Requirements for PostgreSQL

Contents

PostgreSQL/Aurora

Capture	Hub	Integrate
✓	✓	✓

This section describes the requirements, access privileges, and other features of HVR when using PostgreSQL/Aurora PostgreSQL for replication. For information about compatibility and supported versions of PostgreSQL with HVR platforms, see Platform Compatibility Matrix.

For the Capabilities supported by HVR on PostgreSQL, and Aurora PostgreSQL, see Capabilities for PostgreSQL and Capabilities for Aurora PostgreSQL respectively.

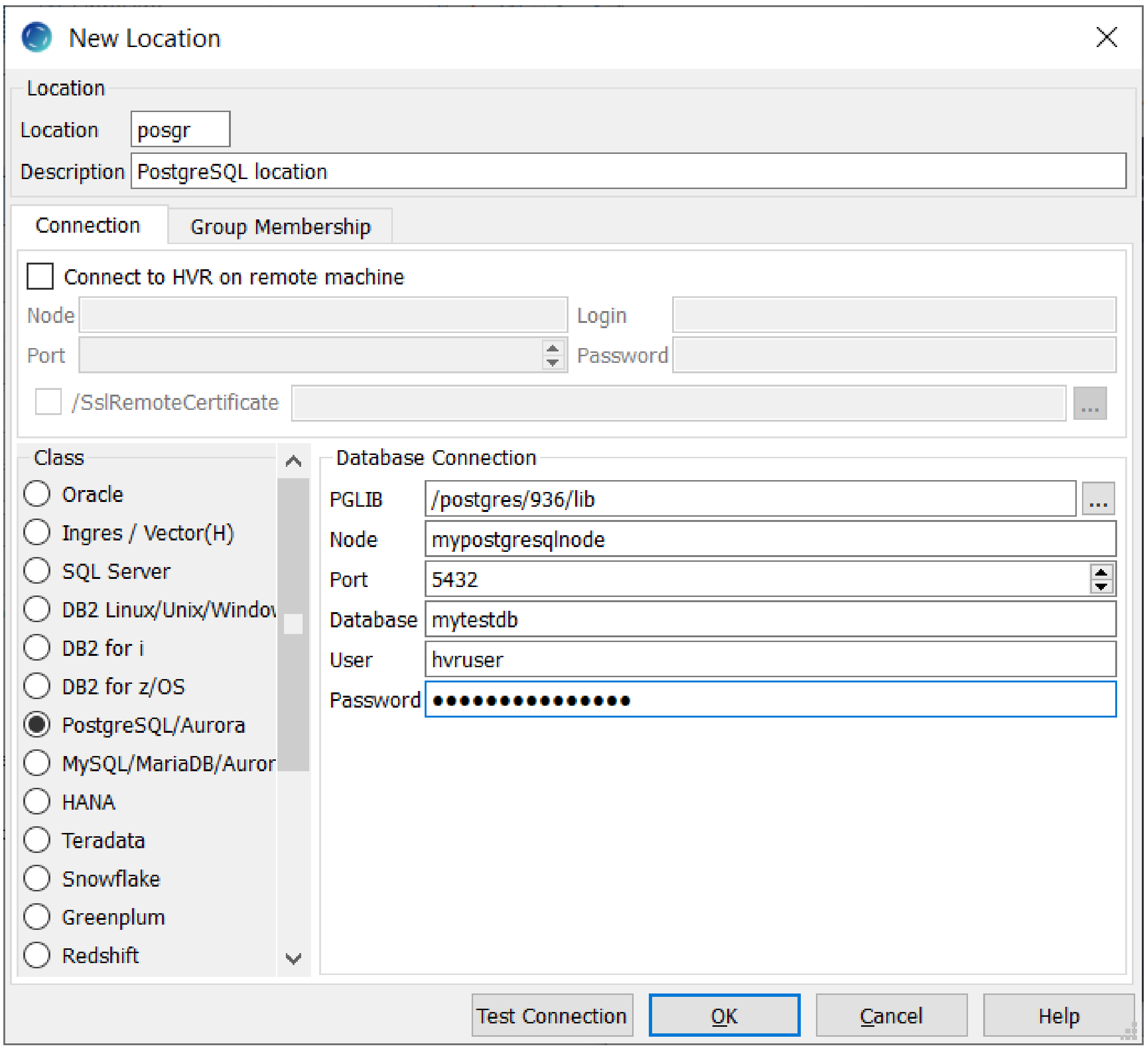
For information about the supported data types and mapping of data types in source DBMS to the corresponding data types in target DBMS or file format, see Data Type Mapping.

Database Connection

HVR requires that the PostgreSQL native LIBPQ (i.e. **libpq.so.5** and its dependencies) is installed on the machine from which HVR will connect to the PostgreSQL database. For information about the client versions required for connecting to the PostgreSQL server, refer to the HVR release notes (**hvr.rel**) available in **hvr_home** directory or the download page.

Location Connection

This section lists and describes the connection details required for creating PostgreSQL/Aurora PostgreSQL location in HVR.


New Location

Location

Location: posgr

Description: PostgreSQL location

Connection | Group Membership

[ ] Connect to HVR on remote machine

Node | Login

Port | Password

[ ] /SslRemoteCertificate

Class

- ( ) Oracle
- ( ) Ingres / Vector(H)
- ( ) SQL Server
- ( ) DB2 Linux/Unix/Windov
- ( ) DB2 for i
- ( ) DB2 for z/OS
- (●) PostgreSQL/Aurora
- ( ) MySQL/MariaDB/Auror
- ( ) HANA
- ( ) Teradata
- ( ) Snowflake
- ( ) Greenplum
- ( ) Redshift

Database Connection

PGLIB	/postgres/936/lib
Node	mypostgresqlnode
Port	5432
Database	mytestdb
User	hvruser
Password	●●●●●●●●●●●●●●●

Test Connection | OK | Cancel | Help

Field	Description
PGLIB	The optional directory path of the PostgreSQL client. **Example:** /postgres/935/lib
Node	The hostname or IP-address of the machine on which the PostgreSQL server is running. **Example:** mypostgresnode
Port	The port on which the PostgreSQL server is expecting connections. **Example:** 5432
Database	The name of the PostgreSQL database. **Example:** mytestdb
User	The username to connect HVR to PostgreSQL **Database**. **Example:** hvruser
Password	The password of the **User** to connect HVR to PostgreSQL **Database**.

Hub

HVR allows you to create a hub database in PostgreSQL/Aurora PostgreSQL. The hub database is a small database that HVR uses to control its replication activities. This database stores HVR catalog tables that hold all specifications of replication such as the names of the replicated databases, the list of replicated tables, and the replication direction.

Grants for Hub

To capture changes from a source database or to integrate changes into a target database, the **User** sh ould have a permission to create and drop HVR catalog tables.

Capture

HVR supports capturing changes from PostgreSQL (includes regular PostgreSQL, Amazon RDS for PostgreSQL ,and Aurora PostgreSQL) location. HVR uses PostgreSQL native LIBPQ for capturing changes from PostgreSQL. For the list of supported PostgreSQL or Aurora PostgreSQL versions from which HVR can capture changes, see Capture changes from location in Capabilities.

Logical replication is only available with Aurora PostgreSQL version 2.2.0 (compatible with PostgreSQL 10.6) and later. For more information, refer to AWS Documentation.

Table Types

HVR supports capture from regular tables in PostgreSQL.

Grants for Log-Based Capture

In order to perform log-based capture the following statement must be executed for the replicated tables:

```
alter table tbl replica identity full
```

HVR Initialize with option **Supplemental Logging** will run these queries. This requires the **User** to be either **superuser** or the owner of the replicated tables. Alternatively, these statements can be performed by a DBA and **HVR Initialize** should be run without option **Suppl emental Logging**.

Log Read Method - DIRECT

HVR captures changes using the **DIRECT** log read method (**Capture** **/LogReadMethod**=**DIRECT**). This capture method is supported to capture changes only from regular PostgreSQL. In this method, HVR reads transaction log records directly from the DBMS log file using the file I/O. This section describes the configuration requirements for capturing changes using **DIRECT** log read method:

1. The HVR agent must be installed on the PostgreSQL source database server.
2. PostgreSQL configuration file **postgresql.conf** should have the following settings:
   - **wal_level** = **logical**

     ```
     show wal_level;
     alter system set wal_level=logical; -- server restart needed
     ```

   - **archive_mode = on**

     ```
     show archive_mode;
     alter system set archive_mode = on; -- server restart needed
     ```

   - **archive_command**
     The value of **archive_command** depends on the location of the archive directory, the operating system and the way archiving is done in a PostgreSQL installation. For example:
     In Unix & Linux

```
 show archive_command;
 alter system set archive_command = 'test ! -f /var/lib/pgsql/9.
5/data/archive/%f && cp %p /var/lib/pgsql/9.5/data/archive/%
f';   -- server restart needed
```

In Windows

```
 show archive_command;
 alter system set archive_command = 'copy "%p" "c:\\Program
Files\\PostgreSQL\\9.5\\data\\archive\\%f"';   -- server
restart needed
```

3. HVR action **Capture /XLogDirectory** should be defined. Parameter **/XLogDirectory** should contain the directory path to the PostgreSQL transaction log file directory. The operating system user as which HVR is running when connecting to PostgreSQL should have read permission to the files in this directory either directly, by running HVR as the DBMS owner (**postgres**) or via a trusted executable **$HVR_HOME/sbin/hvr_postgres**.

4. HVR action **Environment /Name /Value** should be defined. Parameter **/Name** should be set to **HVR_LOG_RELEASE_DIR**. Parameter **/Value** should contain the directory path to the directory where the PostgreSQL transaction log files are archived (for example: /distr/postgres/935/archive). The operating system user as which HVR is running when connecting to PostgreSQL should have read permission to the files in this directory either directly, by running HVR as the DBMS owner (**postgres**), or via a trusted executable **$HVR_HOME/sbin/hvr_postgres**.

   a. To create a **hvr_postgres** executable, execute the following commands while logged in as the DBMS owner (**postgres**):

   ```
   $ cd /usr/hvr/hvr_home
   $ cp bin/hvr sbin/hvr_postgres
   $ chmod 4755 sbin/hvr_postgres
   ```

   b. When user **postgres** does not have permission to write to the HVR installation directories, the following commands can be executed as user **root**:

   ```
   $ cd /usr/hvr/hvr_home
   $ cp /usr/hvr/hvr_home/bin/hvr /usr/hvr/hvr_home/sbin
   /hvr_postgres
   $ chown postgres:postgres /usr/hvr/hvr_home/sbin/hvr_postgres
   $ chmod 4755 /usr/hvr/hvr_home/sbin/hvr_postgres
   ```

   c. Additionally, on Linux the trusted executable needs to be patched using:

   ```
   $ /usr/hvr/hvr_home/lib/patchelf --set-rpath /usr/hvr/hvr_home
   /lib --force-rpath /usr/hvr/hvr_home/sbin/hvr_postgres
   ```

Log Read Method - SQL

HVR captures changes using the **SQL** log read method (**Capture /LogReadMethod**=**SQL**). This capture method supports capturing changes from regular PostgreSQL, Amazon RDS for PostgreSQL, and Aurora PostgreSQL. In this method, HVR reads transaction log records using a special SQL function.

Replication Slots

Capture/LogReadMethod = **SQL** uses PostgreSQL **replication slots**. The names for these slots have to be unique for an entire PostgreSQL cluster.

HVR uses the following naming convention for these replication slots:

hvr_*hub-name_channel-name_location-name*

For example: **hvr_hubdb_mychn_src**

This should allow multi capture in most situations. This includes multiple HVR capture jobs and also coexistence with other replication products.

PostgreSQL will not remove transaction log files for which changes exist that have not been processed by a replication slot. For this reason, replication slots have to be removed when a channel is no longer needed. This can be done manually or by running **hvrinit -d** (**Drop Object** option in GUI).

To retrieve existing replication slots, execute:

```
select slot_name from pg_replication_slots;
```

To manually remove a specific replication slot:

```
select pg_drop_replication_slot('slot_name');
```

For example:

```
    select pg_drop_replication_slot('hvr_hubdb_mychn_src');
```

Capture from Regular PostgreSQL

This section describes the configuration requirements for capturing changes from regular on-premises PostgreSQL using **SQL** log read method:

1. PostgreSQL configuration file **postgresql.conf** should have the following settings:
   - **wal_level** = **logical**

     ```
     show wal_level;
     alter system set wal_level = logical;   -- server restart needed
     ```

   - **max_replication_slots** = *number_of_slots*

     ```
     show max_replication_slots;

     alter system set max_replication_slots = number_of_slots; --
     server restart needed
     ```

     number_of_slots should be set to at least the number of channels multiplied by the number of capture locations in this PostgreSQL installation.

2. The **User** should either have replication permission or be superuser:

   ```
   alter user hvruser replication;
   ```

3. The replication plug-in **test_decoding** should be installed and **User** should have permission to use it. This plug-in is typically installed in **$PG_DATA/lib**. To test whether the plug-in is installed and **User** has the required permissions to execute the following SQL commands:

```
select pg_create_logical_replication_slot('hvr', 'test_decoding');
select pg_drop_replication_slot('hvr');
```

When capturing using **SQL** log read method:

- PostgreSQL versions before 9.4.12 should be avoided due to a PostgreSQL bug (detected in 9.4.6) which affects this log read method.
- Capture rewind (**hvrinit -i**) is not supported
- **SQL** log read method is the only option when capturing from Aurora PostgreSQL machines because the HVR agent cannot be installed on them.

Capture from Amazon RDS for PostgreSQL and Aurora PostgreSQL

HVR supports capturing changes from PostgreSQL at Amazon RDS for PostgreSQL and Aurora PostgreSQL using the log read method **SQL** (**Capture /LogReadMethod**=**SQL**).

To get the required settings and permissions the Parameter Group assigned to the Instance should have **rds.logical_replication=1**. Changing this needs to be followed by a restart of PostgreSQL.

Limitations

Only **insert**, **update** and **delete** changes are captured, **truncate** is not captured.

Integrate and Refresh Target

HVR supports integrating changes into PostgreSQL (includes regular PostgreSQL, Amazon RDS for PostgreSQL, and Aurora PostgreSQL) location. This section describes the configuration requirements for integrating changes (using **Integrate** and **refresh**) into PostgreSQL location. For the list of supported PostgreSQL or Aurora PostgreSQL versions, into which HVR can integrate changes, see Integrate changes into location in Capabilities.

HVR uses the following interfaces to write data to PostgreSQL during **Integrate** and **Refresh**:

- PostgreSQL native LIBPQ, used for continuous **Integrate** and row-wise **Refresh**.
- PostgreSQL "**copy from stdin**" command via native LIBPQ, used for **Integrate** with **/Burst** and Bulk **Refresh**.

Grants for Integrate and Refresh

- The **User** should have permission to read and change replicated tables.

  ```
  grant select, insert, update, delete on tbl to hvruser
  ```

- The **User** should have permission to create and drop HVR state tables.

Compare and Refresh Source

- The **User** should have permission to read replicated tables.

  ```
  grant select on tbl to hvruser
  ```