NAME

pg_receivewal - stream write-ahead logs from a PostgreSQL server

SYNOPSIS

pg_receivewal [*option*...]

DESCRIPTION

pg_receivewal is used to stream the write-ahead log from a running PostgreSQL cluster. The write-ahead log is streamed using the streaming replication protocol, and is written to a local directory of files. This directory can be used as the archive location for doing a restore using point-in-time recovery (see Section 26.3).

pg_receivewal streams the write-ahead log in real time as it's being generated on the server, and does not wait for segments to complete like archive_command and archive_library do. For this reason, it is not necessary to set archive_timeout when using pg_receivewal.

Unlike the WAL receiver of a PostgreSQL standby server, pg_receivewal by default flushes WAL data only when a WAL file is closed. The option **--synchronous** must be specified to flush WAL data in real time. Since pg_receivewal does not apply WAL, you should not allow it to become a synchronous standby when synchronous_commit equals remote_apply. If it does, it will appear to be a standby that never catches up, and will cause transaction commits to block. To avoid this, you should either configure an appropriate value for synchronous_standby_names, or specify *application_name* for pg_receivewal that does not match it, or change the value of *synchronous_commit* to something other than remote_apply.

The write-ahead log is streamed over a regular PostgreSQL connection and uses the replication protocol. The connection must be made with a user having REPLICATION permissions (see Section 22.2) or a superuser, and pg_hba.conf must permit the replication connection. The server must also be configured with max_wal_senders set high enough to leave at least one session available for the stream.

The starting point of the write-ahead log streaming is calculated when pg_receivewal starts:

1. scan the directory where the WAL segment files are written and find the newest completed segment file, using as the starting point the beginning of the next WAL segment file.

2. a starting point cannot be calculated with the previous method, and if a replication slot is used, an extra **READ_REPLICATION_SLOT** command is issued to retrieve the slot's restart_lsn to use as the starting

point. This option is only available when streaming write-ahead logs from PostgreSQL 15 and up.

3.
a starting point cannot be calculated with the previous method, the latest WAL flush location is used as reported by the server from an IDENTIFY_SYSTEM command.

If the connection is lost, or if it cannot be initially established, with a non-fatal error, pg_receivewal will retry the connection indefinitely, and reestablish streaming as soon as possible. To avoid this behavior, use the -n parameter.

In the absence of fatal errors, pg_receivewal will run until terminated by the SIGINT signal (Control+C).

OPTIONS

-D *directory*
--directory=*directory*
Directory to write the output to.

This parameter is required.

-E *lsn*
--endpos=*lsn*
Automatically stop replication and exit with normal exit status 0 when receiving reaches the specified LSN.

If there is a record with LSN exactly equal to *lsn*, the record will be processed.

--if-not-exists
Do not error out when **--create-slot** is specified and a slot with the specified name already exists.

-n
--no-loop
Don’t loop on connection errors. Instead, exit right away with an error.

--no-sync
This option causes **pg_receivewal** to not force WAL data to be flushed to disk. This is faster, but means that a subsequent operating system crash can leave the WAL segments corrupt. Generally, this option is useful for testing but should not be used when doing WAL archiving on a production deployment.

This option is incompatible with --synchronous.

-s *interval*
--status-interval=*interval*

Specifies the number of seconds between status packets sent back to the server. This allows for easier monitoring of the progress from server. A value of zero disables the periodic status updates completely, although an update will still be sent when requested by the server, to avoid timeout disconnect. The default value is 10 seconds.

-S *slotname*
--slot=*slotname*

Require pg_receivewal to use an existing replication slot (see Section 27.2.6). When this option is used, pg_receivewal will report a flush position to the server, indicating when each segment has been synchronized to disk so that the server can remove that segment if it is not otherwise needed.

When the replication client of pg_receivewal is configured on the server as a synchronous standby, then using a replication slot will report the flush position to the server, but only when a WAL file is closed. Therefore, that configuration will cause transactions on the primary to wait for a long time and effectively not work satisfactorily. The option --synchronous (see below) must be specified in addition to make this work correctly.

--synchronous

Flush the WAL data to disk immediately after it has been received. Also send a status packet back to the server immediately after flushing, regardless of --status-interval.

This option should be specified if the replication client of pg_receivewal is configured on the server as a synchronous standby, to ensure that timely feedback is sent to the server.

-v
--verbose

Enables verbose mode.

-Z *level*
-Z *method***[:***detail***]**
--compress=*level*
--compress=*method***[:***detail***]**

Enables compression of write-ahead logs.

The compression method can be set to gzip, lz4 (if PostgreSQL was compiled with **--with-lz4**) or none for no compression. A compression detail string can optionally be specified. If the detail string is an integer, it specifies the compression level. Otherwise, it should be a comma-separated list of items, each of the form keyword or keyword=value. Currently, the only supported keyword is level.

If no compression level is specified, the default compression level will be used. If only a level is specified without mentioning an algorithm, gzip compression will be used if the level is greater than 0, and no compression will be used if the level is 0.

The suffix .gz will automatically be added to all filenames when using gzip, and the suffix .lz4 is added when using lz4.

The following command-line options control the database connection parameters.

-d *connstr*
--dbname=*connstr*
Specifies parameters used to connect to the server, as a connection string; these will override any conflicting command line options.

The option is called --dbname for consistency with other client applications, but because pg_receivewal doesn’t connect to any particular database in the cluster, database name in the connection string will be ignored.

-h *host*
--host=*host*
Specifies the host name of the machine on which the server is running. If the value begins with a slash, it is used as the directory for the Unix domain socket. The default is taken from the **PGHOST** environment variable, if set, else a Unix domain socket connection is attempted.

-p *port*
--port=*port*
Specifies the TCP port or local Unix domain socket file extension on which the server is listening for connections. Defaults to the **PGPORT** environment variable, if set, or a compiled-in default.

-U *username*
--username=*username*
User name to connect as.

-w
--no-password
Never issue a password prompt. If the server requires password authentication and a password is

not available by other means such as a .pgpass file, the connection attempt will fail. This option can be useful in batch jobs and scripts where no user is present to enter a password.

-W
--password
Force pg_receivewal to prompt for a password before connecting to a database.

This option is never essential, since pg_receivewal will automatically prompt for a password if the server demands password authentication. However, pg_receivewal will waste a connection attempt finding out that the server wants a password. In some cases it is worth typing **-W** to avoid the extra connection attempt.

pg_receivewal can perform one of the two following actions in order to control physical replication slots:

--create-slot
Create a new physical replication slot with the name specified in **--slot**, then exit.

--drop-slot
Drop the replication slot with the name specified in **--slot**, then exit.

Other options are also available:

-V
--version
Print the pg_receivewal version and exit.

-?
--help
Show help about pg_receivewal command line arguments, and exit.

EXIT STATUS

pg_receivewal will exit with status 0 when terminated by the SIGINT signal. (That is the normal way to end it. Hence it is not an error.) For fatal errors or other signals, the exit status will be nonzero.

ENVIRONMENT

This utility, like most other PostgreSQL utilities, uses the environment variables supported by libpq (see Section 34.15).

The environment variable **PG_COLOR** specifies whether to use color in diagnostic messages. Possible values are always, auto and never.

NOTES

When using pg_receivewal instead of archive_command or archive_library as the main WAL backup method, it is strongly recommended to use replication slots. Otherwise, the server is free to recycle or remove write-ahead log files before they are backed up, because it does not have any information, either from archive_command or archive_library or the replication slots, about how far the WAL stream has been archived. Note, however, that a replication slot will fill up the server's disk space if the receiver does not keep up with fetching the WAL data.

pg_receivewal will preserve group permissions on the received WAL files if group permissions are enabled on the source cluster.

EXAMPLES

To stream the write-ahead log from the server at mydbserver and store it in the local directory /usr/local/pgsql/archive:

```
$ pg_receivewal -h mydbserver -D /usr/local/pgsql/archive
```

SEE ALSO

pg_basebackup(1)