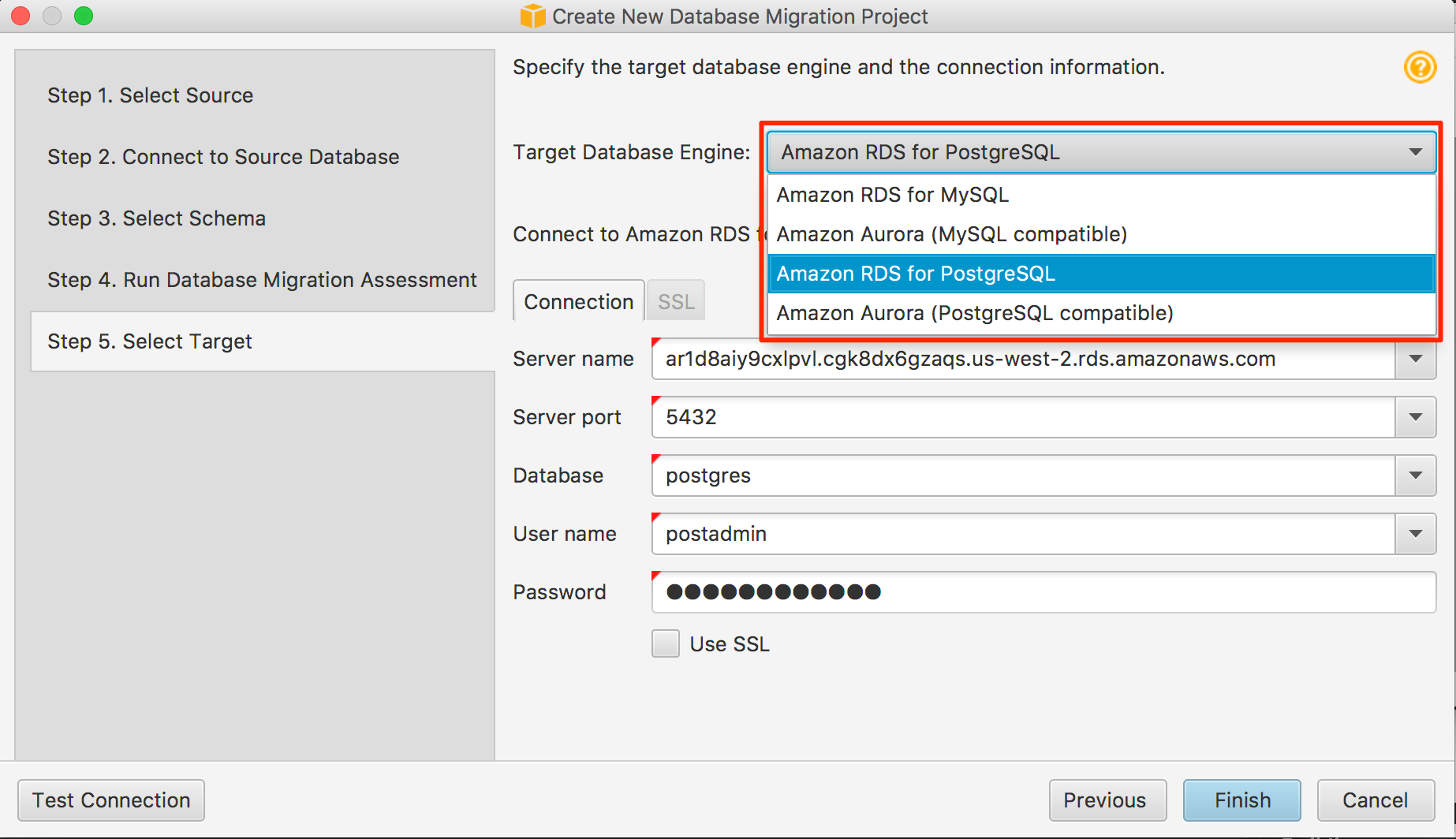
* Step 3 – Select Schema
  + Select ‘**DBMASTER’** as a schema for SCT to analyze
  + Click ‘**Next**’ to proceed



* Step 4 – Run Database Migration Assessment
  + Review the database migration – read through the report
  + Click ‘Next’ to proceed

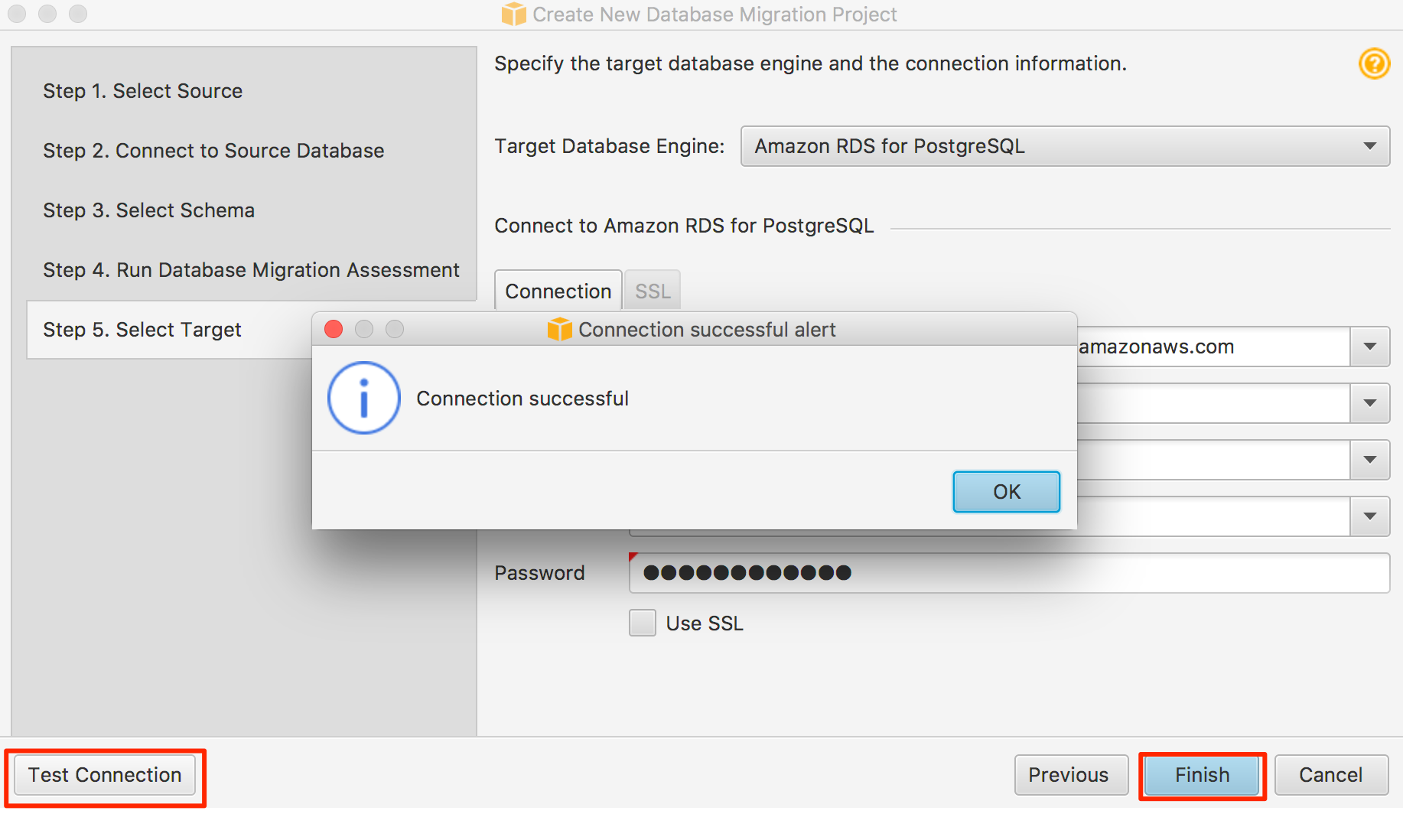


* Step 5 – Select Target
  + Target Database Engine: Amazon RDS for PostgreSQL



|  |  |
| --- | --- |
| Server Name | DNS name of your RDS PostgreSQL instance |
| Database | postgres |
| User name | postadmin |
| Password | postadmin123 |

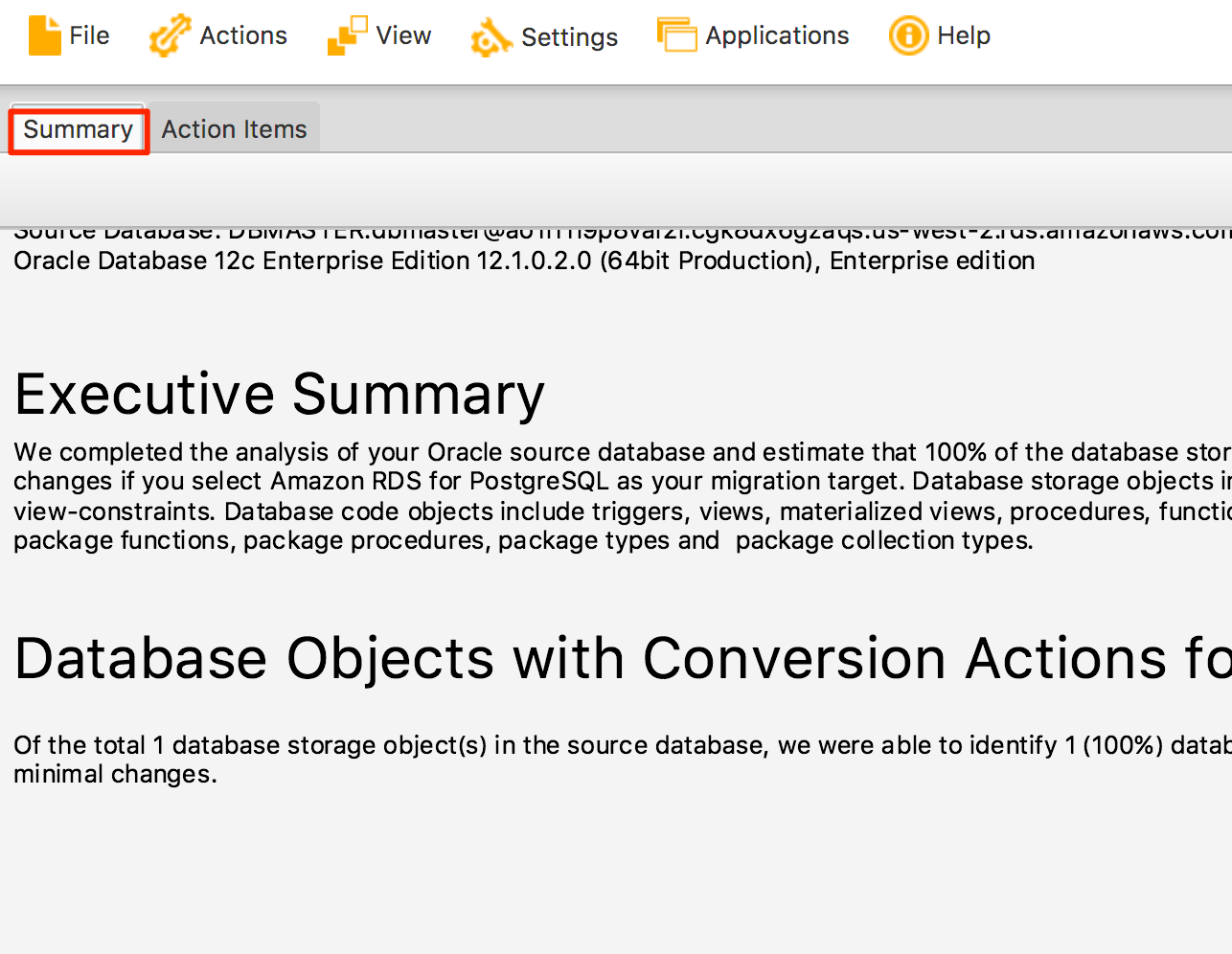
* + Click ‘Test Connection’ – Make sure you get a ‘Connection Successful’ message
  + Click ‘Finish’ to proceed



Run Schema Conversation In SCT

Review the project screen and familiarize yourself

* Uncheck all schemas on the left except for the DMS\_SAMPLE schema.
* Click ‘**Actions’** > ‘**Create Report**’
* Go to the ‘**Summary'** tab on the top and review the generated report



* Look through what Oracle objects could be automatically converted and what could not be. Now, right click and click “**Convert schema**”. The schema will be converted and shown on the PostgreSQL instance (it has not been applied yet).
* Take a few minutes to review the objects being converted.
* Since the majority of the objects which could not be converted are secondary objects like functions or procedures, right click on the created schema on the **Right Panel** and click “**Apply to database**”. This will apply all those converted objects in the PostgreSQL target.
* The above steps will convert all your Oracle objects into PostgreSQL objects. Objects which could not be converted automatically must be taken care of manually after migration at a later time.
* At this point, most of the objects from your source Oracle databased has been converted to PostgreSQL target

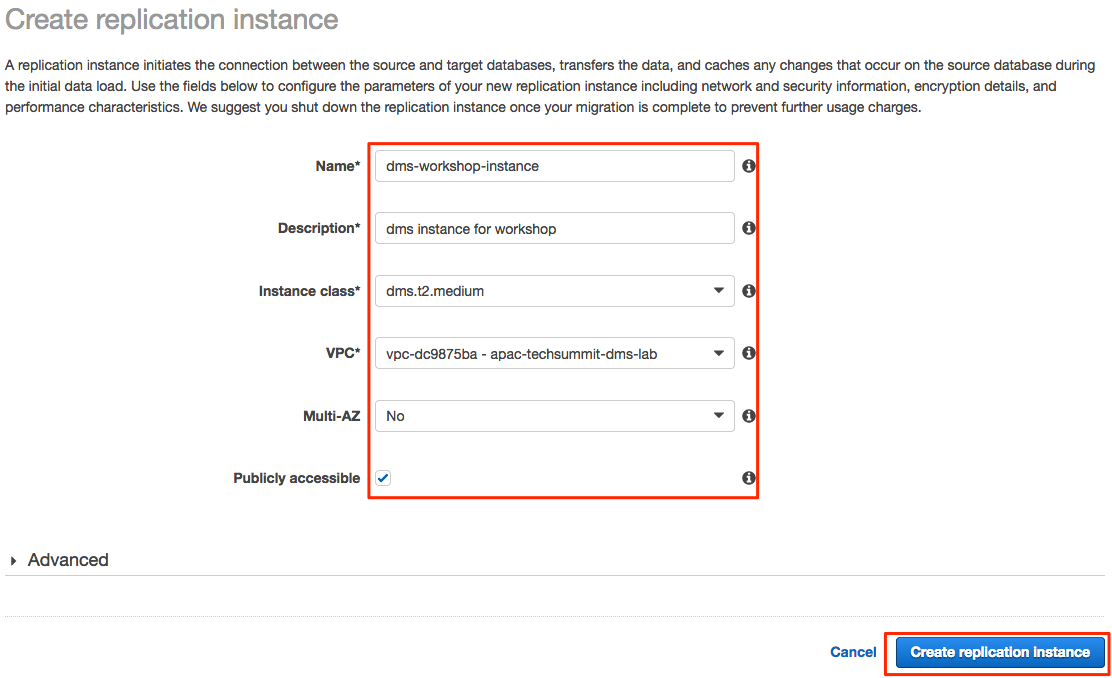
Database migration Service

# Create Database Migration Instance

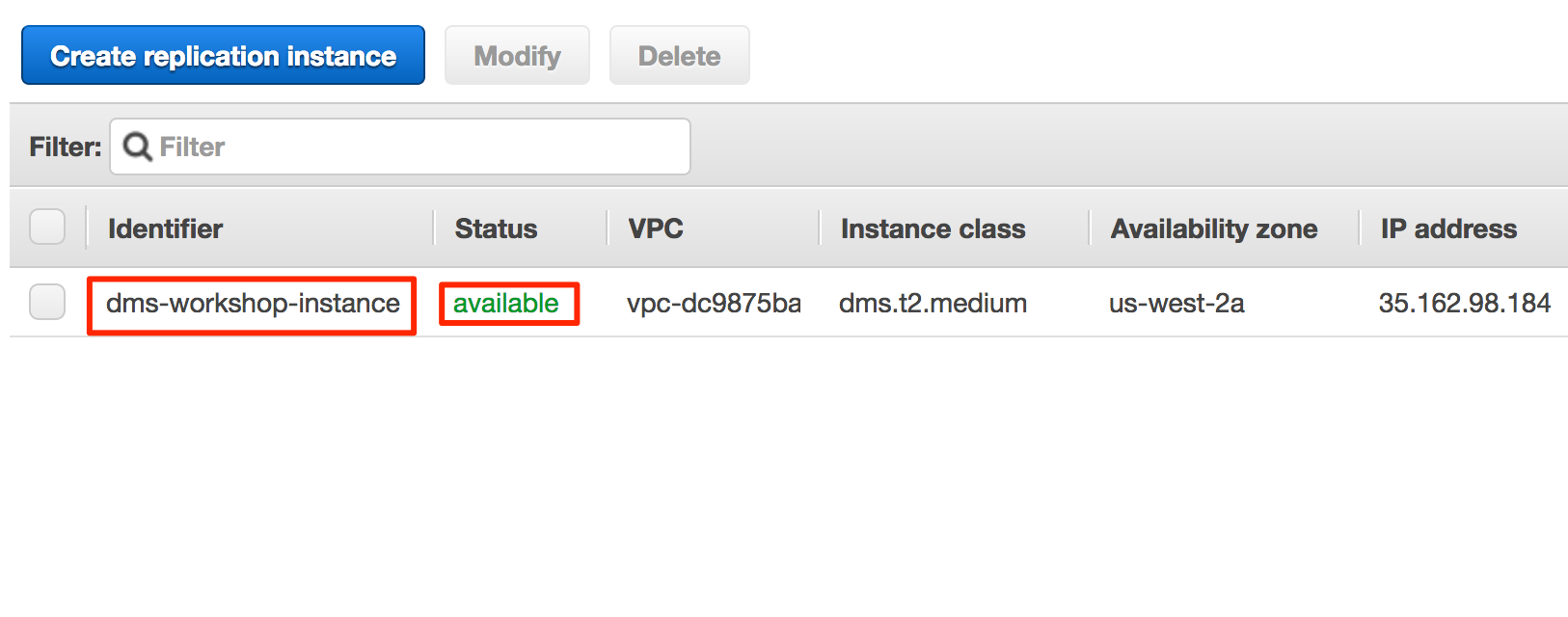
* Navigate to: https://ap-northeast-1.console.aws.amazon.com/dms/home?region=ap-northeast-1#replication-instances:
* Click on ‘**Create Replication Instance**’
* Populate the following values on this page

|  |  |
| --- | --- |
| Name | dms-workshop-instance |
| Description | dms instance for workshop |
| Instance Class | dms.t2.medium |
| VPC | apac-techsummit-dms-lab |
| Multi-AZ | No |
| Publicly accessible | Checked |

* + Click ‘**Create Replication Instance**’ to proceed



Wait for a couple of minutes for the migration instance to start and change the status to ‘**available’**



# Create Source / Target Endpoints

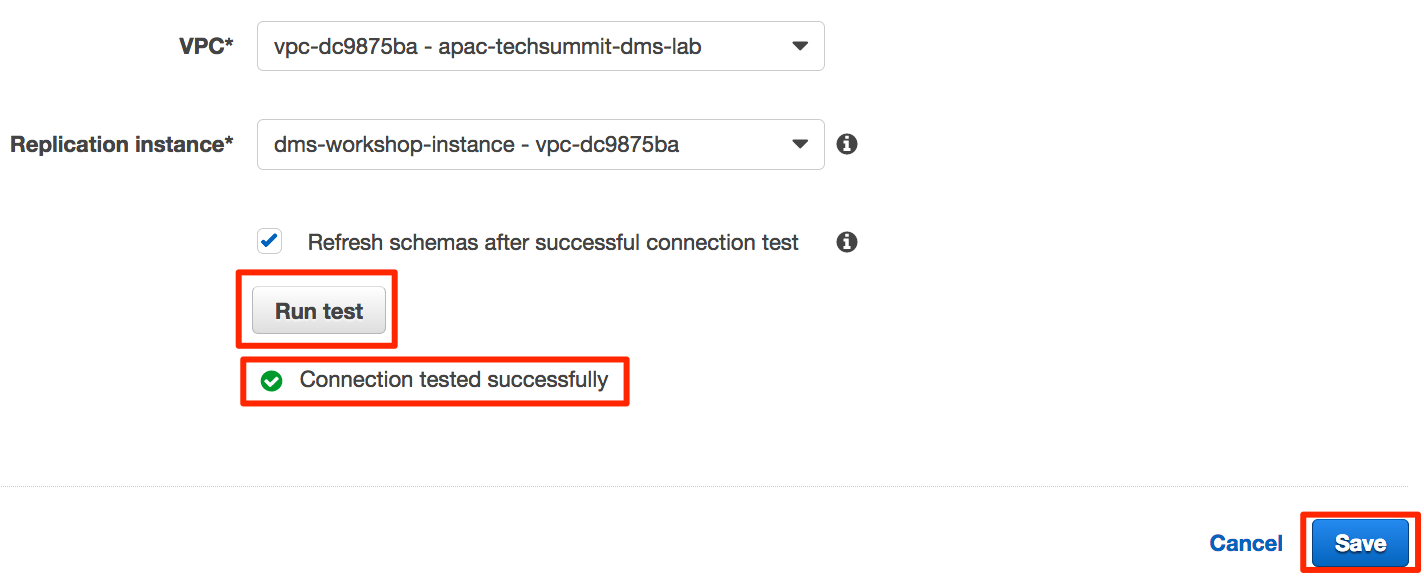
## Create Source Endpoint

* Navigate to: https://ap-northeast-1.console.aws.amazon.com/dms/home?region=ap-northeast-1#endpoints:
* Click on ‘**Create Endpoint**’
* Enter these Details

|  |  |
| --- | --- |
| Endpoint Type | Source |
| Endpoint identifier | dms-workshop-oracle |
| Source engine: | oracle |
| Server name | <oralce-rds-dns-endpoint>  get this from here https://ap-northeast-1.console.aws.amazon.com/rds/home?region=ap-northeast-1#dbinstances |
| Port | 1521 |
| SSL Mode | none |
| User name | dbmaster |
| Password | oraadmin123 |
| SID | ORCL |
| VPC | apac-techsummit-dms-lab |
| Replication instance | dms-workshop-instance |
| Refresh schemas after successful connection test | Checked |



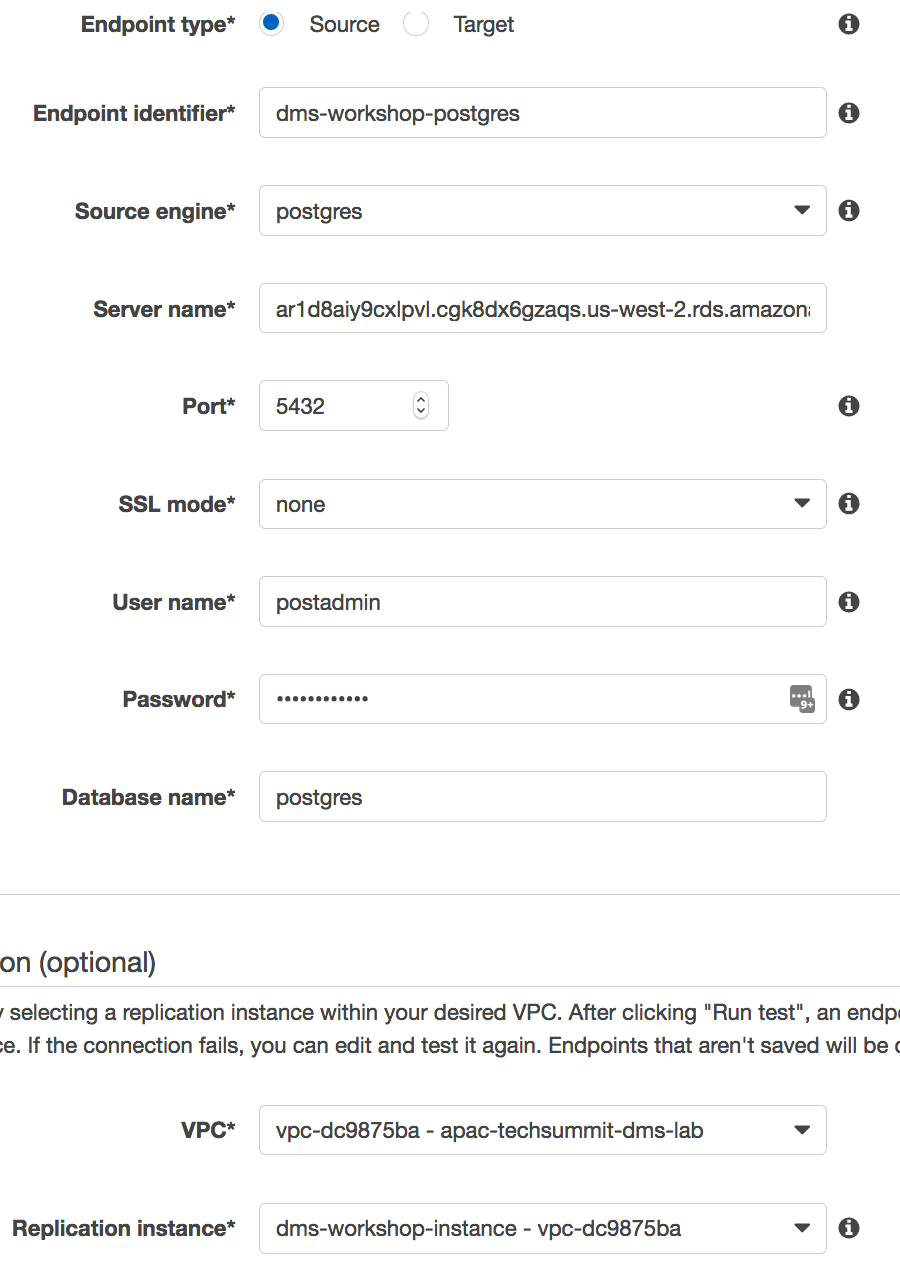
* + Click 'Run Test' -> ensure you get the ‘Connection tested successfully’ message
  + Click on ‘Save’ to proceed



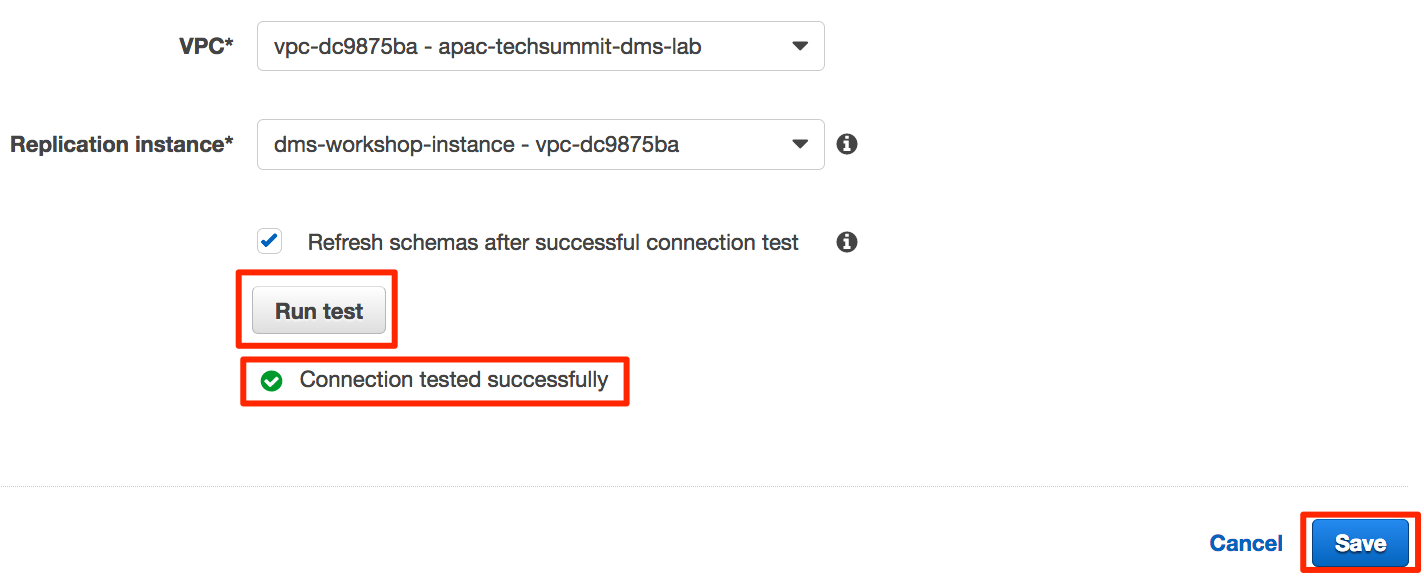
## Create Target Endpoint

* Navigate to: https://ap-northeast-1.console.aws.amazon.com/dms/home?region=ap-northeast-1#endpoints
* Click on ‘**Create Endpoint**’
* Enter these Details

|  |  |
| --- | --- |
| Endpoint Type | Target |
| Endpoint identifier | dms-workshop- postgres |
| Source engine: | postgres |
| Server name | < postgres-rds-dns-endpoint>  get this from here https://ap-northeast-1.console.aws.amazon.com/rds/home?region=ap-northeast-1#dbinstances |
| Port | 5432 |
| SSL Mode | none |
| User name | postadmin |
| Password | postadmin123 |
| Database Name | postgres |
| VPC | apac-techsummit-dms-lab |
| Replication instance | dms-workshop-instance |
| Refresh schemas after successful connection test | Checked |



* + Click 'Run Test' -> ensure you get the ‘Connection tested successfully’ message
  + Click on ‘Save’ to proceed



* Once all both source and target database endpoints have been created and successfully tested, you can proceed to the next step.

Create DMS Migration Task

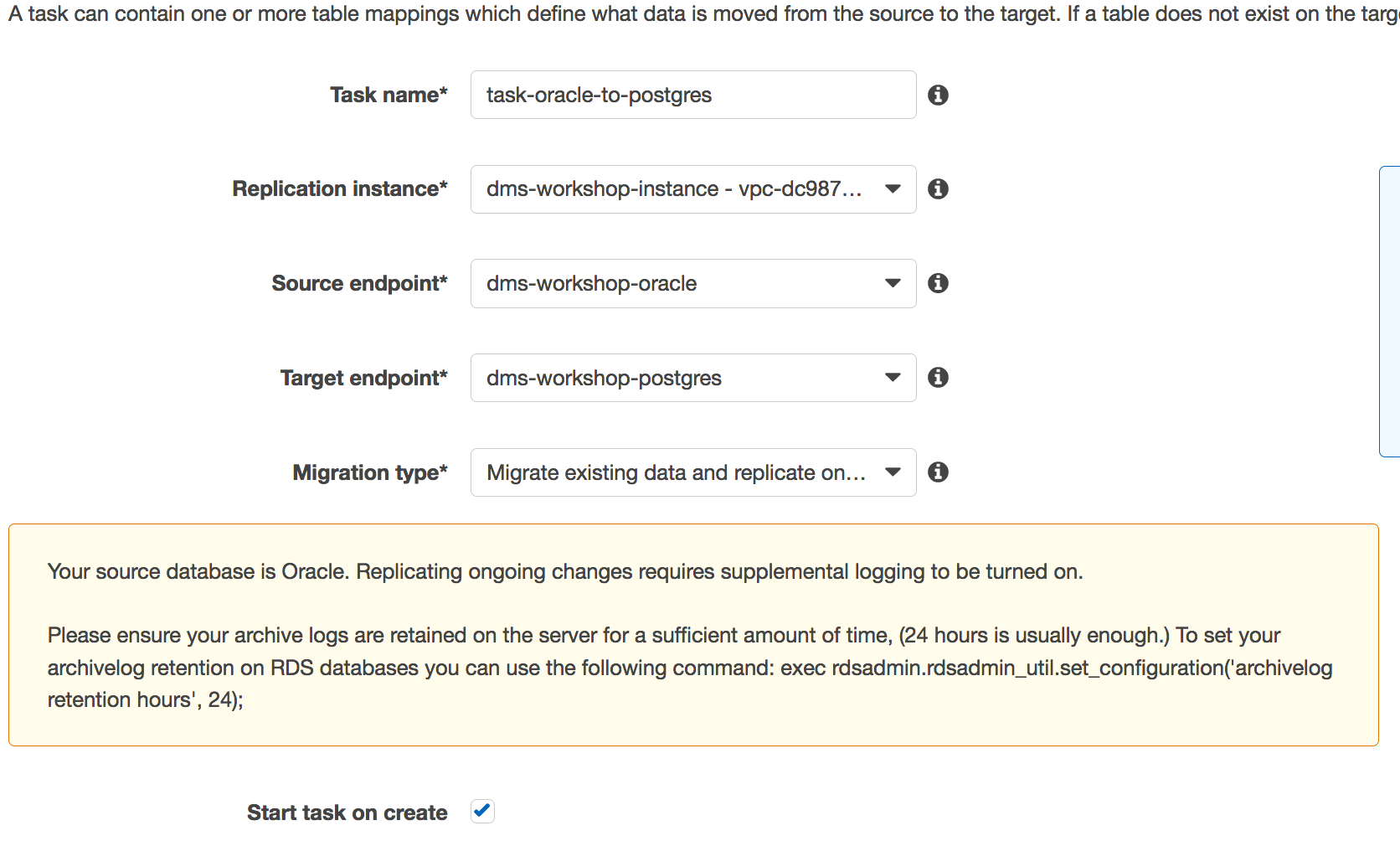
* Navigate to https://ap-northeast-1.console.aws.amazon.com/dms/home?region=ap-northeast-1#tasks:
* Click on ‘**Create Task**’

Enter these Details

* Basic Info

Make sure your configuration looks like the image below

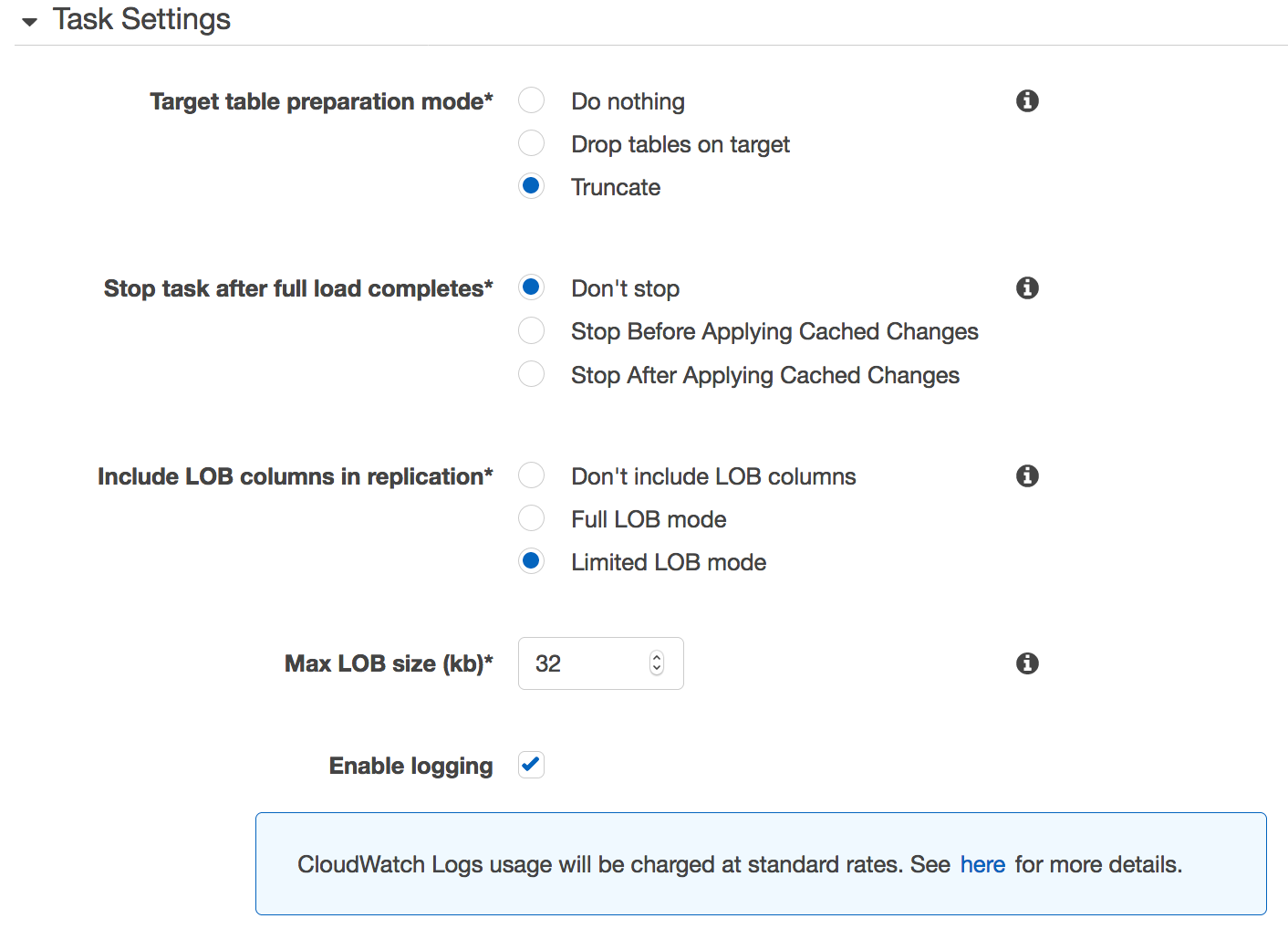
|  |  |
| --- | --- |
| Replication instance | dms-workshop-instance |
| Source endpoint | dms-workshop-oracle |
| Target endpoint | dms-workshop-postgres |
| Migration type | Migrate existing data and replicate ongoing changes |
| Start task on create | Checked |



* Task Settings

|  |  |
| --- | --- |
| Target table preparation mode | Truncate |
| Stop task after full load completes | Don't Stop |
| Include LOB columns in replication | Limited LOB Mode |
| Max LOB size (kb) | 32KB |
| Enable logging | Checked |

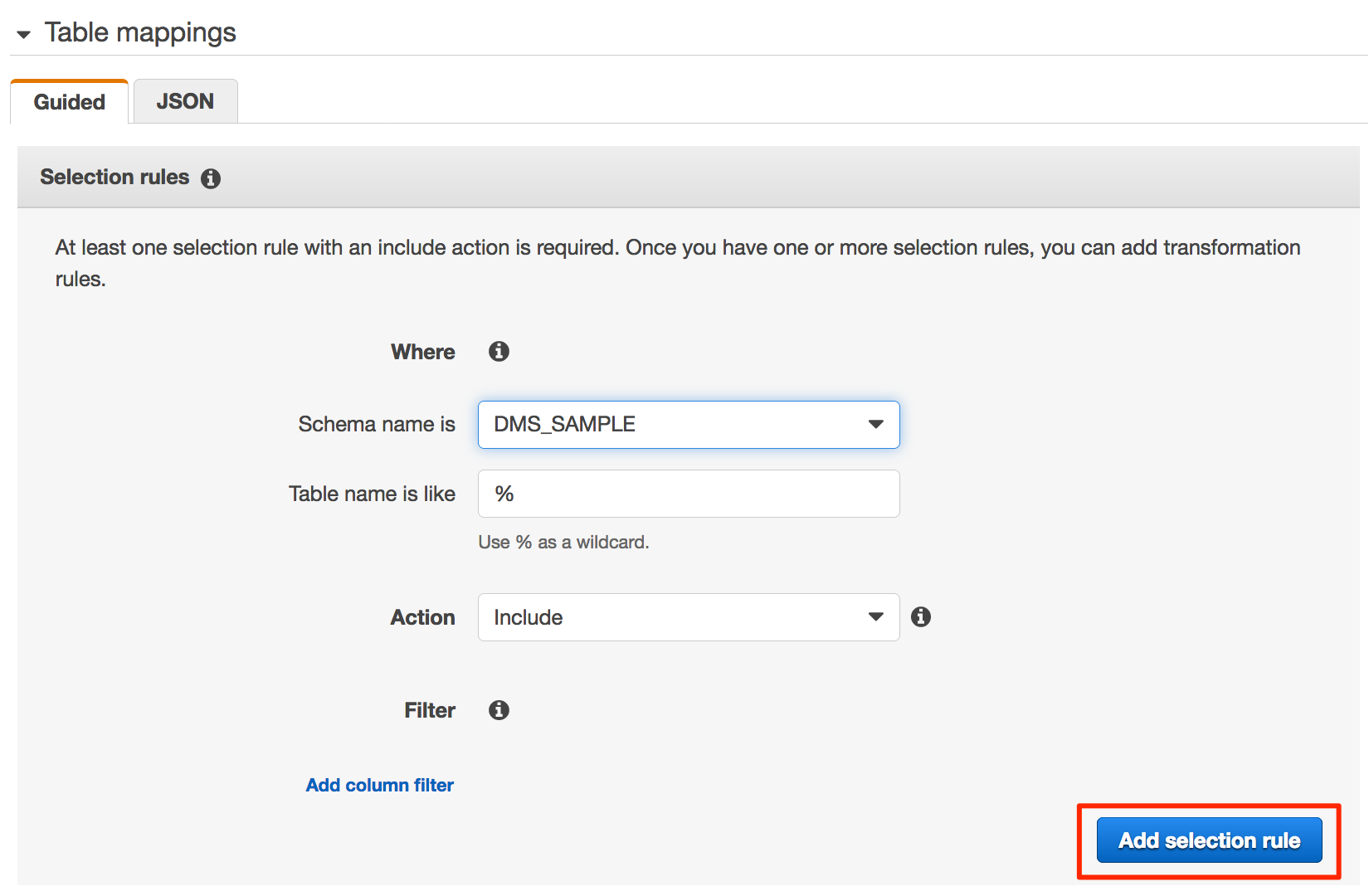
Make sure your configuration looks like the image below



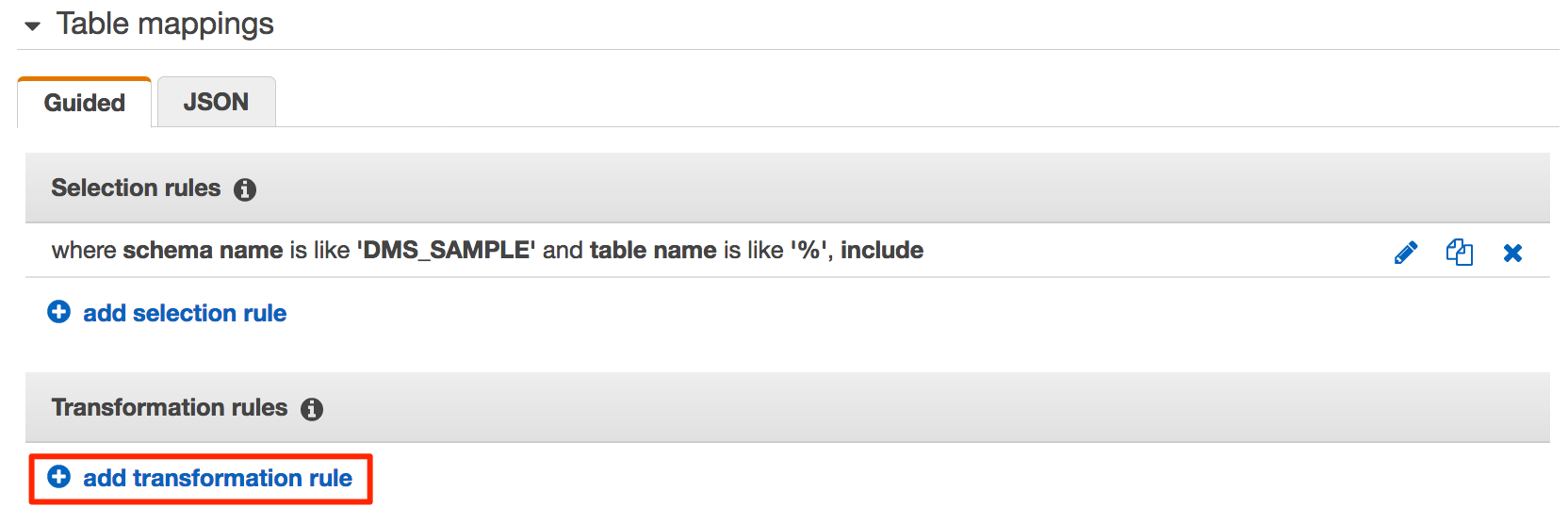
* Table Mappings

|  |  |
| --- | --- |
| Schema Name is | DMS\_SAMPLE |
| Table name is like | % |
| Action | Include |

* + Click '**Add Selection Rule**'



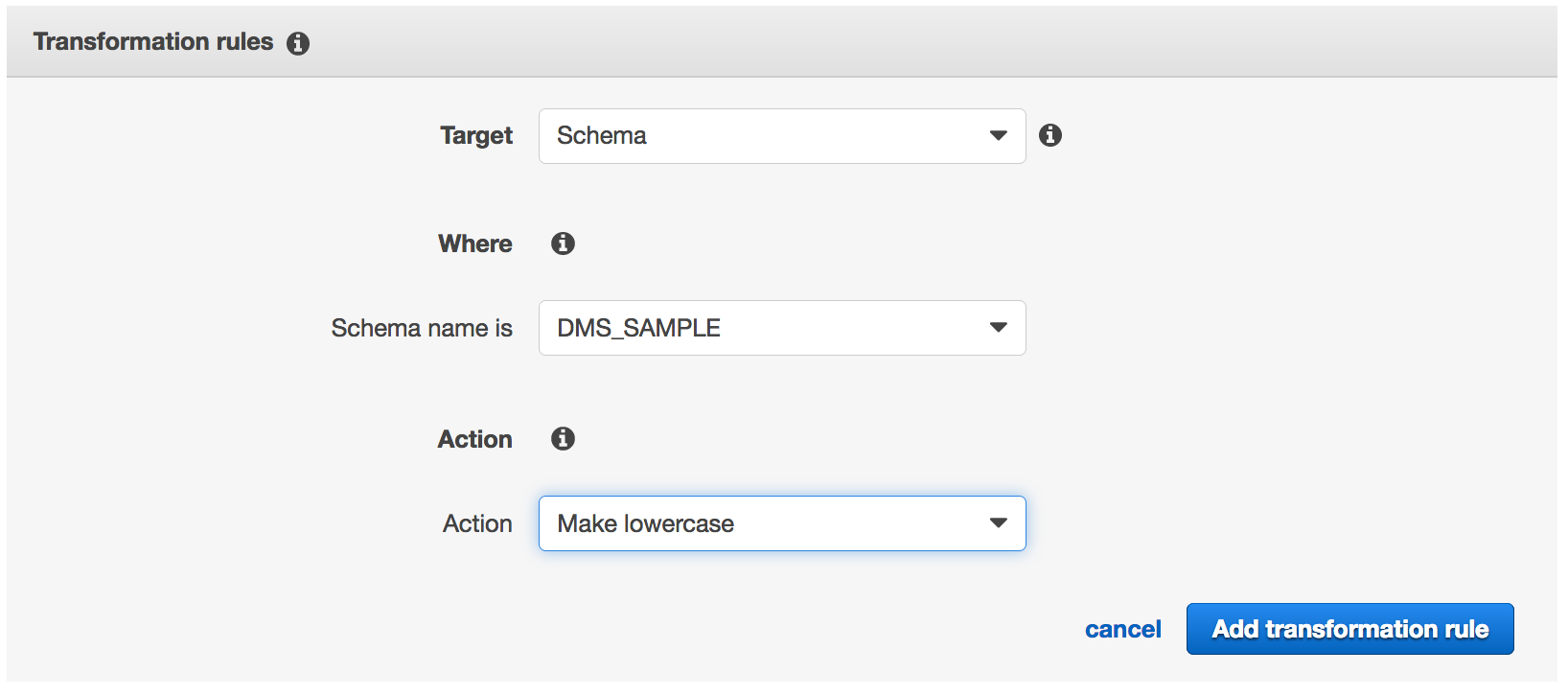
* Under 'Transformation rules' section click on 'add transformation rule' (we will be creating 3 rules here)



* + Rule 1:
    - Target: 'Schema'
    - Schema name is: 'DMS\_SAMPLE'
    - Action: 'make lower case'

|  |  |
| --- | --- |
| Target | Schema |
| Schema Name is | DMS\_SAMPLE |
| Action | Make lowercase |

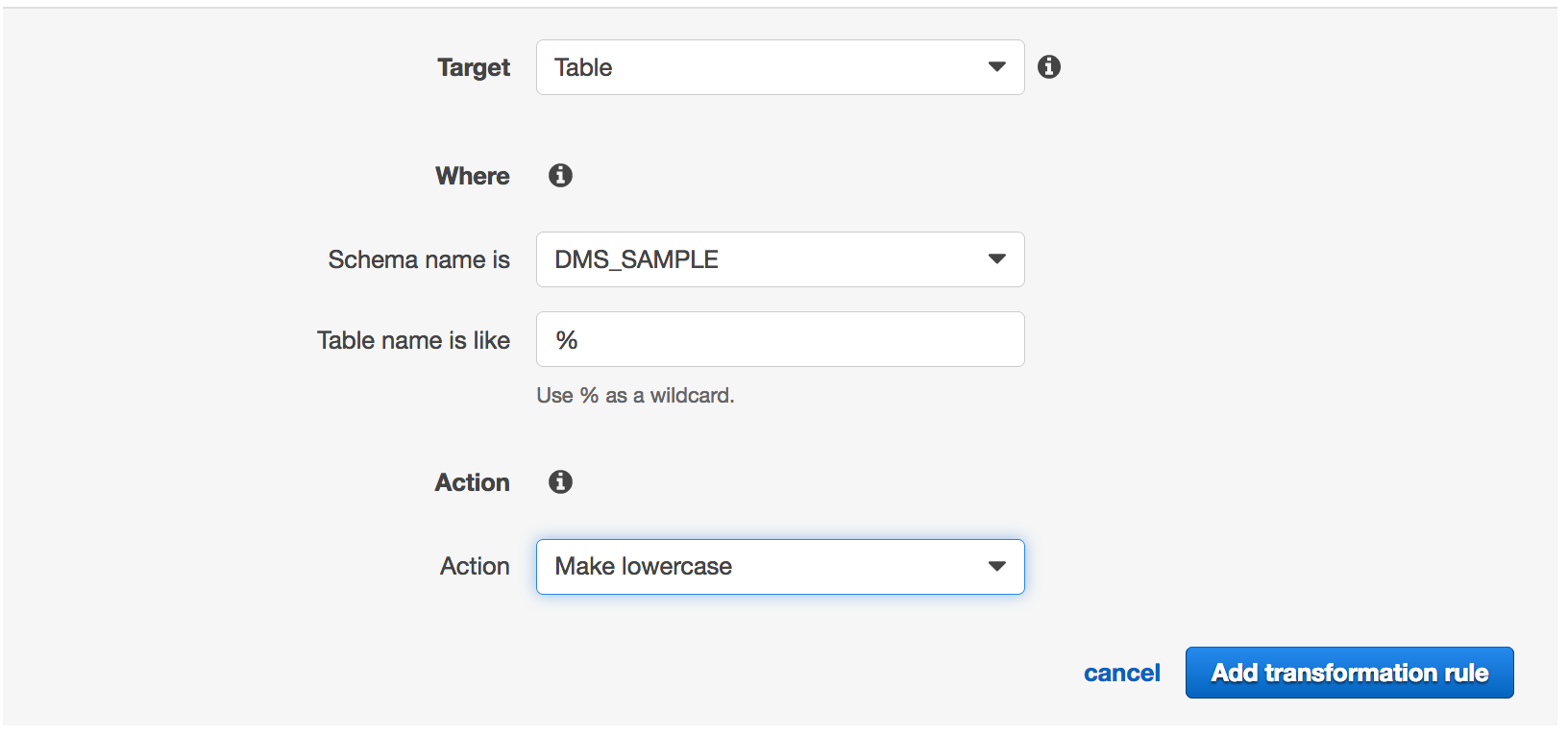
* + - Click '**Add transformation rule**'.



* + Rule 2:

|  |  |
| --- | --- |
| Target | Table |
| Schema Name is | DMS\_SAMPLE |
| Table Name is like | % |
| Action | Make lowercase |

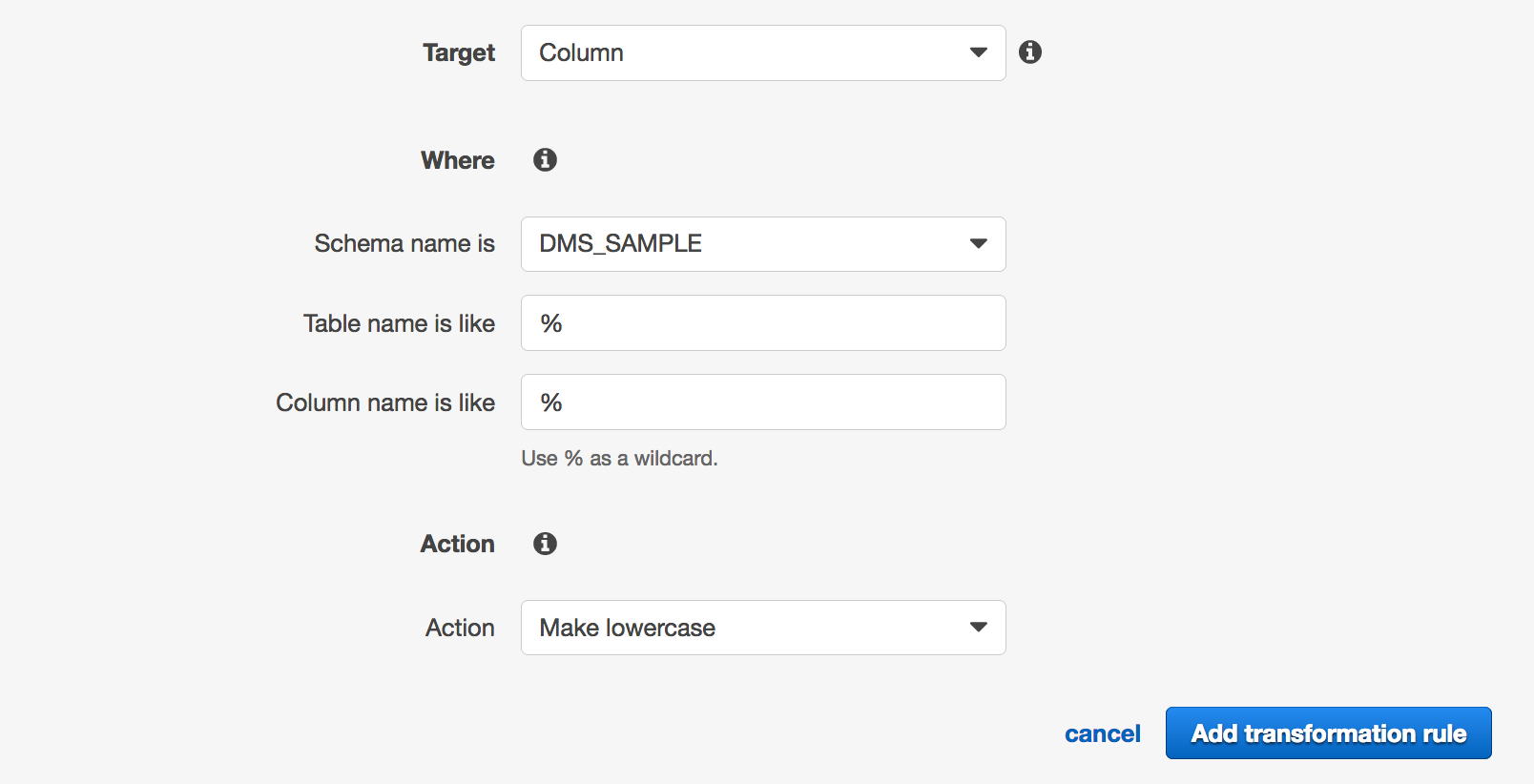
* + - Click 'Add transformation rule'.



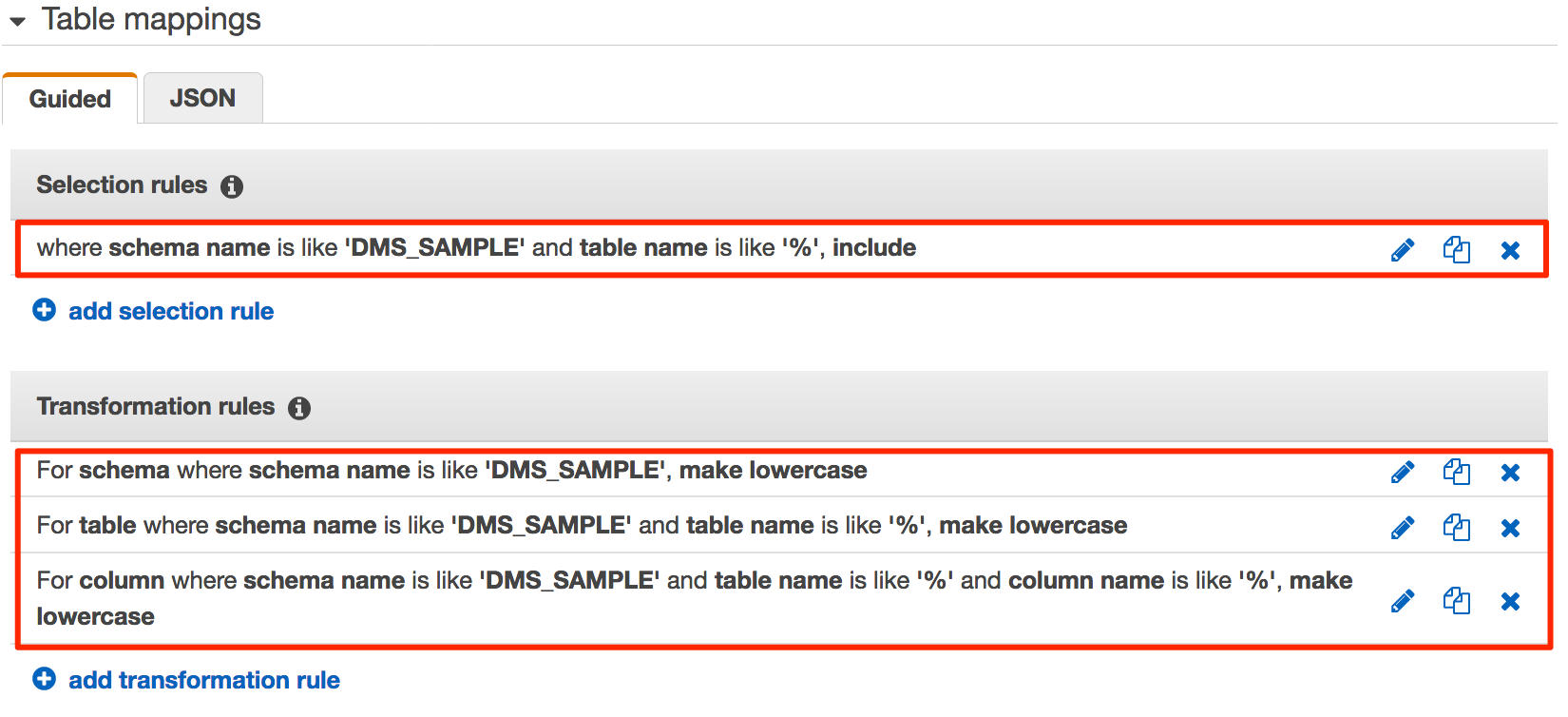
* + Rule 3:

|  |  |
| --- | --- |
| Target | Column |
| Schema Name is | DMS\_SAMPLE |
| Table Name is like | % |
| Column name is like | % |
| Action | Make lowercase |

* + - Click 'Add transformation rule'.



* Make sure your configuration looks like the image below.
* Take few minutes to review the JSON text generated



* Click on 'Create Task'

Wait for the task to get created and start running.

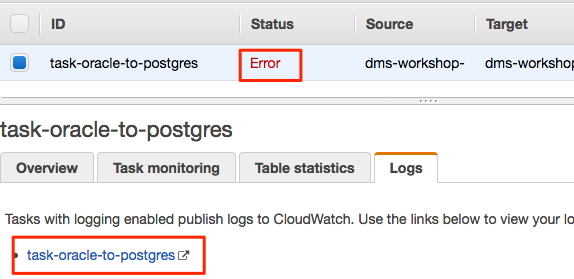
Shorty you will notice that the is still running with status ‘Error’.

Troubleshooting Initial Load

Now, let’s see what happened here why this task failed.

Stop the task.

GoTo the DMS task’s logs tab and click on the task name. This will open a new window with cloud watch entries for this DMS task.



In cloud watch logs:

* Filter all logs for last 1 week ‘**1w**’
* Display all log entries in ‘**Text**’ (this makes it easily readable)
* Filter text with **"E:"** (with double quotes) – this will filter & show messages with log level Error



Now look through the error messages and find error messages that are causing the database migration task to fail. You will notice that there are 2 types of error messages that appear in Coudwatch Logs



Let’s address them one by one.

ERROR: “cannot truncate a table referenced in a foreign key constraint”

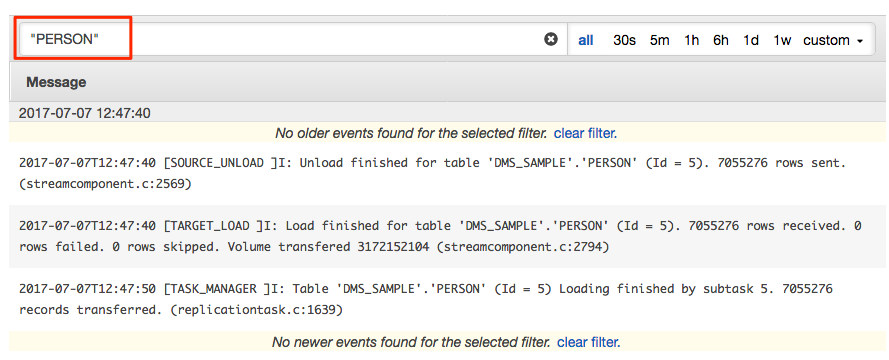
* Why am I seeing this error?
  + The initial load in DMS is done table by table which means that the target tables cannot have active foreign key constraints. As we are using SCT to convert Oracle source objects into PostgreSQL target objects, all secondary objects were created as part of the process. This means that we would need to disable all foreign key constraints on the target for the initial full load to be successful. Foreign keys or referential integrity constraints in PostgreSQL are implemented using triggers. One way to disable foreign keys is to disable all triggers temporarily from the instance and do the loads.
  + One of the ways to do this is to use the session\_replication\_role parameter in PostgreSQL. Triggers also have a state in PostgreSQL (Origin, replica, always or disabled). When the session\_replication\_role parameter is set to replica, only triggers of the state replica will be active and are fired when called. If not, the triggers remain inactive. We have already setup the parameter group on the target to set this role to replica which means all foreign key constraints (innately triggers in the origin state) will not be active. However, PostgreSQL has a failsafe mechanism of not letting a table truncate even with this role set. As we are using prepopulated tables on the target and cannot truncate the table, we need to use do\_nothing for the target table prep mode. More details in this awesome blog post: <http://blog.endpoint.com/2015/01/postgressessionreplication-role.html>
* How do I solve this?
  + Delete the cloudwatch log stream for dms-task – this will give you a clean environment to work with - https://ap-northeast-1.console.aws.amazon.com/cloudwatch/home?region=ap-northeast-1#logStream:group=dms-tasks-dms-workshop-instance;streamFilter=typeLogStreamPrefix
  + Go back to DMS tasks page: https://ap-northeast-1.console.aws.amazon.com/dms/home?region=ap-northeast-1 - tasks:
  + **Stop** the database migration task
  + **Modify** the task
    - Under task setting > change ‘**Target table preparation mode**’ to ‘**Do Nothing**’
    - **Save** the task settings > **Restart** the task. This will try to perform an initial load from the beginning.

Give it few minutes to complete the load.

You can check the migration progress for each table.

For example, let’s check status for ‘PERSON’ table

* Go to cloud watch logs for dms-task & search for "PERSON" (including double quotes)
* You can see 3 set of messages for each table:
  + Scan and unloading of records by DMS from Oracle
  + Records being loaded to Postgres
  + Confirmation of records loaded into Postgres



After fixing this issue, you will still see errors in the logs for the supplemental logging issue.

Let’s take a look at why we are facing this issue and fix it.

Error: “Supplemental logging for table 'DMS\_SAMPLE.MLB\_DATA' is not enabled properly”

* What am I dealing with?
  + Filter cloud watch logs with for **"Supplemental logging"** (with double quotes)
* Why am I seeing this error?
  + The 3 tables which failed full load had supplemental logging configurational issues.
    - MLB\_DATA
    - NFL\_STADIUM\_DATA
    - NFL\_DATA
* How do I solve this?
  + **Stop** the DMS migration task
  + By a rule of thumb, if a table does not have primary key or unique index on any column, you will need to enable supplemental logging on all columns in the table.
  + Login to your SQL Client of choice > connect to your **source Oracle RDS** database and run these SQL statements to enable supplemental logging on all columns

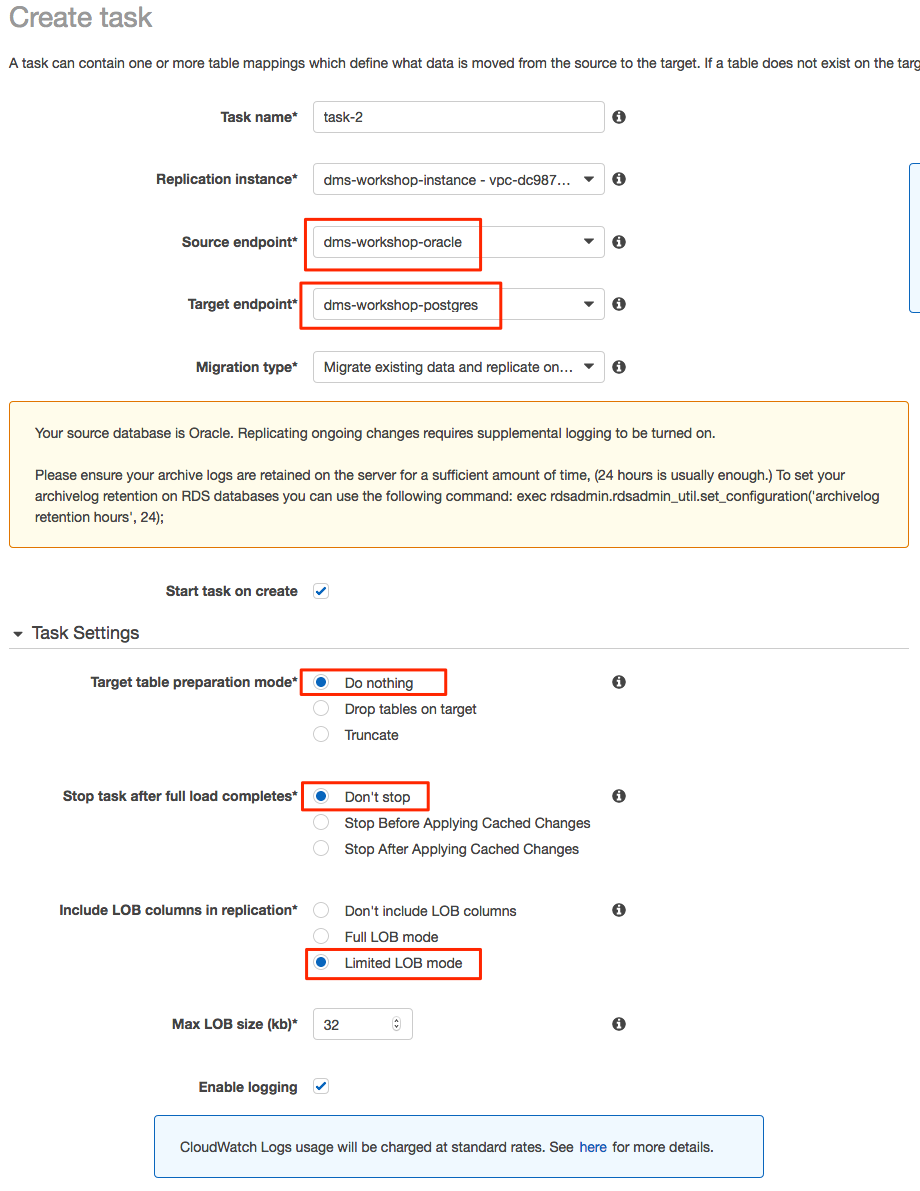
|  |
| --- |
| ALTER TABLE DMS\_SAMPLE.MLB\_DATA ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;  ALTER TABLE DMS\_SAMPLE.NFL\_STADIUM\_DATA ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;  ALTER TABLE DMS\_SAMPLE.NFL\_DATA ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS; |

At this stage, we have fixed the issues in our source database, next we will recreate the schema in Postgres target.

* Go to Schema Conversion Tool
  + On **left** panel, right click on **DMS\_SAMPLE** > **Refresh from Database**
  + On **left** panel, right click on **DMS\_SAMPLE** > **Convert Schema**
  + On **right** panel, right click on **dms\_sample** > **Apply to database**

Now, you now have a clean database in Postgres target.

* Let’s create a **new database migration task**:
  + Ensure your **existing DMS task is stopped**
  + On DMS task page > Click ‘**Modify’** for yourDMS task
    - Under ‘Table Mapping’ > go to JSON tab > copy the json text (we will use this in a new task)
  + On DMS task page > **Create Task**
  + Basic Info
    - Task name: retry-task-oracle-to-postgres
    - Replication instance: **dms-workshop-instance**
    - Source endpoint: **dms-workshop-oracle**
    - Target endpoint: **dms-workshop-postgres**
    - Migration type: **Migrate existing data and replicate ongoing changes**
    - Start task on create: **Checked**
  + Task Settings
    - Target table preparation mode: **Do nothing**
    - Stop task after full load completes: **Don’t stop**
    - Include LOB columns in replication: **Limited LOB mode – 32KB**
    - Enable logging: **Checked**
  + Table Mappings
    - Go to **JSON** Tab
    - Check – Enable JSON editing
    - Paste the previously copied JSON mapping configuration from your old DMS task
  + Click on **Create Task** (Ensure your configuration looks like screenshots below)





At this stage, the database migration task should load 100% of data from Oracle to Postgres. (This will usually take few 10s of minutes)

* Monitoring the progress for your database migration task
  + Select your newly create database migration task
  + Click on ‘Task monitoring’ tab & review the cloud watch metrics for your task
  + Click on ‘Table statistics’ tab & review table level stats for your migration



Executing transactions on the source to test CDC

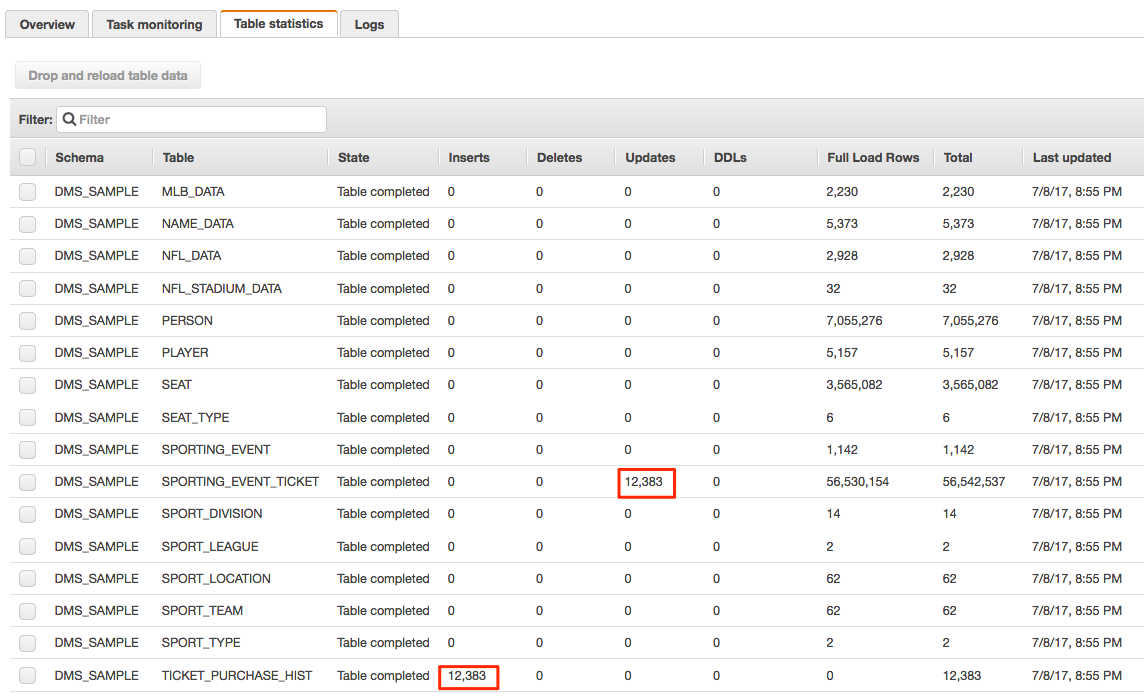
Once your task’s initial load is completed. You might want to execute a few transactions on the source. So, connect to your source database as dbmaster using your favorite tool: SQLDeveloper, DBeaver or even SQL\*Plus!

* Execute the following to sell some tickets:
  + This is a stored procedure in Oracle, it will take ~ 3mins to perform 1000 transactions

|  |
| --- |
| exec ticketManagement.generateTicketActivity(0.01,1000); |

Once the transactions are committed on source, you should see them on the target.

Check the status on your console > Task > Table Statistics



Once you’ve “sold” some tickets, you can execute the following to “transfer” some tickets

|  |
| --- |
| exec ticketManagement.generateTransferActivity(0.1,1000); |

Clean Up Your Lab Environment:

DO NOT FORGET TAKE DOWN YOUR ENVIRONMENT

1. Stop and delete your database migration tasks in DMS
2. Delete the source/target endpoints in DMS
3. Delete your DMS replication instance
4. Delete the cloud formation template

Delete CloudFormation stack from the CloudFormation console.

Appendix

|  |
| --- |
| Oracle - Get row count for all tables |
| SELECT table\_name, num\_rows  FROM dba\_tables  WHERE owner = 'DMS\_SAMPLE'  ORDER BY table\_name; |
| Postgres |
| SELECT relname AS table\_name, n\_live\_tup AS num\_rows  FROM pg\_stat\_user\_tables  WHERE schemaname = 'dms\_sample'  ORDER BY table\_name |

|  |
| --- |
| Oracle - Command to get database size on disk |
| SELECT owner, SUM(bytes) / 1024 / 1024 Size\_MB  FROM dba\_segments  WHERE owner = 'DMS\_SAMPLE'  group by owner; |
| Postgres - Command to get database size on disk |
| SELECT pg\_size\_pretty(CAST((SELECT SUM(pg\_total\_relation\_size (table\_schema || '.' || table\_name)) FROM information\_schema.tables WHERE table\_schema = 'dms\_sample') AS BIGINT)) AS tables\_schema\_size |