**AWS Pop-up Loft – Database Week**

Hand-on Lab: A Simple Database Migration

**Abstract:**

Description: This workshop will focus on hands-on journey for existing Oracle DBA to Aurora PostgreSQL. Attendees will deploy an instance of Aurora PostgreSQL, migrate or generate a test workload on PostgreSQL, and manually monitor the database to understand the workload. They will review a few ways to track queries and their execution plans, and what can be done to optimize the queries. Attendees will also learn how to do the same on Aurora PostgreSQL using Performance Insights for query analysis and tuning.

The following instructions help you launch a CloudFormation stack which sets up the following resources and migrate a sample database from Oracle to Aurora PostgreSQL using DMS and SCT –

1. Oracle RDS source within a VPC with all networking resources configured with a sample “HR” schema installed from [Sample HR Schema installation](https://docs.oracle.com/cd/E11882_01/server.112/e10831/installation.htm#COMSC00004)
2. Aurora PostgreSQL target within same VPC as #1 with all networking resources configured
3. EC2 Bastion instance with preinstalled PSQL and PGBENCH utilities
4. The CloudFormation will provision the necessary security groups with access based on the IP range provided in the template for port 1521 and 5432 respectively for Oracle and PostgreSQL instances.

**Pre-requisites:**

1. Active AWS account with Admin privileges. (IAM user should have administrator access). Please refer the link on how to create IAM administrator user [here](http://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started_create-admin-group.html#w2ab1c10c19b9)
2. Existing EC2 key pair created in the AWS region you are launching the CloudFormation template in. Please refer below on how to first create a new Key pair as shown [here](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html#having-ec2-create-your-key-pair)

Please follow the instructions below for connecting to EC2 Bastion instance.

|  |  |
| --- | --- |
| Your computer | How to connect |
| Linux | |  |  | | --- | --- | |  | [Connecting to Your Linux Instance Using SSH](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html) | |
| Window | [Connecting to Your Linux Instance from Windows Using PuTTY](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html) |
| All | [Connecting to Your Linux Instance Using MindTerm](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/mindterm.html) |

1. Pre-installed AWS Schema Conversion Tool software on your machine. Details on how to download and install AWS Schema Conversion Tool shown below
2. Install and launch SCT on your local machine from <http://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_SchemaConversionTool.Installing.html>
3. Download required drivers from links in the “Installing the Required Database Drivers” section from the above link. You will need to download Oracle and PostgreSQL drivers for this workshop. Alternatively, you can download the required drivers for this lab from –

<http://bit.ly/2phVpPk> -> Oracle JDBC driver

<http://bit.ly/2pt04ZT> -> PostgreSQL JDBC driver

On the Mac, please make sure the file permission for the drivers is set to 755

You can update the path to drivers in Settings -> Global settings -> Drivers

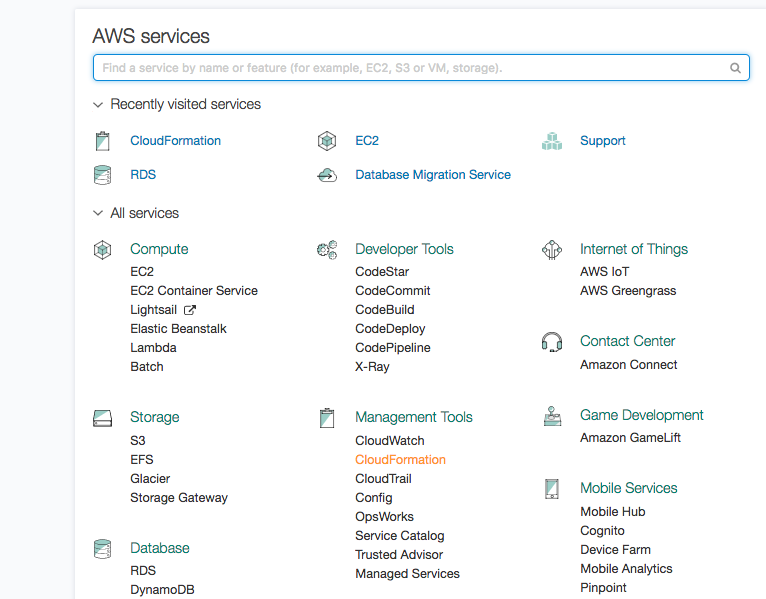
**High level steps:**

1. Log into your AWS account
2. Launch CloudFormation stack
3. Click on create a new stack and “Choose Specify an Amazon S3 template URL” and specify the S3 location for the cloud formation template
4. Provide user defined Stack name
5. Under Parameters section, for the parameter key name, select the preexisting key for your account in this region
6. Click create and wait for the stack to be completed, once it is completed look under the tab “outputs” to get the connection details for Oracle and Aurora PostgreSQL and EC2 instance.
7. Navigate to the DMS service (under Migration)
8. Launch a replication instance on the same VPC as source and target
9. Use SCT to convert source Oracle schema objects to target PostgreSQL
10. Create a source endpoint (Source is launched using the CloudFormation stack)
11. Create a target endpoint (Target is launched using CloudFormation stack)
12. Create a task to migrate the data from the source database to the target database
13. Execute some transactions on the source database (using a script we’ve provided)

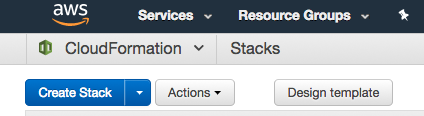
|  |
| --- |
| Please execute this LAB using the us-east-1 (N. Virginia) or us-west-2 (Oregon) region. |

**Lab Instructions:**

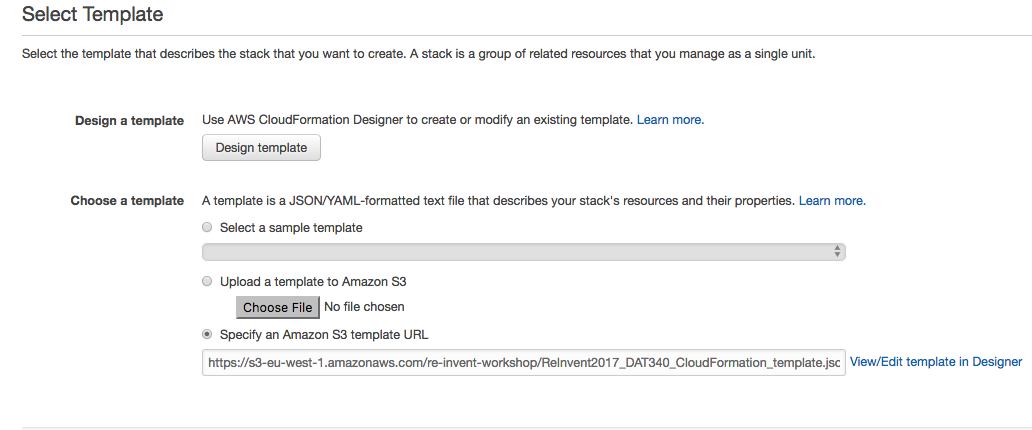
1. Login in to your AWS console and go to the CloudFormation service



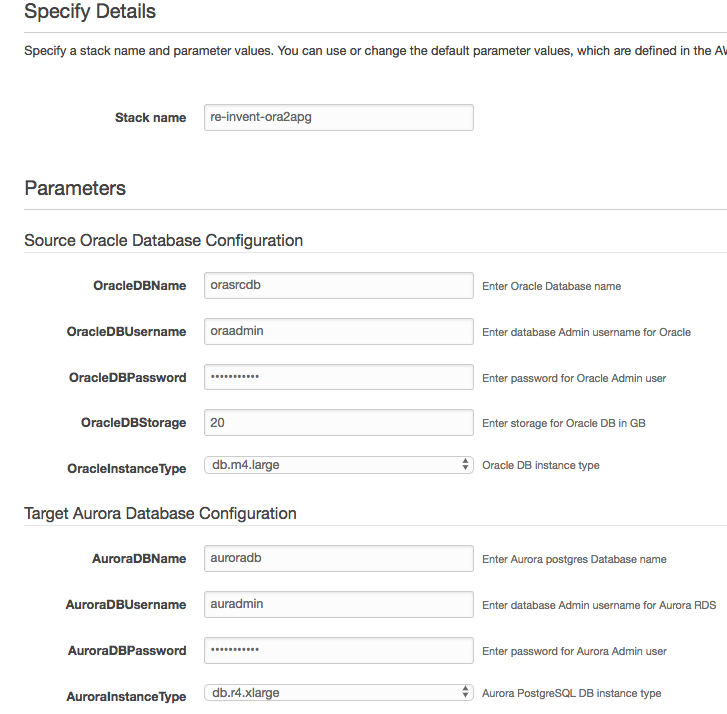
1. Click “Create stack” from the top left



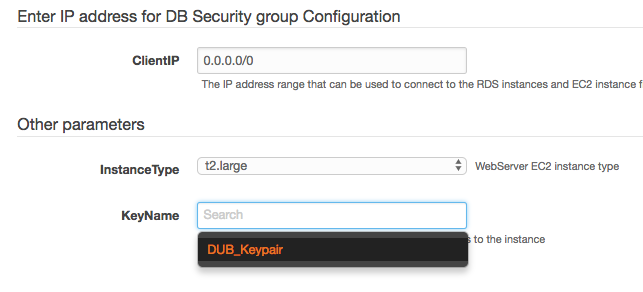
Click create stack, select **‘Specify an Amazon S3 template URL’** and specify the S3 template URL as <https://s3-us-west-2.amazonaws.com/nyc-loft/PostgreSQL_Migration_CloudFormation_template.json>



1. Hit next and give a friendly name such as **ora2pg** for the stack and accept all defaults.

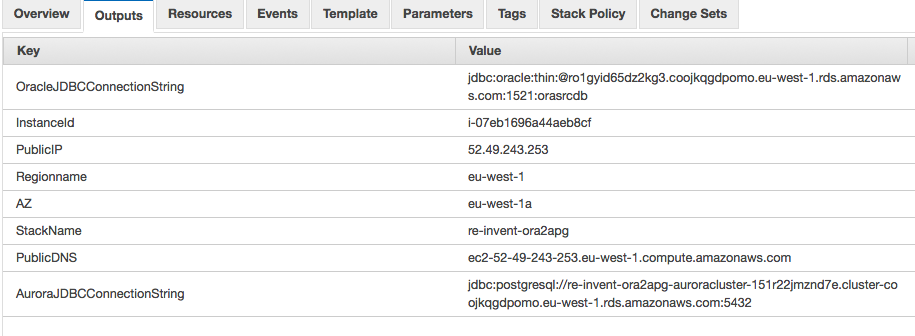


In the **Other Parameters** section choose the key for the EC2 instance as shown below Hit next in the next page and click on create. Wait for the entire stack to launch and become available.

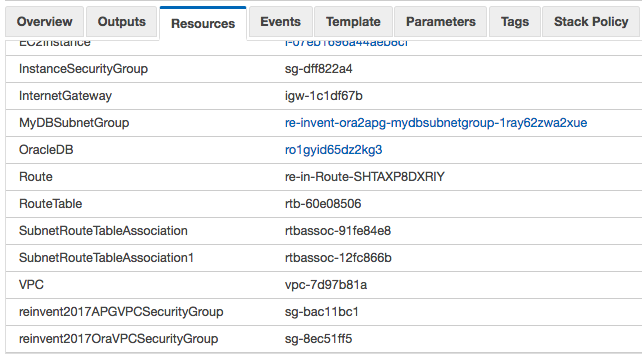


\*Leave everything else default. Once the CloudFormation stack launch is complete, you should see 2 RDS instances in your account in the region you created the stack. Approximately, it will **take 15 to 20 minutes** for creating the Stack.

1. Note down RDS Endpoint details of source and target instances from stack output section as shown below



1. Note down the resources created by the stack from the “Resources” tab, specially note down the VPC created by the stack as shown below

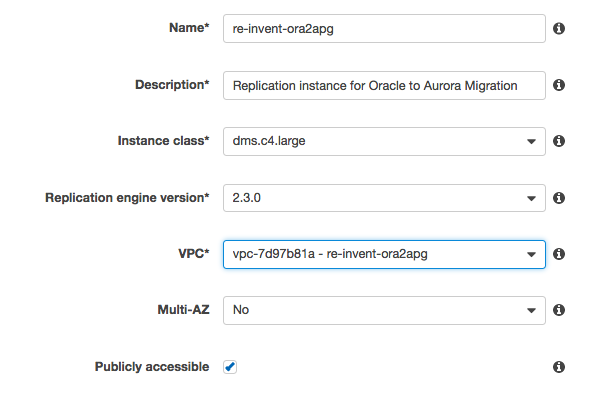


1. Once stack is available, go to the DMS console and launch a replication instance by selecting “Replication Instance” on the left side menu. Choose the same VPC as your stack name (this VPC was created by the CloudFormation Stack).

Also, please check the **“Replication engine version”** to the latest version, this screenshot shows the version 2.3.0, but the users should choose the latest 2.4.3 version.

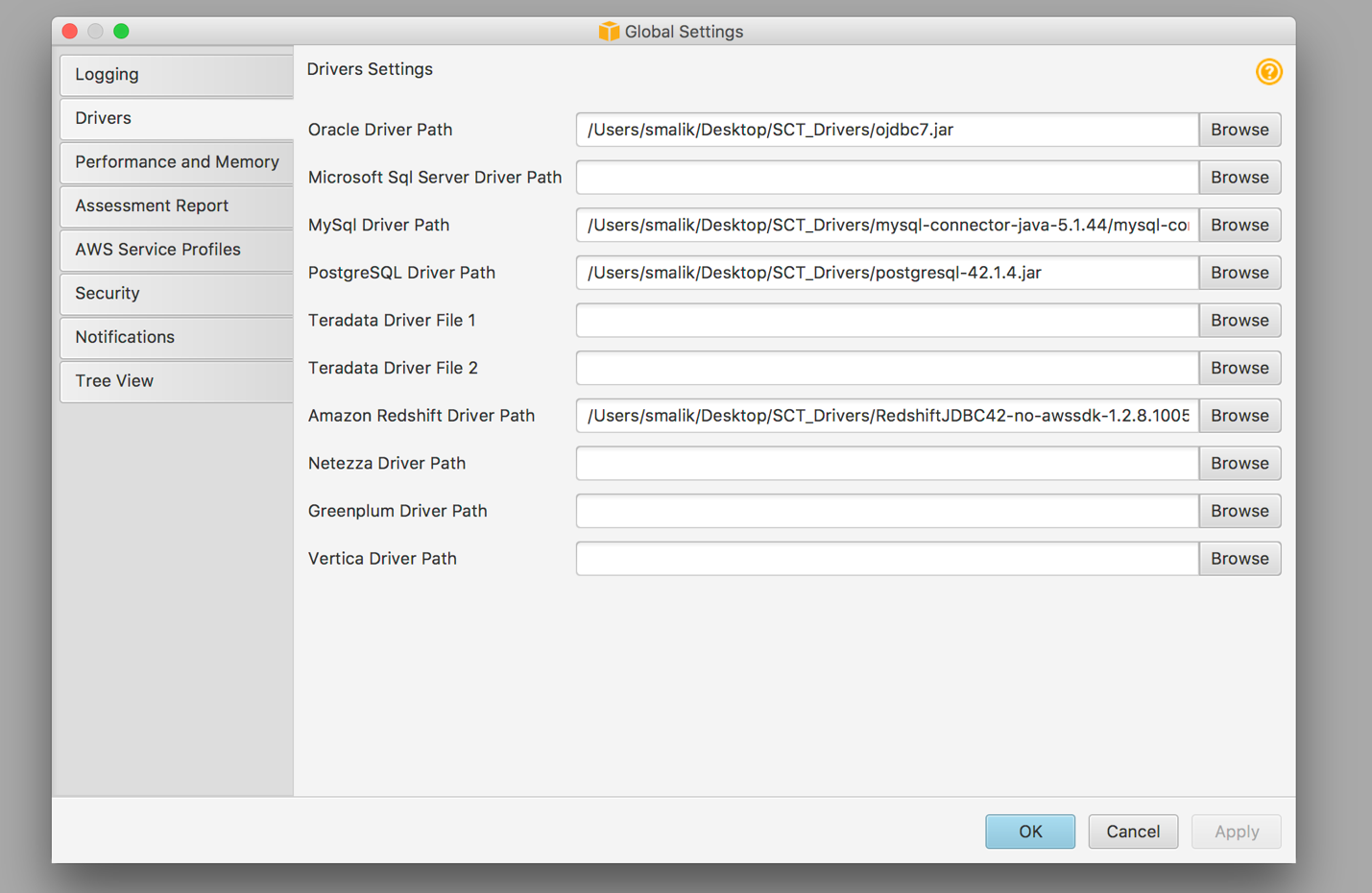
Please leave the rest of the values/advanced settings to the default values.

Here is a screenshot of the create replication instance page –

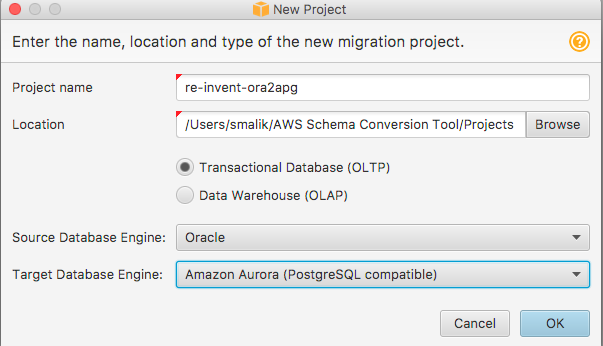


**Schema Conversion:**

1. Launch the AWS Schema Conversion Tool installed on your machine.
2. Make the path to drivers in Settings -> Global settings -> Drivers point to the actual path where you have kept the drivers as show below



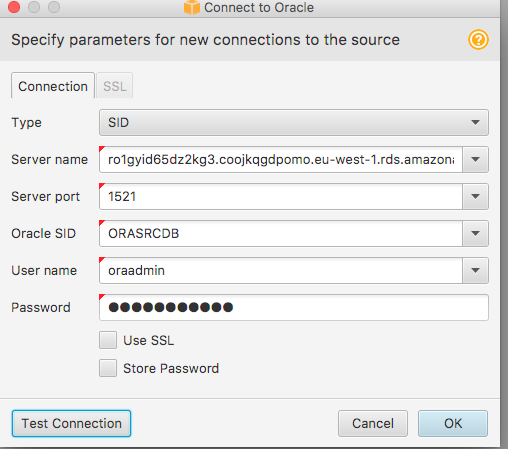
1. Create a New Project using File-> New Project



1. Connect to your assigned Oracle source and PostgreSQL target endpoints. Endpoints can be found under the “Resources” tab on the CloudFormation page or from the RDS Console under instances

**Oracle instance –**

* Type: SID
* Engine: Oracle
* Server Name: RDS Oracle Source endpoint from your account
* Server Port: 1521
* SID: orasrcdb
* User name: oraadmin
* Password: oraadmin123 (Please note that if you changed the default settings when creating the CloudFormation stack, then you need to enter the password that you have provided)

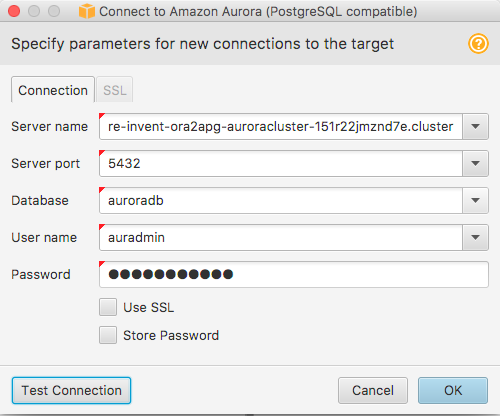


If the defaults were accepted in CloudFormation, the password is oraadmin123. **Substitute the server name with the RDS Oracle Database Endpoint launched in your VPC.**

**NOTE:** If you’re not able to connect, please add the IP address from which you’re trying to connect to the instances from to the inbound security group rules for the security groups attached to the Oracle RDS and Aurora PostgreSQL instances.

**PostgreSQL instance:**

* Server name: Cluster endpoint of Aurora PostgreSQL target instance
* Serve Port: 5432
* Database name: auroradb
* Username: auradmin
* Password: auradmin123 (Please note that if you changed the default settings when creating the CloudFormation stack, then you need to enter the password that you have provided)

****

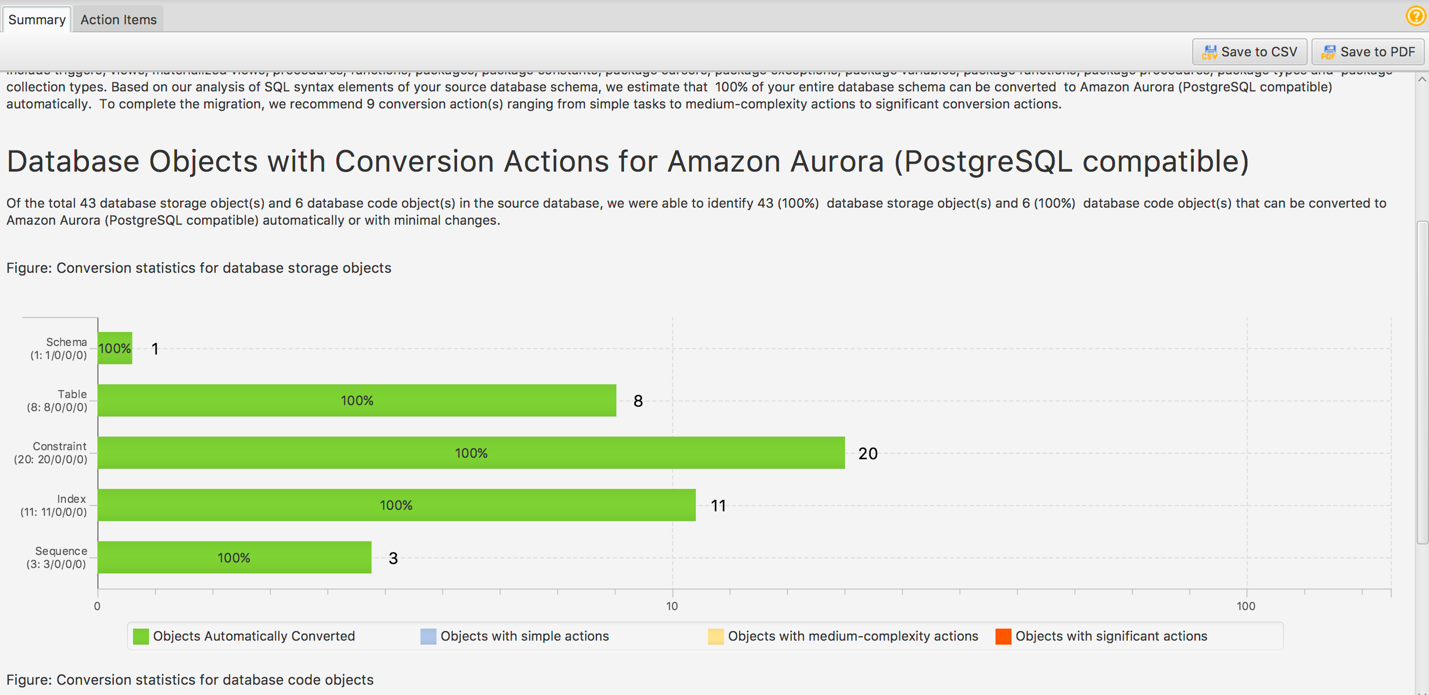
If the defaults were accepted in CloudFormation, the password is **auradmin123**. **Substitute**

**the server name with the RDS Aurora PostgreSQL (cluster endpoint) launched in your VPC.**

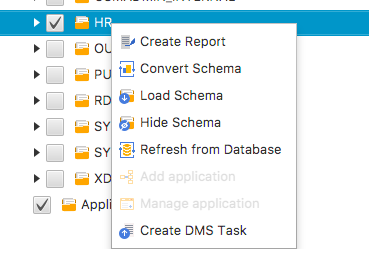
**NOTE:** If you’re not able to connect, please add the IP address from which you’re trying to connect to the instances from to the inbound security group rules for the security groups attached to the Oracle RDS and Aurora PostgreSQL instances.

Please refer troubleshoot connectivity - EC2 security groups [here](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/authorizing-access-to-an-instance.html)

1. Uncheck all schemas on the left except for the **“HR”** schema.
2. Now, right click and click “Create Report”. Review the conversion report – click on the view tab and click assessment report view. Look through what Oracle objects could be automatically converted and what could not be

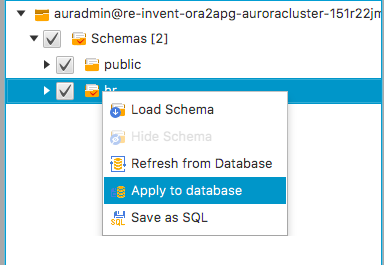


1. Now, right click and click “Convert schema”. The schema will be converted and shown on the PostgreSQL instance (it has not been applied yet).



1. Right click on the created schema on the right 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.



1. Objects which could not be converted automatically must be taken care of manually after migration at a later time.

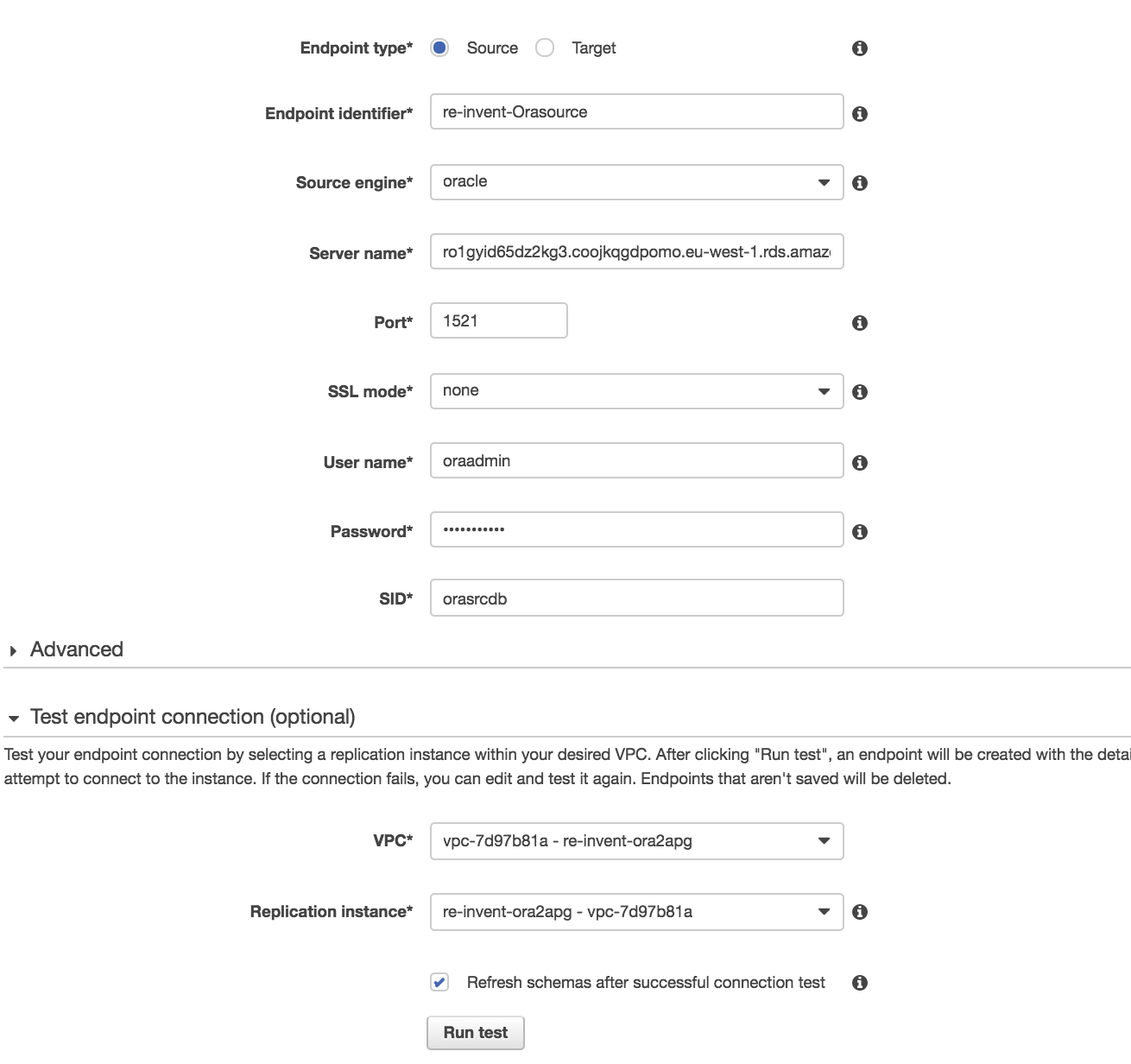
**NOTE:** At this point, all the database objects from the source HR Oracle schema must be converted and created in PostgreSQL in the HR schema in the Aurora postgres database.

**Data Migration:**

**Create a source endpoint:**

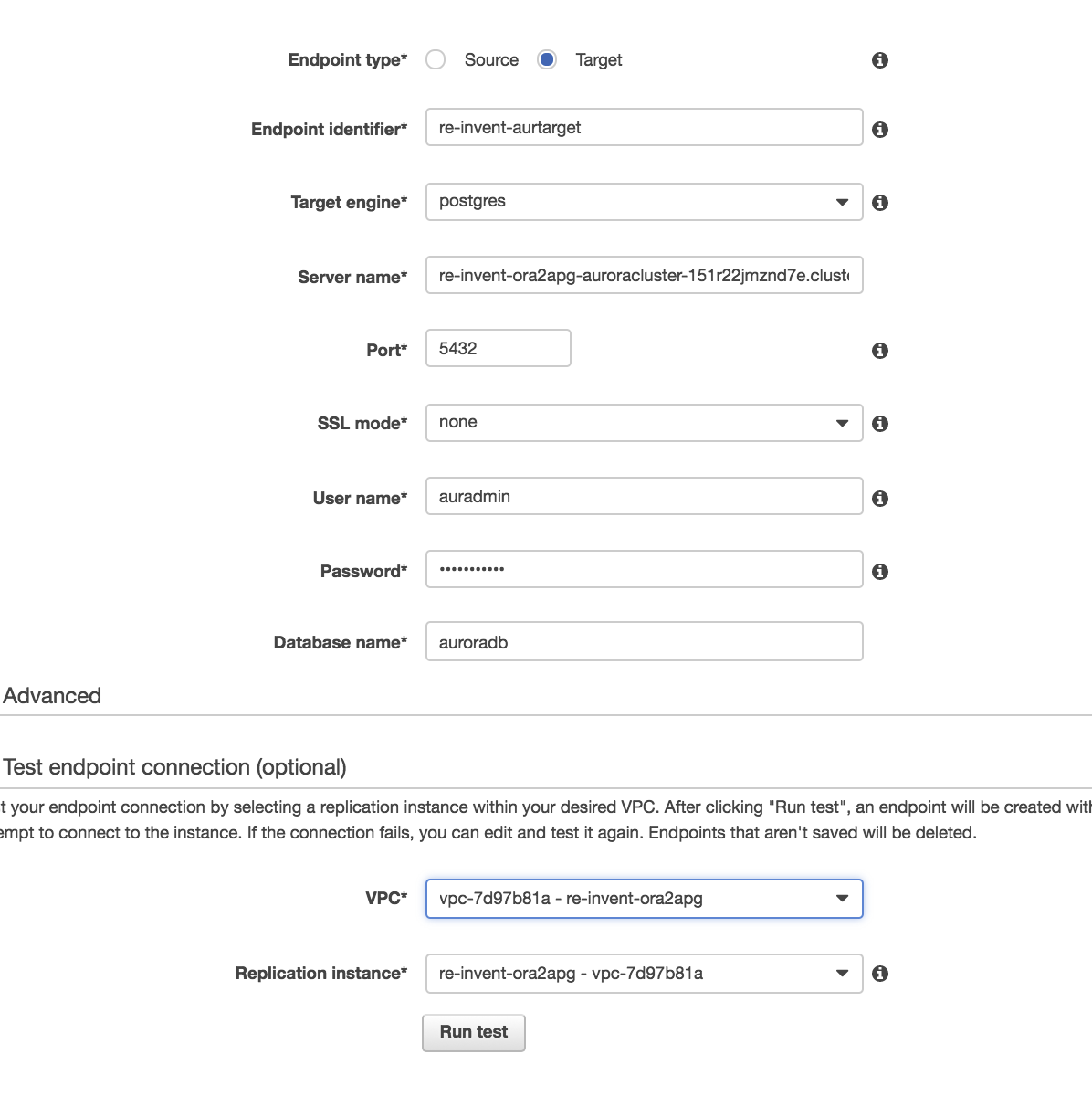
* Select Endpoints from the menu on the left and “click the "create endpoint" button at the top.
* Endpoint identifier: orasource
* Endpoint Type: source
* Engine: Oracle
* Server Name: RDS Oracle Source endpoint from your account
* Port: 1521
* SSL Mode: none
* User name: oraadmin
* Password: oraadmin123 (Please note that if you changed the default settings when creating the CloudFormation stack, then you need to enter the password that you have provided)
* SID: orasrcdb
* VPC: VPC launched by the CloudFormation template
* Replication Instance: choose the instance you created
* Refresh schemas: Yes - check this box

**Test the connection and save it if the test is successful.**

****

**Create a target endpoint:**

* Click the "create endpoint" button at the top
* Endpoint identifier: aurtarget
* Endpoint Type: Target
* Engine: postgres
* Server name: Cluster endpoint of Aurora PostgreSQL target instance
* Port: 5432
* SSL Mode: none
* Username: auradmin
* Password: auradmin123 (Please note that if you changed the default settings when creating the CloudFormation stack, then you need to enter the password that you have provided)
* Database name: auroradb
* VPC: VPC launched by the CloudFormation template
* Replication Instance: choose the instance you created
* Refresh schemas: check this box

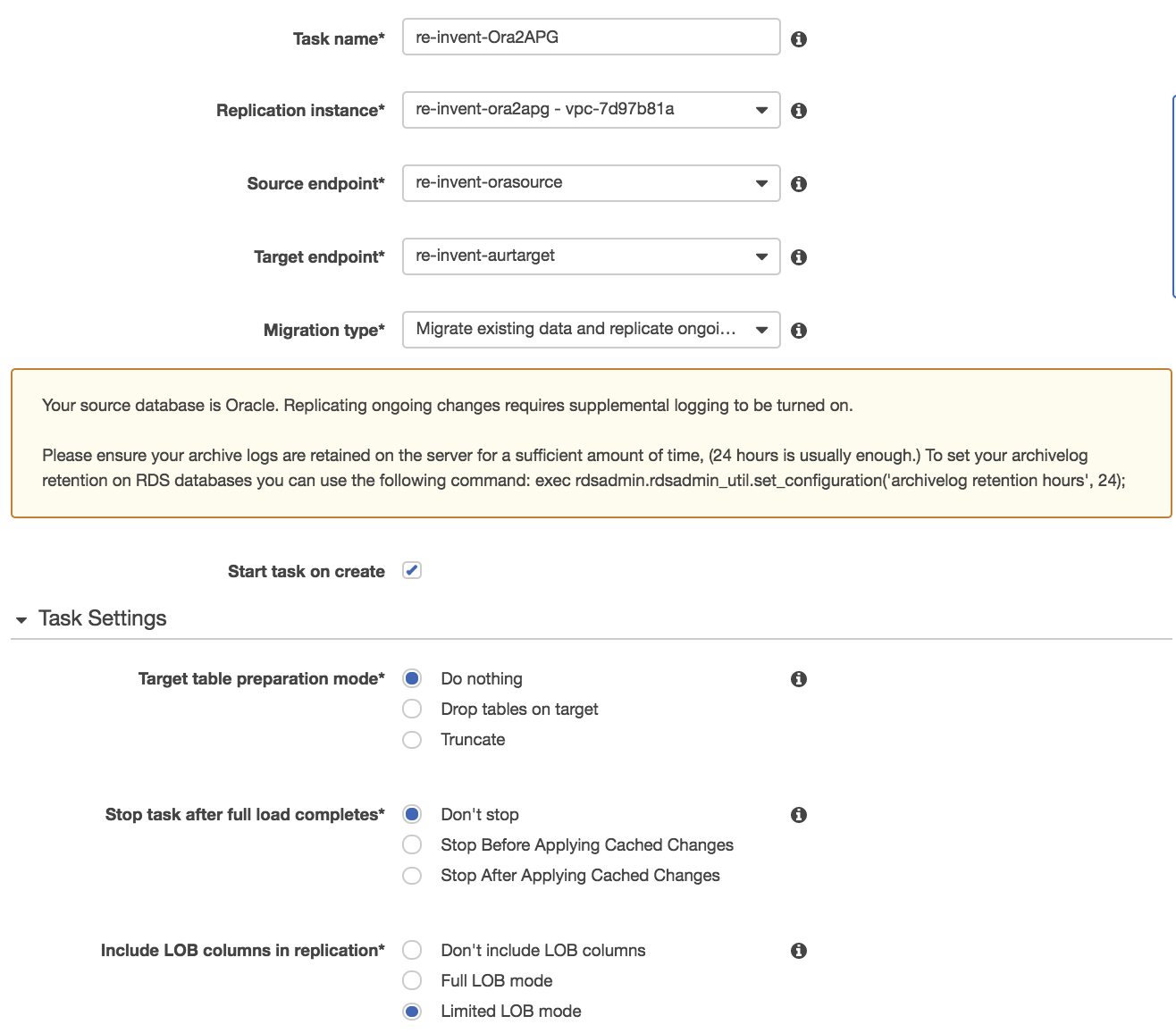
****

**Test the connection and save it if the test is successful**

**CREATE A MIGRATION TASK:**

* Name: Replication Instance: choose the one you created
* Source Endpoint: choose the source endpoint you just created
* Target Endpoint: choose the target endpoint you just created
* Migration Type: Migrate existing data and replicate ongoing changes
* Start task on create: check this (or don't, up to you). If you don’t, you’ll need to start your task Separately.

\*\* Target table preparation mode: Do Nothing



**Why a do-nothing task?**

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 are using do\_nothing for the target table prep mode.

Stop task after full load completes: don't stop

Include LOB columns in replication: Accept the default

Enable logging: YES! (Check this box)

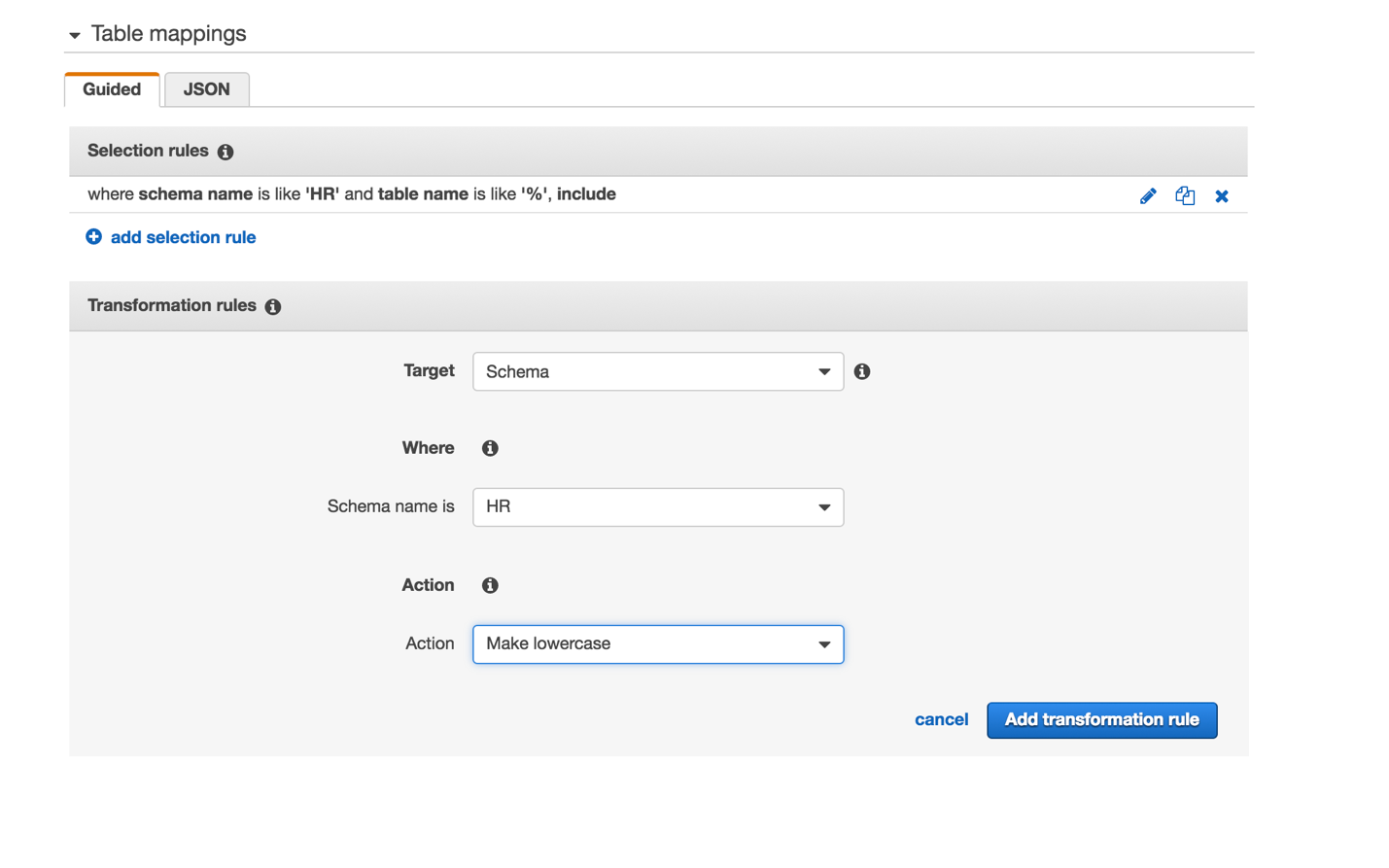
**Table Mappings – Selection rules:**

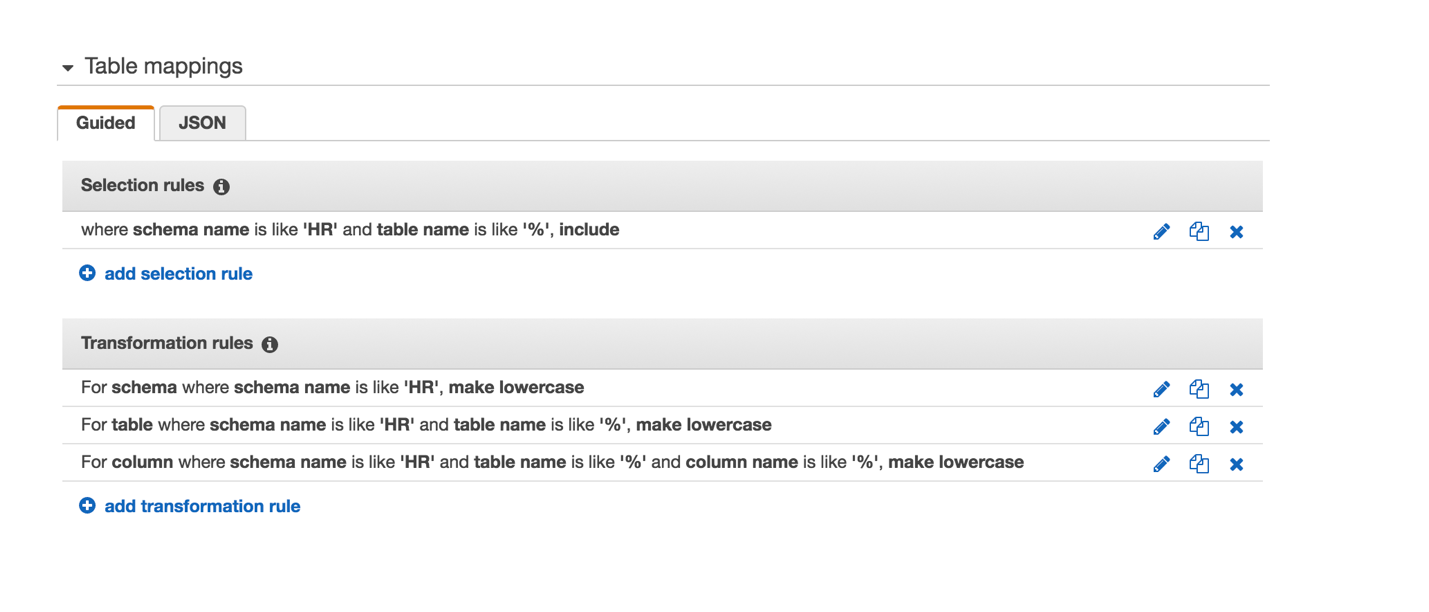
1. Select HR from the drop-down box Click the "Add selection rule" button.

2. Click on “add transformation rule” -> Choose schema for target dropdown -> Schema name is HR -> Choose “make lower case” for action & click add transformation rule.

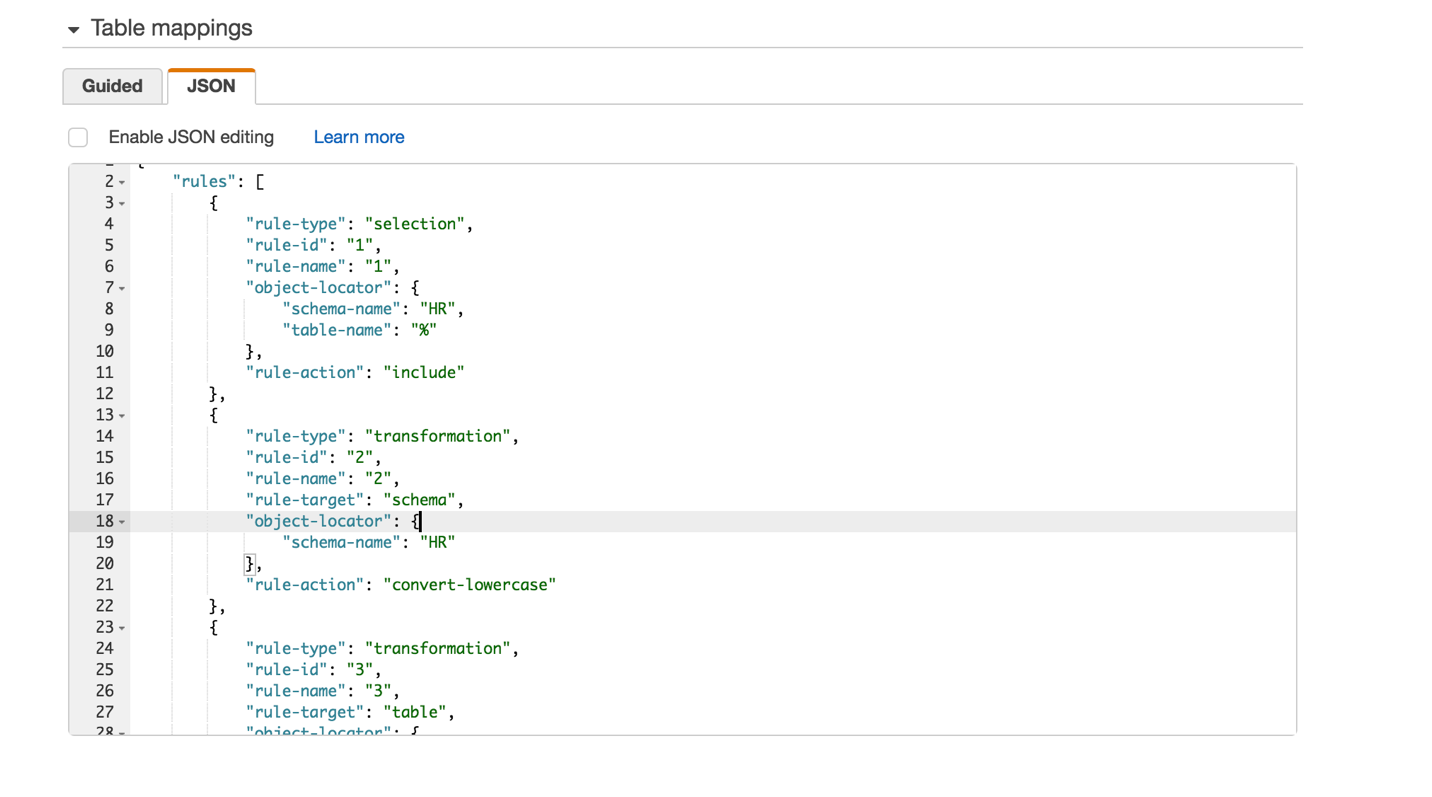
3. Click on “add transformation rule” -> Choose table for target dropdown -> Schema name is HR and table name is % -> Choose “make lower case” for action & click add transformation rule.

4. Click on “add transformation rule” -> Choose column for target dropdown -> schema name is HR, table name is % -> column name is % -> Choose “make lower case” for action & click add transformation rule.





Note – We are doing 2, 3 and 4 above as the schema has been pre-created on the target with lowercase (PostgreSQL convention). Your task json should look like this –



* Click create task now and let the task run.

**Executing transactions on the source to mimic CDC transactions:**

Once your task is running you might want to execute a few transactions on the source. So, connect to your source database as oraadmin using your favorite tool on your local machine: SQLDeveloper, DBeaver or even SQL\*Plus!

You can also choose to use the EC2 instance created as a part of the CloudFormation stack, this EC2 Instance is being created with SQLPLUS pre-installed.

Login to EC2 instance (Refer the EC2 Bastion Instance Public DNS from the Output or Resource tab from the CloudFormation stack) as shown here

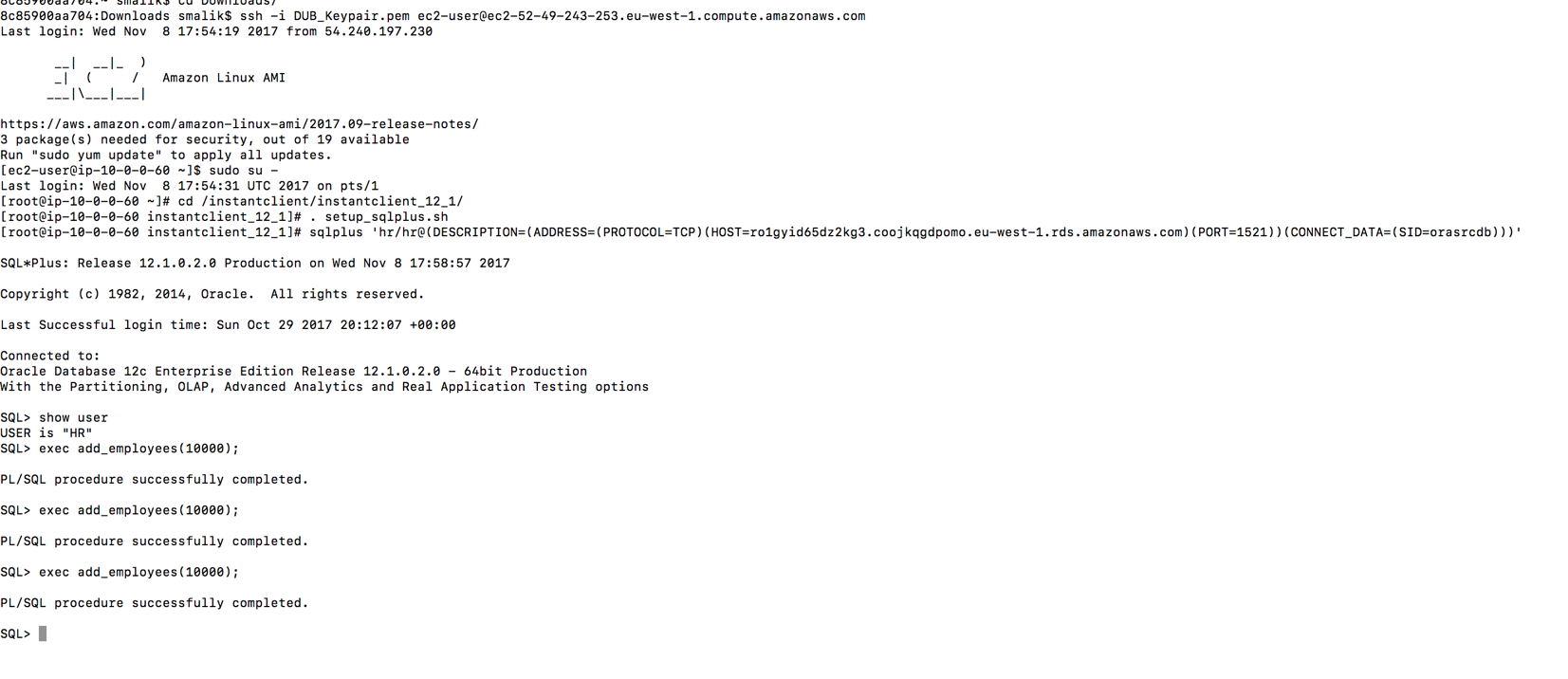
1. ssh –i <keypair.pem> ec2-user@<ec2 host name public DNS>

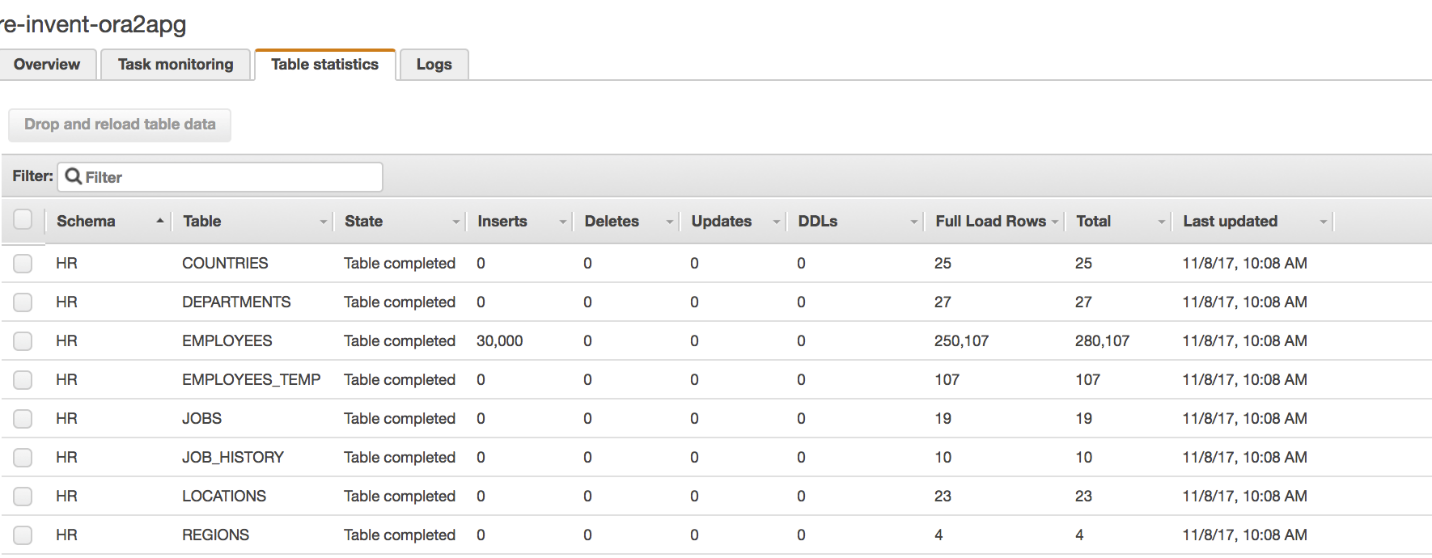
(please note if you are using Putty to connect to EC2 Instance, then please use “.ppk” file instead)

1. Switch to root user (sudo su -)
2. $cd /instantclient/instantclient\_12\_1/
3. $ . setup\_sqlplus.sh

5)sqlplus 'oraadmin@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<RDS Oracle Endpoint>)(PORT=1521))(CONNECT\_DATA=(SID=orasrcdb)))'

6) exec hr.add\_employees(10000);



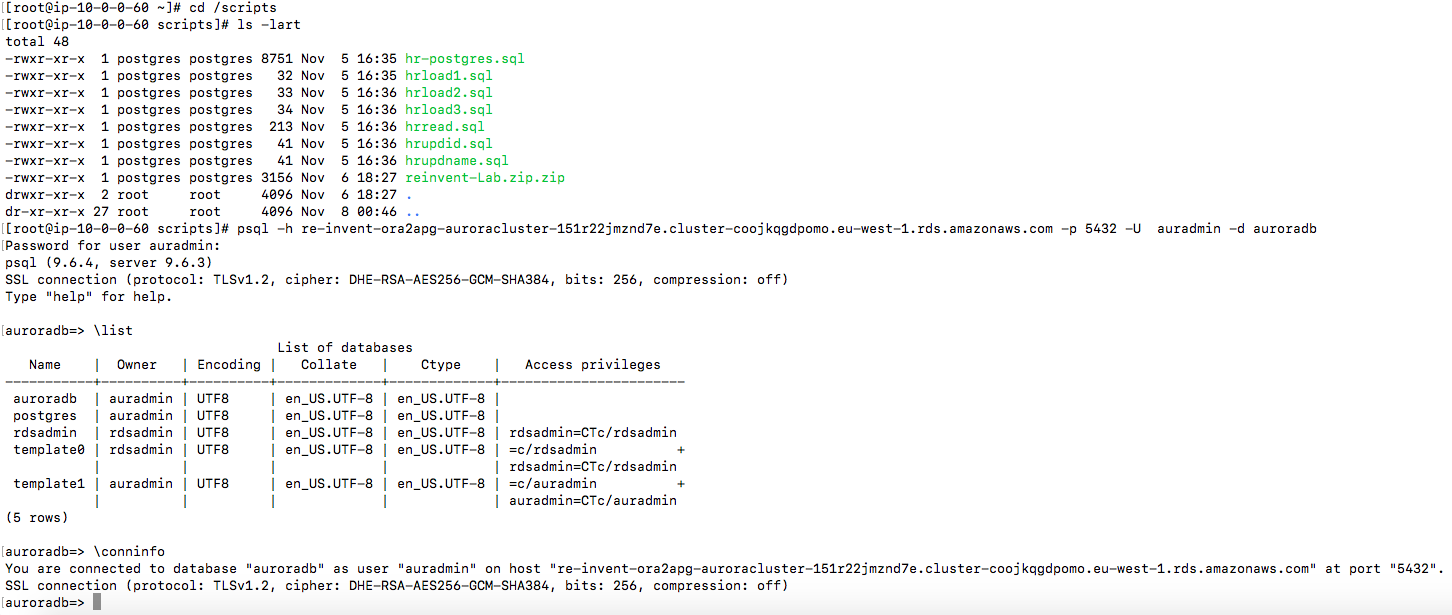


**Steps to connect to Aurora PostgreSQL Database using psql:**

Login to EC2 instance (grab the instance details from the Output or Resource tab from the CloudFormation stack) as shown here

1. ssh –i <keypair.pem> ec2-user@<ec2 host name public DNS>
2. Switch to root user (sudo su -)
3. $ cd /scripts
4. psql -h <replace cluster endpoint of your Aurora DB instance> -p 5432 -U auradmin -d auroradb

provide the password for auradmin user (auradmin123)



# 