Migrating Amazon RDS for PostgreSQL to Azure Database for PostgreSQL using Azure Database Migration Service

This article provides details about how to perform an online migration of data from Amazon RDS for PostgreSQL to Azure Database for PostgreSQL by using Azure Database Migration Service (DMS). It assumes that you have already deployed instances of Amazon RDS for PostgreSQL and Azure Database for PostgreSQL, which are out of scope.

Below are the broad steps required for this migration.

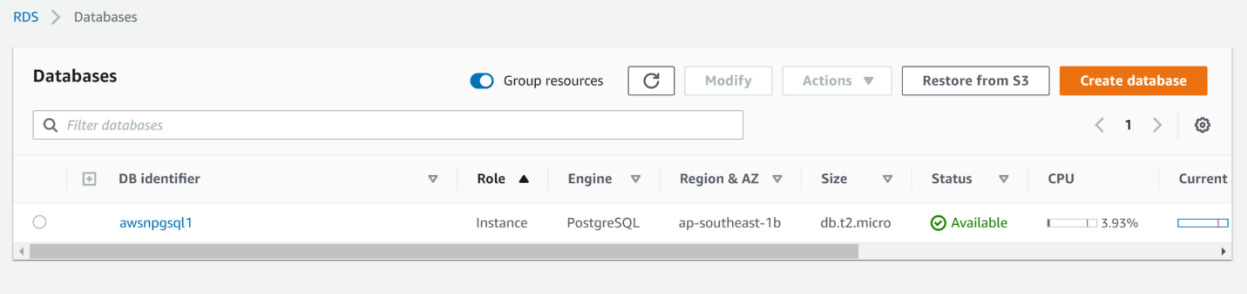
1. AWS RDS connectivity to Azure Database Migration Service using a public IP exposed from DMS and added this IP to AWS RDS security group which is also publicly accessible.
2. Backup schema from AWS RDS and remove RDS specific code and indexes/triggers from the backup script.
3. Restore the script on target Azure DB for PostgreSQL
4. Change logical_replication to 1 in AWS RDS and run the Migration Activity till Databases are sync'ed.

I have following deployed in my AWS account and Azure Subscription.

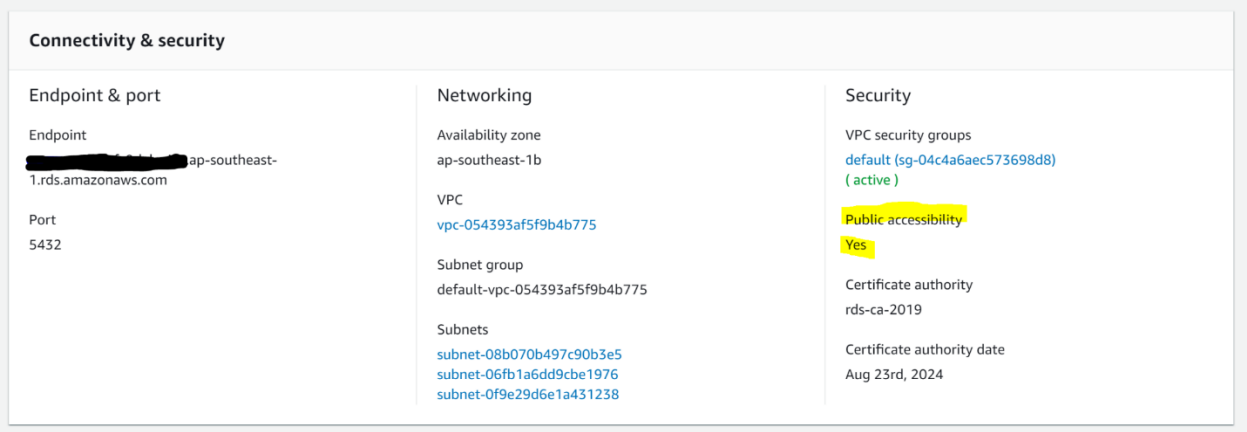
1. AWS RDS for PostgreSQL deployed in a VPC
2. Azure Database for PostgreSQL

I have a sample schema running in AWS RDS which has a single table called Users in a database userdb which we need to migrate. I have a simple insert script which is inserting data into this table in intervals and we will see how to migrate this data using online migration. Once the data is synced we will simulate a cutover to the Azure DB for PostgreSQL.

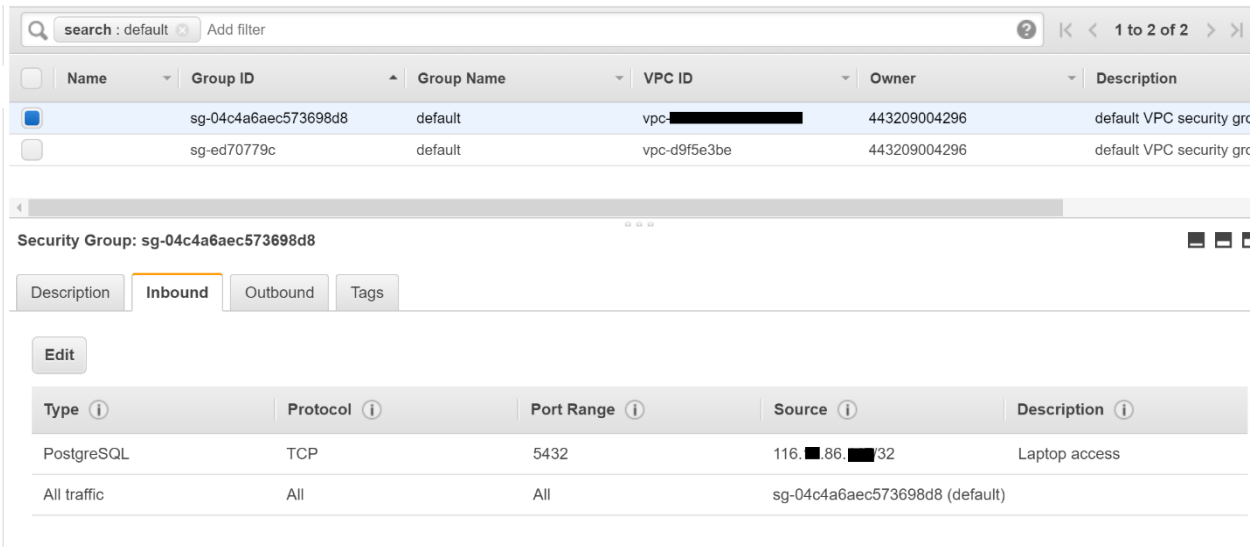
Below screenshot shows the AWS RDS for PostgreSQL provisioned in its own VPC.



Below screenshot shows the connectivity & security configurations.

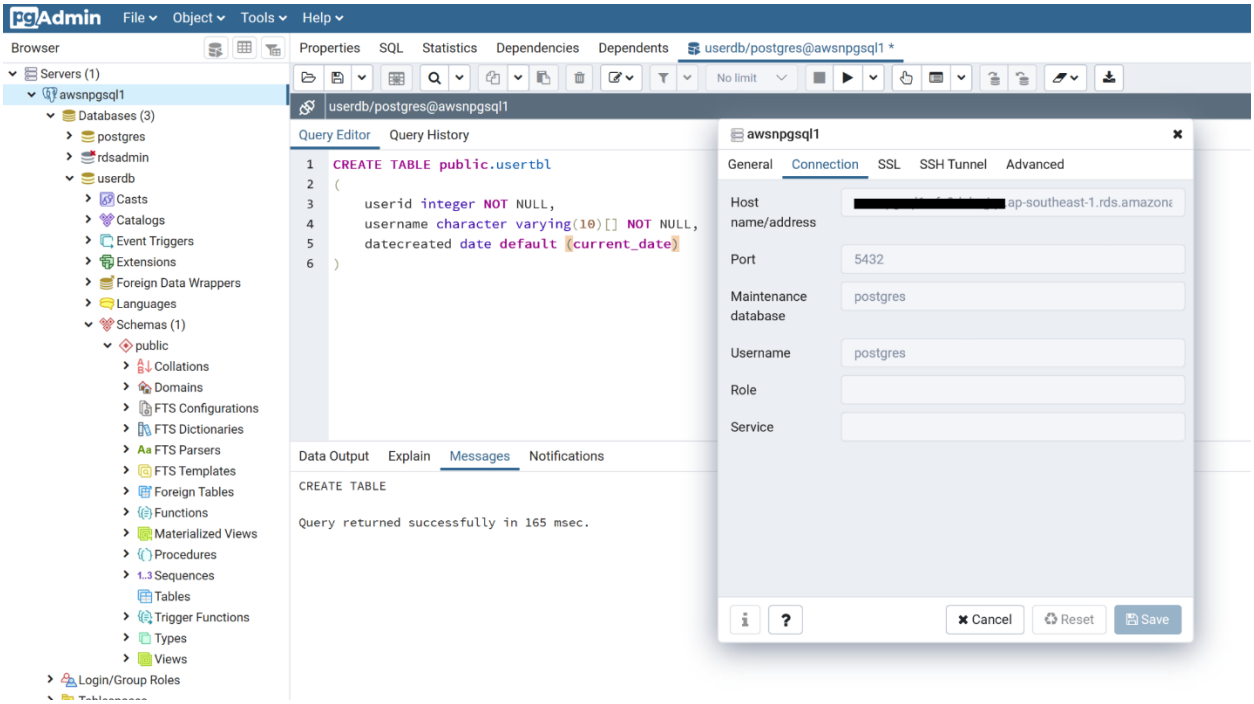


I have made public accessibility of this instance as "yes" so that Azure DMS service can access the same. You can do this by selecting your AWS RDS instance and clicking on modify. And on the next screen choose “Yes” for **Public accessibility** under **Network and Security**. For simplicity, I will use pgadmin to connect and post some transactions on the AWS RDS instance. For that I need to add a security group rule to allow pgadmin to connect from my laptop to this AWS RDS instance. Below are the screenshots of the security group config to allow connections from my laptop.

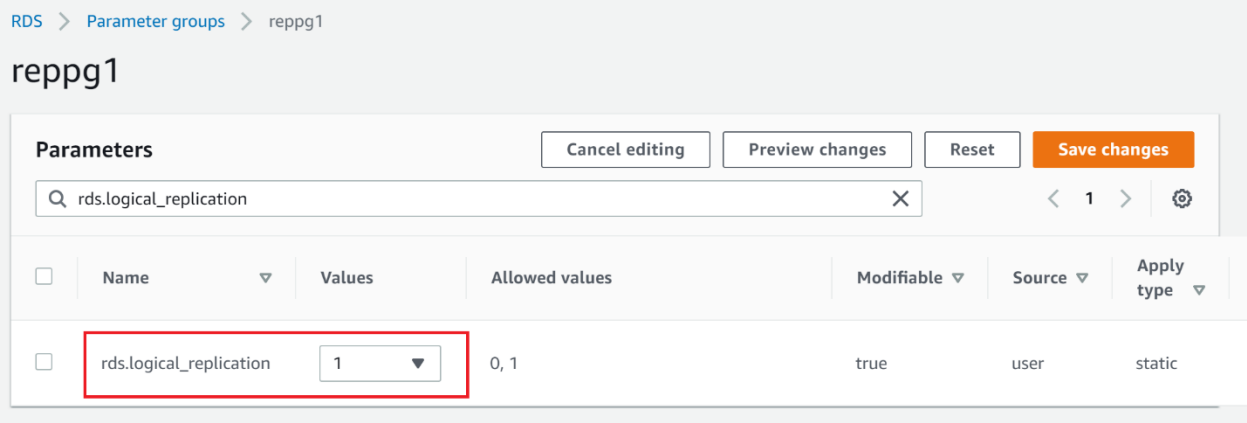


To add the above rule, I will click on the name of VPC security groups under the **Connectivity and Security** and on the next screen add an inbound rule to allow "PostgreSQL" type of traffic with TCP protocol and Port 5432 from "My IP" source. I will give it a description as "Laptop access".

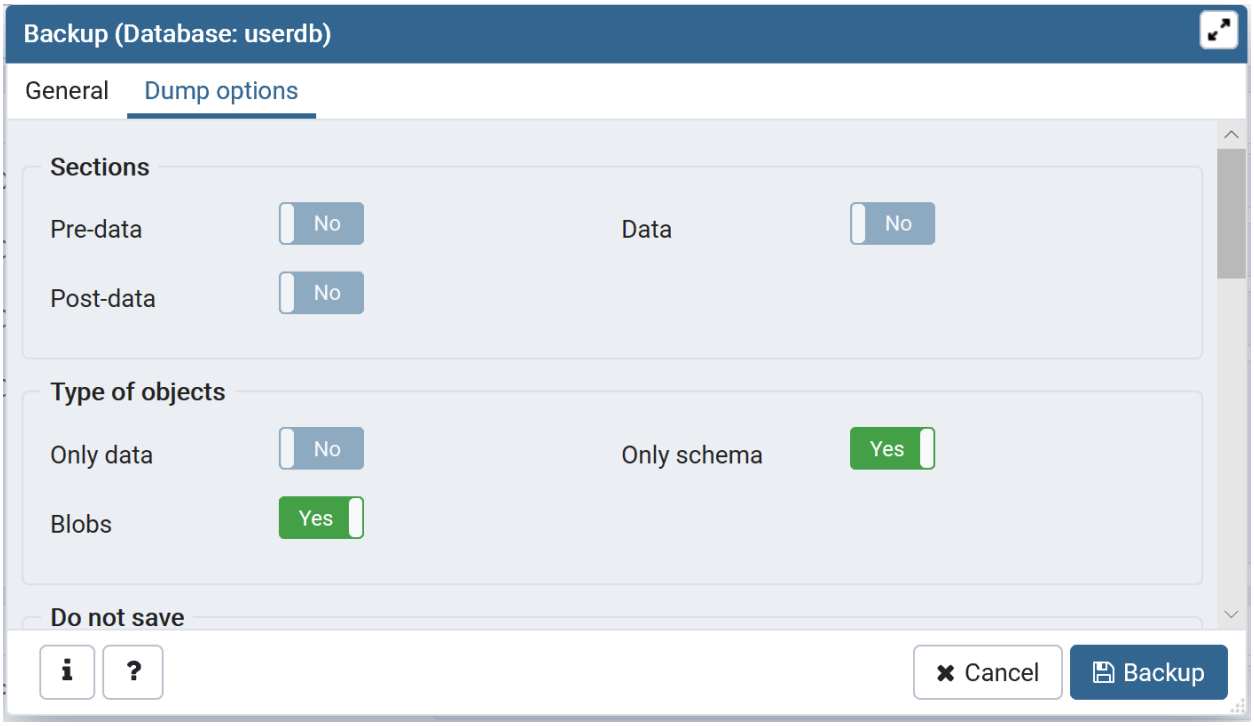
Now let's look at the schema. Below screenshot shows pgadmin connected to the instance and a database called "userdb" with a table called "usertbl".



Now let’s prepare this instance for replication. For Azure DMS to do online migration, we need the wal_level to be set to logical. To achieve this in AWS RDS, I had to change the rds.logical_replication to 1. To do this I will create a new parameter group of postgres11 family. Once created I will modify the rds.logical_replication parameter to 1. Below screenshot shows the same.



After this I will modify the AWS RDS instance to use this parameter group and then reboot the instance to apply the changes. While we do this, remember to select **Apply Immediately** under the **scheduling of modifications** group while modifying the instance and then reboot the instance. Also make sure that under **Configuration** tab, parameter group shows as the new parameter group name that you created and also (in-sync) should be showing against the name and not (pending reboot).



Next step is to create the same schema in the destination as DMS does not create the schema in PostgreSQL online migration. You can use pg_dump or psql to do so. I will take a dump from AWS RDS with schema_only option and execute the SQL script on the destination in Azure Database for PostgreSQL.

First I will create the destination database in Azure and then execute the script taken in the above steps using psql as I took a plain sql dump. I will also have to add the IP of my laptop to firewall of Azure Database for PostgreSQL. Next, I will remove RDS specific statements in the script like the one mentioned below else the script will fail.

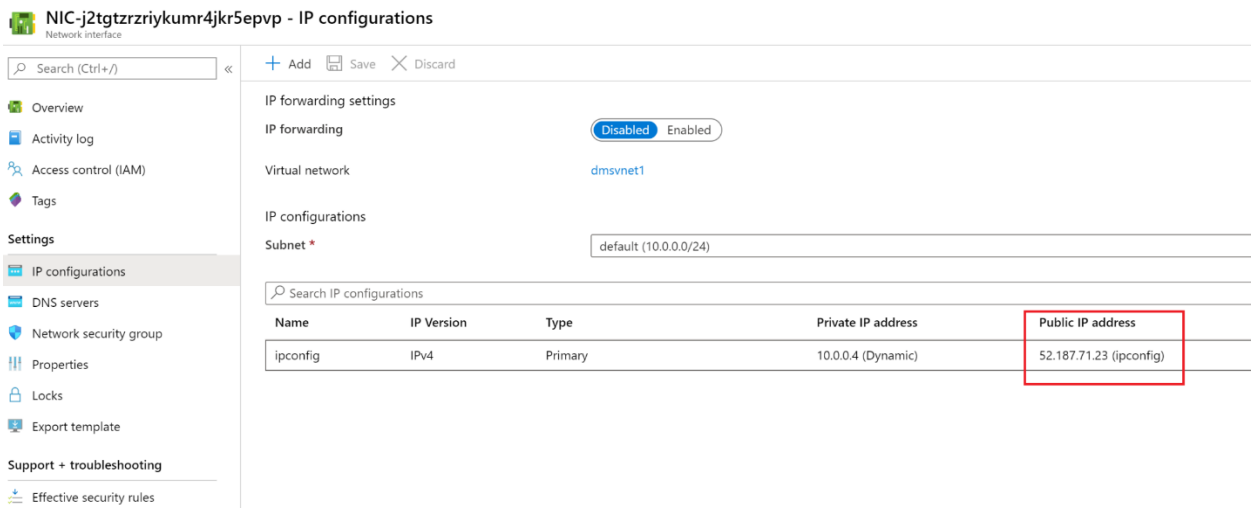
```
REVOKE ALL ON SCHEMA public FROM rdsadmin;
```

DMS needs that we remove indexes and triggers as DMS does not support them due to parallelism. Since my schema is simple, I don't need to do that however when you migrate make sure you remove indexes and triggers and enable them after the migration.

Now, I will create Database migration service in Azure and create a migration project. To create, search for Database Migration Service in all resources and click create. Next, provide inputs like resource group name and migration service name. Make sure you select premium tier as online migration is not supported in basic tier.

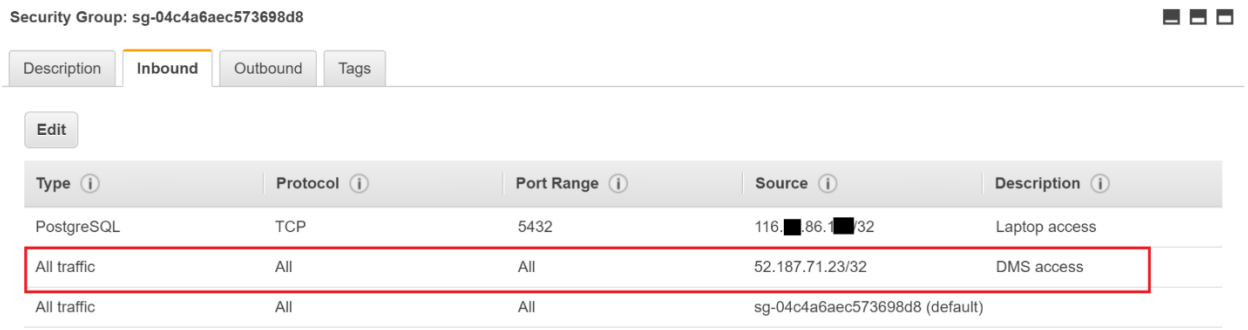
Note: Premium is free for the first 6 months from the service creation date :).

Choose a virtual network or create a new one from the networking screen and click **review + create**.



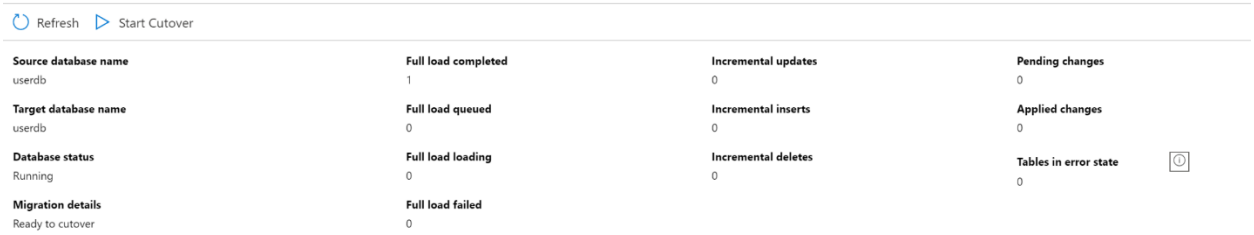
Last thing before we start migration, we need to add a public IP to DMS VNet so that the same can be added to security group in AWS RDS instance's VPC. To do this, I click on the NIC under your DMS resource group, and select IP Configurations. Then select ipconfig and change Public IP address to enabled and under IP address, select or create a new static public IP. Once done you will see the public IP as shown in the screenshot besides.

Next I will take this IP and add it to the security group of the AWS RDS VPC.



Now we are all ready to start the Migration. To start the Migration project, I will go to the DMS and click on **New Migration Project**. Provide a name for the project and select AWS RDS for Postgresql in the **Source server type** and Azure Database for PostgreSQL in the **Target server type** drop down lists. Choose Online data migration under the **Choose type of activity**. Then lets click on **Create and run activity**. Next, Under the **Add source details** blade, I will provide the server name and master login and password for the AWS RDS instance. Under the **Target source details** blade, I will provide the server name and login password for Azure Database for PostgreSQL. Make sure you add the DMS public IP in the firewall for Azure Database for PostgreSQL else it will fail to connect.

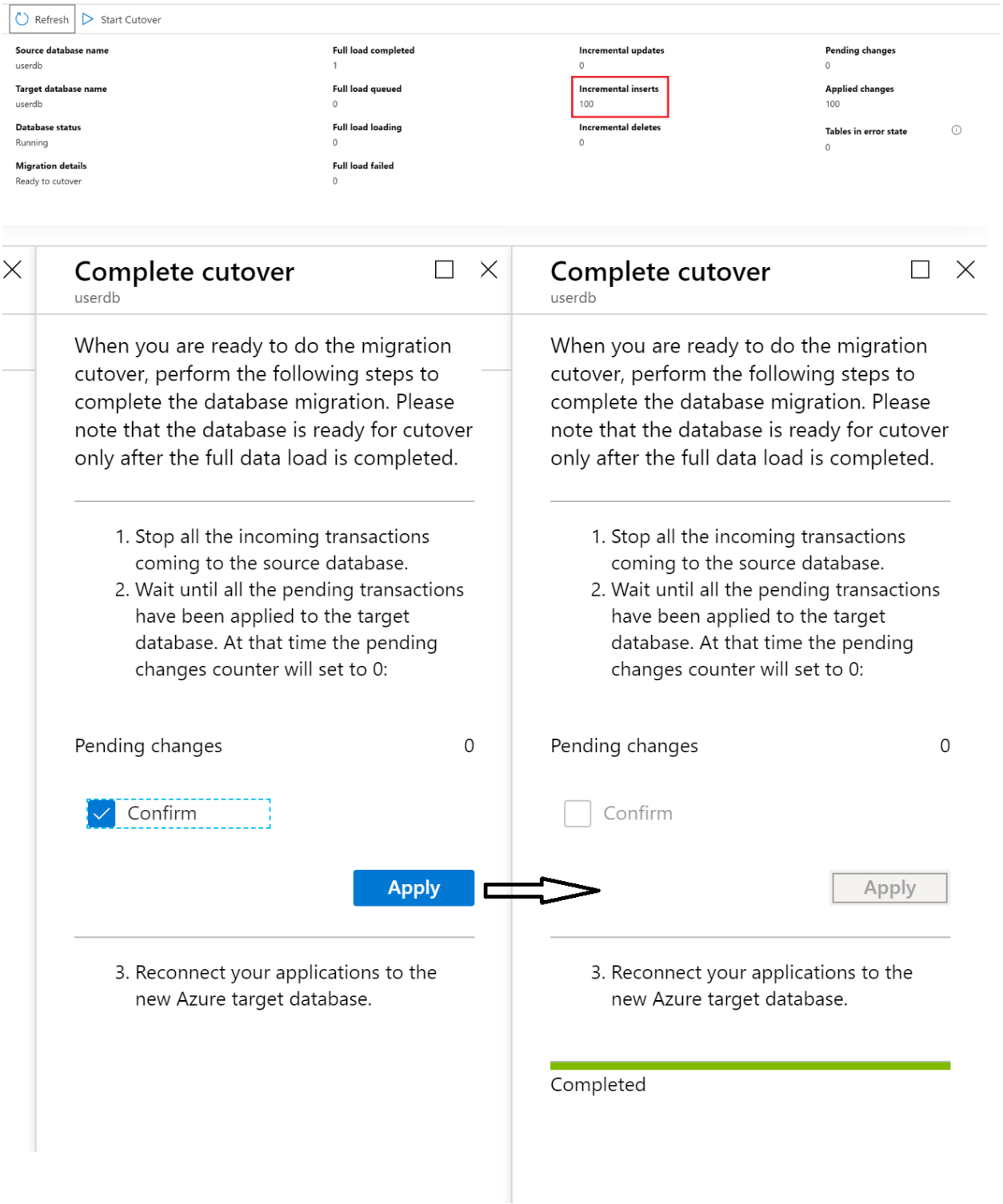
Under **Map to target databases,** I will select userdb, and click save. Under **Migration settings**, you can also choose advanced settings that affect the number of tables for parallel load and LOB data handling. I will use the default values and click save. Next I provide a name for this activity in the **Migration summary** blade and click **Run migration**. You can monitor the Migration activity by clicking on the activity name in the next screen. You can see that for my project, Summary shows that the initial full load is completed and no incremental updates/insert have been sync'ed as yet.



Now I will trigger some transactions in AWS RDS using a simple script as shown below. It adds 100 rows one by one after every 1 second and will terminate after that.

```
do
$DO$
BEGIN
 FOR i in 1..100 LOOP
  INSERT INTO usertbl values(i,'user'||i);
  perform pg_sleep(1);
  commit;
 END LOOP;
END
$DO$;
```

Once this is terminated you will see in the migration activity that DMS successfully migrated 100 rows to Azure DB and we have now established a full sync post which we can now initiate cutover.



Make sure your applications are disconnected from source server while you do this and then redirect the applications to the new migrated instance on Azure. For this I will click on **Start Cutover** and Voila, my Data is migrated from AWS RDS to Azure database.

Hope this helps you in your journey to Azure.

Do let me know of any suggestions or feedback you might have on this post.