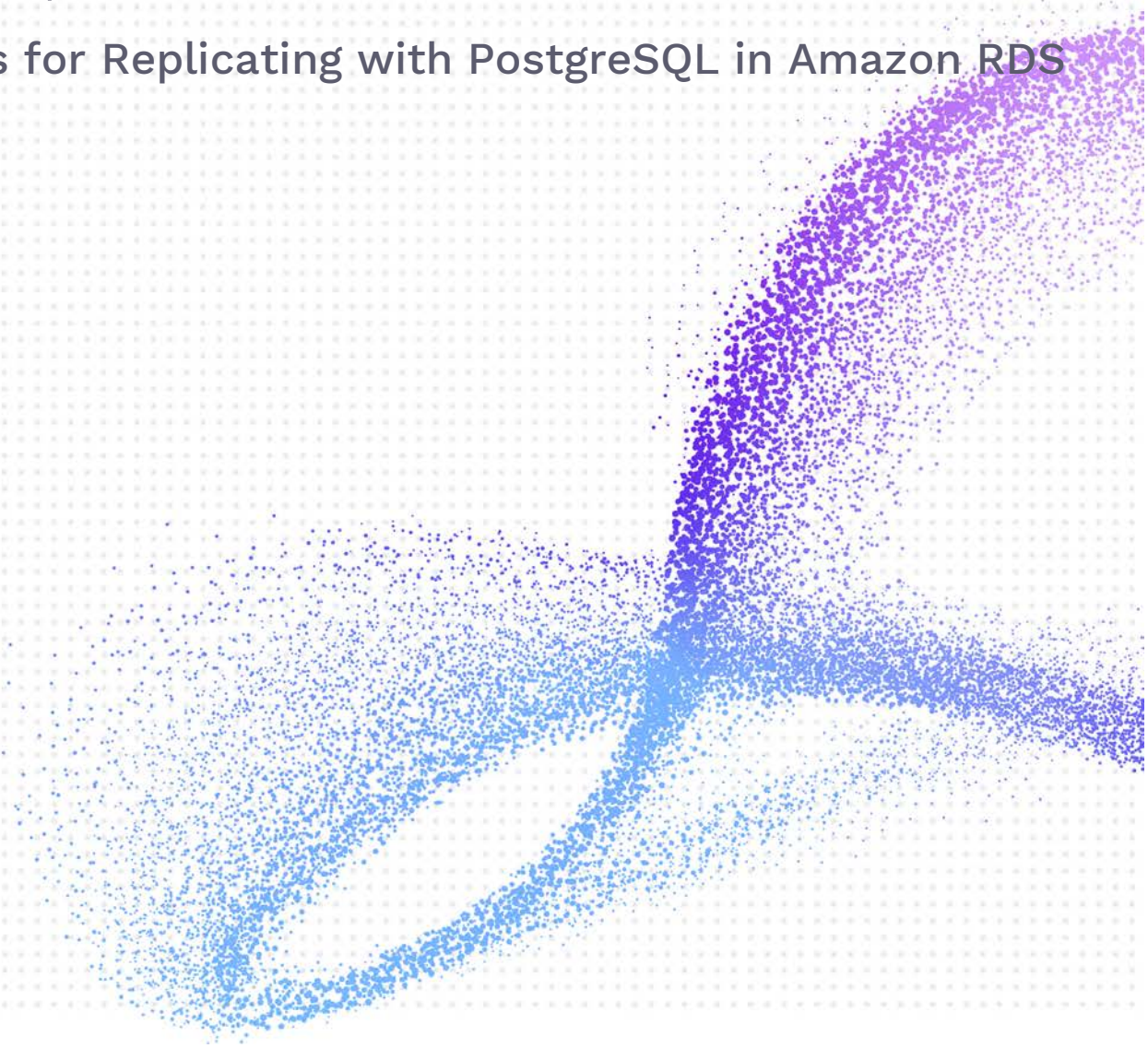
Syniti

Syniti Replicate

Setup Notes for Replicating with PostgreSQL in Amazon RDS

Version 10.3



Syniti

Syniti Replicate

Table of Contents

Syniti Replicate

These notes provide essential information for setting up replications using **Amazon RDS for PostgreSQL**. The setup process for a refresh replication can usually be completed using the Syniti Replicate wizards without additional documentation because it does not involve access to the PostgreSQL log. For complete details on the setup process, check the *Syniti Replicate User Guide* available from the Management Center **Help** menu or the *Syniti Replicate Setup Guide*, available for download in the Help Center.

This guide describes the setup process using the Log Server Agent option for one-way mirroring and synchronization when replicating data from a PostgreSQL database. For mirroring and synchronization replications using PostgreSQL as a source, Syniti Replicate offers:

- **Log Server Agent**: Uses a Windows service and a Log Server component to query the PostgreSQL log for increased performance when dealing with large amounts of data.

Connection Type

PostgreSQL .NET Data Provider recommended by PostgreSQL

Assembly: Npgsql (file name: Npgsql.dll)

Sample path: C:\Npgsql-2.2.3-net40\Npgsql.dll

The provider version number stored in the DBReplicator.exe.config file (in the Syniti Replicate hyper folder) should match the provider version that you are using. To change the version of the provider, specify the "newVersion" in the DBReplicator.exe.config file as follows:

```
<dependentAssembly>
  <assemblyIdentity name="Npgsql" publicKeyToken="5d8b90d52f46fda7"
culture="neutral"></assemblyIdentity>
  <bindingRedirect oldVersion="2.2.3.0" newVersion="2.2.3.0"></bindingRedirect>
</dependentAssembly>
```

PostgreSQL System Settings

Syniti Replicate supports PostgreSQL versions 9.5 and above as a source database for mirroring. It takes advantage of the PostgreSQL replication slot feature by requiring that every connection to PostgreSQL from Syniti Replicate use a dedicated replication slot. The PostgreSQL documentation provides an in-depth explanation of replication slots, but briefly, a single slot represents an ordered stream of changes that occur on a specific database on the origin server. Note that if you enable a replication slot from Syniti Replicate without using it, the slot will consume resources and prevent the database cleanup functionality to purge old records from the transaction log, because they do not appear as consumed by an existing slot. Therefore, it is important to remove replication slots from the database by removing the Syniti Replicate connection if it is not used in replication.

Amazon RDS for PostgreSQL Database Setup

To set up the Amazon RDS for PostgreSQL database as a source for transactional replications (CDC) in Syniti Replicate, use the following link for detailed instructions.

https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.PostgreSQL.html#CHAP_Source.PostgreSQL.RDSPostgreSQL

Specifically, make sure you following the steps in these sections:

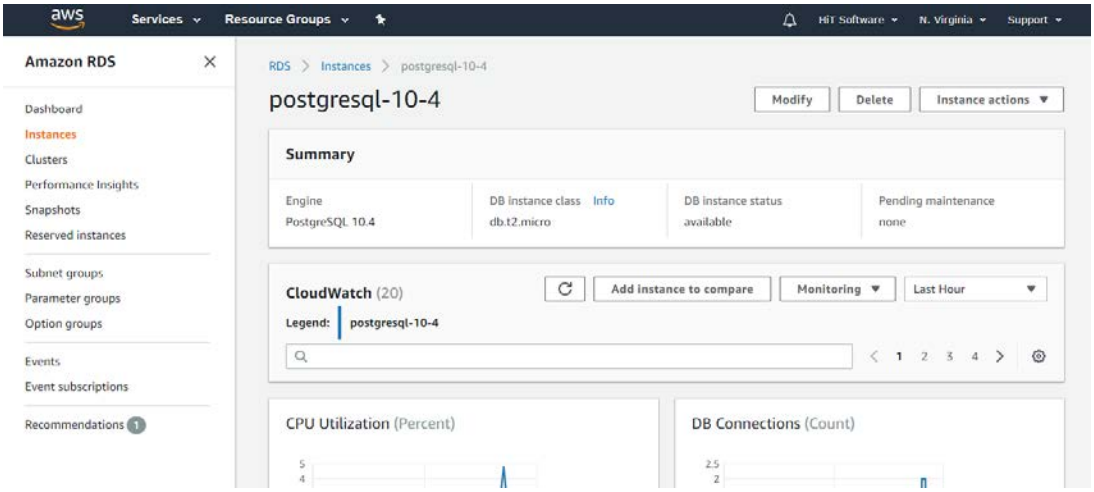
- Prerequisites for Using a PostgreSQL Database as a Source for AWS DMS
- Setting Up an Amazon RDS PostgreSQL DB Instance as a Source (Using CDC with an RDS for PostgreSQL DB Instance section)

Once you have reviewed and completed the above instructions, follow the steps below.

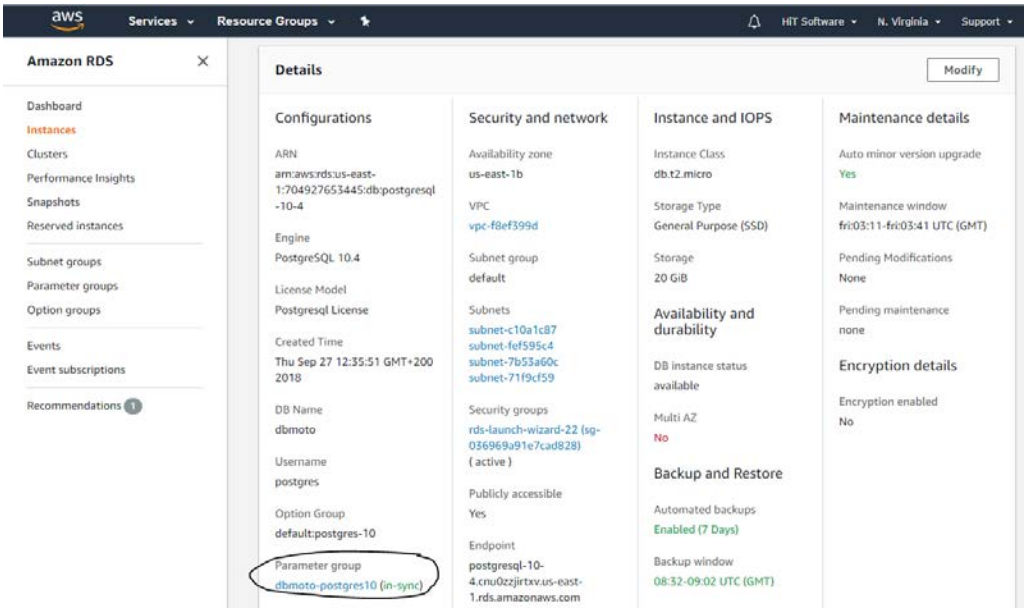
Setting up PostgreSQL RDS for Logical Replication

Here is a step-by-step guide to setting up using the Amazon RDS Console:

1. Log in to aws.amazon.com and go to the RDS database instance that you have created.



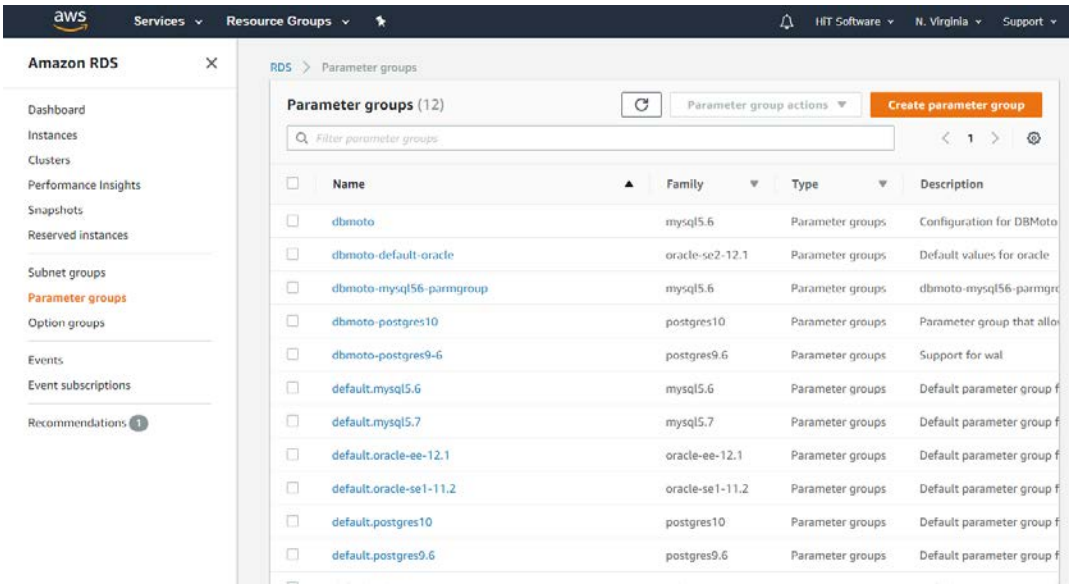
2. Scroll down to the *Details/Configuration* section of your instance and take note of the *Parameter Group* used for your RDBMS:



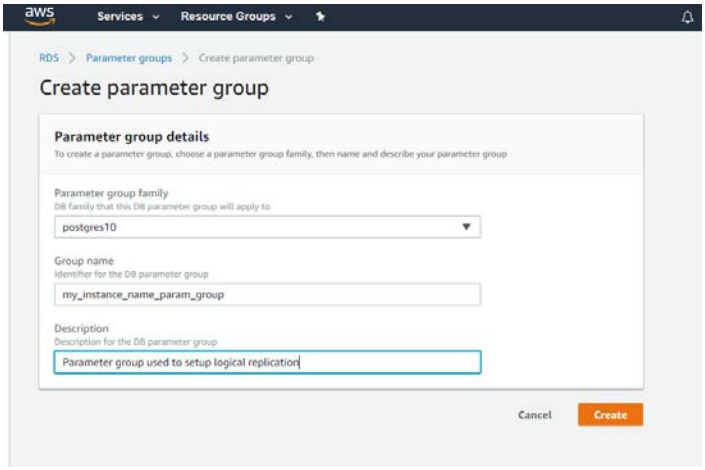
Normally, when an instance is initially created, RDS will configure a set of parameters by assigning a ***Default Parameter Group*** to the instance. A default parameter group is a set of database parameters that is standard and is the same for all instances of that DB type. To customize your parameter settings (as

required to enable transactional replications in Syniti Replicate), create a ***Custom Parameter Group*** and assign it to the instance.

3. Select **Parameter Groups** from the menu bar on the left.
   A list of parameter groups will be shown:

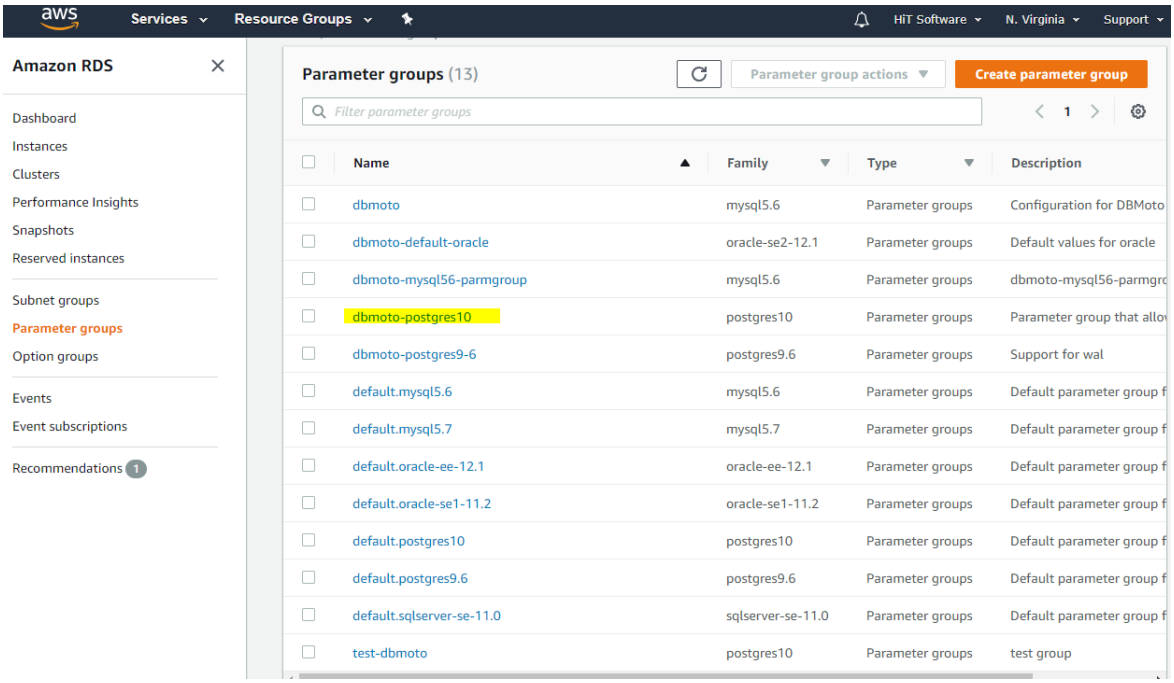


4. Click **Create Parameter Group** to create a custom group. In the process, make sure you select the correct type of parameter group by indicating the type of database (PostgreSQL) and the version that your instance is running:

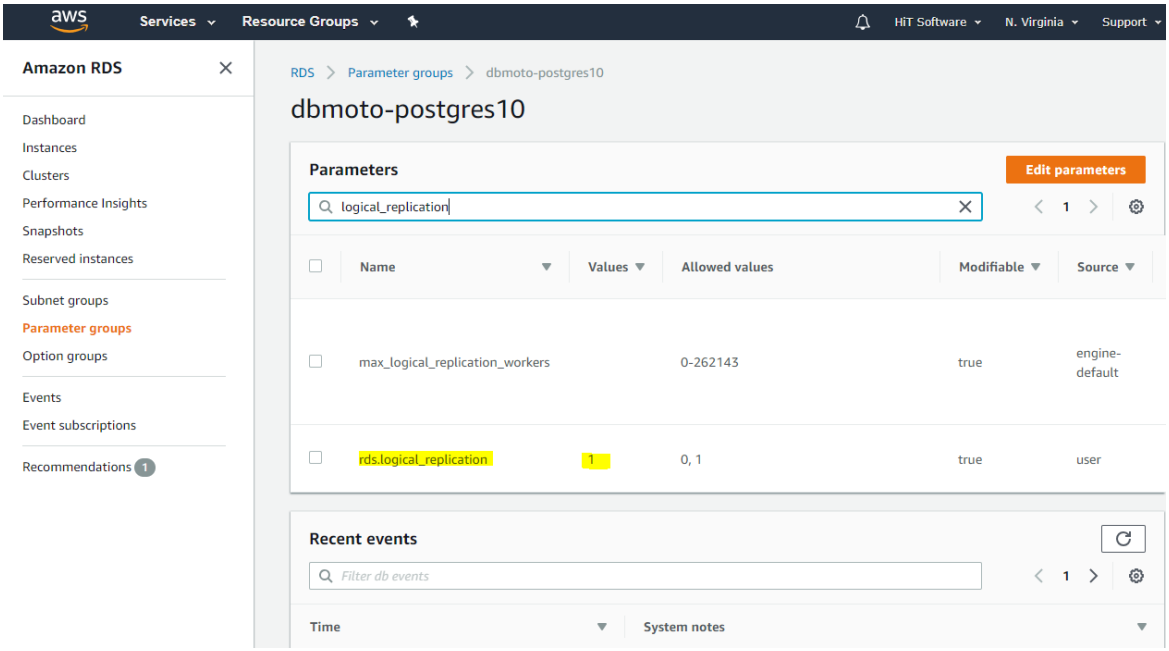


Once the group is created, edit its properties to allow logical replication.

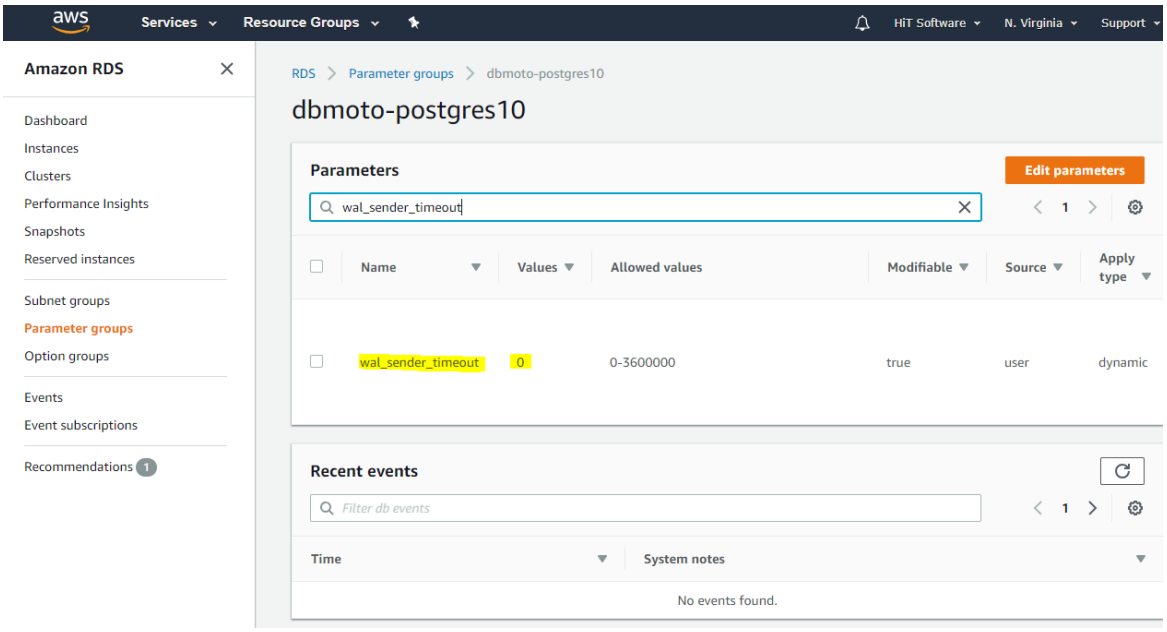
5. Back in the **Parameter Groups** page, select the new group:



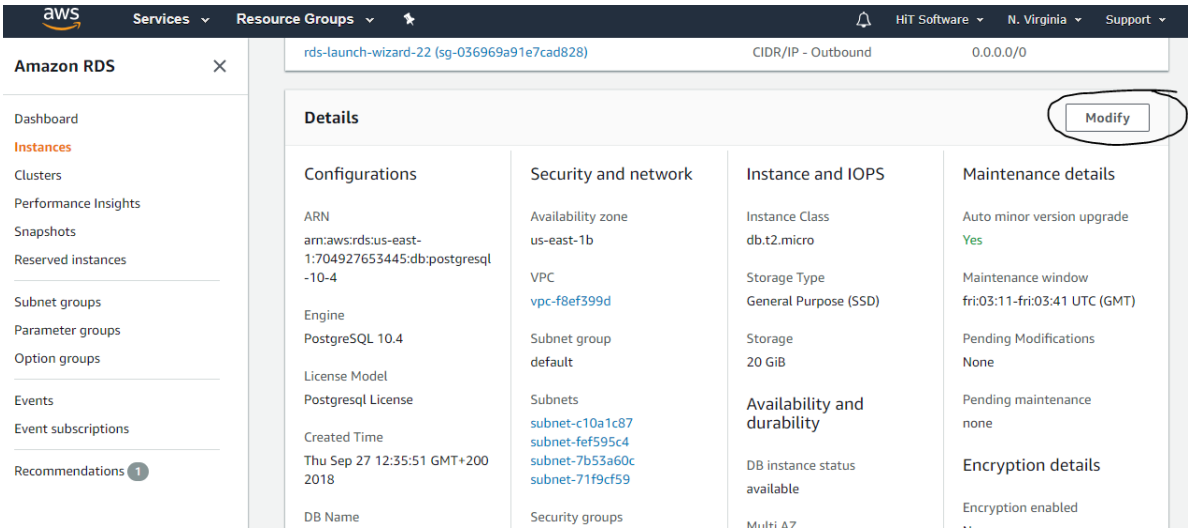
6. Search for the property called *logical_replication*, edit it and set its value to 1:



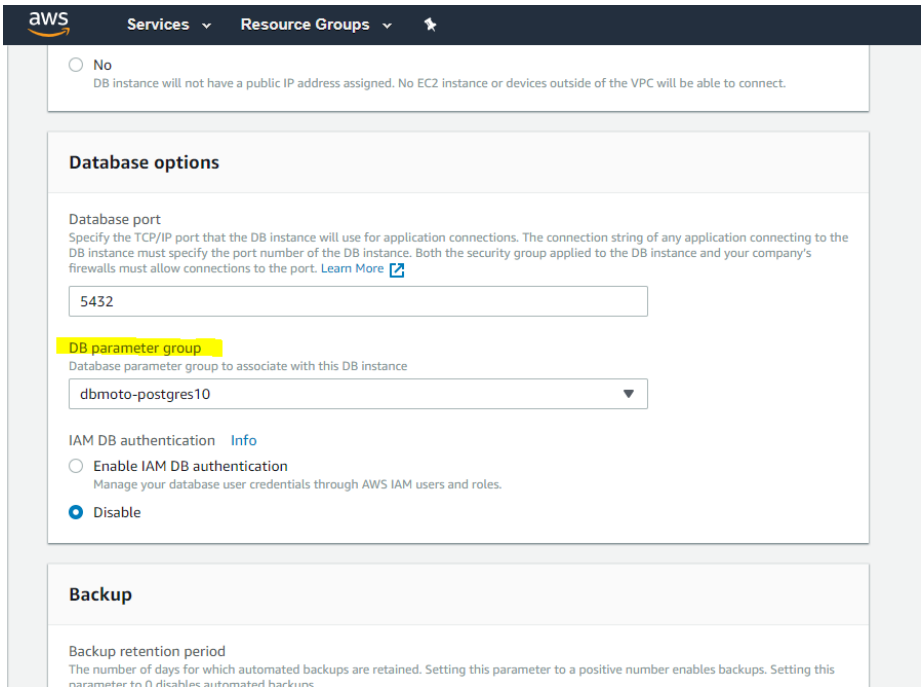
7. Now search for *wal_sender_timeout* and set it to 0:



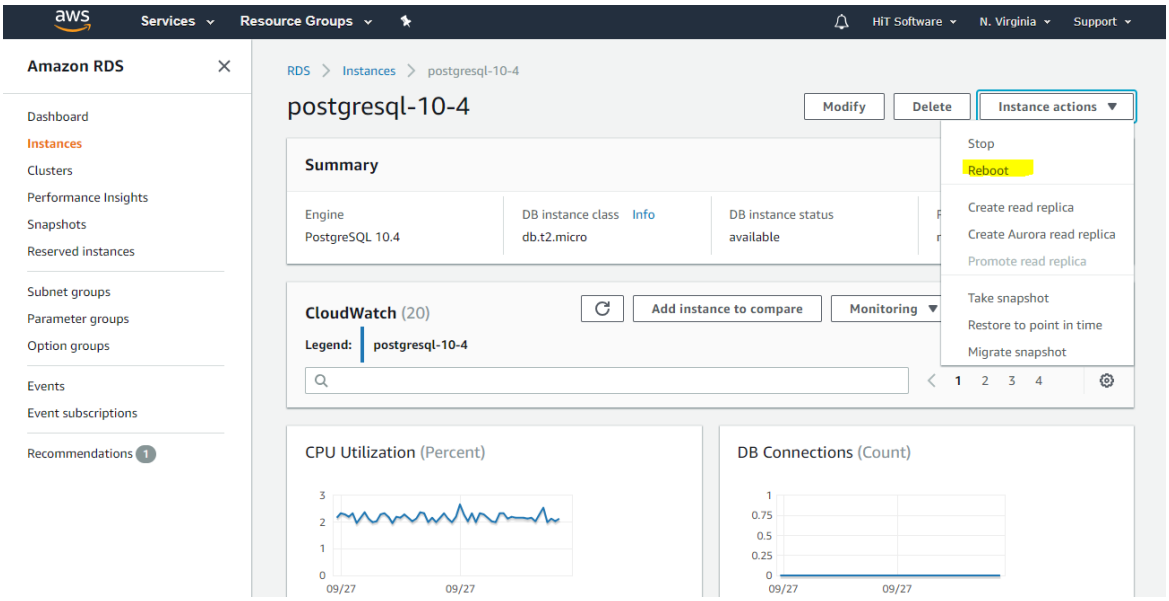
8. Go back to your instance, **Details** section and click **Modify**:



9. In the editing page, scroll down to the **Database Options** section and select the Custom Parameter Group that you just created:



10. Scroll down to the bottom of the page and click **Continue**.
    RDS will then let you save these settings. You can choose the option to save the changes at the next planned maintenance time or immediately. Even if you opt for immediate changes, the database status will be set as unavailable for some time.
11. Once the database is again available, select **Instance Actions** and click **Reboot**. These settings will take effect after the database has been rebooted:



At this point, PostgreSQL on Amazon RDS is ready to be used from Syniti Replicate and you can start setting up the transactional replication from the Management Center.

Recommendations and Restrictions

There are restrictions on CDC replication using PostgreSQL as a source. Although these are listed in the link Amazon RDS for PostgreSQL database or Amazon AWS documentation below, a subset is repeated here as these restrictions apply specifically to use of PostgreSQL with Syniti Replicate.

Limitations on Using a PostgreSQL Database as a Source for AWS DMS

Limitations on Using a PostgreSQL Database as a Source for AWS DMS

The following limitations apply when using PostgreSQL as a source for AWS DMS:

- A captured table must have a primary key. If a table doesn't have a primary key, AWS DMS ignores DELETE and UPDATE record operations for that table.
- Timestamp with a time zone type column is not supported.
- AWS DMS ignores an attempt to update a primary key segment. In these cases, the target identifies the update as one that didn't update any rows. However, because the results of updating a primary key in PostgreSQL are unpredictable, no records are written to the exceptions table.
- Replication of multiple tables with the same name but where each name has a different case (for example table1, TABLE1, and Table1) can cause unpredictable behavior, and therefore AWS DMS doesn't support it.
- AWS DMS doesn't support change processing of TRUNCATE operations.
- The OID LOB data type is not migrated to the target.
- If your source is an on-premises PostgreSQL database or a PostgreSQL database on an Amazon EC2 instance, ensure that the test_decoding output plugin (found in the Postgres contrib package) is installed on your source endpoint. For more information about the test-decoding plugin, see the PostgreSQL documentation.
- AWS DMS doesn't support replication of partitioned tables. When a partitioned table is detected, the following occurs:
  - The endpoint reports a list of parent and child tables.
  - AWS DMS creates the table on the target as a regular table with the same properties as the selected tables.
  - If the parent table in the source database has the same primary key value as its child tables, a "duplicate key" error is generated.

  Note:
  To replicate partitioned tables from a PostgreSQL source to a PostgreSQL target, you first need to manually create the parent and child tables on the target. Then you define a separate task to replicate to those tables. In such a case, you set the task configuration to **Truncate before loading**.

Excerpt from **AWS Database Migration Service User Guide** (Version API Version 2016-01-01)
https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.PostgreSQL.html)

Add Source Connection Wizard

Select Provider Screen

Database

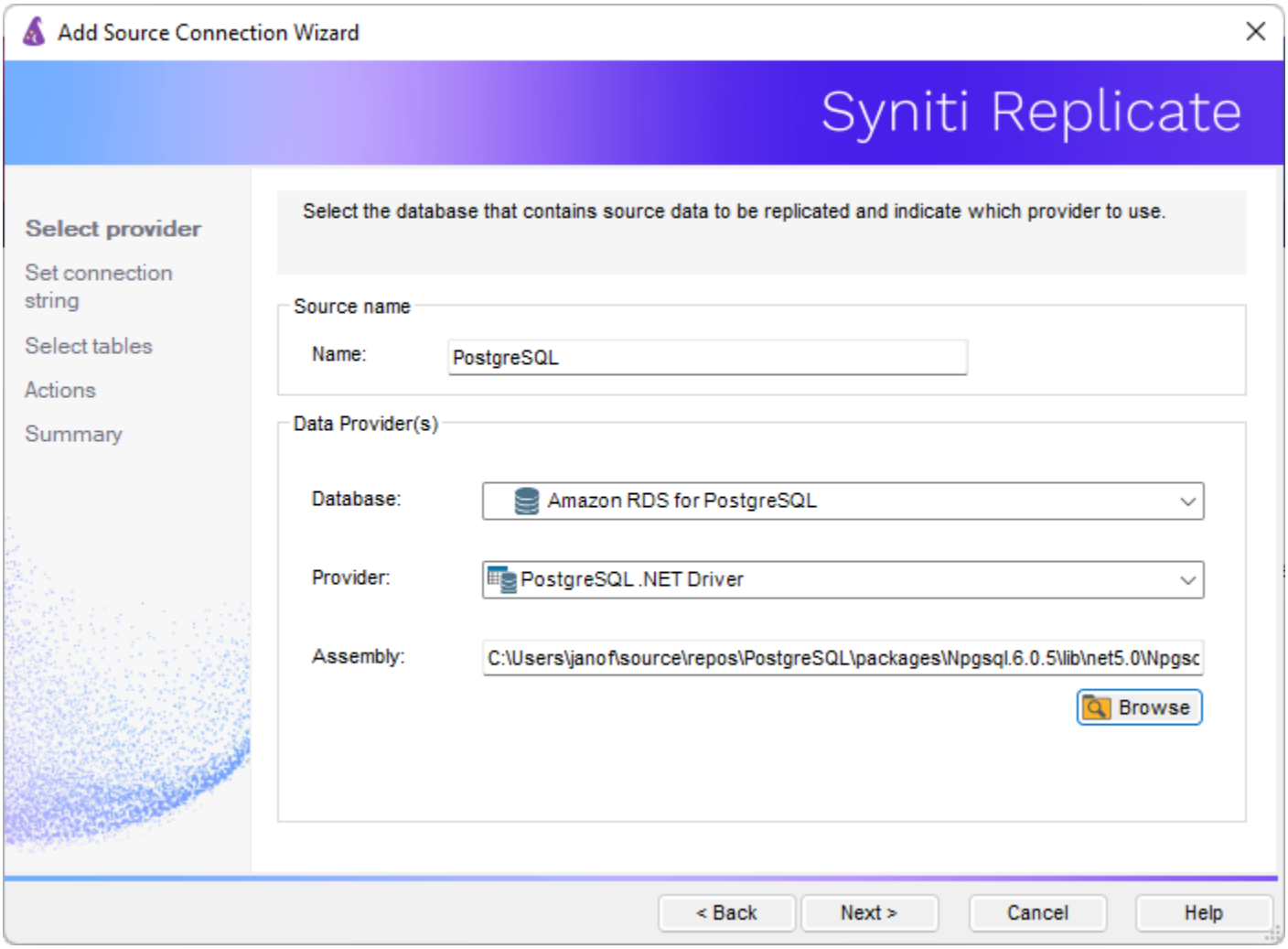
Select PostgreSQL from the drop-down list.

Provider

PostgreSQL .NET Driver

Assembly

Locate the file Npgsql.dll, typically in the installation folder for the PostgreSQL .NET Provider.

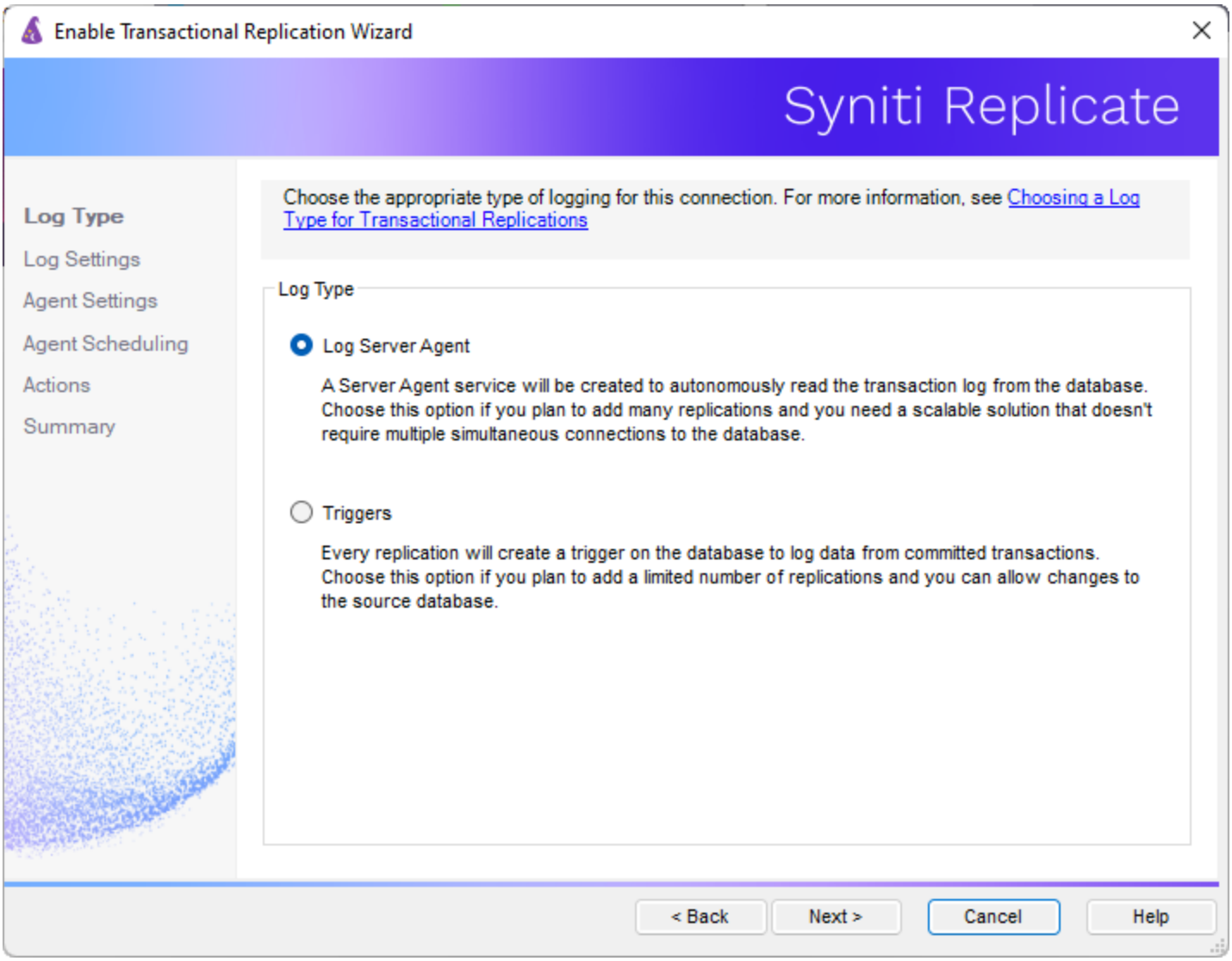


Enable Transactional Replication Wizard

For transactional replications (mirroring and synchronization), use the Enable Transactional Replication wizard after setting up a source connection. The following field(s) require specific information for PostgreSQL.

Log Type Screen

Select the Log Server Agent option.



Log Settings

Replication Slot: Use Existing Slot

Replication slots are channels that you open on the database to allow Syniti Replicate to receive data changes from the log. While the default is to use a new slot specifically for a Syniti Replicate connection, it is possible to reuse an existing slot if it is not consumed by any other connection. A replication slot has to be uniquely assigned to a single Syniti Replicate connection.

Replication Slot: Add New Slot

Syniti Replicate needs to define a “replication_slot” for every connection that it uses to replicate from PostgreSQL. This option allows you to set up a new slot from Syniti Replicate. However, the new slot will be added only if the maximum number of slots has not been exceeded. This value is set in the postgresql.conf file. Be aware that if

slots are created then left unused, they still consume resources, so you should always manage replication slots carefully.

Plugin Type

Make sure that the option **test_decoding** is selected in the drop-down menu.