Contents

[Lab1: Migration using SCT/DMS 2](#_Toc498531667)

[Lab2: Walkthrough of Aurora Console and Read replicas, DB Parameter groups. 23](#_Toc498531668)

[LAB3 : Performance Insights 27](#_Toc498531669)

[**Load Scenario-1 : High volume insert via pgbench** 27](#_Toc498531670)

[**Load Scenario-2: High volume update via pgbench** 28](#_Toc498531671)

[**Load Scenario-3: Connection Load Balancing across replicas** 30](#_Toc498531672)

[Clean up CloudFormation stack: 31](#_Toc498531673)

**Workshop Outline**

**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.

# Lab1: Migration using SCT/DMS

***Oracle RDS to Aurora PostgreSQL migration lab using DMS and SCT:***

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

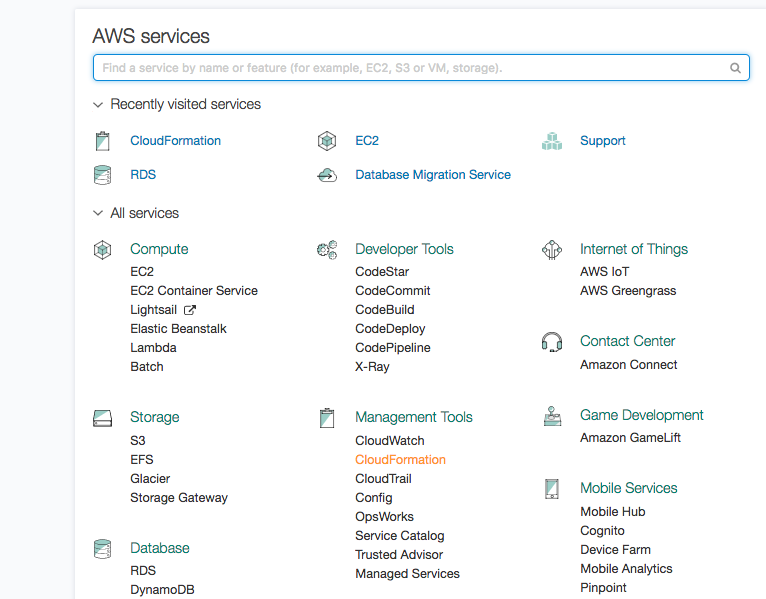
**High level steps:**

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)

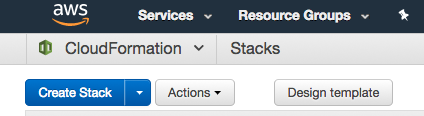
|  |
| --- |
| Please execute this LAB using the us-east-1 (N. Virginia) or us-west-2 (Oregon) region. |

**Lab Instructions:**

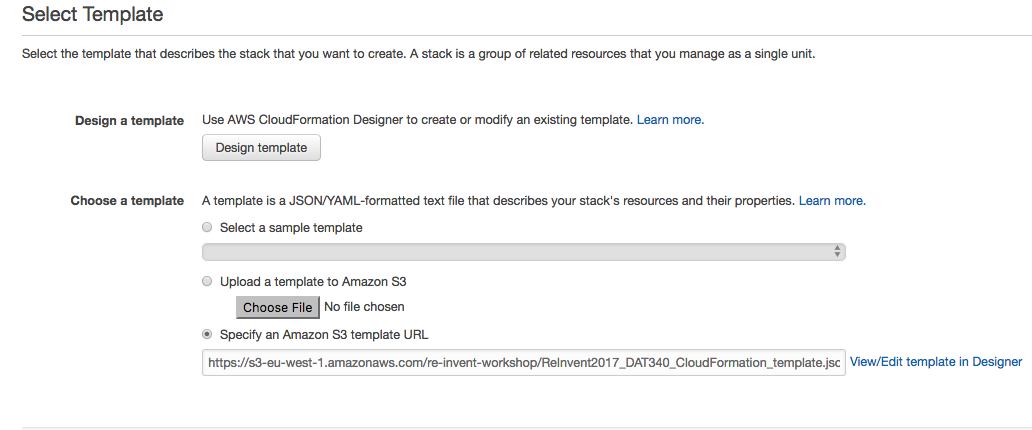
1. Login in to your AWS console and go to the CloudFormation service



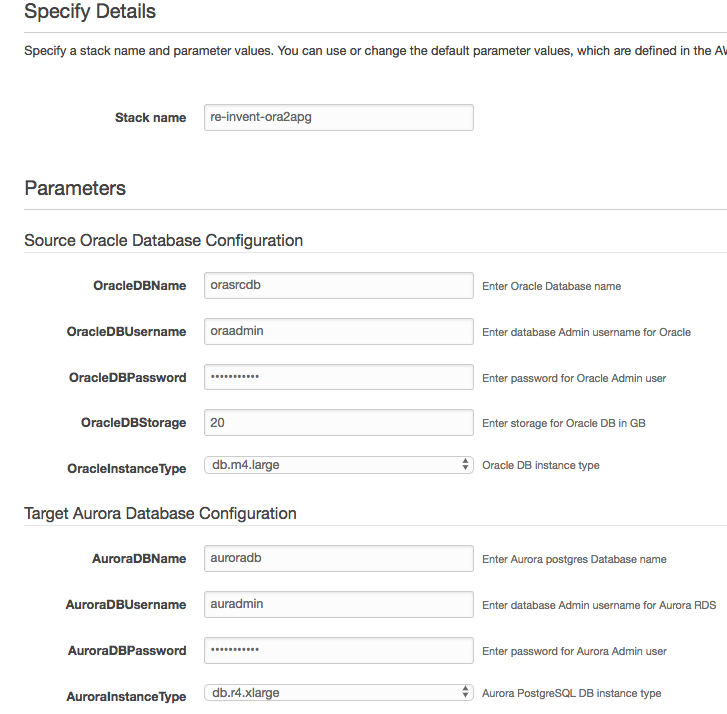
1. Click “Create stack” from the top left



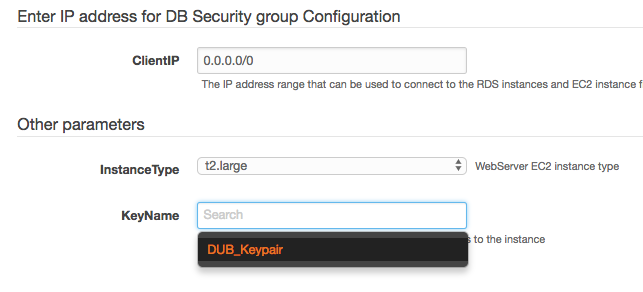
Click create stack, select **‘Specify an Amazon S3 template URL’** and specify the S3 template URL as <https://s3-us-west-2.amazonaws.com/nyc-loft/PostgreSQL_Migration_CloudFormation_template.json>



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

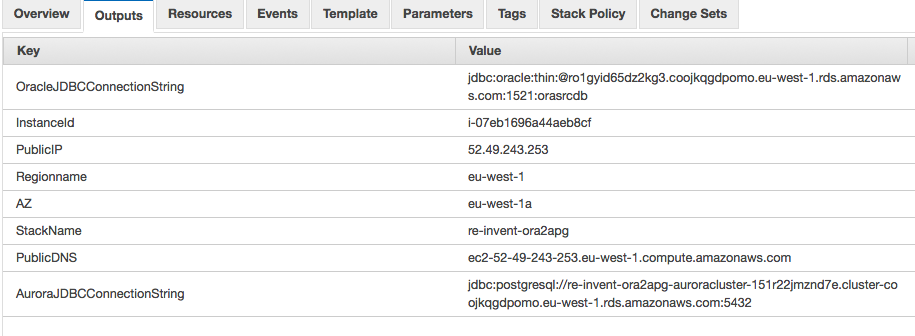


In the **Other Parameters** section choose the key for the EC2 instance as shown below Hit next in the next page and click on create. Wait for the entire stack to launch and become available.

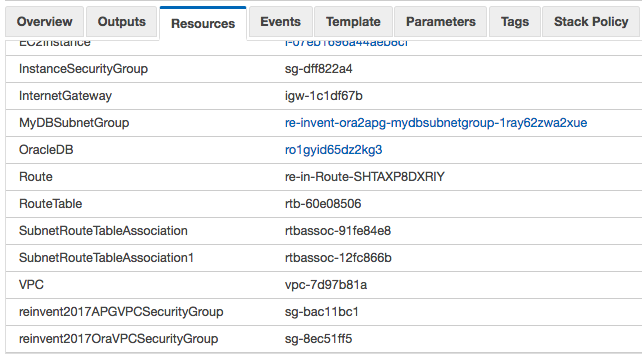


\*Leave everything else default. Once the CloudFormation stack launch is complete, you should see 2 RDS instances in your account in the region you created the stack. Approximately, it will **take 15 to 20 minutes** for creating the Stack.

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

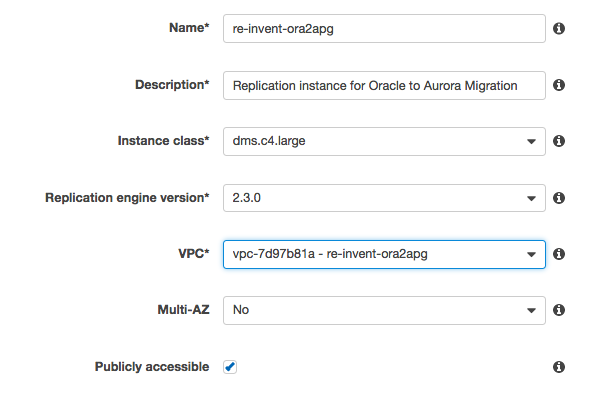


1. Once stack is available, go to the DMS console and launch a replication instance by selecting “Replication Instance” on the left side menu. Choose the same VPC as your stack name (this VPC was created by the CloudFormation Stack).

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 2.4.0 version.

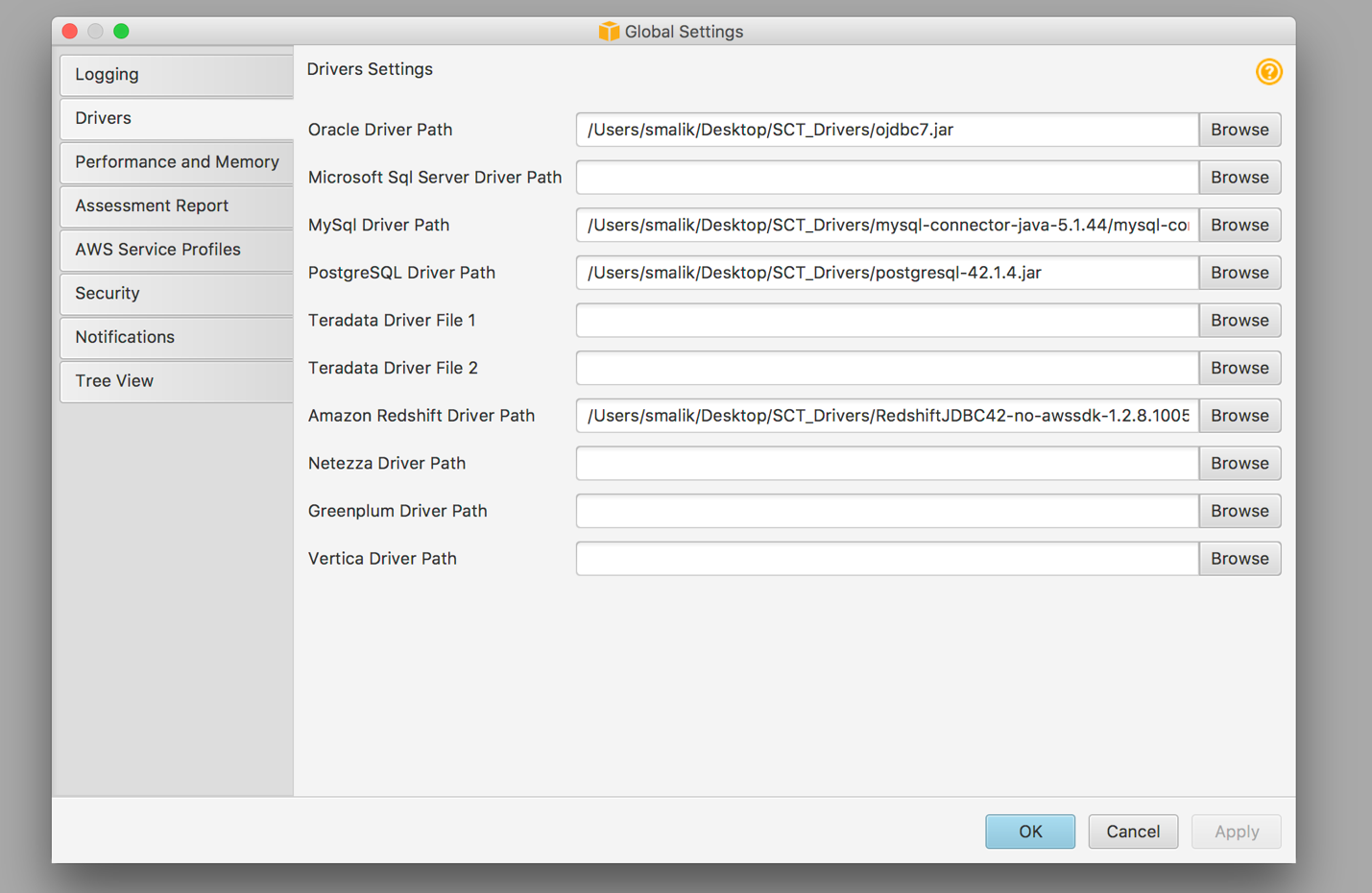
Please leave the rest of the values/advanced settings to the default values.

Here is a screenshot of the create replication instance page –

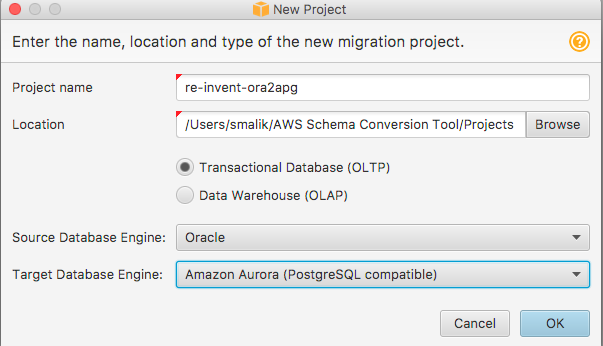


**Schema Conversion:**

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



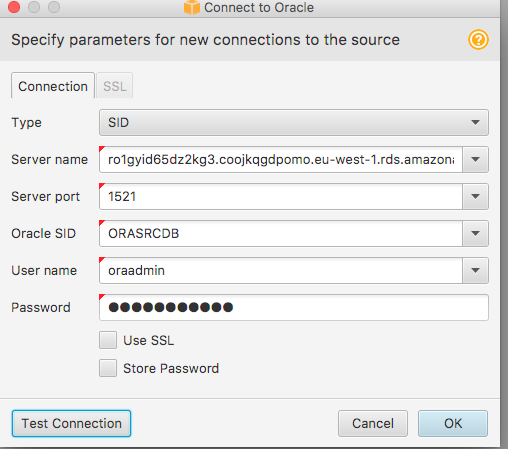
1. Create a New Project using File-> New Project



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)

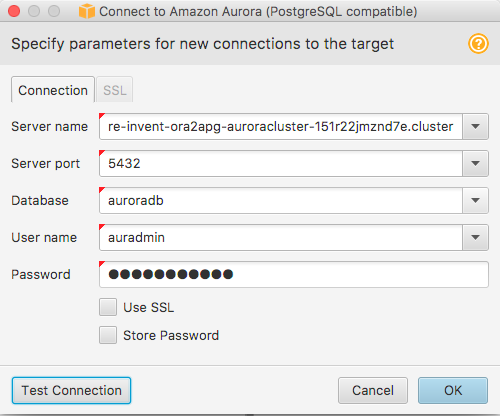


If the defaults were accepted in CloudFormation, the password is oraadmin123. **Substitute the server name with the RDS Oracle Database Endpoint launched in your VPC.**

**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)

****

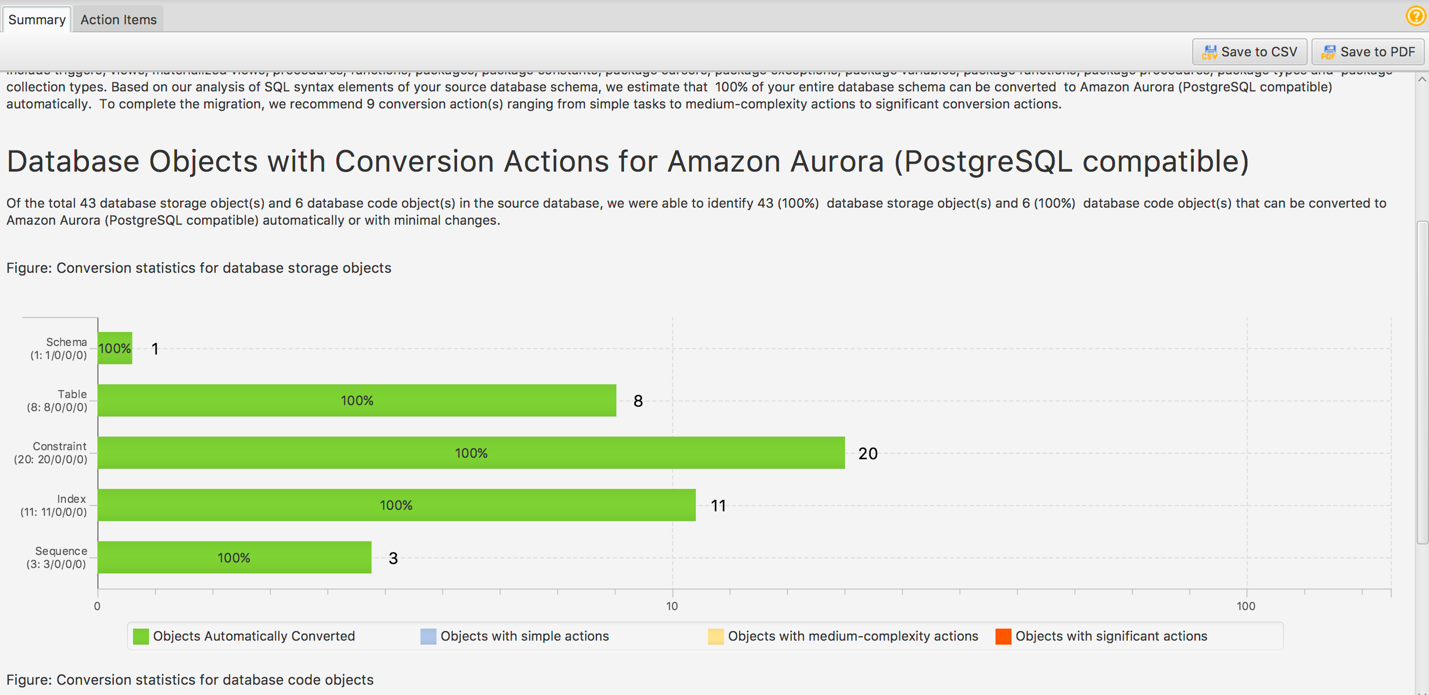
If the defaults were accepted in CloudFormation, the password is **auradmin123**. **Substitute**

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

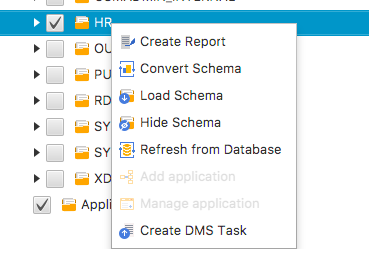
**NOTE:** If you’re not able to connect, please add the IP address from which you’re trying to connect to the instances from to the inbound security group rules for the security groups attached to the Oracle RDS and Aurora PostgreSQL instances.

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

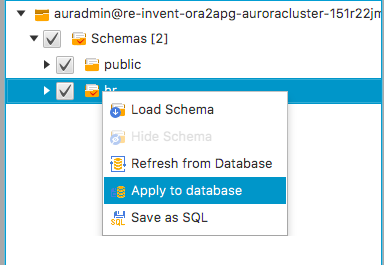


1. Now, right click and click “Convert schema”. The schema will be converted and shown on the PostgreSQL instance (it has not been applied yet).



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.

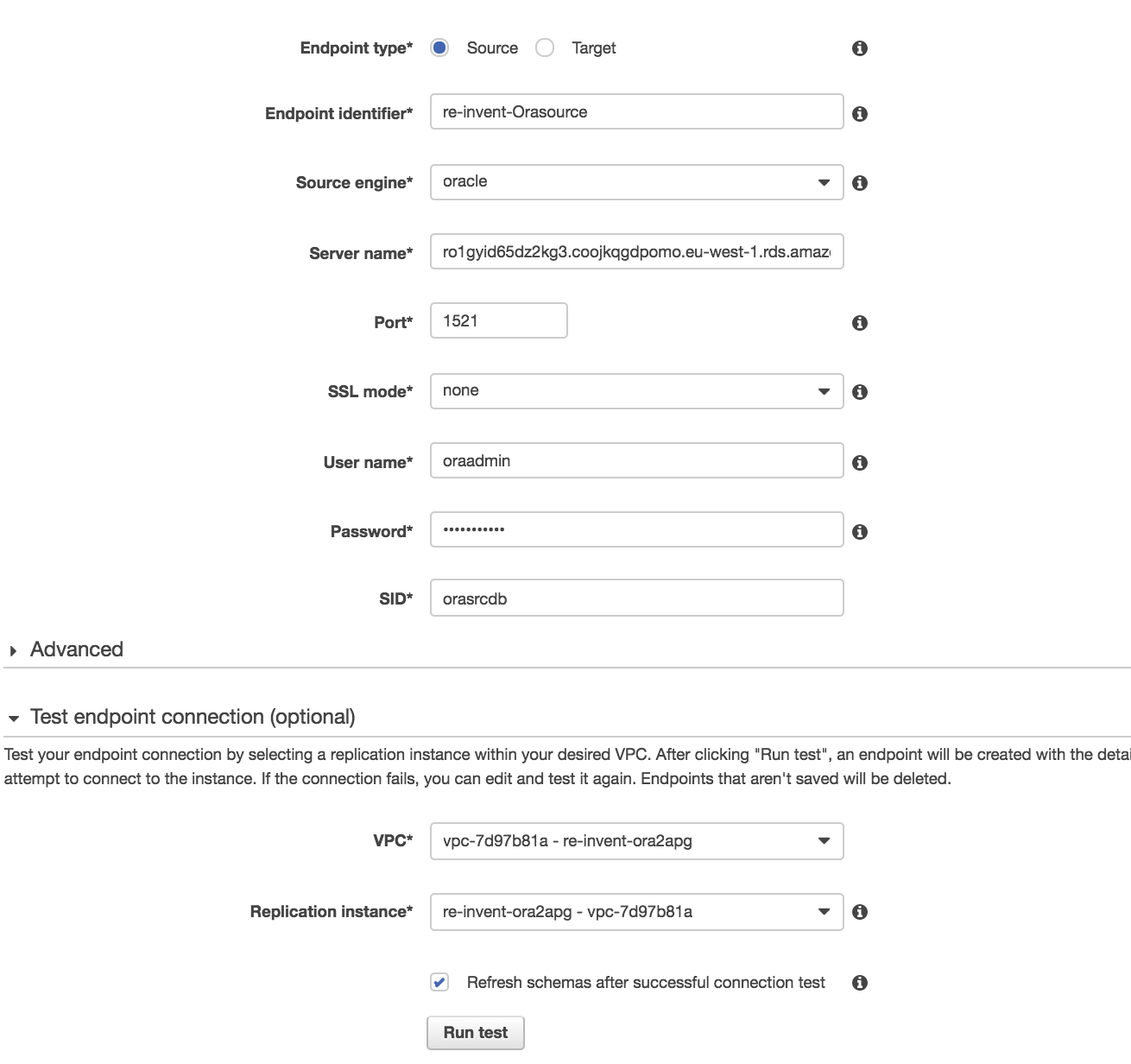
**NOTE:** At this point, all the database objects from the source HR Oracle schema must be converted and created in PostgreSQL in the HR schema in the Aurora postgres database.

**Data Migration:**

**Create a source endpoint:**

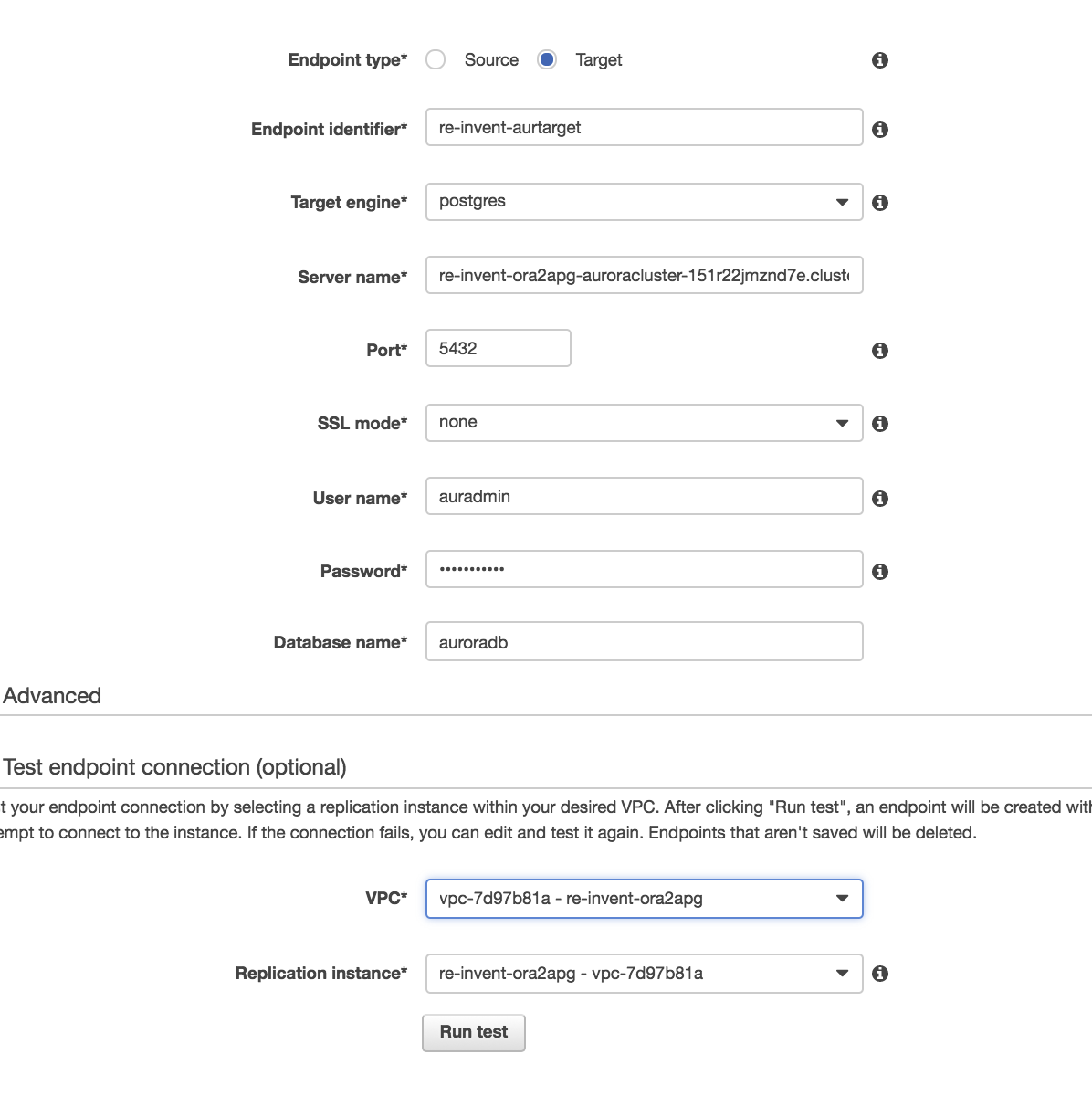
* Select Endpoints from the menu on the left and “click the "create endpoint" button at the top.
* Endpoint identifier: orasource
* Endpoint Type: source
* Engine: Oracle
* Server Name: RDS Oracle Source endpoint from your account
* Port: 1521
* SSL Mode: none
* User name: oraadmin
* Password: oraadmin123 (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

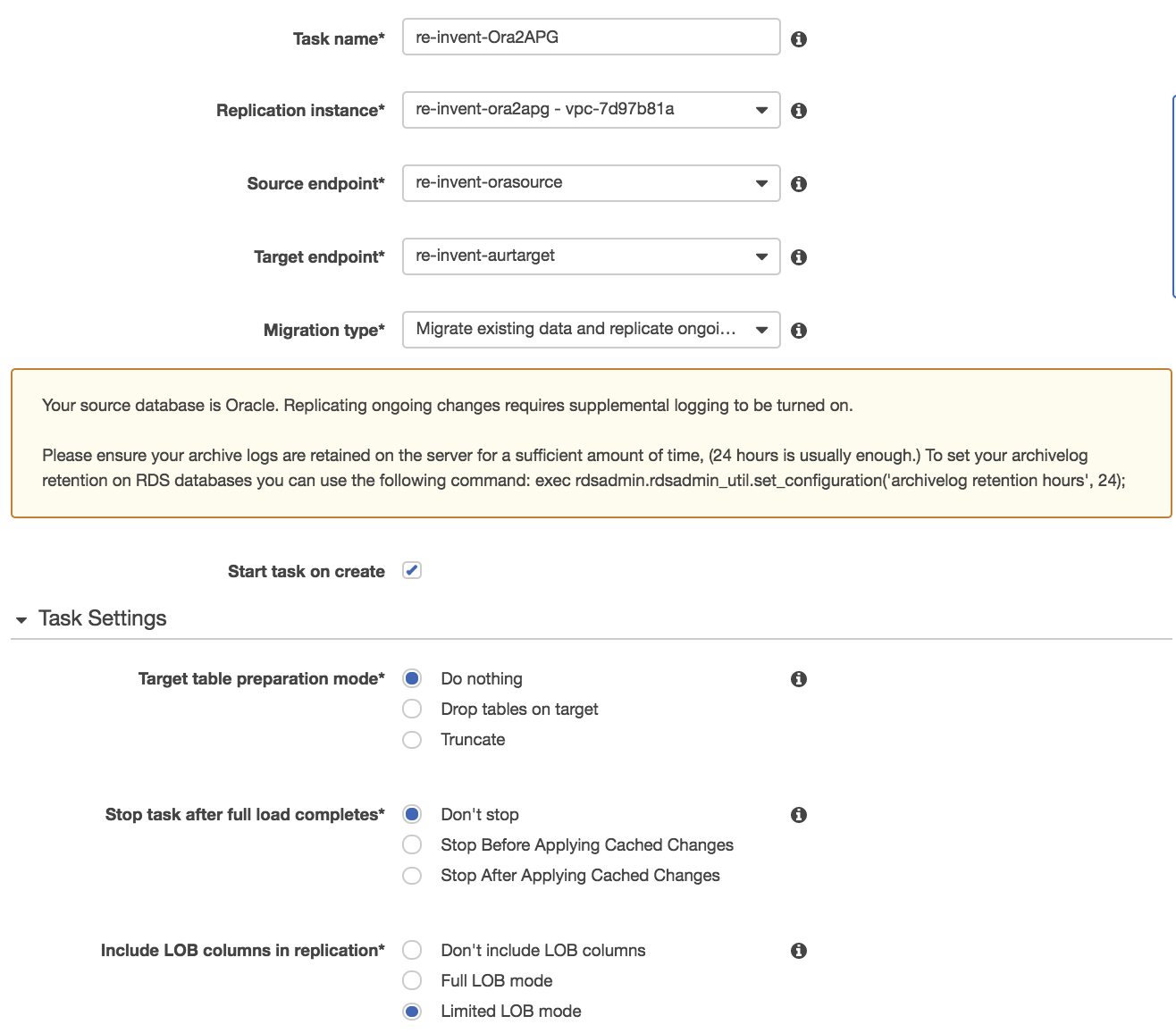
****

**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)

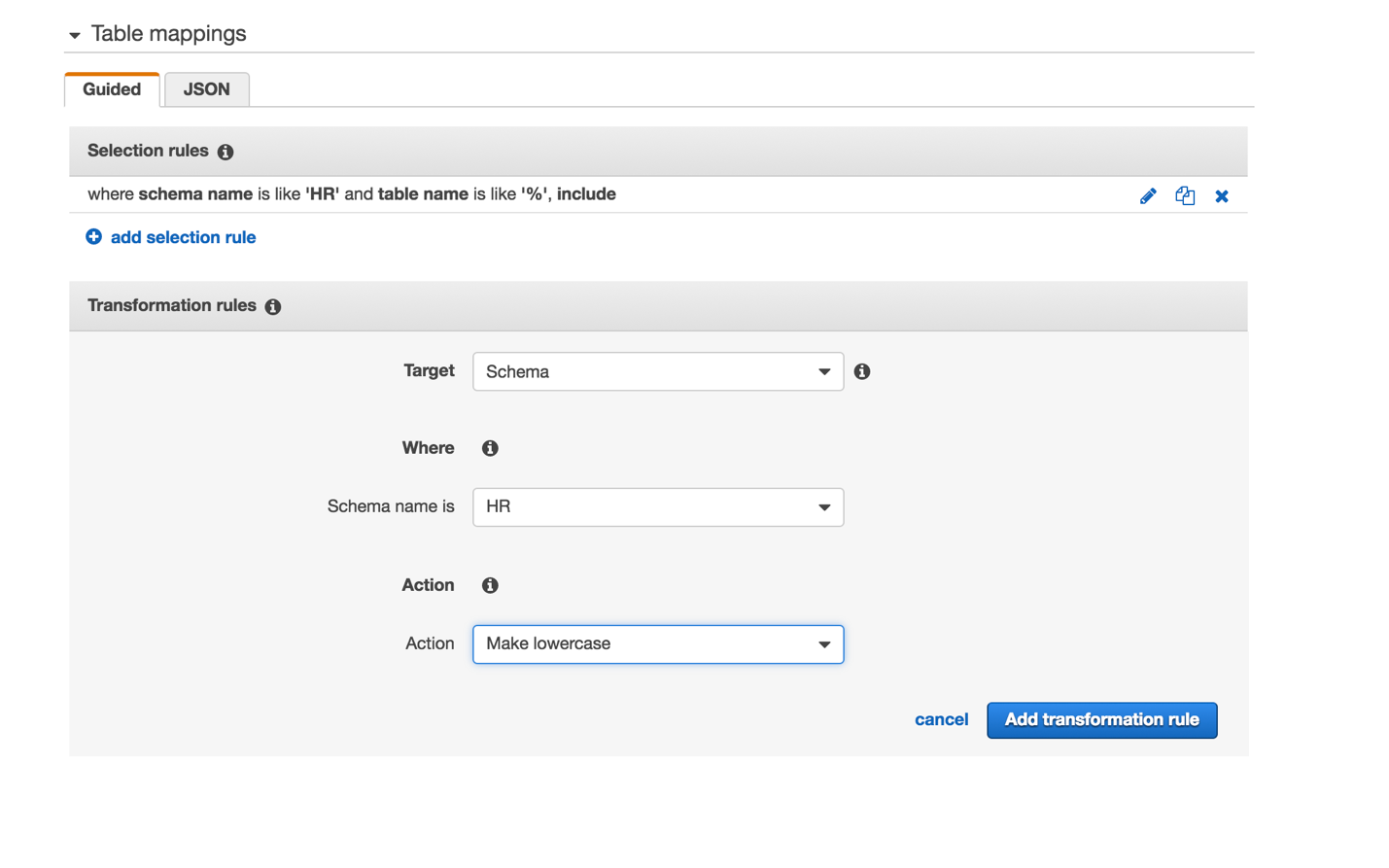
**Table Mappings – Selection rules:**

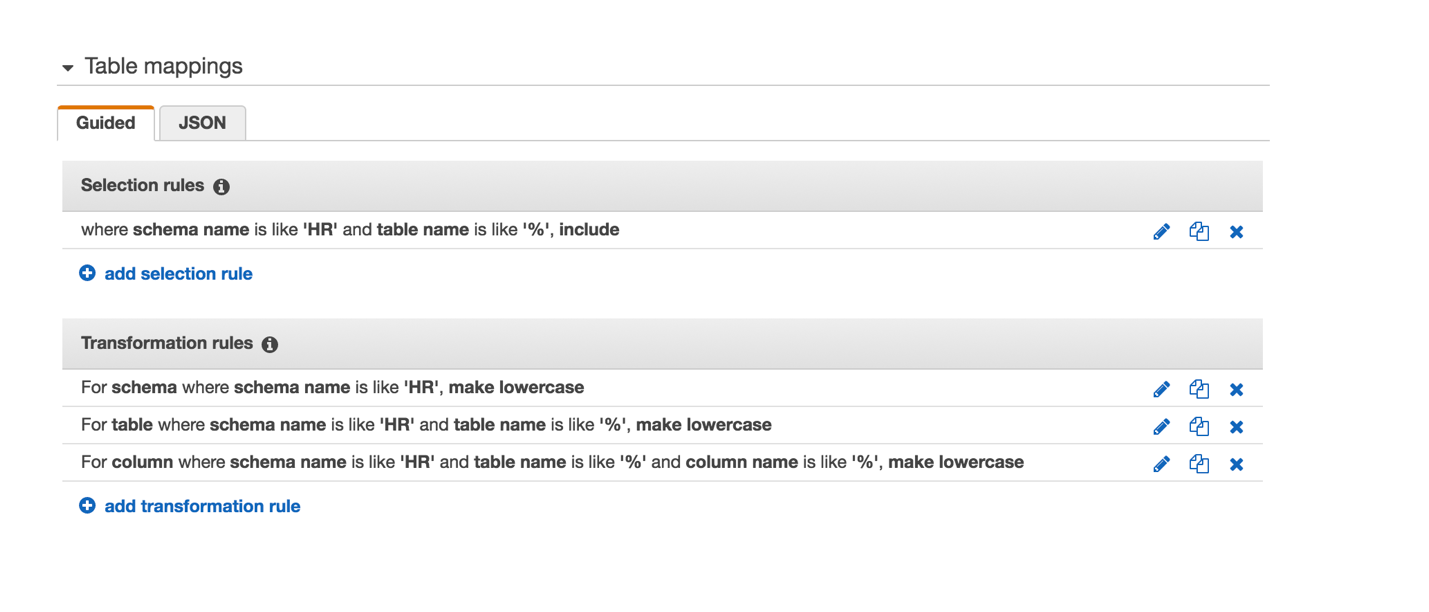
1. Select HR from the drop-down box Click the "Add selection rule" button.

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.

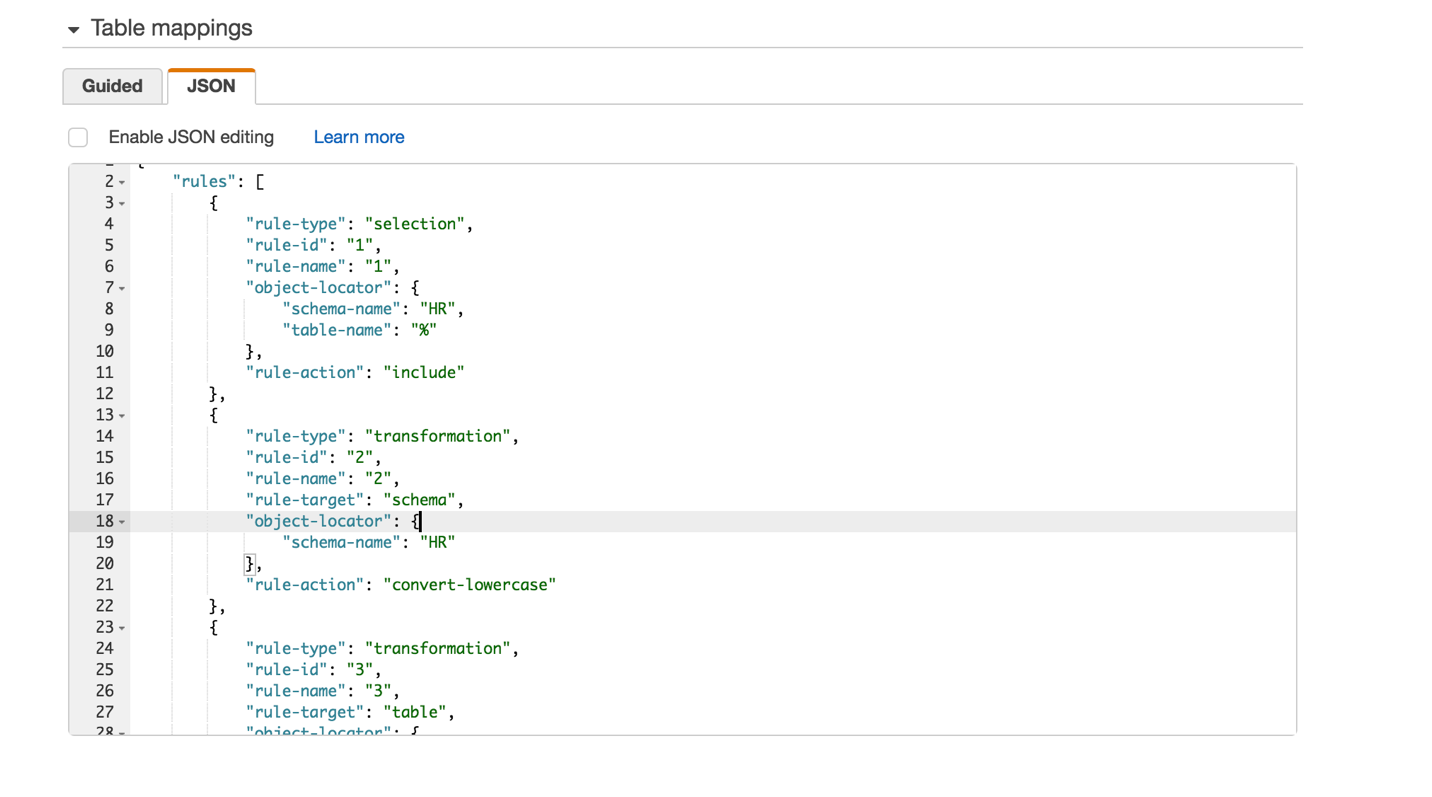
3. Click on “add transformation rule” -> Choose table for target dropdown -> Schema name is HR and table name is % -> Choose “make lower case” for action & click add transformation rule.

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

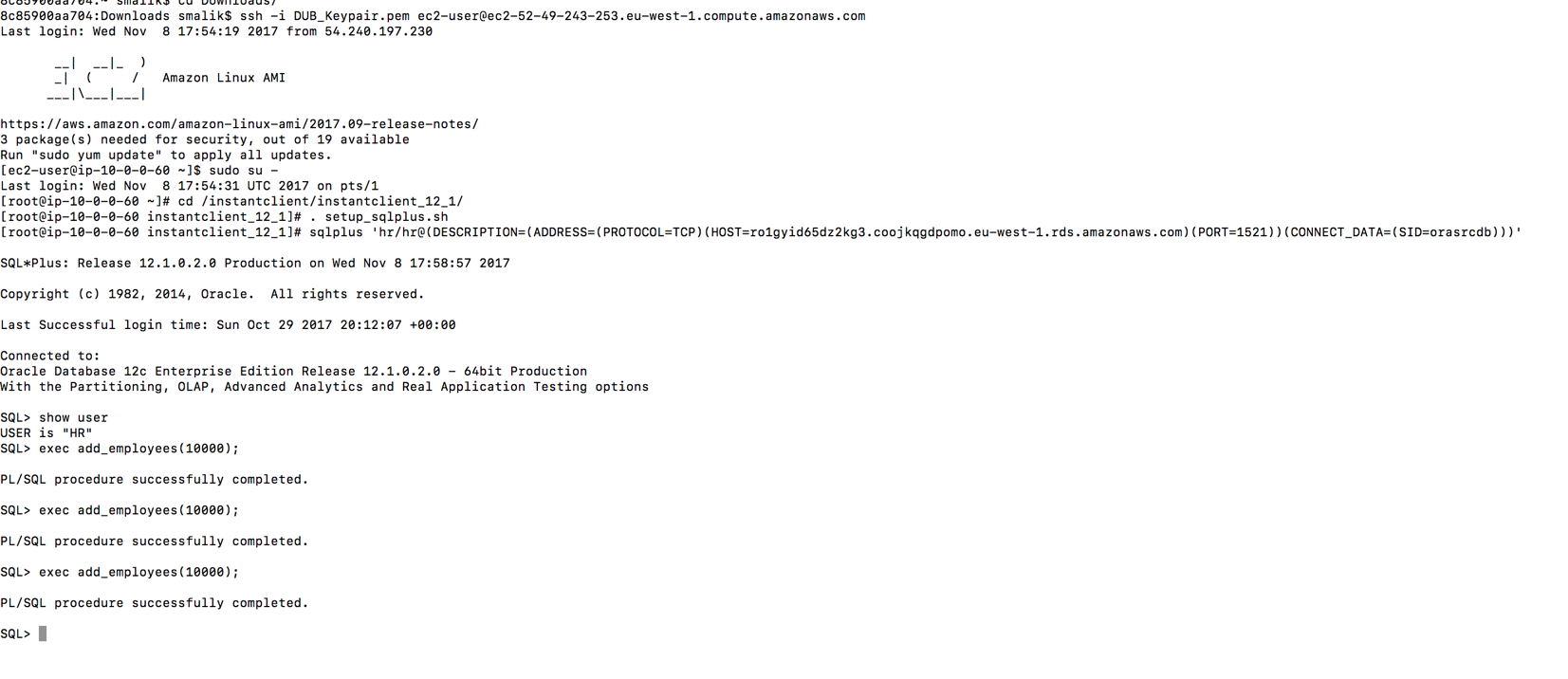
1. ssh –i <keypair.pem> ec2-user@<ec2 host name public DNS>

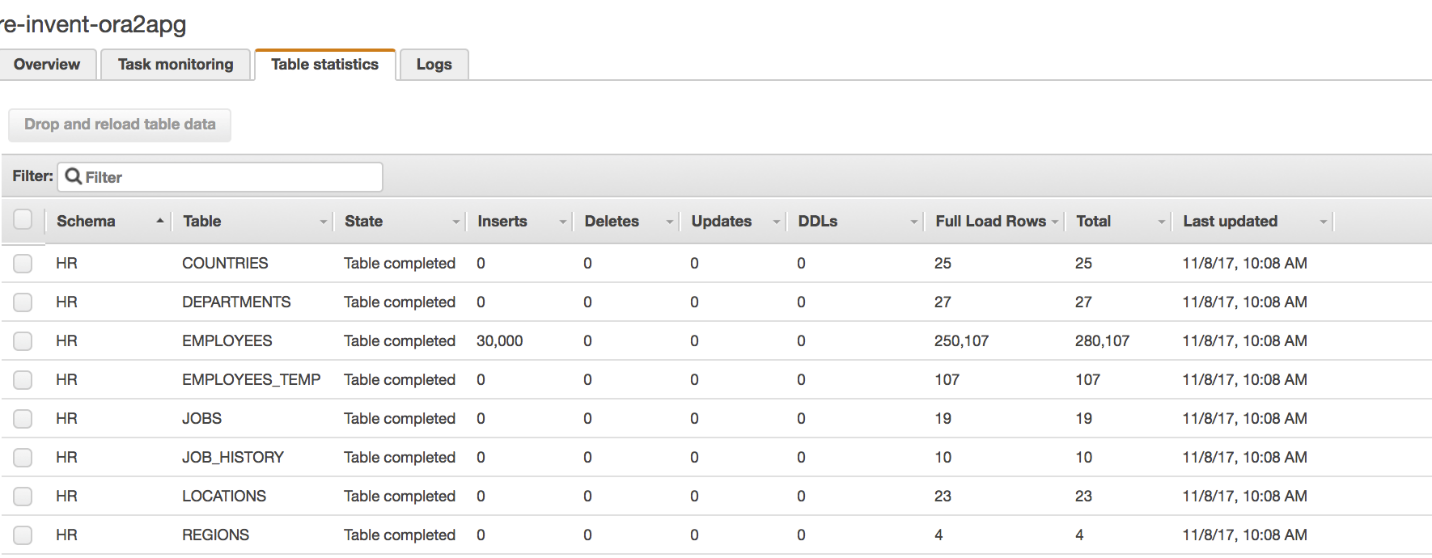
(please note if you are using Putty to connect to EC2 Instance, then please use “.ppk” file instead)

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);



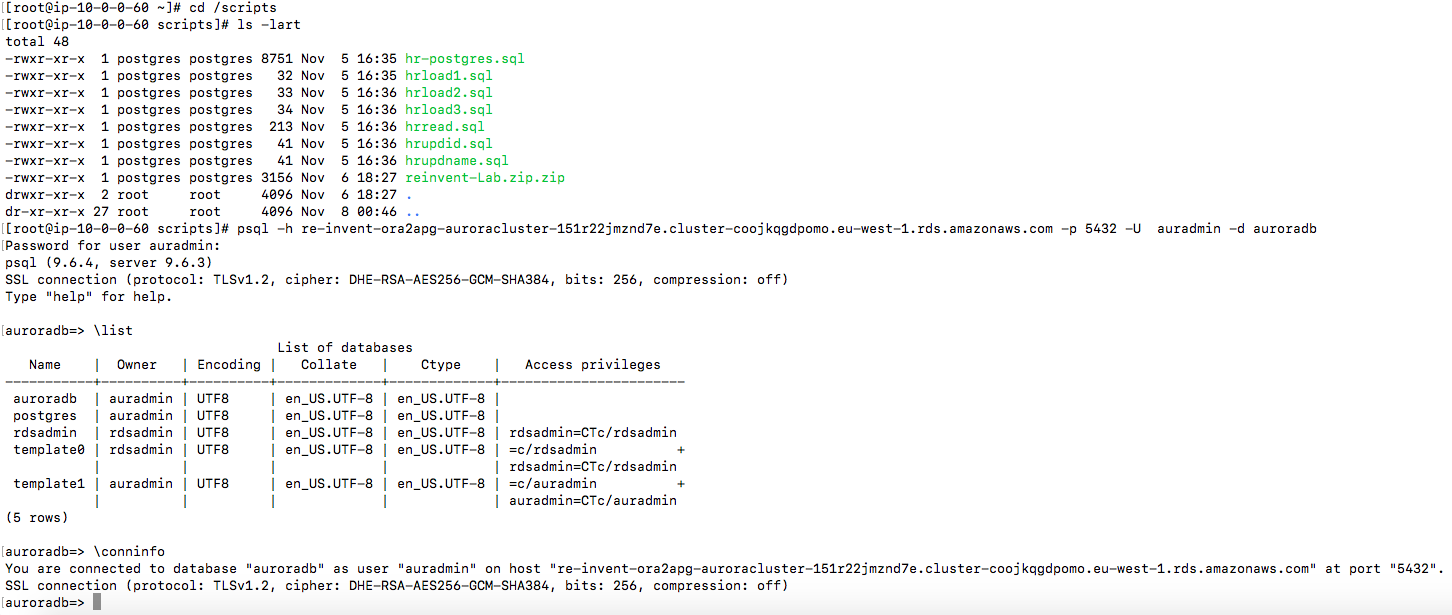


**Steps to connect to Aurora PostgreSQL Database using psql:**

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)



# Lab2: Walkthrough of Aurora Console and Read replicas, DB Parameter groups.

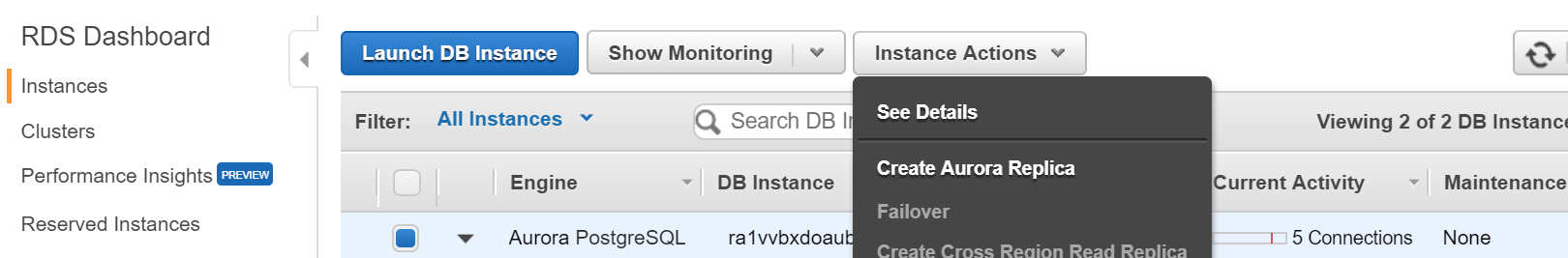
In this exercise, we will walk through the important DBA tasks such as creating read replicas, making changes in DB Parameter groups, enabling performance insights for monitoring.

|  |
| --- |
| Note: We recommend to use the psql installed in the EC2 instance to execute all the commands used in the Labs. However, if you are comfortable with other PostgreSQL client tools such as [pgadmin](https://www.pgadmin.org/), feel free to run the postgreSQL commands.  Set search\_path to ‘hr’; ( similar to alter session set current\_schema in oracle)) |

Add :- Enable Performance insight and Enhanced monitoring

1. Go to RDS Dashboard and click Clusters and then instances.
2. Select the Aurora writer instance (check the column **replication role** in the console which should display writer) and create the read replica by clicking the option as displayed below:

Go to Aurora Instance, click Instance Actions and select Create Aurora Replica.



Use the below parameters in the **Create Aurora Replica** console:

**Instance specifications**

DB Instance Class – db.r4.large

**Settings**

Aurora Replica source - choose the Aurora Cluster instance launched by CloudFormation stack

DB instance identifier – **replica1**

**Network and Security**

Publicly Accessible – Yes

Availability Zone – No preference

**Failover**

Priority - Tier0

**Database Port**

Database ports – 5432

**Monitoring**

Enable Enhanced Monitoring - No

Enable Performance Insights - **Yes**

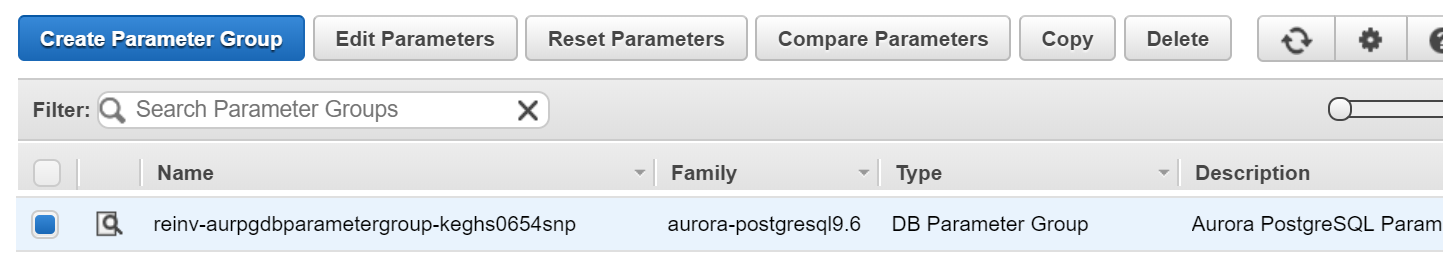
**Maintenance**

Auto-minor version upgrade - Yes

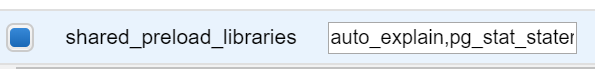
Repeat the above steps mentioned in 3) and create another replica with DB instance identifier as **replica2** and failover priority **as Tier1.**

|  |
| --- |
| Note: replicas may take around 5 to 10 minutes to launch. proceed to the next step. |

1. Go to Parameter groups in RDS Dashboard. Search for the DB Parameter group associated with the Aurora writer instance used in this lab (for example. parameter group name will start with the stack name followed by aurorapgxxx).



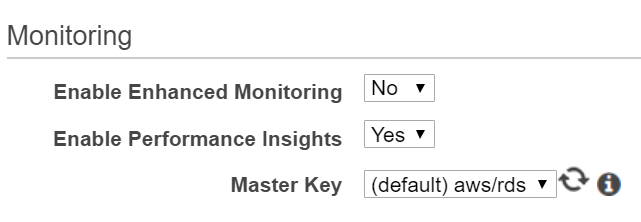
1. Click Edit Parameters
2. Search for the parameter [**shared\_preloaded\_libraries**](https://www.postgresql.org/docs/9.6/static/pgstatstatements.html)
3. Choose and edit the values as auto\_explain,pg\_stat\_statements under shared\_preloaded\_libraries ( This will enable you to use those extensions)



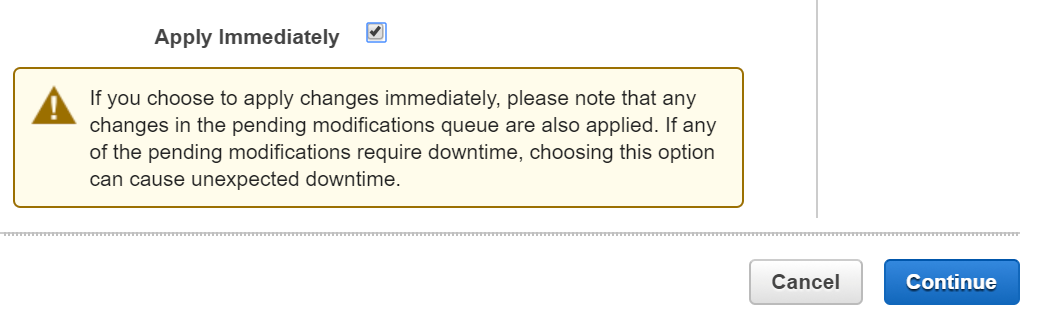
1. Save changes
2. Again, Search for [**auto\_explain.** **log\_**](https://www.postgresql.org/docs/9.6/static/auto-explain.html) and only modify the below parameters as defined below. The first parameter helps to log the SQL statements with execution time > 5s. The second parameter helps you to log the explain plan for the nested SQLs used in the plpgsql function.



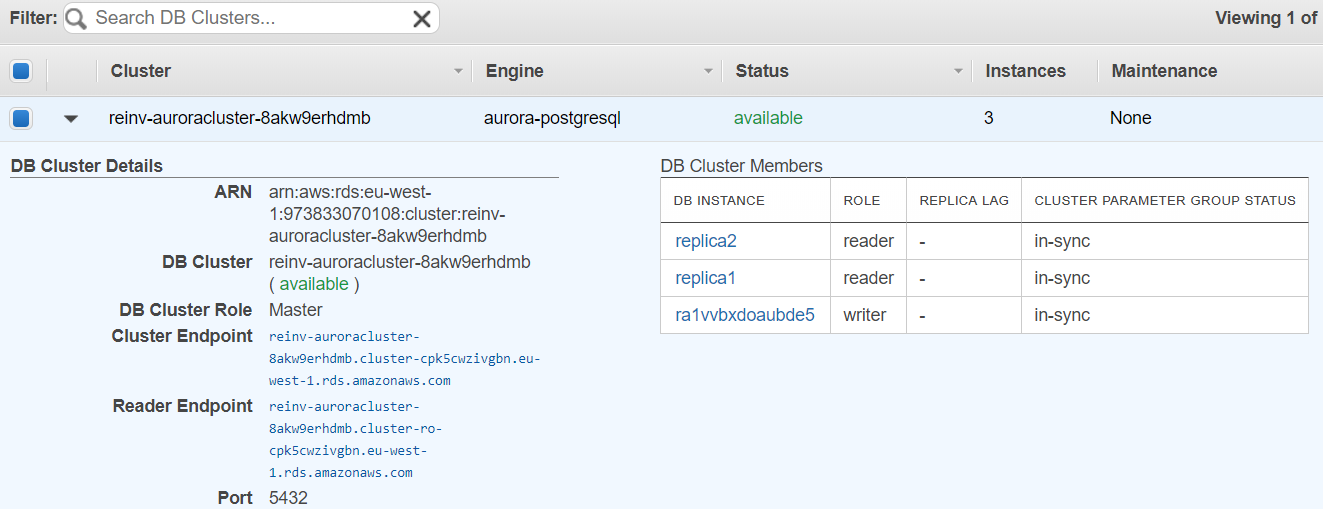
1. Save changes
2. Go to instances under RDS Dashboard and select the Aurora writer instance, click Instance Actions and then Modify.
3. Choose Yes for Enable performance insights section. Don’t make any other changes.



1. Apply the changes immediately by clicking the below option and continue , Click Modify DB Instance.



1. You can also check if modification is completed, by looking at the Events tab.
2. Go to Instances and reboot the Aurora writer instance to complete the parameter changes as you have modified the static parameters. (Go to intance -> instance actions -> reboot)
3. Check if the read replicas are launched. Understand the concepts behind Cluster Endpoint and Reader Endpoint and failover priorities.



|  |
| --- |
| Note: replica instances are launched using default.aurora-postgresql9.6 db parameter group. You can have different DB parameter groups for the writer and reader instance. |

1. Now, lets verify if the extensions are available in your database. Login to to the EC2 instance launched via CloudFormation as mentioned in earlier section.
2. Login as a root (sudo su -) and then issue the below command and provide password for auradmin use

psql -h <**Cluster End point**> -n -p5432 -U auradmin -d auroradb

psql –h ra54dqmy5gadq9.coojkqgdpomo.us-west-2.rds.amazonaws.com -n -p5432 -U auradmin -d auroradb

1. After logging into database via psql, run the below SQL commands.

show shared\_preload\_libraries; (/\* this is to verify the extensions are loaded \*/)

create extension pg\_stat\_statements; (/\* create the extension for pg\_stat\_statements \*/)

SELECT \* FROM pg\_available\_extensions WHERE name = 'pg\_stat\_statements';

show auto\_explain.log\_nested\_statements; /\* it should be on \*/

show auto\_explain.log\_min\_duration; /\* it should be 5s\*/

# LAB3 : Performance Insights

In this exercise, we will learn how to use Performance Insights and PostgreSQL extensions to analyze the top wait events and performance issues. We will run the following load test cases using [pgbench](https://www.postgresql.org/docs/9.6/static/pgbench.html) utility on employees table in the HR schema that is migrated via AWS DMS.

1. Go to Performance insights in RDS Dashbaord
2. Select the Aurora Writer instance and open the PI dashboard in your console and keep it ready for view
3. Login to the EC2 instance and login as a root (sudo su -) and run the below command.

psql -h <Cluster End point> -n -p5432 -U auradmin -d auroradb -f /scripts/hr-postgres.sql

|  |
| --- |
| The hr-postgres.sql ceates the necessary plpgsql functions required for the load test, reset the employee\_seq and analyze the hr schema. |

Now, we are ready to run some load on the Aurora Instance to understand the capabilities of Perormance Insights.

### **Load Scenario-1 : High volume insert via pgbench**

1. Login to EC2 instance and run pgbench workload using the below command.

pgbench -h <Cluster\_Endpoint> -n -U auradmin -d auroradb -c 10 -T 300 -f /scripts/hrload1.sql > /scripts/pgload1-run1.log

|  |
| --- |
| The **hrload1.sql** will ingest employee records using plpgsql function **add\_employee\_data**. This function uses employee\_seq to generate the next employee\_id, randomly generate data including first\_name, salary with department\_id from departments table. Each function call will insert 5 records. This test will be executed for 5 minutes with 10 clients. |

1. Review the PI dashboard and check the top wait events, AAS (Average Active Sessions) for the duration.
2. Note down the key metrics in the pgbench output such as number of transactions, statement latency.
3. Now, lets login to the database and check the top queries.

psql -h <Cluster End point> -p5432 -U auradmin -d auroradb

* 1. Run the below SQL query to understand the load caused by the above run using pg\_stat\_statments extension (Top 5 SQLs by execute time and CPU Consumption)

SELECT substring(query, 1, 50) AS short\_query, round(total\_time::numeric, 2) AS total\_time, calls, round(mean\_time::numeric, 2) AS mean, round((100 \* total\_time /

sum(total\_time::numeric) OVER ())::numeric, 2) AS percentage\_cpu FROM pg\_stat\_statements ORDER BY total\_time DESC LIMIT 5;

1. Lets rerun the same function with 50 inserts per execution and check the impact on wait events. Use hrload2.sql for this run.

pgbench -h <Cluster\_Endpoint> -n -U auradmin -d auroradb -c 10 -T 300 -f /scripts/hrload2.sql > /scripts/pgload1-run2.log

1. Go to PI dashboard and check the top wait events now and see if there are any changes.
2. Login using psql and check the rerun the pg\_stat\_statements query and resource consumption now (optional).
3. Verify which wait event reduced when compared to the previous run?
4. Can you verify whether the overall throughput has increased meaning you were able to insert more employee records by comparing the throughput and latencies reported by pgbench between the runs?

### **Load Scenario-2: High volume update via pgbench**

In this exercise, we will run updates on the employee table using **update\_employee\_data\_fname and update\_employee\_data\_empid** functions**.**

1. Login to EC2 instance and run pgbench workload using the below command.

pgbench -h <Cluster\_Endpoint> -n -U auradmin -d auroradb -c 10 -T 180 -f /scripts/hrupdname.sql > /scripts/pgload2-run1.log

|  |
| --- |
| The hrupdname.sql will update employee salary details of employees using plpgsql function **update\_employee\_data\_fname**. This function randomly selects the employee records and check if their salary is within range (min and max salary of their job), if not update their salary using their first\_name. Each function call will execute 5 records randomly. This test will be executed for 3 minutes with 10 clients. |

1. Go to PI dashboard and check the top wait events and AAS for the run duration. Check the CPU utilization.
2. Let’s look at the performance stats using pg\_stat\_extensions. Connect to Aurora instance using

psql -h <Cluster End point> -n -p5432 -U auradmin –d auroradb

1. Run the below SQL query

SELECT substring(query, 1, 50) AS short\_query, round(total\_time::numeric, 2) AS total\_time, calls, round(mean\_time::numeric, 2) AS mean, round((100 \* total\_time / sum(total\_time::numeric) OVER ())::numeric, 2) AS percentage\_cpu FROM pg\_stat\_statements ORDER BY total\_time DESC LIMIT 5;

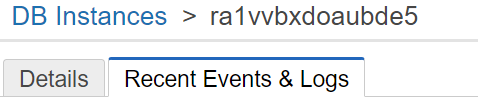
Let’s look at what is the explain plan used in the plpgsql function. In order to capture the explain plan in the logs, set the below DB parameter at your session level.

1. set auto\_explain.log\_min\_duration=10; /\* This will enable to log any SQLs which is taking more than 10ms to be logged in /error/postgres.log \*/
2. show auto\_explain.log\_min\_duration; /\* Verify the setting \*/

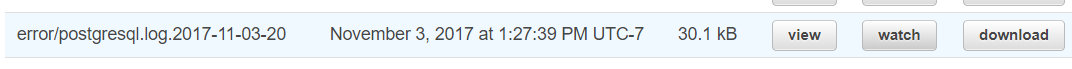
Run the [explain analyze](https://www.postgresql.org/docs/9.6/static/sql-explain.html) to capture the explain plan as well as execute the query and capture the statistics.

1. explain analyze select hr.update\_employee\_data\_fname(10);

1. Go to RDS Dashboard -> Aurora Writer instance -> Instance Actions ->See Details and then click Recent Events & Logs



1. Scroll Down to Logs section and identify the latest log and click Watch



1. Review the explain plan used by the pgplsql function in the logs ( you can watch in the console or download the log and review at your desktop).
2. Check the explain plan for the update statement used in the pgplsql function. Is there an issue with the explain plan?
3. Now, lets run the procedure to use the employee\_id and rerun the load. This time, lets use the procedure hrupdempid.sql which updates the table by indexed column.
4. Login to EC2 instance and run pgbench workload using the below command.

pgbench -h <Cluster\_Endpoint> -n -U auradmin -d auroradb -c 10 -T 180 -f /scripts/hrupdid.sql > /scripts/pgload2-run2.log

|  |
| --- |
| The **hrupdempid.sql** will update employee salary details of employees using pg/plsql function **update\_employee\_data\_empid**. This function randomly selects the employee records and check if their salary is within range (min and max salary of their job), if not update their salary using their employee\_id. Each function call will execute 5 records randomly. This test will be executed for 3 minutes with 10 clients. |

1. Review the PI dashboard and check the top wait events again. Do you see any change in the top wait events? What are the key observations?
2. Compare the execution results using pg\_stat\_statements query.
3. Compare the throughput and latencies reported by pgbench between the runs.

### **Load Scenario-3: Connection Load Balancing across replicas**

This scenario will help you to understand how the connections are load balanced across multiple read replicas. Remember to use Aurora Cluster reader endpoint for this exercise.

1. Login to linux instance and run pgbench workload using the below command.

pgbench -h <**Cluster Reader Endpoint>** -n -U auradmin -d auroradb -C -c 8 -T 180 -f /scripts/hrread.sql > /scripts/pgload3.log

|  |
| --- |
| The **hrread.sql** randomly selects the employee records using emp\_details\_view using the belw SQL query.  WITH empid AS (select max(employee\_id) AS max\_id, min(employee\_id) AS min\_id from hr.employees) select \* from hr.emp\_details\_view where employee\_id = (SELECT floor(random()\*(max\_id-min\_id+  1))+min\_id FROM empid); |

1. Check the load distribution across multiple read replicas using performance insights dashboard. Note down the key observations.

# Clean up CloudFormation stack:

1. After testing, stop the task and delete the task.
2. Delete the source/target endpoints.
3. Delete the replication instance.
4. Delete the subnet group created for DMS.
5. Delete the Aurora Read Replicas (replica 1 and replica2)
6. Delete CloudFormation stack from the CloudFormation console.