Contents

— title: User Guide - Debian & Ubuntu / PostgreSQL 9.4 draft: false —

Introduction

This user guide is intended to be followed sequentially from beginning to end — each section depends on the last. For example, the Backup section relies on setup that is performed in the Quick Start section. Once pgBackRest is up and running then skipping around is possible but following the user guide in order is recommended the first time through.

Although the examples are targeted at Debian/Ubuntu and PostgreSQL 9.4, it should be fairly easy to apply this guide to any Unix distribution and PostgreSQL version. The only OS-specific commands are those to create, start, stop, and drop PostgreSQL clusters. The pgBackRest commands will be the same on any Unix system though the locations to install Perl libraries and executables may vary.

Configuration information and documentation for PostgreSQL can be found in the PostgreSQL Manual .

A somewhat novel approach is taken to documentation in this user guide. Each command is run on a virtual machine when the documentation is built from the XML source. This means you can have a high confidence that the commands work correctly in the order presented. Output is captured and displayed below the command when appropriate. If the output is not included it is because it was deemed not relevant or was considered a distraction from the narrative.

All commands are intended to be run as an unprivileged user that has sudo privileges for both the root and postgres users. It's also possible to run the commands directly as their respective users without modification and in that case the sudo commands can be stripped off.

Concepts

The following concepts are defined as they are relevant to pgBackRest , PostgreSQL , and this user guide.

Backup

A backup is a consistent copy of a database cluster that can be restored to recover from a hardware failure, to perform Point-In-Time Recovery, or to bring up a new standby.

Full Backup : pgBackRest copies the entire contents of the database cluster to the backup server. The first backup of the database cluster is always a Full Backup. pgBackRest is always able to restore a full backup directly. The full backup does not depend on any files outside of the full backup for consistency.

Differential Backup : pgBackRest copies only those database cluster files that have changed since the last full backup. pgBackRest restores a differential backup by copying all of the files in the chosen differential backup and the appropriate unchanged files from the previous full backup. The advantage of a differential backup is that it requires less disk space than a full backup, however, the differential backup and the full backup must both be valid to restore the differential backup.

Incremental Backup : pgBackRest copies only those database cluster files that have changed since the last backup (which can be another incremental backup, a differential backup, or a full backup). As an incremental backup only includes those files changed since the prior backup, they are generally much smaller than full or differential backups. As with the differential backup, the incremental backup depends on other backups to be valid to restore the incremental backup. Since the incremental backup includes only those files since the last backup, all prior incremental backups back to the prior differential, the prior differential backup, and the prior full backup must all be valid to perform a restore of the incremental backup. If no differential backup exists then all prior incremental backups back to the prior full backup, which must exist, and the full backup itself must be valid to restore the incremental backup.

Restore

A restore is the act of copying a backup to a system where it will be started as a live database cluster. A restore requires the backup files and one or more WAL segments in order to work correctly.

Write Ahead Log (WAL)

WAL is the mechanism that PostgreSQL uses to ensure that no committed changes are lost. Transactions are written sequentially to the WAL and a transaction is considered to be committed when those writes are flushed to disk. Afterwards, a background process writes the changes into the main database cluster files (also known as the heap). In the event of a crash, the WAL is replayed to make the database consistent.

WAL is conceptually infinite but in practice is broken up into individual 16MB files called segments. WAL segments follow the naming convention 0000000100000A1E000000FE where the first 8 hexadecimal digits represent the timeline and the next 16 digits are the logical sequence number (LSN).

Encryption

Encryption is the process of converting data into a format that is unrecognizable unless the appropriate password (also referred to as passphrase) is provided.

pgBackRest will encrypt the repository based on a user-provided password, thereby preventing unauthorized access to data stored within the repository.

Installation

A new host named db-primary is created to contain the demo cluster and run pgBackRest examples.

pgBackRest supports 32-bit distributions that build Perl with 64-bit integer support.

db-primary Check for 64-bit integers

```
sudo -u postgres perl -V | grep USE_64_BIT_INT
```

```
                        USE_64_BIT_ALL USE_64_BIT_INT USE_ITHREADS
```

If pgBackRest has been installed before it's best to be sure that no prior copies of it are still installed. Depending on how old the version of pgBackRest is it may have been installed in a few different locations. The following commands will remove all prior versions of pgBackRest.

db-primary Remove prior pgBackRest installations

```
sudo rm -f /usr/bin/pgbackrest
```

```
sudo rm -f /usr/bin/pg_backrest
```

```
sudo rm -rf /usr/lib/perl5/BackRest
```

```
sudo rm -rf /usr/share/perl5/BackRest
```

```
sudo rm -rf /usr/lib/perl5/pgBackRest
```

```
sudo rm -rf /usr/share/perl5/pgBackRest
```

pgBackRest is written in Perl which is included with Debian/Ubuntu by default. Some additional modules must also be installed but they are available as standard packages.

db-primary Install required Perl packages

```
sudo apt-get install libdbd-pg-perl libio-socket-ssl-perl libxml-libxml-perl
```

Debian/Ubuntu packages for pgBackRest are available at apt.postgresql.org . If they are not provided for your distribution/version it is easy to download the source and install manually.

db-primary Download version 1.29 of pgBackRest

```
sudo wget -q -O - \
       https://github.com/pgbackrest/pgbackrest/archive/release/1.29.tar.gz | \
       sudo tar zx -C /root
```

db-primary Install pgBackRest

```
sudo cp -r /root/pgbackrest-release-1.29/lib/pgBackRest \
       /usr/share/perl5
```

```
sudo find /usr/share/perl5/pgBackRest -type f -exec chmod 644 {} +
```

```
sudo find /usr/share/perl5/pgBackRest -type d -exec chmod 755 {} +
```

```
sudo cp /root/pgbackrest-release-1.29/bin/pgbackrest /usr/bin/pgbackrest
```

```
sudo chmod 755 /usr/bin/pgbackrest
```

```
sudo mkdir -m 770 /var/log/pgbackrest
```

```
sudo chown postgres:postgres /var/log/pgbackrest
```

```
sudo touch /etc/pgbackrest.conf
```

```
sudo chmod 640 /etc/pgbackrest.conf
```

```
sudo chown postgres:postgres /etc/pgbackrest.conf
```

pgBackRest includes an optional companion C library that enhances performance and enables the ‘checksum-page‘ option and encryption. Pre-built packages are generally a better option than building the C library manually but the steps required are given below for completeness. Depending on the distribution a number of packages may be required which will not be enumerated here.

db-primary Build and Install C Library

```
sudo sh -c 'cd /root/pgbackrest-release-1.29/libc && \
       perl Makefile.PL INSTALLMAN1DIR=none INSTALLMAN3DIR=none'
```

```
sudo make -C /root/pgbackrest-release-1.29/libc test
```

```
sudo make -C /root/pgbackrest-release-1.29/libc install
```

pgBackRest should now be properly installed but it is best to check. If any dependencies were missed then you will get an error when running pgBackRest from the command line.

db-primary Make sure the installation worked

```
sudo -u postgres pgbackrest
```

```
pgBackRest 1.29 - General help

Usage:
    pgbackrest [options] [command]

Commands:
    archive-get     Get a WAL segment from the archive.
    archive-push    Push a WAL segment to the archive.
    backup          Backup a database cluster.
    check           Check the configuration.
    expire          Expire backups that exceed retention.
    help            Get help.
    info            Retrieve information about backups.
    restore         Restore a database cluster.
    stanza-create   Create the required stanza data.
    stanza-delete   Delete a stanza.
    stanza-upgrade  Upgrade a stanza.
    start           Allow pgBackRest processes to run.
    stop            Stop pgBackRest processes from running.
    version         Get version.

Use 'pgbackrest help [command]' for more information.
```

Quick Start

The Quick Start section will cover basic configuration of pgBackRest and PostgreSQL and introduce the backup , restore , and info commands.

Setup Demo Cluster

Creating the demo cluster is optional but is strongly recommended, especially for new users, since the example commands in the user guide reference the demo cluster; the examples assume the demo cluster is running on the default port (i.e. 5432). The cluster will not be started until a later section because there is still some configuration to do.

db-primary Create the demo cluster

```
sudo -u postgres /usr/lib/postgresql/9.4/bin/initdb \
       -D /var/lib/postgresql/9.4/demo -k -A peer
```

```
sudo pg_createcluster 9.4 demo
```

```
Configuring already existing cluster (configuration: /etc/postgresql/9.4/demo, data:
   /var/lib/postgresql/9.4/demo, owner: 5000:5000)
Ver Cluster Port Status Owner    Data directory              Log file
9.4 demo    5432 down   postgres /var/lib/postgresql/9.4/demo
   /var/log/postgresql/postgresql-9.4-demo.log
```

By default PostgreSQL will only accept local connections. The examples in this guide will require connections from other servers so listen_addresses is configured to listen on all interfaces. This may not be appropriate for secure installations.

db-primary : /etc/postgresql/9.4/demo/postgresql.conf Set listen_addresses

listen_addresses = '*'

For demonstration purposes the log_line_prefix setting will be minimally configured. This keeps the log output as brief as possible to better illustrate important information.

db-primary : /etc/postgresql/9.4/demo/postgresql.conf Set log_line_prefix

listen_addresses = '*'
log_line_prefix = ''

Configure Cluster Stanza

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one Postgres database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

The name 'demo' describes the purpose of this cluster accurately so that will also make a good stanza name.

pgBackRest needs to know where the base data directory for the PostgreSQL cluster is located. The path can be requested from PostgreSQL directly but in a recovery scenario the PostgreSQL process will not be available. During backups the value supplied to pgBackRest will be compared against the path that PostgreSQL is running on and they must be equal or the backup will return an error. Make sure that db-path is exactly equal to data_directory in postgresql.conf .

By default Debian/Ubuntu stores clusters in /var/lib/postgresql/[version]/[cluster] so it is easy to determine the correct path for the data directory.

When creating the /etc/pgbackrest.conf file, the database owner (usually postgres) must be granted read privileges.

db-primary : /etc/pgbackrest.conf Configure the PostgreSQL cluster data directory

[demo]
db-path=/var/lib/postgresql/9.4/demo

pgBackRest configuration files follow the Windows INI convention. Sections are denoted by text in brackets and key/value pairs are contained in each section. Lines beginning with # are ignored and can be used as comments.

Create the Repository

The repository is where pgBackRest stores backup and archives WAL segments.

If you are new to backup then it will be difficult to estimate in advance how much space you'll need. The best thing to do is take some backups then record the size of different types of backups (full/incr/diff) and measure the amount of WAL generated per day. This will give you a general idea of how much space you'll need, though of course requirements will likely change over time as your database evolves.

For this demonstration the repository will be stored on the same host as the PostgreSQL server. This is the simplest configuration and is useful in cases where traditional backup software is employed to backup the database host.

db-primary Create the pgBackRest repository

```
sudo mkdir /var/lib/pgbackrest
```

```
sudo chmod 750 /var/lib/pgbackrest
```

```
sudo chown postgres:postgres /var/lib/pgbackrest
```

The repository path must be configured so pgBackRest knows where to find it.

db-primary : /etc/pgbackrest.conf Configure the pgBackRest repository path

[demo]
db-path=/var/lib/postgresql/9.4/demo

[global]
repo-path=/var/lib/pgbackrest

Configure Archiving

Backing up a running PostgreSQL cluster requires WAL archiving to be enabled. Note that *at least* one WAL segment will be created during the backup process even if no explicit writes are made to the cluster.

db-primary : /etc/postgresql/9.4/demo/postgresql.conf Configure archive settings

archive_command = 'pgbackrest –stanza=demo archive-push %p'
archive_mode = on
listen_addresses = '*'
log_line_prefix = ''
max_wal_senders = 3
wal_level = hot_standby

The wal_level setting must be set to archive at a minimum but hot_standby and logical also work fine for backups. Setting wal_level to hot_standy and increasing max_wal_senders is a good idea even if you do not currently run a hot standby as this will allow them to be added later without restarting the primary cluster.

The PostgreSQL cluster must be restarted after making these changes and before performing a backup.

db-primary Restart the demo cluster

```
sudo pg_ctlcluster 9.4 demo restart
```

When archiving a WAL segment is expected to take more than 60 seconds (the default) to reach the pgBackRest repository, then the pgBackRest archive-timeout option should be increased. Note that this option is not the same as the PostgreSQL archive_timeout option which is used to force a WAL segment switch; useful for databases where there are long periods of inactivity. For more information on the PostgreSQL archive_timeout option, see PostgreSQL Write Ahead Log .

Configure Retention

pgBackRest expires backups based on retention options.

db-primary : /etc/pgbackrest.conf Configure retention to 2 full backups

[demo]
db-path=/var/lib/postgresql/9.4/demo

[global]
repo-path=/var/lib/pgbackrest
retention-full=2

More information about retention can be found in the Retention section.

Configure Repository Encryption

The repository will be configured with a cipher type and key to demonstrate encryption. The companion C library is required for encryption, see Installation .

It is important to use a long, random passphrase for the cipher key. A good way to generate one is to run: openssl rand -base64 48 .

db-primary : /etc/pgbackrest.conf Configure pgBackRest repository encryption

[demo]
db-path=/var/lib/postgresql/9.4/demo

[global]
repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo-cipher-type=aes-256-cbc

repo-path=/var/lib/pgbackrest
retention-full=2

Once the repository has been configured and the stanza created and checked, the repository encryption settings cannot be changed.

Create the Stanza

The stanza-create command must be run on the host where the repository is located to initialize the stanza. It is recommended that the check command be run after stanza-create to ensure archiving and backups are properly configured.

db-primary Create the stanza and check the configuration

```
sudo -u postgres pgbackrest --stanza=demo --log-level-console=info stanza-create
```

```
P00   INFO: stanza-create command begin 1.29: --db1-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --no-log-timestamp --repo-cipher-pass=
   --repo-cipher-type=aes-256-cbc --repo-path=/var/lib/pgbackrest --stanza=demo
```

```
P00   INFO: stanza-create command end: completed successfully
```

Check the Configuration

The check command validates that pgBackRest and the archive_command setting are configured correctly for archiving and backups. It detects misconfigurations, particularly in archiving, that result in incomplete backups because required WAL segments did not reach the archive. The command can be run on the database or the backup host. The command may also be run on the standby host, however, since pg_switch_xlog() / pg_switch_wal() cannot be performed on the standby, the command will only test the repository configuration.

Note that pg_create_restore_point('pgBackRest Archive Check') and pg_switch_xlog() / pg_switch_wal() are called to force PostgreSQL to archive a WAL segment. Restore points are only supported in PostgreSQL >= 9.1 so for older versions the check command may fail if there has been no write activity since the last log rotation, therefore it is recommended that activity be generated by the user if there have been no writes since the last WAL switch before running the check command.

db-primary Check the configuration

```
sudo -u postgres pgbackrest --stanza=demo --log-level-console=info check
```

```
P00   INFO: check command begin 1.29: --db1-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --no-log-timestamp --repo-cipher-pass=
   --repo-cipher-type=aes-256-cbc --repo-path=/var/lib/pgbackrest --stanza=demo
```

```
P00   INFO: WAL segment 000000010000000000000001 successfully stored in the archive at
   '/var/lib/pgbackrest/archive/demo/9.4-1/0000000100000000/000000010000000000000001-a2cb0209004f0a78
```

```
P00   INFO: check command end: completed successfully
```

Perform a Backup

To perform a backup of the PostgreSQL cluster run pgBackRest with the backup command.

db-primary Backup the demo cluster

```
sudo -u postgres pgbackrest --stanza=demo \
       --log-level-console=info backup
```

```
P00   INFO: backup command begin 1.29: --db1-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --no-log-timestamp --repo-cipher-pass=
   --repo-cipher-type=aes-256-cbc --repo-path=/var/lib/pgbackrest --retention-full=2 --stanza=demo
```

```
P00   WARN: no prior backup exists, incr backup has been changed to full
```

```
P00   INFO: execute exclusive pg_start_backup() with label "pgBackRest backup started at
   2018-07-05 22:15:23": backup begins after the next regular checkpoint completes
P00   INFO: backup start archive = 000000010000000000000002, lsn = 0/2000028
       [filtered 754 lines of output]
P01   INFO: backup file /var/lib/postgresql/9.4/demo/base/1/11895 (0B, 100%)
P01   INFO: backup file /var/lib/postgresql/9.4/demo/base/1/11885 (0B, 100%)
```

```
P00   INFO: full backup size = 19.2MB
```

```
P00   INFO: execute exclusive pg_stop_backup() and wait for all WAL segments to archive
P00   INFO: backup stop archive = 000000010000000000000002, lsn = 0/2000128
       [filtered 4 lines of output]
```

By default pgBackRest will attempt to perform an incremental backup. However, an incremental backup must be based on a full backup and since no full backup existed pgBackRest ran a full backup instead.

The type option can be used to specify a full or differential backup.

db-primary Differential backup of the demo cluster

```
sudo -u postgres pgbackrest --stanza=demo --type=diff \
       --log-level-console=info backup
```

```
       [filtered 4 lines of output]
P01   INFO: backup file /var/lib/postgresql/9.4/demo/global/pg_control (8KB, 97%) checksum
   a2b990954b573ece4f51abf6fc7078d7c115cd47
P01   INFO: backup file /var/lib/postgresql/9.4/demo/backup_label (236B, 100%) checksum
   b619ed4bdc754eef7642fae8ecae479b6581a060
```

```
P00   INFO: diff backup size = 8.2KB
```

```
P00   INFO: execute exclusive pg_stop_backup() and wait for all WAL segments to archive
P00   INFO: backup stop archive = 000000010000000000000003, lsn = 0/30000F0
       [filtered 4 lines of output]
```

This time there was no warning because a full backup already existed. While incremental backups can be based on a full *or* differential backup, differential backups must be based on a full backup. A full backup can be performed by running the backup command with –type=full .

More information about the backup command can be found in the Backup section.

Schedule a Backup

Backups can be scheduled with utilities such as cron.

In the following example, two cron jobs are configured to run; full backups are scheduled for 6:30 AM every Sunday with differential backups scheduled for 6:30 AM Monday through Saturday. If this crontab is installed for the first time mid-week, then pgBackRest will run a full backup the first time the differential job is executed, followed the next day by a differential backup.

```
#m h   dom mon dow   command
30 06  *   *   0     pgbackrest --type=full --stanza=demo backup
30 06  *   *   1-6   pgbackrest --type=diff --stanza=demo backup
```

Once backups are scheduled it's important to configure retention so backups are expired on a regular schedule, see Retention .

Backup Information

Use the info command to get information about backups.

db-primary Get info for the demo cluster

```
sudo -u postgres pgbackrest info
```

```
stanza: demo
    status: ok

    db (current)
        wal archive min/max (9.4-1): 000000010000000000000002 / 000000010000000000000003
```

```
        full backup: 20180705-221523F
```

```
            timestamp start/stop: 2018-07-05 22:15:23 / 2018-07-05 22:15:34
            wal start/stop: 000000010000000000000002 / 000000010000000000000002
            database size: 19.2MB, backup size: 19.2MB
            repository size: 2.2MB, repository backup size: 2.2MB
```

```
        diff backup: 20180705-221523F_20180705-221534D

            timestamp start/stop: 2018-07-05 22:15:34 / 2018-07-05 22:15:37
            wal start/stop: 000000010000000000000003 / 000000010000000000000003
            database size: 19.2MB, backup size: 8.2KB
            repository size: 2.2MB, repository backup size: 400B
            backup reference list: 20180705-221523F
```

Each stanza has a separate section and it is possible to limit output to a single stanza with the –stanza option. The stanza ' status 'gives a brief indication of the stanza's health. If this is' ok ' then pgBackRest is functioning normally. The ' wal archive min/max ' shows the minimum and maximum WAL currently stored in the archive. Note that there may be gaps due to archive retention policies or other reasons.

The backups are displayed oldest to newest. The oldest backup will *always* be a full backup (indicated by an F at the end of the label) but the newest backup can be full, differential (ends with D), or incremental (ends with I).

The ' timestamp start/stop 'defines the time period when the backup ran. The' timestamp stop ' can be used to determine the backup to use when performing Point-In-Time Recovery. More information about Point-In-Time Recovery can be found in the Point-In-Time Recovery section.

The ' wal start/stop ' defines the WAL range that is required to make the database consistent when restoring. The backup command will ensure that this WAL range is in the archive before completing.

The ' database size 'is the full uncompressed size of the database while' backup size 'is the amount of data actually backed up (these will be the same for full backups). The' repository size 'includes all the files from this backup and any referenced backups that are required to restore the database while' repository backup size ' includes only the files in this backup (these will also be the same for full backups). Repository sizes reflect compressed file sizes if compression is enabled in pgBackRest or the filesystem.

The ' backup reference list ' contains the additional backups that are required to restore this backup.

Restore a Backup

Backups can protect you from a number of disaster scenarios, the most common of which are hardware failure and data corruption. The easiest way to simulate data corruption is to remove an important PostgreSQL cluster file.

db-primary Stop the demo cluster and delete the pg_control file

```
sudo pg_ctlcluster 9.4 demo stop
```

```
sudo -u postgres rm /var/lib/postgresql/9.4/demo/global/pg_control
```

Starting the cluster without this important file will result in an error.

db-primary Attempt to start the corrupted demo cluster

```
sudo pg_ctlcluster 9.4 demo start
```

```
The PostgreSQL server failed to start. Please check the log output:
```

```
postgres: could not find the database system
```

```
Expected to find it in the directory "/var/lib/postgresql/9.4/demo",
but could not open file "/var/lib/postgresql/9.4/demo/global/pg_control": No such file or directory
```

To restore a backup of the PostgreSQL cluster run pgBackRest with the restore command. The cluster needs to be stopped (in this case it is already stopped) and all files must be removed from the PostgreSQL data directory.

db-primary Remove old files from demo cluster

```
sudo -u postgres find /var/lib/postgresql/9.4/demo -mindepth 1 -delete
```

db-primary Restore the demo cluster and start PostgreSQL

```
sudo -u postgres pgbackrest --stanza=demo restore
```

```
sudo pg_ctlcluster 9.4 demo start
```

This time the cluster started successfully since the restore replaced the missing pg_control file.

More information about the restore command can be found in the Restore section.

Backup

The Backup section introduces additional backup command features.

Fast Start Option

By default pgBackRest will wait for the next regularly scheduled checkpoint before starting a backup. Depending on the checkpoint_timeout and checkpoint_segments settings in PostgreSQL it may be quite some time before a checkpoint completes and the backup can begin.

db-primary Incremental backup of the demo cluster with the regularly scheduled checkpoint

```
sudo -u postgres pgbackrest --stanza=demo --type=incr \
       --log-level-console=info backup
```

```
P00   INFO: backup command begin 1.29: --db1-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --no-log-timestamp --repo-cipher-pass=
   --repo-cipher-type=aes-256-cbc --repo-path=/var/lib/pgbackrest --retention-full=2 --stanza=demo
   --type=incr
P00   INFO: last backup label = 20180705-221523F_20180705-221534D, version = 1.29
```

```
P00   INFO: execute exclusive pg_start_backup() with label "pgBackRest backup started at
   2018-07-05 22:15:56": backup begins after the next regular checkpoint completes
```

```
P00   INFO: backup start archive = 000000020000000000000005, lsn = 0/5000028
P01   INFO: backup file /var/lib/postgresql/9.4/demo/pg_multixact/offsets/0000 (8KB, 33%) checksum
   0631457264ff7f8d5fb1edc2c0211992a67c73e6
       [filtered 10 lines of output]
```

When –start-fast is passed on the command-line or start-fast=y is set in /etc/pgbackrest.conf an immediate checkpoint is requested and the backup will start more quickly. This is convenient for testing and for ad-hoc backups. For instance, if a backup is being taken at the beginning of a release window it makes no sense to wait for a checkpoint. Since regularly scheduled backups generally only happen once per day it is unlikely that enabling the start-fast in /etc/pgbackrest.conf will negatively affect performance, however for high-volume transactional systems you may want to pass –start-fast on the command-line instead. Alternately, it is possible to override the setting in the configuration file by passing –no-start-fast on the command-line.

db-primary : /etc/pgbackrest.conf Enable the start-fast option

```
[demo]
db-path=/var/lib/postgresql/9.4/demo

[global]
repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo-cipher-type=aes-256-cbc
repo-path=/var/lib/pgbackrest
retention-full=2
start-fast=y
```

db-primary Incremental backup of the demo cluster with an immediate checkpoint

```
sudo -u postgres pgbackrest --stanza=demo --type=incr \
       --log-level-console=info backup
```

```
P00   INFO: backup command begin 1.29: --db1-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --no-log-timestamp --repo-cipher-pass=
   --repo-cipher-type=aes-256-cbc --repo-path=/var/lib/pgbackrest --retention-full=2 --stanza=demo
   --start-fast --type=incr
P00   INFO: last backup label = 20180705-221523F_20180705-221556I, version = 1.29
```

```
P00   INFO: execute exclusive pg_start_backup() with label "pgBackRest backup started at
   2018-07-05 22:16:00": backup begins after the requested immediate checkpoint completes
```

```
P00   INFO: backup start archive = 000000020000000000000006, lsn = 0/6000028
P01   INFO: backup file /var/lib/postgresql/9.4/demo/global/pg_control (8KB, 97%) checksum
   7df60ef9b13f809b2b0548c6508da704b7d4dcb7
       [filtered 8 lines of output]
```

Automatic Stop Option

Sometimes pgBackRest will exit unexpectedly and the backup in progress on the PostgreSQL cluster will not be properly stopped. pgBackRest exits as quickly as possible when an error occurs so that the cause can be reported accurately and is not masked by another problem that might happen during a more extensive cleanup.

Here an error is intentionally caused by removing repository permissions.

db-primary Revoke write privileges in the pgBackRest repository and attempt a backup

```
sudo chmod 550 /var/lib/pgbackrest/backup/demo/
```

```
sudo -u postgres pgbackrest --stanza=demo --type=incr \
       --log-level-console=info backup
```

```
       [filtered 2 lines of output]
P00   INFO: execute exclusive pg_start_backup() with label "pgBackRest backup started at
   2018-07-05 22:16:04": backup begins after the requested immediate checkpoint completes
P00   INFO: backup start archive = 000000020000000000000007, lsn = 0/7000028
```

```
P00  ERROR: [047]: unable to create path
   '/var/lib/pgbackrest/backup/demo/20180705-221523F_20180705-221604I': Permission denied
```

```
P00   INFO: backup command end: aborted with exception [047]
```

Even when the permissions are fixed pgBackRest will still be unable to perform a backup because the PostgreSQL cluster is stuck in backup mode.

db-primary Restore write privileges in the pgBackRest repository and attempt a backup

```
sudo chmod 750 /var/lib/pgbackrest/backup/demo/
```

```
sudo -u postgres pgbackrest --stanza=demo --type=incr \
       --log-level-console=info backup
```

```
P00   INFO: backup command begin 1.29: --db1-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --no-log-timestamp --repo-cipher-pass=
   --repo-cipher-type=aes-256-cbc --repo-path=/var/lib/pgbackrest --retention-full=2 --stanza=demo
   --start-fast --type=incr
P00   INFO: last backup label = 20180705-221523F_20180705-221600I, version = 1.29
P00   INFO: execute exclusive pg_start_backup() with label "pgBackRest backup started at
   2018-07-05 22:16:06": backup begins after the requested immediate checkpoint completes
```

```
P00  ERROR: [057]: ERROR:  a backup is already in progress
```

```
            HINT:  Run pg_stop_backup() and try again.:
            select to_char(current_timestamp, 'YYYY-MM-DD HH24:MI:SS.US TZ'),
               pg_xlogfile_name(lsn), lsn::text from pg_start_backup('pgBackRest backup started at
               2018-07-05 22:16:06', true) as lsn
```

Enabling the stop-auto option allows pgBackRest to stop the current backup if it detects that no other pgBackRest backup process is running.

db-primary : /etc/pgbackrest.conf Enable the stop-auto option

[demo]
db-path=/var/lib/postgresql/9.4/demo

[global]
repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo-cipher-type=aes-256-cbc
repo-path=/var/lib/pgbackrest
retention-full=2
start-fast=y
stop-auto=y

Now pgBackRest will stop the old backup and start a new one so the process completes successfully.

db-primary Perform an incremental backup

```
sudo -u postgres pgbackrest --stanza=demo --type=incr \
       --log-level-console=info backup
```

```
P00   INFO: backup command begin 1.29: --db1-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --no-log-timestamp --repo-cipher-pass=
   --repo-cipher-type=aes-256-cbc --repo-path=/var/lib/pgbackrest --retention-full=2 --stanza=demo
   --start-fast --stop-auto --type=incr
P00   INFO: last backup label = 20180705-221523F_20180705-221600I, version = 1.29
```

```
P00   WARN: the cluster is already in backup mode but no pgBackRest backup process is running.
   pg_stop_backup() will be called so a new backup can be started.
```

```
P00   INFO: execute exclusive pg_stop_backup() and wait for all WAL segments to archive
```

```
P00   INFO: execute exclusive pg_start_backup() with label "pgBackRest backup started at
   2018-07-05 22:16:08": backup begins after the requested immediate checkpoint completes
```

```
P00   INFO: backup start archive = 000000020000000000000008, lsn = 0/8000028
P01   INFO: backup file /var/lib/postgresql/9.4/demo/global/pg_control (8KB, 97%) checksum
   390ede54c87382c2138b9af6bdaa0d1a00aa009a
       [filtered 8 lines of output]
```

Although useful this feature may not be appropriate when another third-party backup solution is being used to take online backups as pgBackRest will not recognize that the other software is running and may terminate a backup started by that software. However, it would be unusual to run more than one third-party backup solution at the same time so this is not likely to be a problem.

Note that pg_dump and pg_base_backup do not take online backups so are not affected. It is safe to run them in conjunction with pgBackRest .

Archive Timeout

During an online backup pgBackRest waits for WAL segments that are required for backup consistency to be archived. This wait time is governed by the pgBackRest archive-timeout option which defaults to 60 seconds. If archiving an individual segment is known to take longer then this option should be increased.

Retention

Generally it is best to retain as many backups as possible to provide a greater window for Point-in-Time Recovery , but practical concerns such as disk space must also be considered. Retention options remove older backups once they are no longer needed.

Full Backup Retention

Set retention-full to the number of full backups required. New backups must be completed before expiration will occur — that means if retention-full=2 then there will be three full backups stored before the oldest one is expired.

db-primary : /etc/pgbackrest.conf Configure retention-full

[demo]
db-path=/var/lib/postgresql/9.4/demo

[global]
repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo-cipher-type=aes-256-cbc
repo-path=/var/lib/pgbackrest
retention-full=2
start-fast=y
stop-auto=y

Backup retention-full=2 but currently there is only one full backup so the next full backup to run will not expire any full backups.

db-primary Perform a full backup

```
sudo -u postgres pgbackrest --stanza=demo --type=full \
       --log-level-console=detail backup
```

```
       [filtered 763 lines of output]
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 1.29: --log-level-console=detail --log-level-stderr=off
   --no-log-timestamp --repo-cipher-pass= --repo-cipher-type=aes-256-cbc
   --repo-path=/var/lib/pgbackrest --retention-archive=2 --retention-full=2 --stanza=demo
```

```
P00 DETAIL: archive retention on backup 20180705-221523F, archiveId = 9.4-1, start =
   000000010000000000000002
```

```
P00 DETAIL: no archive to remove, archiveId = 9.4-1
P00   INFO: expire command end: completed successfully
```

Archive *is* expired because WAL segments were generated before the oldest backup. These are not useful for recovery — only WAL segments generated after a backup can be used to recover that backup.

db-primary Perform a full backup

```
sudo -u postgres pgbackrest --stanza=demo --type=full \
       --log-level-console=info backup
```

```
       [filtered 763 lines of output]
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 1.29: --log-level-console=info --log-level-stderr=off
   --no-log-timestamp --repo-cipher-pass= --repo-cipher-type=aes-256-cbc
   --repo-path=/var/lib/pgbackrest --retention-archive=2 --retention-full=2 --stanza=demo
```

```
P00   INFO: expire full backup set: 20180705-221523F, 20180705-221523F_20180705-221534D,
   20180705-221523F_20180705-221556I, 20180705-221523F_20180705-221600I,
   20180705-221523F_20180705-221608I
```

```
P00   INFO: remove expired backup 20180705-221523F_20180705-221608I
P00   INFO: remove expired backup 20180705-221523F_20180705-221600I
       [filtered 3 lines of output]
```

The 20180705-221523F full backup is expired and archive retention is based on the 20180705-221614F which is now the oldest full backup.

Differential Backup Retention

Set retention-diff to the number of differential backups required. Differentials only rely on the prior full backup so it is possible to create a "rolling" set of differentials for the last day or more. This allows quick restores to recent points-in-time but reduces overall space consumption.

db-primary : /etc/pgbackrest.conf Configure retention-diff

```
[demo]
db-path=/var/lib/postgresql/9.4/demo

[global]
repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo-cipher-type=aes-256-cbc
repo-path=/var/lib/pgbackrest
retention-diff=1
retention-full=2
start-fast=y
stop-auto=y
```

Backup retention-diff=1 so two differentials will need to be performed before one is expired. An incremental backup is added to demonstrate incremental expiration. Incremental backups cannot be expired independently — they are always expired with their related full or differential backup.

db-primary Perform differential and incremental backups

```
sudo -u postgres pgbackrest --stanza=demo --type=diff backup
```

```
sudo -u postgres pgbackrest --stanza=demo --type=incr backup
```

Now performing a differential backup will expire the previous differential and incremental backups leaving only one differential backup.

db-primary Perform a differential backup

```
sudo -u postgres pgbackrest --stanza=demo --type=diff \
       --log-level-console=info backup
```

```
       [filtered 10 lines of output]
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 1.29: --log-level-console=info --log-level-stderr=off
   --no-log-timestamp --repo-cipher-pass= --repo-cipher-type=aes-256-cbc
   --repo-path=/var/lib/pgbackrest --retention-archive=2 --retention-diff=1 --retention-full=2
   --stanza=demo
```

```
P00   INFO: expire diff backup set: 20180705-221626F_20180705-221638D,
   20180705-221626F_20180705-221643I
```

```
P00   INFO: remove expired backup 20180705-221626F_20180705-221643I
P00   INFO: remove expired backup 20180705-221626F_20180705-221638D
```

Archive Retention

Although pgBackRest automatically removes archived WAL segments when expiring backups (the default expires WAL for full backups based on the retention-full option), it may be useful to expire archive more aggressively to save disk space. Note that full backups are treated as differential backups for the purpose of differential archive retention.

Expiring archive will never remove WAL segments that are required to make a backup consistent. However, since Point-in-Time-Recovery (PITR) only works on a continuous WAL stream, care should be taken when aggressively expiring archive outside of the normal backup expiration process.

db-primary : /etc/pgbackrest.conf Configure retention-diff

[demo]
db-path=/var/lib/postgresql/9.4/demo

[global]
repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo-cipher-type=aes-256-cbc
repo-path=/var/lib/pgbackrest
retention-diff=2
retention-full=2
start-fast=y
stop-auto=y

db-primary Perform differential backup

```
sudo -u postgres pgbackrest --stanza=demo --type=diff \
       --log-level-console=info backup
```

```
       [filtered 7 lines of output]
P00   INFO: execute exclusive pg_stop_backup() and wait for all WAL segments to archive
P00   INFO: backup stop archive = 000000020000000000000011, lsn = 0/110000F0
```

```
P00   INFO: new backup label = 20180705-221626F_20180705-221652D
```

```
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 1.29: --log-level-console=info --log-level-stderr=off
   --no-log-timestamp --repo-cipher-pass= --repo-cipher-type=aes-256-cbc
   --repo-path=/var/lib/pgbackrest --retention-archive=2 --retention-diff=2 --retention-full=2
   --stanza=demo
```

db-primary Expire archive

```
sudo -u postgres pgbackrest --stanza=demo --log-level-console=detail \
       --retention-archive-type=diff --retention-archive=1 expire
```

```
P00   INFO: expire command begin 1.29: --log-level-console=detail --log-level-stderr=off
   --no-log-timestamp --repo-cipher-pass= --repo-cipher-type=aes-256-cbc
   --repo-path=/var/lib/pgbackrest --retention-archive=1 --retention-archive-type=diff
   --retention-diff=2 --retention-full=2 --stanza=demo
```

```
P00 DETAIL: archive retention on backup 20180705-221614F, archiveId = 9.4-1, start =
    000000020000000000000009, stop = 000000020000000000000009
P00 DETAIL: archive retention on backup 20180705-221626F, archiveId = 9.4-1, start =
    00000002000000000000000A, stop = 00000002000000000000000A

P00 DETAIL: archive retention on backup 20180705-221626F_20180705-221646D, archiveId = 9.4-1,
    start = 00000002000000000000000D, stop = 00000002000000000000000D

P00 DETAIL: archive retention on backup 20180705-221626F_20180705-221652D, archiveId = 9.4-1,
    start = 000000020000000000000011

P00 DETAIL: remove archive: archiveId = 9.4-1, start = 00000002000000000000000B, stop =
    00000002000000000000000C
P00 DETAIL: remove archive: archiveId = 9.4-1, start = 00000002000000000000000E, stop =
    000000020000000000000010

P00   INFO: expire command end: completed successfully
```

The 20180705-221626F_20180705-221646D differential backup has archived WAL segments that must be retained to make the older backups consistent even though they cannot be played any further forward with PITR. WAL segments generated after 20180705-221626F_20180705-221646D but before 20180705-221626F_20180705-221652D are removed. WAL segments generated after the new backup 20180705-221626F_20180705-221652D remain and can be used for PITR.

Since full backups are considered differential backups for the purpose of differential archive retention, if a full backup is now performed with the same settings, only the archive for that full backup is retained for PITR.

Restore

The Restore section introduces additional restore command features.

Delta Option

Restore a Backup in Quick Start required the database cluster directory to be cleaned before the restore could be performed. The delta option allows pgBackRest to automatically determine which files in the database cluster directory can be preserved and which ones need to be restored from the backup — it also *removes* files not present in the backup manifest so it will dispose of divergent changes. This is accomplished by calculating a SHA-1 cryptographic hash for each file in the database cluster directory. If the SHA-1 hash does not match the hash stored in the backup then that file will be restored. This operation is very efficient when combined with the process-max option. Since the PostgreSQL server is shut down during the restore, a larger number of processes can be used than might be desirable during a backup when the PostgreSQL server is running.

db-primary Stop the demo cluster, perform delta restore

```
sudo pg_ctlcluster 9.4 demo stop
```

```
sudo -u postgres pgbackrest --stanza=demo --delta \
       --log-level-console=detail restore
```

```
       [filtered 692 lines of output]
P01 DETAIL: restore file /var/lib/postgresql/9.4/demo/base/12134/PG_VERSION - exists and matches
    backup (4B, 99%) checksum 8dbabb96e032b8d9f1993c0e4b9141e71ade01a1
P01 DETAIL: restore file /var/lib/postgresql/9.4/demo/base/1/PG_VERSION - exists and matches
    backup (4B, 99%) checksum 8dbabb96e032b8d9f1993c0e4b9141e71ade01a1

P01 DETAIL: restore file /var/lib/postgresql/9.4/demo/PG_VERSION - exists and matches backup (4B,
    100%) checksum 8dbabb96e032b8d9f1993c0e4b9141e71ade01a1

P01 DETAIL: restore file /var/lib/postgresql/9.4/demo/global/12086 - exists and is zero size (0B,
    100%)
P01 DETAIL: restore file /var/lib/postgresql/9.4/demo/global/12038 - exists and is zero size (0B,
    100%)
       [filtered 83 lines of output]
P01 DETAIL: restore file /var/lib/postgresql/9.4/demo/base/1/11885 - exists and is zero size (0B,
    100%)
P00   INFO: write /var/lib/postgresql/9.4/demo/recovery.conf
```

```
P00   INFO: restore global/pg_control (performed last to ensure aborted restores cannot be started)
```

```
P00   INFO: restore command end: completed successfully
```

db-primary Restart PostgreSQL

```
sudo pg_ctlcluster 9.4 demo start
```

Restore Selected Databases

There may be cases where it is desirable to selectively restore specific databases from a cluster backup. This could be done for performance reasons or to move selected databases to a machine that does not have enough space to restore the entire cluster backup.

To demonstrate this feature two databases are created: test1 and test2. A fresh backup is run so pgBackRest is aware of the new databases.

db-primary Create two test databases and perform a backup

```
sudo -u postgres psql -c "create database test1;"
```

```
CREATE DATABASE
```

```
sudo -u postgres psql -c "create database test2;"
```

```
CREATE DATABASE
```

```
sudo -u postgres pgbackrest --stanza=demo --type=incr backup
```

Each test database will be seeded with tables and data to demonstrate that recovery works with selective restore.

db-primary Create a test table in each database

```
sudo -u postgres psql -c "create table test1_table (id int); \
      insert into test1_table (id) values (1);" test1
```

```
INSERT 0 1
```

```
sudo -u postgres psql -c "create table test2_table (id int); \
      insert into test2_table (id) values (2);" test2
```

```
INSERT 0 1
```

One of the main reasons to use selective restore is to save space. The size of the test1 database is shown here so it can be compared with the disk utilization after a selective restore.

db-primary Show space used by test1 database

```
sudo -u postgres du -sh /var/lib/postgresql/9.4/demo/base/16384
```

```
6.4M    /var/lib/postgresql/9.4/demo/base/16384
```

Stop the cluster and restore only the test2 database. Built-in databases (template0 , template1 , and postgres) are always restored.

db-primary Restore from last backup including only the test2 database

```
sudo pg_ctlcluster 9.4 demo stop
```

```
sudo -u postgres pgbackrest --stanza=demo --delta \
      --db-include=test2 restore
```

```
sudo pg_ctlcluster 9.4 demo start
```

Once recovery is complete the test2 database will contain all previously created tables and data.

db-primary Demonstrate that the test2 database was recovered

```
sudo -u postgres psql -c "select * from test2_table;" test2
```

```
 id
----
  2
(1 row)
```

The test1 database, despite successful recovery, is not accessible. This is because the entire database was restored as sparse, zeroed files. PostgreSQL can successfully apply WAL on the zeroed files but the database as a whole will not be valid because key files contain no data. This is purposeful to prevent the database from being accidentally used when it might contain partial data that was applied during WAL replay.

db-primary Attempting to connect to the test1 database will produce an error

```
sudo -u postgres psql -c "select * from test1_table;" test1
```

```
psql: FATAL:  relation mapping file "base/16384/pg_filenode.map" contains invalid data
```

Since the test1 database is restored with sparse, zeroed files it will only require as much space as the amount of WAL that is written during recovery. While the amount of WAL generated during a backup and applied during recovery can be significant it will generally be a small fraction of the total database size, especially for large databases where this feature is most likely to be useful.

It is clear that the test1 database uses far less disk space during the selective restore than it would have if the entire database had been restored.

db-primary Show space used by test1 database after recovery

```
sudo -u postgres du -sh /var/lib/postgresql/9.4/demo/base/16384
```

```
152K    /var/lib/postgresql/9.4/demo/base/16384
```

At this point the only action that can be taken on the invalid test1 database is drop database . pgBackRest does not automatically drop the database since this cannot be done until recovery is complete and the cluster is accessible.

db-primary Drop the test1 database

```
sudo -u postgres psql -c "drop database test1;"
```

```
DROP DATABASE
```

Now that the invalid test1 database has been dropped only the test2 and built-in databases remain.

db-primary List remaining databases

```
sudo -u postgres psql -c "select oid, datname from pg_database order by oid;"
```

```
  oid  |  datname
-------+-----------
     1 | template1
 12134 | template0
 12139 | postgres

 16385 | test2

(4 rows)
```

Point-in-Time Recovery

Restore a Backup in Quick Start performed default recovery, which is to play all the way to the end of the WAL stream. In the case of a hardware failure this is usually the best choice but for data corruption scenarios (whether machine or human in origin) Point-in-Time Recovery (PITR) is often more appropriate.

Point-in-Time Recovery (PITR) allows the WAL to be played from the last backup to a specified time, transaction id, or recovery point. For common recovery scenarios time-based recovery is arguably the most useful. A typical recovery scenario is to restore a table that was accidentally dropped or data that was accidentally deleted. Recovering a dropped table is more dramatic so that's the example given here but deleted data would be recovered in exactly the same way.

db-primary Backup the demo cluster and create a table with very important data

```
sudo -u postgres pgbackrest --stanza=demo --type=diff backup
```

```
sudo -u postgres psql -c "begin; \
       create table important_table (message text); \
       insert into important_table values ('Important Data'); \
       commit; \
       select * from important_table;"
```

```
    message
----------------

 Important Data

(1 row)
```

It is important to represent the time as reckoned by PostgreSQL and to include timezone offsets. This reduces the possibility of unintended timezone conversions and an unexpected recovery result.

db-primary Get the time from PostgreSQL

```
sudo -u postgres psql -Atc "select current_timestamp"

2018-07-05 22:17:32.812092+00
```

Now that the time has been recorded the table is dropped. In practice finding the exact time that the table was dropped is a lot harder than in this example. It may not be possible to find the exact time, but some forensic work should be able to get you close.

db-primary Drop the important table

```
sudo -u postgres psql -c "begin; \
       drop table important_table; \
       commit; \
       select * from important_table;"

ERROR:  relation "important_table" does not exist

LINE 1: ...le important_table;     commit;     select * from important_...
                                                            ^
```

Now the restore can be performed with time-based recovery to bring back the missing table.

db-primary Stop PostgreSQL , restore the demo cluster to 2018-07-05 22:17:32.812092+00 , and display recovery.conf

```
sudo pg_ctlcluster 9.4 demo stop

sudo -u postgres pgbackrest --stanza=demo --delta \
       --type=time "--target=2018-07-05 22:17:32.812092+00" restore

sudo -u postgres cat /var/lib/postgresql/9.4/demo/recovery.conf

restore_command = '/usr/bin/pgbackrest --stanza=demo archive-get %f "%p"'

recovery_target_time = '2018-07-05 22:17:32.812092+00'
```

The recovery.conf file has been automatically generated by pgBackRest so PostgreSQL can be started immediately. Once PostgreSQL has finished recovery the table will exist again and can be queried.

db-primary Start PostgreSQL and check that the important table exists

```
sudo pg_ctlcluster 9.4 demo start

sudo -u postgres psql -c "select * from important_table"

    message
----------------

 Important Data

(1 row)
```

The PostgreSQL log also contains valuable information. It will indicate the time and transaction where the recovery stopped and also give the time of the last transaction to be applied.

db-primary Examine the PostgreSQL log output

```
sudo -u postgres cat /var/log/postgresql/postgresql-9.4-demo.log

LOG:  database system was interrupted; last known up at 2018-07-05 22:17:26 UTC
```

```
LOG:  starting point-in-time recovery to 2018-07-05 22:17:32.812092+00

LOG:  restored log file "00000004.history" from archive
LOG:  restored log file "000000040000000000000016" from archive
       [filtered 2 lines of output]
LOG:  incomplete startup packet
LOG:  restored log file "000000040000000000000017" from archive

LOG:  recovery stopping before commit of transaction 686, time 2018-07-05 22:17:33.066969+00

LOG:  redo done at 0/170157F0

LOG:  last completed transaction was at log time 2018-07-05 22:17:32.510815+00

LOG:  selected new timeline ID: 5
LOG:  restored log file "00000004.history" from archive
       [filtered 5 lines of output]
```

This example was rigged to give the correct result. If a backup after the required time is chosen then PostgreSQL will not be able to recover the lost table. PostgreSQL can only play forward, not backward. To demonstrate this the important table must be dropped (again).

db-primary Drop the important table (again)

```
sudo -u postgres psql -c "begin; \
       drop table important_table; \
       commit; \
       select * from important_table;"

ERROR:  relation "important_table" does not exist

LINE 1: ...le important_table;     commit;     select * from important_...
                                                            ^
```

Now take a new backup and attempt recovery from the new backup.

db-primary Perform a backup then attempt recovery from that backup

```
sudo -u postgres pgbackrest --stanza=demo --type=incr backup

sudo pg_ctlcluster 9.4 demo stop

sudo -u postgres pgbackrest --stanza=demo --delta \
       --type=time "--target=2018-07-05 22:17:32.812092+00" restore

sudo pg_ctlcluster 9.4 demo start

sudo -u postgres psql -c "select * from important_table"

ERROR:  relation "important_table" does not exist

LINE 1: select * from important_table
                      ^
```

Looking at the log output it's not obvious that recovery failed to restore the table. The key is to look for the presence of the "recovery stopping before…" and "last completed transaction…" log messages. If they are not present then the recovery to the specified point-in-time was not successful.

db-primary Examine the PostgreSQL log output to discover the recovery was not successful

```
sudo -u postgres cat /var/log/postgresql/postgresql-9.4-demo.log

LOG:  database system was interrupted; last known up at 2018-07-05 22:17:45 UTC

LOG:  starting point-in-time recovery to 2018-07-05 22:17:32.812092+00

LOG:  restored log file "00000005.history" from archive
LOG:  restored log file "000000050000000000000018" from archive
LOG:  redo starts at 0/18000028
```

```
LOG:  consistent recovery state reached at 0/180000F0
```

```
LOG:  incomplete startup packet
FATAL:  the database system is starting up
       [filtered 11 lines of output]
```

Using an earlier backup will allow PostgreSQL to play forward to the correct time. The info command can be used to find the next to last backup.

db-primary Get backup info for the demo cluster

```
sudo -u postgres pgbackrest info
```

```
stanza: demo
    status: ok

    db (current)
        wal archive min/max (9.4-1): 000000020000000000000009 / 000000050000000000000018

        full backup: 20180705-221614F
            timestamp start/stop: 2018-07-05 22:16:14 / 2018-07-05 22:16:26
            wal start/stop: 000000020000000000000009 / 000000020000000000000009
            database size: 19.2MB, backup size: 19.2MB
            repository size: 2.2MB, repository backup size: 2.2MB

        full backup: 20180705-221626F
            timestamp start/stop: 2018-07-05 22:16:26 / 2018-07-05 22:16:37
            wal start/stop: 00000002000000000000000A / 00000002000000000000000A
            database size: 19.2MB, backup size: 19.2MB
            repository size: 2.2MB, repository backup size: 2.2MB

        diff backup: 20180705-221626F_20180705-221652D
            timestamp start/stop: 2018-07-05 22:16:52 / 2018-07-05 22:16:55
            wal start/stop: 000000020000000000000011 / 000000020000000000000011
            database size: 19.2MB, backup size: 8.2KB
            repository size: 2.2MB, repository backup size: 400B
            backup reference list: 20180705-221626F

        incr backup: 20180705-221626F_20180705-221705I
            timestamp start/stop: 2018-07-05 22:17:05 / 2018-07-05 22:17:14
            wal start/stop: 000000030000000000000013 / 000000030000000000000013
            database size: 31.7MB, backup size: 12.5MB
            repository size: 3.7MB, repository backup size: 1.5MB
            backup reference list: 20180705-221626F
```

```
        diff backup: 20180705-221626F_20180705-221725D
```

```
            timestamp start/stop: 2018-07-05 22:17:25 / 2018-07-05 22:17:32
            wal start/stop: 000000040000000000000016 / 000000040000000000000016
            database size: 25.5MB, backup size: 6.3MB
            repository size: 3MB, repository backup size: 771.6KB
            backup reference list: 20180705-221626F

        incr backup: 20180705-221626F_20180705-221743I
            timestamp start/stop: 2018-07-05 22:17:43 / 2018-07-05 22:17:47
            wal start/stop: 000000050000000000000018 / 000000050000000000000018
            database size: 25.5MB, backup size: 1.7MB
            repository size: 3MB, repository backup size: 200.8KB
```

```
            backup reference list: 20180705-221626F, 20180705-221626F_20180705-221725D
```

The default behavior for restore is to use the last backup but an earlier backup can be specified with the –set option.

db-primary Stop PostgreSQL , restore from the selected backup, and start PostgreSQL

```
sudo pg_ctlcluster 9.4 demo stop
```

```
sudo -u postgres pgbackrest --stanza=demo --delta \
       --type=time "--target=2018-07-05 22:17:32.812092+00" \
       --set=20180705-221626F_20180705-221725D restore
```

```
sudo pg_ctlcluster 9.4 demo start
```

```
sudo -u postgres psql -c "select * from important_table"
```

```
    message
---------------

 Important Data

(1 row)
```

Now the the log output will contain the expected “recovery stopping before…” and “last completed transaction…” messages showing that the recovery was successful.

db-primary Examine the PostgreSQL log output for log messages indicating success

```
sudo -u postgres cat /var/log/postgresql/postgresql-9.4-demo.log
```

```
LOG:  database system was interrupted; last known up at 2018-07-05 22:17:26 UTC
```

```
LOG:  starting point-in-time recovery to 2018-07-05 22:17:32.812092+00
```

```
LOG:  restored log file "00000004.history" from archive
LOG:  restored log file "000000040000000000000016" from archive
       [filtered 2 lines of output]
LOG:  incomplete startup packet
LOG:  restored log file "000000040000000000000017" from archive
```

```
LOG:  recovery stopping before commit of transaction 686, time 2018-07-05 22:17:33.066969+00
```

```
LOG:  redo done at 0/170157F0
```

```
LOG:  last completed transaction was at log time 2018-07-05 22:17:32.510815+00
```

```
LOG:  restored log file "00000005.history" from archive
LOG:  restored log file "00000006.history" from archive
       [filtered 7 lines of output]
```

S3 Support

pgBackRest supports storing repositories in Amazon S3 . The bucket used to store the repository must be created in advance — pgBackRest will not do it automatically.

db-primary : /etc/pgbackrest.conf Configure S3

[demo]
db-path=/var/lib/postgresql/9.4/demo

[global]
process-max=4
repo-cipher-type=none
repo-path=/
repo-s3-bucket=demo-bucket
repo-s3-endpoint=s3.amazonaws.com
repo-s3-key=accessKey1
repo-s3-key-secret=verySecretKey1
repo-s3-region=us-east-1
repo-type=s3
retention-diff=2
retention-full=2
start-fast=y
stop-auto=y

Commands are run exactly as if the repository were stored on a local disk.

db-primary Create the stanza

```
sudo -u postgres pgbackrest --stanza=demo --log-level-console=info stanza-create
```

```
P00   INFO: stanza-create command begin 1.29: --db1-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --no-log-timestamp --repo-cipher-type=none
   --repo-path=/ --repo-s3-bucket=demo-bucket --repo-s3-endpoint=s3.amazonaws.com --repo-s3-key=
   --repo-s3-key-secret= --repo-s3-region=us-east-1 --no-repo-s3-verify-ssl --repo-type=s3
   --stanza=demo
```

```
P00   INFO: stanza-create command end: completed successfully
```

File creation time in S3 is relatively slow so commands benefit by increasing process-max to parallelize file creation.

db-primary Backup the demo cluster

```
sudo -u postgres pgbackrest --stanza=demo \
       --log-level-console=info backup
```

```
P00   INFO: backup command begin 1.29: --db1-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --no-log-timestamp --process-max=4
   --repo-cipher-type=none --repo-path=/ --repo-s3-bucket=demo-bucket
   --repo-s3-endpoint=s3.amazonaws.com --repo-s3-key= --repo-s3-key-secret=
   --repo-s3-region=us-east-1 --no-repo-s3-verify-ssl --repo-type=s3 --retention-diff=2
   --retention-full=2 --stanza=demo --start-fast --stop-auto
```

```
P00   WARN: no prior backup exists, incr backup has been changed to full
```

```
P00   INFO: execute exclusive pg_start_backup() with label "pgBackRest backup started at
   2018-07-05 22:18:11": backup begins after the requested immediate checkpoint completes
P00   INFO: backup start archive = 000000070000000000000018, lsn = 0/18000028
       [filtered 995 lines of output]
P02   INFO: backup file /var/lib/postgresql/9.4/demo/base/1/11895 (0B, 100%)
P01   INFO: backup file /var/lib/postgresql/9.4/demo/base/1/11885 (0B, 100%)
```

```
P00   INFO: full backup size = 25.5MB
```

```
P00   INFO: execute exclusive pg_stop_backup() and wait for all WAL segments to archive
P00   INFO: backup stop archive = 000000070000000000000018, lsn = 0/18000128
       [filtered 4 lines of output]
```

Delete a Stanza

The stanza-delete command removes data in the repository associated with a stanza. Use this command with caution — it will permanently remove all backups and archives from the pgBackRest repository for the specified stanza.

To delete a stanza:

- Shut down the PostgreSQL cluster associated with the stanza (or use –force to override).
- Run the stop command on the backup host (the host where the repository is mounted).
- Run the stanza-delete command on the backup host.

Once the command successfully completes, it is the responsibility of the user to remove the stanza from all pgBackRest configuration files.

db-primary Stop PostgreSQL cluster to be removed

```
sudo pg_ctlcluster 9.4 demo stop
```

db-primary Stop pgBackRest for the stanza

```
sudo -u postgres pgbackrest --stanza=demo --log-level-console=info stop
```

```
P00   INFO: stop command begin 1.29: --log-level-console=info --log-level-stderr=off
   --no-log-timestamp --repo-cipher-type=none --repo-path=/ --repo-s3-bucket=demo-bucket
   --repo-s3-endpoint=s3.amazonaws.com --repo-s3-key= --repo-s3-key-secret=
   --repo-s3-region=us-east-1 --no-repo-s3-verify-ssl --repo-type=s3 --stanza=demo
```

```
P00   INFO: stop command end: completed successfully
```

db-primary Delete the stanza

```
sudo -u postgres pgbackrest --stanza=demo --log-level-console=info stanza-delete
```

```
P00   INFO: stanza-delete command begin 1.29: --db1-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --no-log-timestamp --repo-cipher-type=none
   --repo-path=/ --repo-s3-bucket=demo-bucket --repo-s3-endpoint=s3.amazonaws.com --repo-s3-key=
   --repo-s3-key-secret= --repo-s3-region=us-east-1 --no-repo-s3-verify-ssl --repo-type=s3
   --stanza=demo
```

```
P00   INFO: stanza-delete command end: completed successfully
```

Dedicated Backup Host

The configuration described in Quickstart is suitable for simple installations but for enterprise configurations it is more typical to have a dedicated backup host. This separates the backups and WAL archive from the database server so database host failures have less impact. It is still a good idea to employ traditional backup software to backup the backup host.

Installation

A new host named backup is created to store the cluster backups.

The backrest user is created to own the pgBackRest repository. Any user can own the repository but it is best not to use postgres (if it exists) to avoid confusion.

backup Create backrest

```
sudo adduser --disabled-password --gecos "" backrest
```

pgBackRest is written in Perl which is included with Debian/Ubuntu by default. Some additional modules must also be installed but they are available as standard packages.

backup Install required Perl packages

```
sudo apt-get install libdbd-pg-perl libio-socket-ssl-perl libxml-libxml-perl
```

Debian/Ubuntu packages for pgBackRest are available at apt.postgresql.org . If they are not provided for your distribution/version it is easy to download the source and install manually.

backup Download version 1.29 of pgBackRest

```
sudo wget -q -O - \
       https://github.com/pgbackrest/pgbackrest/archive/release/1.29.tar.gz | \
       sudo tar zx -C /root
```

backup Install pgBackRest

```
sudo cp -r /root/pgbackrest-release-1.29/lib/pgBackRest \
       /usr/share/perl5
```

```
sudo find /usr/share/perl5/pgBackRest -type f -exec chmod 644 {} +
```

```
sudo find /usr/share/perl5/pgBackRest -type d -exec chmod 755 {} +
```

```
sudo cp /root/pgbackrest-release-1.29/bin/pgbackrest /usr/bin/pgbackrest
```

```
sudo chmod 755 /usr/bin/pgbackrest
```

```
sudo mkdir -m 770 /var/log/pgbackrest
```

```
sudo chown backrest:backrest /var/log/pgbackrest
```

```
sudo touch /etc/pgbackrest.conf
```

```
sudo chmod 640 /etc/pgbackrest.conf
```

```
sudo chown backrest:backrest /etc/pgbackrest.conf
```

pgBackRest includes an optional companion C library that enhances performance and enables the ‘checksum-page‘ option and encryption. Pre-built packages are generally a better option than building the C library manually but the steps required are given below for completeness. Depending on the distribution a number of packages may be required which will not be enumerated here.

db-primary Build and Install C Library

```
sudo sh -c 'cd /root/pgbackrest-release-1.29/libc && \
       perl Makefile.PL INSTALLMAN1DIR=none INSTALLMAN3DIR=none'
```

```
sudo make -C /root/pgbackrest-release-1.29/libc test
```

```
sudo make -C /root/pgbackrest-release-1.29/libc install
```

backup Create the pgBackRest repository

```
sudo mkdir /var/lib/pgbackrest
```

```
sudo chmod 750 /var/lib/pgbackrest
```

```
sudo chown backrest:backrest /var/lib/pgbackrest
```

Setup Trusted SSH

pgBackRest requires trusted (no password) SSH to enable communication between the hosts.

backup Create backup host key pair

```
sudo -u backrest mkdir -m 750 /home/backrest/.ssh
```

```
sudo -u backrest ssh-keygen -f /home/backrest/.ssh/id_rsa -t rsa -b 4096 -N ""
```

db-primary Create db-primary host key pair

```
sudo -u postgres mkdir -m 750 -p /home/postgres/.ssh
```

```
sudo -u postgres ssh-keygen -f /home/postgres/.ssh/id_rsa -t rsa -b 4096 -N ""
```

Exchange keys between backup and db-primary .

backup Copy db-primary public key to backup

```
sudo ssh root@db-primary cat /home/postgres/.ssh/id_rsa.pub | \
       sudo -u backrest tee -a /home/backrest/.ssh/authorized_keys
```

db-primary Copy backup public key to db-primary

```
sudo ssh root@backup cat /home/backrest/.ssh/id_rsa.pub | \
       sudo -u postgres tee -a /home/postgres/.ssh/authorized_keys
```

Test that connections can be made from backup to db-primary and vice versa.

backup Test connection from backup to db-primary

```
sudo -u backrest ssh postgres@db-primary
```

db-primary Test connection from db-primary to backup

```
sudo -u postgres ssh backrest@backup
```

Configuration

The backup host must be configured with the db-primary host/user and database path. The primary will be configured as db1 to allow a standby to be added later.

backup : /etc/pgbackrest.conf Configure db1-host / db1-user and db1-path

```
[demo]
db1-host=db-primary
db1-path=/var/lib/postgresql/9.4/demo
db1-user=postgres

[global]
repo-path=/var/lib/pgbackrest
retention-full=2
start-fast=y
```

The database host must be configured with the backup host/user. The default for the backup-user option is backrest . If the postgres user does restores on the backup host it is best not to also allow the postgres user to perform backups. However, the postgres user can read the repository directly if it is in the same group as the backrest user.

db-primary : /etc/pgbackrest.conf Configure backup-host / backup-user

```
[demo]
db-path=/var/lib/postgresql/9.4/demo

[global]
backup-host=backup
backup-user=backrest
log-level-file=detail
```

Commands are run the same as on a single host configuration except that some commands such as backup and expire are run from the backup host instead of the database host.

Create the stanza in the new repository.

backup Create the stanza

```
sudo -u backrest pgbackrest --stanza=demo stanza-create
```

Check that the configuration is correct on both the database and backup hosts. More information about the check command can be found in Check the Configuration .

db-primary Check the configuration

```
sudo -u postgres pgbackrest --stanza=demo check
```

backup Check the configuration

```
sudo -u backrest pgbackrest --stanza=demo check
```

Perform a Backup

To perform a backup of the PostgreSQL cluster run pgBackRest with the backup command on the backup host.

backup Backup the demo cluster

```
sudo -u backrest pgbackrest --stanza=demo backup
```

```
P00   WARN: no prior backup exists, incr backup has been changed to full
```

Since a new repository was created on the backup host the warning about the incremental backup changing to a full backup was emitted.

Restore a Backup

To perform a restore of the PostgreSQL cluster run pgBackRest with the restore command on the database host.

db-primary Stop the demo cluster, restore, and restart PostgreSQL

```
sudo pg_ctlcluster 9.4 demo stop
```

```
sudo -u postgres pgbackrest --stanza=demo --delta restore
```

```
sudo pg_ctlcluster 9.4 demo start
```

A new backup must be performed due to the timeline switch.

backup Backup the demo cluster

```
sudo -u backrest pgbackrest --stanza=demo backup
```

Asynchronous Archiving

The archive-async option offloads WAL archiving to a separate process (or processes) to improve throughput. It works by "looking ahead" to see which WAL segments are ready to be archived beyond the request that PostgreSQL is currently making via the archive_command . WAL segments are transferred to the archive directly from the pg_xlog / pg_wal directory and success is only returned by the archive_command when the WAL segment has been safely stored in the archive.

The spool directory is created to hold the current status of WAL archiving. Status files written into the spool directory are typically zero length and should consume a minimal amount of space (a few MB at most) and very little IO. All the information in this directory can be recreated so it is not necessary to preserve the spool directory if the cluster is moved to new hardware.

NOTE: In the original implementation of asynchronous archiving, WAL segments were copied to the spool directory before compression and transfer. The new implementation copies WAL directly from the pg_xlog directory. If asynchronous archiving was utilized in v1.12 or prior, read the v1.13 release notes carefully before upgrading.

db-primary Create the spool directory

```
sudo mkdir -m 750 /var/spool/pgbackrest
```

```
sudo chown postgres:postgres /var/spool/pgbackrest
```

The spool path must be configured and asynchronous archiving enabled. Asynchronous archiving automatically confers some benefit by reducing the number of ssh connections made to the backup server, but setting process-max can drastically improve performance. Be sure not to set process-max so high that it affects normal database operations.

db-primary : /etc/pgbackrest.conf Configure the spool path and asynchronous archiving

```
[demo]
db-path=/var/lib/postgresql/9.4/demo

[global]
archive-async=y
backup-host=backup
backup-user=backrest
log-level-file=detail
spool-path=/var/spool/pgbackrest

[global:archive-push]
process-max=2
```

The archive-async.log file can be used to monitor the activity of the asynchronous process. A good way to test this is to quickly push a number of WAL segments.

db-primary Test parallel asynchronous archiving

```
sudo -u postgres psql -c " \
       select pg_create_restore_point('test async push'); select pg_switch_xlog(); \
       select pg_create_restore_point('test async push'); select pg_switch_xlog(); \
       select pg_create_restore_point('test async push'); select pg_switch_xlog(); \
       select pg_create_restore_point('test async push'); select pg_switch_xlog(); \
       select pg_create_restore_point('test async push'); select pg_switch_xlog();"
```

```
sudo -u postgres pgbackrest --stanza=demo --log-level-console=info check
```

```
P00   INFO: check command begin 1.29: --backup-host=backup --backup-user=backrest
   --db1-path=/var/lib/postgresql/9.4/demo --log-level-console=info --log-level-file=detail
   --log-level-stderr=off --no-log-timestamp --stanza=demo
```

```
P00   INFO: WAL segment 000000080000000000000024 successfully stored in the archive at
   '/var/lib/pgbackrest/archive/demo/9.4-1/0000000800000000/000000080000000000000024-5778280824c975d3
```

```
P00   INFO: check command end: completed successfully
```

Now the log file will contain parallel, asynchronous activity.

db-primary Check results in the log

```
sudo -u postgres cat /var/log/pgbackrest/demo-archive-async.log
```

```
-------------------PROCESS START-------------------
P00   INFO: archive-push command begin 1.29: --archive-async --backup-host=backup
   --backup-user=backrest --db1-path=/var/lib/postgresql/9.4/demo --log-level-file=detail
   --log-level-stderr=off --no-log-timestamp --process-max=2 --spool-path=/var/spool/pgbackrest
   --stanza=demo
```

```
P00   INFO: push 2 WAL file(s) to archive: 000000080000000000000001F...000000080000000000000020
P01 DETAIL: pushed WAL file 000000080000000000000001F to archive
P02 DETAIL: pushed WAL file 000000080000000000000020 to archive
```

```
P00   INFO: archive-push command end: completed successfully

-------------------PROCESS START-------------------
P00   INFO: archive-push command begin 1.29: --archive-async --backup-host=backup
   --backup-user=backrest --db1-path=/var/lib/postgresql/9.4/demo --log-level-file=detail
   --log-level-stderr=off --no-log-timestamp --process-max=2 --spool-path=/var/spool/pgbackrest
   --stanza=demo
```

```
P00   INFO: push 3 WAL file(s) to archive: 000000080000000000000021...000000080000000000000023
P01 DETAIL: pushed WAL file 000000080000000000000021 to archive
P02 DETAIL: pushed WAL file 000000080000000000000022 to archive
P01 DETAIL: pushed WAL file 000000080000000000000023 to archive
```

```
P00   INFO: archive-push command end: completed successfully

-------------------PROCESS START-------------------
P00   INFO: archive-push command begin 1.29: --archive-async --backup-host=backup
   --backup-user=backrest --db1-path=/var/lib/postgresql/9.4/demo --log-level-file=detail
   --log-level-stderr=off --no-log-timestamp --process-max=2 --spool-path=/var/spool/pgbackrest
   --stanza=demo
```

```
P00   INFO: push 1 WAL file(s) to archive: 000000080000000000000024
P01 DETAIL: pushed WAL file 000000080000000000000024 to archive
```

```
P00   INFO: archive-push command end: completed successfully
```

Parallel Backup / Restore

pgBackRest offers parallel processing to improve performance of compression and transfer. The number of processes to be used for this feature is set using the –process-max option.

It is usually best not to use more than 25% of available CPUs for the backup command. Backups don't have to run that fast as long as they are performed regularly and the backup process should not impact database performance, if at all possible.

The restore command can and should use all available CPUs because during a restore the PostgreSQL cluster is shut down and there is generally no other important work being done on the host. If the host contains multiple clusters then that should be considered when setting restore parallelism.

backup Perform a backup with single process

```
sudo -u backrest pgbackrest --stanza=demo --type=full backup
```

backup : /etc/pgbackrest.conf Configure pgBackRest to use multiple backup processes

[demo]
db1-host=db-primary
db1-path=/var/lib/postgresql/9.4/demo
db1-user=postgres

[global]

```
process-max=3
repo-path=/var/lib/pgbackrest
retention-full=2
start-fast=y
```

backup Perform a backup with multiple processes

```
sudo -u backrest pgbackrest --stanza=demo --type=full backup
```

backup Get backup info for the demo cluster

```
sudo -u backrest pgbackrest info
```

```
stanza: demo
    status: ok

    db (current)
        wal archive min/max (9.4-1): 000000080000000000000026 / 000000080000000000000028

        full backup: 20180705-221949F

            timestamp start/stop: 2018-07-05 22:19:49 / 2018-07-05 22:20:04

            wal start/stop: 000000080000000000000026 / 000000080000000000000026
            database size: 25.5MB, backup size: 25.5MB
            repository size: 3MB, repository backup size: 3MB

        full backup: 20180705-222005F

            timestamp start/stop: 2018-07-05 22:20:05 / 2018-07-05 22:20:15

            wal start/stop: 000000080000000000000028 / 000000080000000000000028
            database size: 25.5MB, backup size: 25.5MB
            repository size: 3MB, repository backup size: 3MB
```

The performance of the last backup should be improved by using multiple processes. For very small backups the difference may not be very apparent, but as the size of the database increases so will time savings.

Starting and Stopping

Sometimes it is useful to prevent pgBackRest from running on a system. For example, when failing over from a primary to a standby it's best to prevent pgBackRest from running on the old primary in case PostgreSQL gets restarted or can't be completely killed. This will also prevent pgBackRest from running on cron .

db-primary Stop the pgBackRest services

```
sudo -u postgres pgbackrest stop
```

New pgBackRest processes will no longer run.

backup Attempt a backup

```
sudo -u backrest pgbackrest --stanza=demo backup
```

```
P00  ERROR: [062]: raised from remote process on 'db-primary': stop file exists for all stanzas
```

Specify the –force option to terminate any pgBackRest process that are currently running. If pgBackRest is already stopped then stopping again will generate a warning.

db-primary Stop the pgBackRest services again

```
sudo -u postgres pgbackrest stop
```

```
P00   WARN: stop file already exists for all stanzas
```

Start pgBackRest processes again with the start command.

db-primary Start the pgBackRest services

```
sudo -u postgres pgbackrest start
```

It is also possible to stop pgBackRest for a single stanza.

db-primary Stop pgBackRest services for the demo stanza

```
sudo -u postgres pgbackrest --stanza=demo stop
```

New pgBackRest processes for the specified stanza will no longer run.

backup Attempt a backup

```
sudo -u backrest pgbackrest --stanza=demo backup
```

```
P00  ERROR: [062]: raised from remote process on 'db-primary': stop file exists for stanza demo
```

The stanza must also be specified when starting the pgBackRest processes for a single stanza.

db-primary Start the pgBackRest services for the demo stanza

```
sudo -u postgres pgbackrest --stanza=demo start
```

Replication

Replication allows multiple copies of a PostgreSQL cluster (called standbys) to be created from a single primary. The standbys are useful for balancing reads and to provide redundancy in case the primary host fails.

Installation

A new host named db-standby is created to run the standby.

pgBackRest is written in Perl which is included with Debian/Ubuntu by default. Some additional modules must also be installed but they are available as standard packages.

db-standby Install required Perl packages

```
sudo apt-get install libdbd-pg-perl libio-socket-ssl-perl libxml-libxml-perl
```

Debian/Ubuntu packages for pgBackRest are available at apt.postgresql.org . If they are not provided for your distribution/version it is easy to download the source and install manually.

db-standby Download version 1.29 of pgBackRest

```
sudo wget -q -O - \
       https://github.com/pgbackrest/pgbackrest/archive/release/1.29.tar.gz | \
       sudo tar zx -C /root
```

db-standby Install pgBackRest

```
sudo cp -r /root/pgbackrest-release-1.29/lib/pgBackRest \
       /usr/share/perl5
```

```
sudo find /usr/share/perl5/pgBackRest -type f -exec chmod 644 {} +
```

```
sudo find /usr/share/perl5/pgBackRest -type d -exec chmod 755 {} +
```

```
sudo cp /root/pgbackrest-release-1.29/bin/pgbackrest /usr/bin/pgbackrest
```

```
sudo chmod 755 /usr/bin/pgbackrest
```

```
sudo mkdir -m 770 /var/log/pgbackrest
```

```
sudo chown postgres:postgres /var/log/pgbackrest
```

```
sudo touch /etc/pgbackrest.conf
```

```
sudo chmod 640 /etc/pgbackrest.conf
```

```
sudo chown postgres:postgres /etc/pgbackrest.conf
```

pgBackRest includes an optional companion C library that enhances performance and enables the ‘checksum-page‘ option and encryption. Pre-built packages are generally a better option than building the C library manually but the steps required are given below for completeness. Depending on the distribution a number of packages may be required which will not be enumerated here.

db-primary Build and Install C Library

```
sudo sh -c 'cd /root/pgbackrest-release-1.29/libc && \
       perl Makefile.PL INSTALLMAN1DIR=none INSTALLMAN3DIR=none'
```

```
sudo make -C /root/pgbackrest-release-1.29/libc test
```

```
sudo make -C /root/pgbackrest-release-1.29/libc install
```

The demo cluster must be created even though it will be overwritten later.

db-standby Create demo cluster

```
sudo pg_createcluster 9.4 demo
```

Setup Trusted SSH

pgBackRest requires trusted (no password) SSH to enable communication between the hosts.

db-standby Create db-standby host key pair

```
sudo -u postgres mkdir -m 750 -p /home/postgres/.ssh
```

```
sudo -u postgres ssh-keygen -f /home/postgres/.ssh/id_rsa -t rsa -b 4096 -N ""
```

Exchange keys between backup and db-standby .

backup Copy db-standby public key to backup

```
sudo ssh root@db-standby cat /home/postgres/.ssh/id_rsa.pub | \
       sudo -u backrest tee -a /home/backrest/.ssh/authorized_keys
```

db-standby Copy backup public key to db-standby

```
sudo ssh root@backup cat /home/backrest/.ssh/id_rsa.pub | \
       sudo -u postgres tee -a /home/postgres/.ssh/authorized_keys
```

Test that connections can be made from backup to db-standby and vice versa.

backup Test connection from backup to db-standby

```
sudo -u backrest ssh postgres@db-standby
```

db-standby Test connection from db-standby to backup

```
sudo -u postgres ssh backrest@backup
```

Hot Standby

A hot standby performs replication using the WAL archive and allows read-only queries.

pgBackRest configuration is very similar to db-primary except that the standby_mode setting will be enabled to keep the cluster in recovery mode when the end of the WAL stream has been reached.

db-standby : /etc/pgbackrest.conf Configure pgBackRest on the standby

[demo]
db-path=/var/lib/postgresql/9.4/demo
recovery-option=standby_mode=on

[global]
backup-host=backup

Now the standby can be created with the restore command.

db-standby Restore the demo standby cluster

```
sudo -u postgres pgbackrest --stanza=demo --delta restore
```

```
sudo -u postgres cat /var/lib/postgresql/9.4/demo/recovery.conf
```

```
standby_mode = 'on'
restore_command = '/usr/bin/pgbackrest --stanza=demo archive-get %f "%p"'
```

Note that the standby_mode setting has been written into the recovery.conf file. Configuring recovery settings in pgBackRest means that the recovery.conf file does not need to be stored elsewhere since it will be properly recreated with each restore. The –type=preserve option can be used with the restore to leave the existing recovery.conf file in place if that behavior is preferred.

The hot_standby setting must be enabled before starting PostgreSQL to allow read-only connections on db-standby . Otherwise, connection attempts will be refused.

db-standby : /etc/postgresql/9.4/demo/postgresql.conf Enable hot_standby

hot_standby = on
log_filename = 'postgresql.log'
log_line_prefix = ''

db-standby Start PostgreSQL

```
sudo pg_ctlcluster 9.4 demo start
```

The PostgreSQL log gives valuable information about the recovery. Note especially that the cluster has entered standby mode and is ready to accept read-only connections.

db-standby Examine the PostgreSQL log output for log messages indicating success

```
sudo -u postgres cat /var/log/postgresql/postgresql-9.4-demo.log
```

```
LOG:  could not bind IPv6 socket: Cannot assign requested address
HINT:  Is another postmaster already running on port 5432? If not, wait a few seconds and retry.
LOG:  database system was interrupted; last known up at 2018-07-05 22:20:06 UTC
```

```
LOG:  entering standby mode
```

```
LOG:  restored log file "00000008.history" from archive
LOG:  incomplete startup packet
LOG:  restored log file "000000080000000000000028" from archive
LOG:  redo starts at 0/28000028
LOG:  consistent recovery state reached at 0/280000F0
```

```
LOG:  database system is ready to accept read only connections
```

An easy way to test that replication is properly configured is to create a table on db-primary .

db-primary Create a new table on the primary

```
sudo -u postgres psql -c " \
       begin; \
       create table replicated_table (message text); \
       insert into replicated_table values ('Important Data'); \
       commit; \
       select * from replicated_table";
```

```
    message
----------------
```

```
 Important Data
```

```
(1 row)
```

And then query the same table on db-standby .

db-standby Query new table on the standby

```
sudo -u postgres psql -c "select * from replicated_table;"
```

```
ERROR:  relation "replicated_table" does not exist
```

```
LINE 1: select * from replicated_table;
                      ^
```

So, what went wrong? Since PostgreSQL is pulling WAL segments from the archive to perform replication, changes won't be seen on the standby until the WAL segment that contains those changes is pushed from db-primary .

This can be done manually by calling pg_switch_xlog() which pushes the current WAL segment to the archive (a new WAL segment is created to contain further changes).

db-primary Call pg_switch_xlog()

```
sudo -u postgres psql -c "select *, current_timestamp from pg_switch_xlog()";
```

```
 pg_switch_xlog |              now
----------------+-------------------------------
 0/290199E0     | 2018-07-05 22:20:49.401096+00
(1 row)
```

Now after a short delay the table will appear on db-standby .

db-standby Now the new table exists on the standby (may require a few retries)

```
sudo -u postgres psql -c " \
       select *, current_timestamp from replicated_table"
```

```
    message     |              now
----------------+-------------------------------

 Important Data | 2018-07-05 22:20:50.956462+00

(1 row)
```

Check the standby configuration for access to the repository.

db-standby Check the configuration

```
sudo -u postgres pgbackrest --stanza=demo --log-level-console=info check
```

```
P00   INFO: check command begin 1.29: --backup-host=backup --db1-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --no-log-timestamp --stanza=demo

P00   INFO: switch xlog cannot be performed on the standby, all other checks passed successfully

P00   INFO: check command end: completed successfully
```

Streaming Replication

Instead of relying solely on the WAL archive, streaming replication makes a direct connection to the primary and applies changes as soon as they are made on the primary. This results in much less lag between the primary and standby.

Streaming replication requires a user with the replication privilege.

db-primary Create replication user

```
sudo -u postgres psql -c " \
       create user replicator password 'jw8s0F4' replication";
```

```
CREATE ROLE
```

The pg_hba.conf file must be updated to allow the standby to connect as the replication user. Be sure to replace the IP address below with the actual IP address of your db-primary . A reload will be required after modifying the pg_hba.conf file.

db-primary Create pg_hba.conf entry for replication user

```
sudo -u postgres sh -c 'echo \
       "host    replication     replicator      172.17.0.5/32           md5" \
       >> /etc/postgresql/9.4/demo/pg_hba.conf'
```

```
sudo pg_ctlcluster 9.4 demo reload
```

The standby needs to know how to contact the primary so the primary_conninfo setting will be configured in pgBackRest .

db-standby : /etc/pgbackrest.conf Set primary_conninfo

```
[demo]
db-path=/var/lib/postgresql/9.4/demo
recovery-option=standby_mode=on
recovery-option=primary_conninfo=host=172.17.0.3 port=5432 user=replicator

[global]
backup-host=backup
```

It is possible to configure a password in the primary_conninfo setting but using a .pgpass file is more flexible and secure.

db-standby Configure the replication password in the .pgpass file.

```
sudo -u postgres sh -c 'echo \
       "172.17.0.3:*:replication:replicator:jw8s0F4" \
       >> /home/postgres/.pgpass'
```

```
sudo -u postgres chmod 600 /home/postgres/.pgpass
```

Now the standby can be created with the restore command.

db-standby Stop PostgreSQL and restore the demo standby cluster

```
sudo pg_ctlcluster 9.4 demo stop
```

```
sudo -u postgres pgbackrest --stanza=demo --delta restore
```

```
sudo -u postgres cat /var/lib/postgresql/9.4/demo/recovery.conf
```

```
primary_conninfo = 'host=172.17.0.3 port=5432 user=replicator'
standby_mode = 'on'
restore_command = '/usr/bin/pgbackrest --stanza=demo archive-get %f "%p"'
```

db-standby Start PostgreSQL

```
sudo pg_ctlcluster 9.4 demo start
```

The PostgreSQL log will confirm that streaming replication has started.

db-standby Examine the PostgreSQL log output for log messages indicating success

```
sudo -u postgres cat /var/log/postgresql/postgresql-9.4-demo.log
```

```
       [filtered 10 lines of output]
LOG:  database system is ready to accept read only connections
LOG:  restored log file "000000080000000000000029" from archive
```

```
LOG:  started streaming WAL from primary at 0/2A000000 on timeline 8
```

Now when a table is created on db-primary it will appear on db-standby quickly and without the need to call pg_switch_xlog() .

db-primary Create a new table on the primary

```
sudo -u postgres psql -c " \
       begin; \
       create table stream_table (message text); \
       insert into stream_table values ('Important Data'); \
       commit; \
       select *, current_timestamp from stream_table";
```

```
    message     |              now
----------------+-------------------------------
```

```
 Important Data | 2018-07-05 22:21:03.489564+00
```

```
(1 row)
```

db-standby Query table on the standby

```
sudo -u postgres psql -c " \
       select *, current_timestamp from stream_table"
```

```
    message     |             now
----------------+-------------------------------
```

```
 Important Data | 2018-07-05 22:21:03.7855+00
```

```
(1 row)
```

Backup from a Standby

pgBackRest can perform backups on a standby instead of the primary. Standby backups require the db-standby host to be configured and the backup-standby option enabled. If more than one standby is configured then the first running standby found will be used for the backup.

backup : /etc/pgbackrest.conf Configure db2-host / db2-user and db2-path

[demo]
db1-host=db-primary
db1-path=/var/lib/postgresql/9.4/demo
db1-user=postgres
db2-host=db-standby
db2-path=/var/lib/postgresql/9.4/demo
db2-user=postgres

[global]
backup-standby=y
process-max=3
repo-path=/var/lib/pgbackrest
retention-full=2
start-fast=y

Both the primary and standby databases are required to perform the backup, though the vast majority of the files will be copied from the standby to reduce load on the primary. The database hosts can be configured in any order. pgBackRest will automatically determine which is the primary and which is the standby.

backup Backup the demo cluster from db-standby

```
sudo -u backrest pgbackrest --stanza=demo --log-level-console=detail backup
```

```
       [filtered 2 lines of output]
P00   INFO: execute exclusive pg_start_backup() with label "pgBackRest backup started at
   2018-07-05 22:21:04": backup begins after the requested immediate checkpoint completes
P00   INFO: backup start archive = 000000080000000000000002B, lsn = 0/2B000028
```

```
P00   INFO: wait for replay on the standby to reach 0/2B000028
P00   INFO: replay on the standby reached 0/2B0000C8
```

```
P04   INFO: backup file db-standby:/var/lib/postgresql/9.4/demo/base/12139/12009 (240KB, 12%)
   checksum 14a2c1b5fe6f7155aaeaaec0ca7eb60c46aa5ac1
P02   INFO: backup file db-standby:/var/lib/postgresql/9.4/demo/base/12139/12007 (392KB, 32%)
   checksum 87f10131e163854653b5968b390075c2dbfbb735
P04   INFO: backup file db-standby:/var/lib/postgresql/9.4/demo/base/12139/12010 (232KB, 44%)
   checksum 3dbbb29edd9a48fd214a407048690b5dee1b9f75
P02   INFO: backup file db-standby:/var/lib/postgresql/9.4/demo/base/12139/11891 (128KB, 51%)
   checksum c111ec02806557817e44dceb037f760d69e6e58a
P04   INFO: backup file db-standby:/var/lib/postgresql/9.4/demo/base/12139/11892 (88KB, 55%)
   checksum 5c5d1c6a0b94c76eee597f9812fc105863097dbc
```

```
P01   INFO: backup file db-primary:/var/lib/postgresql/9.4/demo/global/pg_control (8KB, 56%)
   checksum c3017ecef8bf1b95a65ab87e0a323d6f1035ba92
```

```
P02   INFO: backup file db-standby:/var/lib/postgresql/9.4/demo/base/12139/11900 (64KB, 59%)
   checksum 85f2904959a51142abfeee4ac714c0b83768ebc4
```

```
P01   INFO: backup file db-primary:/var/lib/postgresql/9.4/demo/backup_label (238B, 59%) checksum
   839bb33be4e25f8a2bda51cdda80cd5761fe934a
```

```
P02   INFO: backup file db-standby:/var/lib/postgresql/9.4/demo/base/12139/11903 (56KB, 62%)
   checksum 6469c3d1fc92d2d1792b2f66dab2266660bdd09f
P03   INFO: backup file db-standby:/var/lib/postgresql/9.4/demo/base/12139/11889 (344KB, 80%)
   checksum 95f362e9fd6b1863138b8c3e9d5a6bf872c9dc15
       [filtered 29 lines of output]
```

This incremental backup shows that most of the files are copied from the db-standby host and only a few are copied from the db-primary host.

pgBackRest creates a standby backup that is identical to a backup performed on the primary. It does this by starting/stopping the backup on the db-primary host, copying only files that are replicated from the db-standby host, then copying the remaining few files from the db-primary host. This means that logs and statistics from the primary database will be included in the backup.

Upgrading PostgreSQL

Immediately after upgrading PostgreSQL to a newer major version, the db-path for all pgBackRest configurations must be set to the new database location and the stanza-upgrade run on the backup host. If the database is offline use the –no-online option.

The following instructions are not meant to be a comprehensive guide for upgrading PostgreSQL , rather they will outline the general process for upgrading a primary and standby with the intent of demonstrating the steps required to reconfigure pgBackRest . It is recommended that a backup be taken prior to upgrading.

db-primary Install new PostgreSQL version

```
sudo apt-get install postgresql-9.5
```

Create the new cluster. If the PostgreSQL install creates a default cluster, then remove it to avoid confusion.

db-primary Drop default cluster and create the new demo cluster

```
sudo pg_dropcluster 9.5 main
```

```
sudo -u postgres /usr/lib/postgresql/9.5/bin/initdb \
       -D /var/lib/postgresql/9.5/demo -k -A peer
```

```
sudo pg_createcluster 9.5 demo
```

```
Configuring already existing cluster (configuration: /etc/postgresql/9.5/demo, data:
   /var/lib/postgresql/9.5/demo, owner: 5000:5000)
Ver Cluster Port Status Owner    Data directory              Log file
9.5 demo    5433 down   postgres /var/lib/postgresql/9.5/demo
   /var/log/postgresql/postgresql-9.5-demo.log
```

Stop the old cluster on the standby since it will be restored from the newly upgraded cluster to ensure the database system id is identical on both the primary and standby.

db-standby Stop old cluster and drop default cluster if created

```
sudo pg_ctlcluster 9.4 demo stop
```

```
sudo apt-get install postgresql-9.5
```

```
sudo pg_dropcluster 9.5 main
```

Stop the old cluster on the primary and perform the upgrade.

db-primary Stop old cluster and perform the upgrade

```
sudo pg_ctlcluster 9.4 demo stop
```

```
sudo -u postgres sh -c 'cd /var/lib/postgresql && \
       /usr/lib/postgresql/9.5/bin/pg_upgrade \
       --old-bindir=/usr/lib/postgresql/9.4/bin \
       --new-bindir=/usr/lib/postgresql/9.5/bin \
       --old-datadir=/var/lib/postgresql/9.4/demo \
       --new-datadir=/var/lib/postgresql/9.5/demo \
       --old-options=" -c config_file=/etc/postgresql/9.4/demo/postgresql.conf" \
       --new-options=" -c config_file=/etc/postgresql/9.5/demo/postgresql.conf"'
```

```
       [filtered 68 lines of output]
Creating script to delete old cluster                         ok

Upgrade Complete

----------------
Optimizer statistics are not transferred by pg_upgrade so,
       [filtered 4 lines of output]
```

Configure the new cluster settings and port.

db-primary : /etc/postgresql/9.5/demo/postgresql.conf Configure PostgreSQL

archive_command = 'pgbackrest –stanza=demo archive-push %p'
archive_mode = on
listen_addresses = '*'
log_line_prefix = ''
max_wal_senders = 3
port = 5432
wal_level = hot_standby

Update the pgBackRest configuration on all systems to point to the new cluster.

db-primary : /etc/pgbackrest.conf Upgrade the db-path

[demo]
db-path=/var/lib/postgresql/9.5/demo

[global]
archive-async=y
backup-host=backup
backup-user=backrest
log-level-file=detail
spool-path=/var/spool/pgbackrest

[global:archive-push]
process-max=2

db-standby : /etc/pgbackrest.conf Upgrade the db-path

[demo]
db-path=/var/lib/postgresql/9.5/demo
recovery-option=standby_mode=on
recovery-option=primary_conninfo=host=172.17.0.3 port=5432 user=replicator

[global]
backup-host=backup

backup : /etc/pgbackrest.conf Upgrade db1-path and db2-path , disable backup from standby

[demo]
db1-host=db-primary
db1-path=/var/lib/postgresql/9.5/demo
db1-user=postgres
db2-host=db-standby
db2-path=/var/lib/postgresql/9.5/demo
db2-user=postgres

[global]
backup-standby=n
process-max=3
repo-path=/var/lib/pgbackrest
retention-full=2
start-fast=y

db-primary Copy hba configuration

```
sudo cp /etc/postgresql/9.4/demo/pg_hba.conf \
       /etc/postgresql/9.5/demo/pg_hba.conf
```

Before starting the new cluster, the stanza-upgrade command must be run on the server where the pgBackRest repository is located.

backup Upgrade the stanza

```
sudo -u backrest pgbackrest --stanza=demo --no-online \
       --log-level-console=info stanza-upgrade
```

```
P00   INFO: stanza-upgrade command begin 1.29: --no-backup-standby --db1-host=db-primary
   --db1-path=/var/lib/postgresql/9.5/demo --db1-user=postgres --db2-host=db-standby
   --db2-path=/var/lib/postgresql/9.5/demo --db2-user=postgres --log-level-console=info
   --log-level-stderr=off --no-log-timestamp --no-online --repo-path=/var/lib/pgbackrest
   --stanza=demo
```

```
P00   INFO: stanza-upgrade command end: completed successfully
```

Start the new cluster and confirm it is successfully installed.

db-primary Start new cluster

```
sudo pg_ctlcluster 9.5 demo start
```

Test configuration using the check command. The warning on the backup host regarding the standby being down is expected and can be ignored.

db-primary Check configuration

```
sudo -u postgres pg_lsclusters
```

```
Ver Cluster Port Status Owner    Data directory               Log file
9.4 demo    5432 down   postgres /var/lib/postgresql/9.4/demo
   /var/log/postgresql/postgresql-9.4-demo.log
9.5 demo    5432 online postgres /var/lib/postgresql/9.5/demo
   /var/log/postgresql/postgresql-9.5-demo.log
```

```
sudo -u postgres pgbackrest --stanza=demo check
```

backup Check configuration

```
sudo -u backrest pgbackrest --stanza=demo check
```

```
P00   WARN: [056]: raised from remote process on 'db-standby': could not connect to server: No
   such file or directory
               Is the server running locally and accepting
               connections on Unix domain socket "/var/run/postgresql/.s.PGSQL.5432"?
```

Remove the old cluster.

db-primary Remove old cluster

```
sudo pg_dropcluster 9.4 demo
```

Run a full backup on the new cluster and then restore the standby from the backup. The backup type will automatically be changed to full if incr or diff is requested.

backup Run a full backup

```
sudo -u backrest pgbackrest --stanza=demo --type=full backup
```

Install the new PostgreSQL binaries on the standby and create the cluster.

db-standby Remove old cluster and initialize new one

```
sudo pg_dropcluster 9.4 demo
```

```
sudo -u postgres /usr/lib/postgresql/9.5/bin/initdb \
       -D /var/lib/postgresql/9.5/demo -k -A peer
```

```
sudo pg_createcluster 9.5 demo
```

```
Configuring already existing cluster (configuration: /etc/postgresql/9.5/demo, data:
   /var/lib/postgresql/9.5/demo, owner: 5000:5000)
Ver Cluster Port Status Owner    Data directory               Log file
9.5 demo    5432 down   postgres /var/lib/postgresql/9.5/demo
   /var/log/postgresql/postgresql-9.5-demo.log
```

db-standby Restore the demo standby cluster

```
sudo -u postgres pgbackrest --stanza=demo --delta restore
```

db-standby : /etc/postgresql/9.5/demo/postgresql.conf Configure PostgreSQL

hot_standby = on

db-standby Start PostgreSQL

```
sudo pg_ctlcluster 9.5 demo start
```

Backup from standby can be enabled now that the standby is restored.

backup : /etc/pgbackrest.conf Reenable backup from standby

[demo]
db1-host=db-primary
db1-path=/var/lib/postgresql/9.5/demo
db1-user=postgres
db2-host=db-standby
db2-path=/var/lib/postgresql/9.5/demo
db2-user=postgres

[global]
backup-standby=y
process-max=3
repo-path=/var/lib/pgbackrest
retention-full=2
start-fast=y

— title: Command Reference draft: false —

Introduction

Commands are used to execute the various pgBackRest functions. Here the command options are listed exhaustively, that is, each option applicable to a command is listed with that command even if it applies to one or more other commands. This includes all the options that may also configured in pgbackrest.conf .

Archive Get Command (archive-get)

WAL segments are required for restoring a PostgreSQL cluster or maintaining a replica.

General Options

Buffer Size Option (–buffer-size) Buffer size for file operations.

Set the buffer size used for copy, compress, and uncompress functions. A maximum of 3 buffers will be in use at a time per process. An additional maximum of 256K per process may be used for zlib buffers.

```
default: 4194304
example: --buffer-size=32768
```

SSH client command Option (–cmd-ssh) Path to ssh client executable.

Use a specific SSH client when an alternate is desired or the ssh executable is not in $PATH.

```
default: ssh
example: --cmd-ssh=/usr/bin/ssh
```

Compress Option (–compress) Use gzip file compression.

Backup files are compatible with command-line gzip tools.

```
default: y
example: --no-compress
```

Compress Level Option (–compress-level) Compression level for stored files.

Sets the zlib level to be used for file compression when compress=y .

```
default: 6
allowed: 0-9
example: --compress-level=9
```

Network Compress Level Option (–compress-level-network) Compression level for network transfer when compress=n .

Sets the zlib level to be used for protocol compression when compress=n and the database cluster is not on the same host as the backup. Protocol compression is used to reduce network traffic but can be disabled by setting compress-level-network=0 . When compress=y the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once. SSH compression is always disabled.

```
default: 3
allowed: 0-9
example: --compress-level-network=1
```

Config Option (–config) pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: /etc/pgbackrest.conf
example: --config=/var/lib/backrest/pgbackrest.conf
```

Database Timeout Option (–db-timeout) Database query timeout.

Sets the timeout, in seconds, for queries against the database. This includes the pg_start_backup() and pg_stop_backup() functions which can each take a substantial amount of time. Because of this the timeout should be kept high unless you know that these functions will return quickly (i.e. if you have set startfast=y and you know that the database cluster will not generate many WAL segments during the backup).

```
default: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

Lock Path Option (–lock-path) Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: --lock-path=/backup/db/lock
```

Neutral Umask Option (–neutral-umask) Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640. The lock and log directories set the directory and file mode to 0770 and 0660 respectively.

To use the executing user's umask instead specify neutral-umask=n in the config file or –no-neutral-umask on the command line.

```
default: y
example: --no-neutral-umask
```

Protocol Timeout Option (–protocol-timeout) Protocol timeout.

Sets the timeout, in seconds, that the local or remote process will wait for a new message to be received on the protocol layer. This prevents processes from waiting indefinitely for a message. The protocol-timeout option must be greater than the db-timeout option.

```
default: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

Stanza Option (–stanza) Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one Postgres database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

```
example: --stanza=main
```

Log Options

Console Log Level Option (–log-level-console) Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-console=error
```

File Log Level Option (–log-level-file) Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: --log-level-file=debug
```

Std Error Log Level Option (–log-level-stderr) Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr .

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-stderr=error
```

Log Path Option (–log-path) Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=none then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Timestamp Option (–log-timestamp) Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Repository Options

Backup Host Command Option (–backup-cmd) pgBackRest exe path on the backup host.

Required only if the path to pgbackrest is different on the local and backup hosts. If not defined, the backup host exe path will be set the same as the local exe path.

```
example: --backup-cmd=/usr/lib/backrest/bin/pgbackrest
```

Backup Host Configuration Option (–backup-config) pgBackRest backup host configuration file.

Sets the location of the configuration file on the backup host. This is only required if the backup host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --backup-config=/etc/pgbackrest_backup.conf
```

Backup Host Option (–backup-host) Backup host when operating remotely via SSH.

Make sure that trusted SSH authentication is configured between the db host and the backup host.

When backing up to a locally mounted network filesystem this setting is not required.

```
example: --backup-host=backup.domain.com
```

Backup SSH Port Option (–backup-ssh-port) Backup server SSH port when backup-host is set.

Use this option to specify a non-default SSH port for the backup server.

```
example: --backup-ssh-port=25
```

Backup User Option (–backup-user) Backup host user when backup-host is set.

Defines the user that will be used for operations on the backup server. Preferably this is not the postgres user but rather some other user like backrest . If PostgreSQL runs on the backup server the postgres user can be placed in the backrest group so it has read permissions on the repository without being able to damage the contents accidentally.

```
default: backrest
example: --backup-user=backrest
```

Repository Cipher Passphrase Option (–repo-cipher-pass) Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

```
example: --repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

Repository Cipher Type Option (–repo-cipher-type) Cipher used to encrypt the repository.

The following repository types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

```
default: none
example: --repo-cipher-type=aes-256-cbc
```

Repository Path Option (–repo-path) Repository path where WAL segments and backups stored.

The repository is where pgBackRest stores backup and archives WAL segments.

If you are new to backup then it will be difficult to estimate in advance how much space you'll need. The best thing to do is take some backups then record the size of different types of backups (full/incr/diff) and measure the amount of WAL generated per day. This will give you a general idea of how much space you'll need, though of course requirements will likely change over time as your database evolves.

```
default: /var/lib/pgbackrest
example: --repo-path=/backup/db/backrest
```

S3 Repository Bucket Option (–repo-s3-bucket) S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo , so logs and other AWS generated content can also be stored in the bucket.

```
example: --repo-s3-bucket=db-backup
```

S3 SSL CA File Option (–repo-s3-ca-file) S3 SSL CA File.

Use a CA file other than the system default.

```
example: --repo-s3-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

S3 SSL CA Path Option (–repo-s3-ca-path) S3 SSL CA Path.

Use a CA path other than the system default.

```
example: --repo-s3-ca-path=/etc/pki/tls/certs
```

S3 Repository Endpoint Option (–repo-s3-endpoint) S3 repository endpoint.

The AWS end point should be valid for the selected region.

```
example: --repo-s3-endpoint=s3.amazonaws.com
```

S3 Repository Host Option (–repo-s3-host) S3 repository host.

Connect to a host other than the end point. This is typically used for testing.

```
example: --repo-s3-host=127.0.0.1
```

S3 Repository Access Key Option (–repo-s3-key) S3 repository access key.

AWS key used to access this bucket.

```
example: --repo-s3-key=AKIAIOSFODNN7EXAMPLE
```

S3 Repository Secret Access Key Option (–repo-s3-key-secret) S3 repository secret access key.

AWS secret key used to access this bucket.

```
example: --repo-s3-key-secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

S3 Repository Region Option (–repo-s3-region) S3 repository region.

The AWS region where the bucket was created.

```
example: --repo-s3-region=us-east-1
```

S3 Repository Verify SSL Option (–repo-s3-verify-ssl) Verify S3 server certificate.

Disables verification of the S3 server certificate. This should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo-s3-verify-ssl
```

Repository Type Option (–repo-type) Type of storage used for the repository.

The following repository types are supported:

- cifs - Like posix , but disables links and directory fsyncs
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service

```
default: posix
example: --repo-type=cifs
```

Stanza Options

Database Path Option (–db-path) Cluster data directory.

This should be the same as the data_directory setting in postgresql.conf . Even though this value can be read from postgresql.conf or the database cluster it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

The db-path option is tested against the value reported by PostgreSQL on every online backup so it should always be current.

```
example: --db-path=/data/db
```

Archive Push Command (archive-push)

The WAL segment may be pushed immediately to the archive or stored locally depending on the value of archive-async

Command Options

Asynchronous Archiving Option (–archive-async) Archive WAL segments asynchronously.

WAL segments will be copied to the local repo, then a process will be forked to compress the segment and transfer it to the remote repo if configured. Control will be returned to PostgreSQL as soon as the WAL segment is copied locally.

```
default: n
example: --archive-async
```

Maximum Archive Queue Size Option (–archive-queue-max) Limit size (in bytes) of the PostgreSQL archive queue.

After the limit is reached, the following will happen:

1. pgBackRest will notify PostgreSQL that the WAL was successfully archived, then **DROP IT** .
2. A warning will be output to the Postgres log.

If this occurs then the archive log stream will be interrupted and PITR will not be possible past that point. A new backup will be required to regain full restore capability.

In asynchronous mode the entire queue will be dropped to prevent spurts of WAL getting through before the queue limit is exceeded again.

The purpose of this feature is to prevent the log volume from filling up at which point Postgres will stop completely. Better to lose the backup than have PostgreSQL go down.

```
example: --archive-queue-max=1073741824
```

Archive Timeout Option (–archive-timeout) Archive timeout.

Set maximum time, in seconds, to wait for each WAL segment to reach the pgBackRest archive repository. The timeout applies to the check and backup commands when waiting for WAL segments required for backup consistency to be archived.

```
default: 60
allowed: 0.1-86400
example: --archive-timeout=30
```

General Options

Buffer Size Option (–buffer-size) Buffer size for file operations.

Set the buffer size used for copy, compress, and uncompress functions. A maximum of 3 buffers will be in use at a time per process. An additional maximum of 256K per process may be used for zlib buffers.

```
default: 4194304
example: --buffer-size=32768
```

SSH client command Option (–cmd-ssh) Path to ssh client executable.

Use a specific SSH client when an alternate is desired or the ssh executable is not in $PATH.

```
default: ssh
example: --cmd-ssh=/usr/bin/ssh
```

Compress Option (–compress) Use gzip file compression.

Backup files are compatible with command-line gzip tools.

```
default: y
example: --no-compress
```

Compress Level Option (–compress-level) Compression level for stored files.

Sets the zlib level to be used for file compression when compress=y .

```
default: 6
allowed: 0-9
example: --compress-level=9
```

Network Compress Level Option (–compress-level-network) Compression level for network transfer when compress=n .

Sets the zlib level to be used for protocol compression when compress=n and the database cluster is not on the same host as the backup. Protocol compression is used to reduce network traffic but can be disabled by setting compress-level-network=0 . When compress=y the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once. SSH compression is always disabled.

```
default: 3
allowed: 0-9
example: --compress-level-network=1
```

Config Option (–config) pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: /etc/pgbackrest.conf
example: --config=/var/lib/backrest/pgbackrest.conf
```

Database Timeout Option (–db-timeout) Database query timeout.

Sets the timeout, in seconds, for queries against the database. This includes the pg_start_backup() and pg_stop_backup() functions which can each take a substantial amount of time. Because of this the timeout should be kept high unless you know that these functions will return quickly (i.e. if you have set startfast=y and you know that the database cluster will not generate many WAL segments during the backup).

```
default: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

Lock Path Option (–lock-path) Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: --lock-path=/backup/db/lock
```

Neutral Umask Option (–neutral-umask) Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640. The lock and log directories set the directory and file mode to 0770 and 0660 respectively.

To use the executing user's umask instead specify neutral-umask=n in the config file or –no-neutral-umask on the command line.

```
default: y
example: --no-neutral-umask
```

Process Maximum Option (–process-max) Max processes to use for compress/transfer.

Each process will perform compression and transfer to make the command run faster, but don't set process-max so high that it impacts database performance.

```
default: 1
allowed: 1-96
example: --process-max=4
```

Protocol Timeout Option (–protocol-timeout) Protocol timeout.

Sets the timeout, in seconds, that the local or remote process will wait for a new message to be received on the protocol layer. This prevents processes from waiting indefinitely for a message. The protocol-timeout option must be greater than the db-timeout option.

```
default: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

Spool Path Option (–spool-path) Path where transient data is stored.

This path is used to store acknowledgements from the asynchronous archive-push process. These files are generally very small (zero to a few hundred bytes) so not much space is required.

The data stored in the spool path is not strictly temporary since it can and should survive a reboot. However, loss of the data in the spool path is not a problem. pgBackRest will simply recheck each WAL segment to ensure it is safely archived.

```
default: /var/spool/pgbackrest
example: --spool-path=/backup/db/spool
```

Stanza Option (–stanza) Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one Postgres database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

```
example: --stanza=main
```

Log Options

Console Log Level Option (–log-level-console) Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-console=error
```

File Log Level Option (–log-level-file) Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: --log-level-file=debug
```

Std Error Log Level Option (–log-level-stderr) Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr .

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-stderr=error
```

Log Path Option (–log-path) Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=none then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Timestamp Option (–log-timestamp) Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Repository Options

Backup Host Command Option (–backup-cmd) pgBackRest exe path on the backup host.

Required only if the path to pgbackrest is different on the local and backup hosts. If not defined, the backup host exe path will be set the same as the local exe path.

```
example: --backup-cmd=/usr/lib/backrest/bin/pgbackrest
```

Backup Host Configuration Option (–backup-config) pgBackRest backup host configuration file.

Sets the location of the configuration file on the backup host. This is only required if the backup host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --backup-config=/etc/pgbackrest_backup.conf
```

Backup Host Option (–backup-host) Backup host when operating remotely via SSH.

Make sure that trusted SSH authentication is configured between the db host and the backup host.

When backing up to a locally mounted network filesystem this setting is not required.

```
example: --backup-host=backup.domain.com
```

Backup SSH Port Option (–backup-ssh-port) Backup server SSH port when backup-host is set.

Use this option to specify a non-default SSH port for the backup server.

```
example: --backup-ssh-port=25
```

Backup User Option (–backup-user) Backup host user when backup-host is set.

Defines the user that will be used for operations on the backup server. Preferably this is not the postgres user but rather some other user like backrest . If PostgreSQL runs on the backup server the postgres user can be placed in the backrest group so it has read permissions on the repository without being able to damage the contents accidentally.

```
default: backrest
example: --backup-user=backrest
```

Repository Cipher Passphrase Option (–repo-cipher-pass) Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

```
example: --repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

Repository Cipher Type Option (–repo-cipher-type) Cipher used to encrypt the repository.

The following repository types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

```
default: none
example: --repo-cipher-type=aes-256-cbc
```

Repository Path Option (–repo-path) Repository path where WAL segments and backups stored.

The repository is where pgBackRest stores backup and archives WAL segments.

If you are new to backup then it will be difficult to estimate in advance how much space you'll need. The best thing to do is take some backups then record the size of different types of backups (full/incr/diff) and measure the amount of WAL generated per day. This will give you a general idea of how much space you'll need, though of course requirements will likely change over time as your database evolves.

```
default: /var/lib/pgbackrest
example: --repo-path=/backup/db/backrest
```

S3 Repository Bucket Option (–repo-s3-bucket) S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo , so logs and other AWS generated content can also be stored in the bucket.

```
example: --repo-s3-bucket=db-backup
```

S3 SSL CA File Option (–repo-s3-ca-file) S3 SSL CA File.

Use a CA file other than the system default.

```
example: --repo-s3-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

S3 SSL CA Path Option (–repo-s3-ca-path) S3 SSL CA Path.

Use a CA path other than the system default.

```
example: --repo-s3-ca-path=/etc/pki/tls/certs
```

S3 Repository Endpoint Option (–repo-s3-endpoint) S3 repository endpoint.

The AWS end point should be valid for the selected region.

```
example: --repo-s3-endpoint=s3.amazonaws.com
```

S3 Repository Host Option (–repo-s3-host) S3 repository host.

Connect to a host other than the end point. This is typically used for testing.

```
example: --repo-s3-host=127.0.0.1
```

S3 Repository Access Key Option (–repo-s3-key) S3 repository access key.

AWS key used to access this bucket.

```
example: --repo-s3-key=AKIAIOSFODNN7EXAMPLE
```

S3 Repository Secret Access Key Option (–repo-s3-key-secret) S3 repository secret access key.

AWS secret key used to access this bucket.

```
example: --repo-s3-key-secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

S3 Repository Region Option (–repo-s3-region) S3 repository region.

The AWS region where the bucket was created.

```
example: --repo-s3-region=us-east-1
```

S3 Repository Verify SSL Option (–repo-s3-verify-ssl) Verify S3 server certificate.

Disables verification of the S3 server certificate. This should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo-s3-verify-ssl
```

Repository Type Option (–repo-type) Type of storage used for the repository.

The following repository types are supported:

- cifs - Like posix , but disables links and directory fsyncs
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service

```
default: posix
example: --repo-type=cifs
```

Stanza Options

Database Host Option (–db-host) Cluster host for operating remotely via SSH.

Used for backups where the database cluster host is different from the backup host.

```
example: --db-host=db.domain.com
```

Database Path Option (–db-path) Cluster data directory.

This should be the same as the data_directory setting in postgresql.conf . Even though this value can be read from postgresql.conf or the database cluster it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

The db-path option is tested against the value reported by PostgreSQL on every online backup so it should always be current.

```
example: --db-path=/data/db
```

Database SSH Port Option (–db-ssh-port) Database server SSH port when db-host is set.

Use this option to specify a non-default SSH port for a database server.

```
example: --db-ssh-port=25
```

Backup Command (backup)

pgBackRest does not have a built-in scheduler so it's best to run it from cron or some other scheduling mechanism.

Command Options

Check Archive Option (–archive-check) Check that WAL segments are present in the archive before backup completes.

Checks that all WAL segments required to make the backup consistent are present in the WAL archive. It's a good idea to leave this as the default unless you are using another method for archiving.

This option must be enabled if archive-copy is enabled.

```
default: y
example: --no-archive-check
```

Copy Archive Option (–archive-copy) Copy WAL segments needed for consistency to the backup.

This slightly paranoid option protects against corruption in the WAL segment archive by storing the WAL segments required for consistency directly in the backup. WAL segments are still stored in the archive so this option will use additional space.

On restore, the WAL segments will be present in pg_xlog/pg_wal and PostgreSQL will use them in preference to calling the restore_command .

The archive-check option must be enabled if archive-copy is enabled.

```
default: n
example: --archive-copy
```

Archive Timeout Option (–archive-timeout) Archive timeout.

Set maximum time, in seconds, to wait for each WAL segment to reach the pgBackRest archive repository. The timeout applies to the check and backup commands when waiting for WAL segments required for backup consistency to be archived.

```
default: 60
allowed: 0.1-86400
example: --archive-timeout=30
```

Backup from Standby Option (–backup-standby) Backup from the standby cluster.

Enable backup from standby to reduce load on the primary cluster. This option requires that both the primary and standby hosts be configured.

```
default: n
example: --backup-standby
```

Page Checksums Option (–checksum-page) Validate data page checksums.

Directs pgBackRest to validate all data page checksums while backing up a cluster. This option will be automatically enabled when the required C library is present and checksums are enabled on the cluster.

Failures in checksum validation will not abort a backup. Rather, warnings will be emitted in the log (and to the console with default settings) and the list of invalid pages will be stored in the backup manifest.

```
example: --no-checksum-page
```

Force Option (–force) Force an offline backup.

When used with –no-start-stop a backup will be run even if pgBackRest thinks that PostgreSQL is running. **This option should be used with extreme care as it will likely result in a bad backup.**

There are some scenarios where a backup might still be desirable under these conditions. For example, if a server crashes and the database cluster volume can only be mounted read-only, it would be a good idea to take a backup even if postmaster.pid is present. In this case it would be better to revert to the prior backup and replay WAL, but possibly there is a very important transaction in a WAL segment that did not get archived.

```
default: n
example: --force
```

Hardlink Option (–hardlink) Hardlink files between backups.

Enable hard-linking of files in differential and incremental backups to their full backups. This gives the appearance that each backup is a full backup. Be careful, though, because modifying files that are hard-linked can affect all the backups in the set.

```
default: n
example: --hardlink
```

Manifest Save Threshold Option (–manifest-save-threshold) Manifest save threshold during backup.

Defines how often the manifest will be saved during a backup (in bytes). Saving the manifest is important because it stores the checksums and allows the resume function to work efficiently. The actual threshold used is 1% of the backup size or manifest-save-threshold , whichever is greater.

```
default: 1073741824
example: --manifest-save-threshold=5368709120
```

Online Option (–online) Perform an online backup.

Specifying –no-online prevents pgBackRest from running pg_start_backup() and pg_stop_backup() on the database cluster. In order for this to work PostgreSQL should be shut down and pgBackRest will generate an error if it is not.

The purpose of this option is to allow offline backups. The pg_xlog / pg_wal directory is copied as-is and archive-check is automatically disabled for the backup.

```
default: y
example: --no-online
```

Resume Option (–resume) Allow resume of failed backup.

Defines whether the resume feature is enabled. Resume can greatly reduce the amount of time required to run a backup after a previous backup of the same type has failed. It adds complexity, however, so it may be desirable to disable in environments that do not require the feature.

```
default: y
example: --no-resume
```

Start Fast Option (–start-fast) Force a checkpoint to start backup quickly.

Forces a checkpoint (by passing y to the fast parameter of pg_start_backup()) so the backup begins immediately. Otherwise the backup will start after the next regular checkpoint.

This feature only works in PostgreSQL >= 8.4 .

```
default: n
example: --start-fast
```

Stop Auto Option (–stop-auto) Stop prior failed backup on new backup.

This will only be done if an exclusive advisory lock can be acquired to demonstrate that the prior failed backup process has really stopped.

This feature relies on pg_is_in_backup() so only works on PostgreSQL >= 9.3 .

The setting is disabled by default because it assumes that pgBackRest is the only process doing exclusive online backups. It depends on an advisory lock that only pgBackRest sets so it may abort other processes that do exclusive online backups. Note that base_backup and pg_dump are safe to use with this setting because they do not call pg_start_backup() so are not exclusive.

```
default: n
example: --stop-auto
```

Type Option (–type) Backup type.

The following backup types are supported:

- full - all database cluster files will be copied and there will be no dependencies on previous backups.
- incr - incremental from the last successful backup.
- diff - like an incremental backup but always based on the last full backup.

```
default: incr
example: --type=full
```

Expire Options

Archive Retention Option (–retention-archive) Number of backups worth of continuous WAL to retain.

Note that the WAL segments required to make a backup consistent are always retained until the backup is expired regardless of how this option is configured.

If this value is not set, then the archive to expire will default to the retention-full (or retention-diff) value corresponding to the retention-archive-type if set to full (or diff). This will ensure that WAL is only expired for backups that are already expired.

This option must be set if retention-archive-type is set to incr . If disk space is at a premium, then this setting, in conjunction with retention-archive-type , can be used to aggressively expire WAL segments. However, doing so negates the ability to perform PITR from the backups with expired WAL and is therefore **not** recommended.

```
allowed: 1-9999999
example: --retention-archive=2
```

Archive Retention Type Option (–retention-archive-type) Backup type for WAL retention.

If set to full pgBackRest will keep archive logs for the number of full backups defined by retention-archive . If set to diff (differential) pgBackRest will keep archive logs for the number of full and differential backups defined by retention-archive , meaning if the last backup taken was a full backup, it will be counted as a differential for the purpose of retention. If set to incr (incremental) pgBackRest will keep archive logs for the number of full, differential, and incremental backups defined by retention-archive . It is recommended that this setting not be changed from the default which will only expire WAL in conjunction with expiring full backups.

```
default: full
example: --retention-archive-type=diff
```

Differential Retention Option (–retention-diff) Number of differential backups to retain.

When a differential backup expires, all incremental backups associated with the differential backup will also expire. When not defined all differential backups will be kept until the full backups they depend on expire.

```
allowed: 1-9999999
example: --retention-diff=3
```

Full Retention Option (–retention-full) Number of full backups to retain.

When a full backup expires, all differential and incremental backups associated with the full backup will also expire. When the option is not defined a warning will be issued. If indefinite retention is desired then set the option to the max value.

```
allowed: 1-9999999
example: --retention-full=2
```

General Options

Buffer Size Option (–buffer-size) Buffer size for file operations.

Set the buffer size used for copy, compress, and uncompress functions. A maximum of 3 buffers will be in use at a time per process. An additional maximum of 256K per process may be used for zlib buffers.

```
default: 4194304
example: --buffer-size=32768
```

SSH client command Option (–cmd-ssh) Path to ssh client executable.

Use a specific SSH client when an alternate is desired or the ssh executable is not in $PATH.

```
default: ssh
example: --cmd-ssh=/usr/bin/ssh
```

Compress Option (–compress) Use gzip file compression.

Backup files are compatible with command-line gzip tools.

```
default: y
example: --no-compress
```

Compress Level Option (–compress-level) Compression level for stored files.

Sets the zlib level to be used for file compression when compress=y .

```
default: 6
allowed: 0-9
example: --compress-level=9
```

Network Compress Level Option (–compress-level-network) Compression level for network transfer when compress=n .

Sets the zlib level to be used for protocol compression when compress=n and the database cluster is not on the same host as the backup. Protocol compression is used to reduce network traffic but can be disabled by setting compress-level-network=0 . When compress=y the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once. SSH compression is always disabled.

```
default: 3
allowed: 0-9
example: --compress-level-network=1
```

Config Option (–config) pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: /etc/pgbackrest.conf
example: --config=/var/lib/backrest/pgbackrest.conf
```

Database Timeout Option (–db-timeout) Database query timeout.

Sets the timeout, in seconds, for queries against the database. This includes the pg_start_backup() and pg_stop_backup() functions which can each take a substantial amount of time. Because of this the timeout should be kept high unless you know that these functions will return quickly (i.e. if you have set startfast=y and you know that the database cluster will not generate many WAL segments during the backup).

```
default: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

Lock Path Option (–lock-path) Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: --lock-path=/backup/db/lock
```

Neutral Umask Option (–neutral-umask) Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640. The lock and log directories set the directory and file mode to 0770 and 0660 respectively.

To use the executing user's umask instead specify neutral-umask=n in the config file or –no-neutral-umask on the command line.

```
default: y
example: --no-neutral-umask
```

Process Maximum Option (–process-max) Max processes to use for compress/transfer.

Each process will perform compression and transfer to make the command run faster, but don't set process-max so high that it impacts database performance.

```
default: 1
allowed: 1-96
example: --process-max=4
```

Protocol Timeout Option (–protocol-timeout) Protocol timeout.

Sets the timeout, in seconds, that the local or remote process will wait for a new message to be received on the protocol layer. This prevents processes from waiting indefinitely for a message. The protocol-timeout option must be greater than the db-timeout option.

```
default: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

Stanza Option (–stanza) Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one Postgres database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

```
example: --stanza=main
```

Log Options

Console Log Level Option (–log-level-console) Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-console=error
```

File Log Level Option (–log-level-file) Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: --log-level-file=debug
```

Std Error Log Level Option (–log-level-stderr) Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr .

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-stderr=error
```

Log Path Option (–log-path) Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=none then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Timestamp Option (–log-timestamp) Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Repository Options

Repository Cipher Passphrase Option (–repo-cipher-pass) Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

```
example: --repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

Repository Cipher Type Option (–repo-cipher-type) Cipher used to encrypt the repository.

The following repository types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

```
default: none
example: --repo-cipher-type=aes-256-cbc
```

Repository Path Option (–repo-path) Repository path where WAL segments and backups stored.

The repository is where pgBackRest stores backup and archives WAL segments.

If you are new to backup then it will be difficult to estimate in advance how much space you'll need. The best thing to do is take some backups then record the size of different types of backups (full/incr/diff) and measure the amount of WAL generated per day. This will give you a general idea of how much space you'll need, though of course requirements will likely change over time as your database evolves.

```
default: /var/lib/pgbackrest
example: --repo-path=/backup/db/backrest
```

S3 Repository Bucket Option (–repo-s3-bucket) S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo , so logs and other AWS generated content can also be stored in the bucket.

```
example: --repo-s3-bucket=db-backup
```

S3 SSL CA File Option (–repo-s3-ca-file) S3 SSL CA File.

Use a CA file other than the system default.

```
example: --repo-s3-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

S3 SSL CA Path Option (–repo-s3-ca-path) S3 SSL CA Path.

Use a CA path other than the system default.

```
example: --repo-s3-ca-path=/etc/pki/tls/certs
```

S3 Repository Endpoint Option (–repo-s3-endpoint) S3 repository endpoint.

The AWS end point should be valid for the selected region.

```
example: --repo-s3-endpoint=s3.amazonaws.com
```

S3 Repository Host Option (–repo-s3-host) S3 repository host.

Connect to a host other than the end point. This is typically used for testing.

```
example: --repo-s3-host=127.0.0.1
```

S3 Repository Access Key Option (–repo-s3-key) S3 repository access key.

AWS key used to access this bucket.

```
example: --repo-s3-key=AKIAIOSFODNN7EXAMPLE
```

S3 Repository Secret Access Key Option (–repo-s3-key-secret) S3 repository secret access key.

AWS secret key used to access this bucket.

```
example: --repo-s3-key-secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

S3 Repository Region Option (–repo-s3-region) S3 repository region.

The AWS region where the bucket was created.

```
example: --repo-s3-region=us-east-1
```

S3 Repository Verify SSL Option (–repo-s3-verify-ssl) Verify S3 server certificate.

Disables verification of the S3 server certificate. This should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo-s3-verify-ssl
```

Repository Type Option (–repo-type) Type of storage used for the repository.

The following repository types are supported:

- cifs - Like posix , but disables links and directory fsyncs
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service

```
default: posix
example: --repo-type=cifs
```

Stanza Options

Database Host Command Option (–db-cmd) pgBackRest exe path on the database host.

Required only if the path to pgbackrest is different on the local and database hosts. If not defined, the database host exe path will be set the same as the local exe path.

```
example: --db-cmd=/usr/lib/backrest/bin/pgbackrest
```

Database Host Configuration Option (–db-config) pgBackRest database host configuration file.

Sets the location of the configuration file on the database host. This is only required if the database host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --db-config=/etc/pgbackrest_db.conf
```

Database Host Option (–db-host) Cluster host for operating remotely via SSH.

Used for backups where the database cluster host is different from the backup host.

```
example: --db-host=db.domain.com
```

Database Path Option (–db-path) Cluster data directory.

This should be the same as the data_directory setting in postgresql.conf . Even though this value can be read from postgresql.conf or the database cluster it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

The db-path option is tested against the value reported by PostgreSQL on every online backup so it should always be current.

```
example: --db-path=/data/db
```

Database Port Option (–db-port) Cluster port.

Port that PostgreSQL is running on. This usually does not need to be specified as most database clusters run on the default port.

```
default: 5432
example: --db-port=6543
```

Database Socket Path Option (–db-socket-path) Cluster unix socket path.

The unix socket directory that was specified when PostgreSQL was started. pgBackRest will automatically look in the standard location for your OS so there usually no need to specify this setting unless the socket directory was explicitly modified with the unix_socket_directory setting in postgressql.conf .

```
example: --db-socket-path=/var/run/postgresql
```

Database SSH Port Option (–db-ssh-port) Database server SSH port when db-host is set.

Use this option to specify a non-default SSH port for a database server.

```
example: --db-ssh-port=25
```

Database User Option (–db-user) Cluster host logon user when db-host is set.

This user will also own the remote pgBackRest process and will initiate connections to PostgreSQL . For this to work correctly the user should be the PostgreSQL database cluster owner which is generally postgres , the default.

```
default: postgres
example: --db-user=db_owner
```

Check Command (check)

The check command validates that pgBackRest and the archive_command setting are configured correctly for archiving and backups. It detects misconfigurations, particularly in archiving, that result in incomplete backups because required WAL segments did not reach the archive. The command can be run on the database or the backup host. The command may also be run on the standby host, however, since pg_switch_xlog() / pg_switch_wal() cannot be performed on the standby, the command will only test the repository configuration.

Note that pg_create_restore_point('pgBackRest Archive Check') and pg_switch_xlog() / pg_switch_wal() are called to force PostgreSQL to archive a WAL segment. Restore points are only supported in PostgreSQL >= 9.1 so for older versions the check command may fail if there has been no write activity since the last log rotation, therefore it is recommended that activity be generated by the user if there have been no writes since the last WAL switch before running the check command.

Command Options

Check Archive Option (–archive-check) Check that WAL segments are present in the archive before backup completes.

Checks that all WAL segments required to make the backup consistent are present in the WAL archive. It's a good idea to leave this as the default unless you are using another method for archiving.

This option must be enabled if archive-copy is enabled.

```
default: y
example: --no-archive-check
```

Archive Timeout Option (–archive-timeout) Archive timeout.

Set maximum time, in seconds, to wait for each WAL segment to reach the pgBackRest archive repository. The timeout applies to the check and backup commands when waiting for WAL segments required for backup consistency to be archived.

```
default: 60
allowed: 0.1-86400
example: --archive-timeout=30
```

Backup from Standby Option (–backup-standby) Backup from the standby cluster.

Enable backup from standby to reduce load on the primary cluster. This option requires that both the primary and standby hosts be configured.

```
default: n
example: --backup-standby
```

Online Option (–online) Check an online cluster.

Specifying –no-online prevents pgBackRest from connecting to PostgreSQL and will disable some checks.

```
default: y
example: --no-online
```

General Options

Buffer Size Option (–buffer-size) Buffer size for file operations.

Set the buffer size used for copy, compress, and uncompress functions. A maximum of 3 buffers will be in use at a time per process. An additional maximum of 256K per process may be used for zlib buffers.

```
default: 4194304
example: --buffer-size=32768
```

SSH client command Option (–cmd-ssh) Path to ssh client executable.

Use a specific SSH client when an alternate is desired or the ssh executable is not in $PATH.

```
default: ssh
example: --cmd-ssh=/usr/bin/ssh
```

Compress Level Option (–compress-level) Compression level for stored files.

Sets the zlib level to be used for file compression when compress=y .

```
default: 6
allowed: 0-9
example: --compress-level=9
```

Network Compress Level Option (–compress-level-network) Compression level for network transfer when compress=n .

Sets the zlib level to be used for protocol compression when compress=n and the database cluster is not on the same host as the backup. Protocol compression is used to reduce network traffic but can be disabled by setting compress-level-network=0 . When compress=y the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once. SSH compression is always disabled.

```
default: 3
allowed: 0-9
example: --compress-level-network=1
```

Config Option (–config) pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: /etc/pgbackrest.conf
example: --config=/var/lib/backrest/pgbackrest.conf
```

Database Timeout Option (–db-timeout) Database query timeout.

Sets the timeout, in seconds, for queries against the database. This includes the pg_start_backup() and pg_stop_backup() functions which can each take a substantial amount of time. Because of this the timeout should be kept high unless you know that these functions will return quickly (i.e. if you have set startfast=y and you know that the database cluster will not generate many WAL segments during the backup).

```
default: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

Neutral Umask Option (–neutral-umask) Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640. The lock and log directories set the directory and file mode to 0770 and 0660 respectively.

To use the executing user's umask instead specify neutral-umask=n in the config file or –no-neutral-umask on the command line.

```
default: y
example: --no-neutral-umask
```

Protocol Timeout Option (–protocol-timeout) Protocol timeout.

Sets the timeout, in seconds, that the local or remote process will wait for a new message to be received on the protocol layer. This prevents processes from waiting indefinitely for a message. The protocol-timeout option must be greater than the db-timeout option.

```
default: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

Stanza Option (–stanza) Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one Postgres database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

```
example: --stanza=main
```

Log Options

Console Log Level Option (–log-level-console) Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-console=error
```

File Log Level Option (–log-level-file) Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: --log-level-file=debug
```

Std Error Log Level Option (–log-level-stderr) Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr .

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-stderr=error
```

Log Path Option (–log-path) Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=none then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Timestamp Option (–log-timestamp) Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Repository Options

Backup Host Command Option (–backup-cmd) pgBackRest exe path on the backup host.

Required only if the path to pgbackrest is different on the local and backup hosts. If not defined, the backup host exe path will be set the same as the local exe path.

```
example: --backup-cmd=/usr/lib/backrest/bin/pgbackrest
```

Backup Host Configuration Option (–backup-config) pgBackRest backup host configuration file.

Sets the location of the configuration file on the backup host. This is only required if the backup host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --backup-config=/etc/pgbackrest_backup.conf
```

Backup Host Option (–backup-host) Backup host when operating remotely via SSH.

Make sure that trusted SSH authentication is configured between the db host and the backup host.

When backing up to a locally mounted network filesystem this setting is not required.

```
example: --backup-host=backup.domain.com
```

Backup SSH Port Option (–backup-ssh-port) Backup server SSH port when backup-host is set.

Use this option to specify a non-default SSH port for the backup server.

```
example: --backup-ssh-port=25
```

Backup User Option (–backup-user) Backup host user when backup-host is set.

Defines the user that will be used for operations on the backup server. Preferably this is not the postgres user but rather some other user like backrest . If PostgreSQL runs on the backup server the postgres user can be placed in the backrest group so it has read permissions on the repository without being able to damage the contents accidentally.

```
default: backrest
example: --backup-user=backrest
```

Repository Cipher Passphrase Option (–repo-cipher-pass) Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

```
example: --repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

Repository Cipher Type Option (–repo-cipher-type) Cipher used to encrypt the repository.

The following repository types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

```
default: none
example: --repo-cipher-type=aes-256-cbc
```

Repository Path Option (–repo-path) Repository path where WAL segments and backups stored.

The repository is where pgBackRest stores backup and archives WAL segments.

If you are new to backup then it will be difficult to estimate in advance how much space you'll need. The best thing to do is take some backups then record the size of different types of backups (full/incr/diff) and measure the amount of WAL generated per day. This will give you a general idea of how much space you'll need, though of course requirements will likely change over time as your database evolves.

```
default: /var/lib/pgbackrest
example: --repo-path=/backup/db/backrest
```

S3 Repository Bucket Option (–repo-s3-bucket) S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo , so logs and other AWS generated content can also be stored in the bucket.

```
example: --repo-s3-bucket=db-backup
```

S3 SSL CA File Option (–repo-s3-ca-file) S3 SSL CA File.

Use a CA file other than the system default.

```
example: --repo-s3-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

S3 SSL CA Path Option (–repo-s3-ca-path) S3 SSL CA Path.

Use a CA path other than the system default.

```
example: --repo-s3-ca-path=/etc/pki/tls/certs
```

S3 Repository Endpoint Option (–repo-s3-endpoint) S3 repository endpoint.

The AWS end point should be valid for the selected region.

```
example: --repo-s3-endpoint=s3.amazonaws.com
```

S3 Repository Host Option (–repo-s3-host) S3 repository host.

Connect to a host other than the end point. This is typically used for testing.

```
example: --repo-s3-host=127.0.0.1
```

S3 Repository Access Key Option (–repo-s3-key) S3 repository access key.

AWS key used to access this bucket.

```
example: --repo-s3-key=AKIAIOSFODNN7EXAMPLE
```

S3 Repository Secret Access Key Option (–repo-s3-key-secret) S3 repository secret access key.

AWS secret key used to access this bucket.

```
example: --repo-s3-key-secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

S3 Repository Region Option (–repo-s3-region) S3 repository region.

The AWS region where the bucket was created.

```
example: --repo-s3-region=us-east-1
```

S3 Repository Verify SSL Option (–repo-s3-verify-ssl) Verify S3 server certificate.

Disables verification of the S3 server certificate. This should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo-s3-verify-ssl
```

Repository Type Option (–repo-type) Type of storage used for the repository.

The following repository types are supported:

- cifs - Like posix , but disables links and directory fsyncs
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service

```
default: posix
example: --repo-type=cifs
```

Stanza Options

Database Host Command Option (–db-cmd) pgBackRest exe path on the database host.

Required only if the path to pgbackrest is different on the local and database hosts. If not defined, the database host exe path will be set the same as the local exe path.

```
example: --db-cmd=/usr/lib/backrest/bin/pgbackrest
```

Database Host Configuration Option (–db-config) pgBackRest database host configuration file.

Sets the location of the configuration file on the database host. This is only required if the database host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --db-config=/etc/pgbackrest_db.conf
```

Database Host Option (–db-host) Cluster host for operating remotely via SSH.

Used for backups where the database cluster host is different from the backup host.

```
example: --db-host=db.domain.com
```

Database Path Option (–db-path) Cluster data directory.

This should be the same as the data_directory setting in postgresql.conf . Even though this value can be read from postgresql.conf or the database cluster it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

The db-path option is tested against the value reported by PostgreSQL on every online backup so it should always be current.

```
example: --db-path=/data/db
```

Database Port Option (–db-port) Cluster port.

Port that PostgreSQL is running on. This usually does not need to be specified as most database clusters run on the default port.

```
default: 5432
example: --db-port=6543
```

Database Socket Path Option (–db-socket-path) Cluster unix socket path.

The unix socket directory that was specified when PostgreSQL was started. pgBackRest will automatically look in the standard location for your OS so there usually no need to specify this setting unless the socket directory was explicitly modified with the unix_socket_directory setting in postgressql.conf .

```
example: --db-socket-path=/var/run/postgresql
```

Database SSH Port Option (–db-ssh-port) Database server SSH port when db-host is set.

Use this option to specify a non-default SSH port for a database server.

```
example: --db-ssh-port=25
```

Database User Option (–db-user) Cluster host logon user when db-host is set.

This user will also own the remote pgBackRest process and will initiate connections to PostgreSQL . For this to work correctly the user should be the PostgreSQL database cluster owner which is generally postgres , the default.

```
default: postgres
example: --db-user=db_owner
```

Expire Command (expire)

pgBackRest does backup rotation but is not concerned with when the backups were created. If two full backups are configured for retention, pgBackRest will keep two full backups no matter whether they occur two hours or two weeks apart.

Command Options

Archive Retention Option (–retention-archive) Number of backups worth of continuous WAL to retain.

Note that the WAL segments required to make a backup consistent are always retained until the backup is expired regardless of how this option is configured.

If this value is not set, then the archive to expire will default to the retention-full (or retention-diff) value corresponding to the retention-archive-type if set to full (or diff). This will ensure that WAL is only expired for backups that are already expired.

This option must be set if retention-archive-type is set to incr . If disk space is at a premium, then this setting, in conjunction with retention-archive-type , can be used to aggressively expire WAL segments. However, doing so negates the ability to perform PITR from the backups with expired WAL and is therefore **not** recommended.

```
allowed: 1-9999999
example: --retention-archive=2
```

Archive Retention Type Option (–retention-archive-type) Backup type for WAL retention.

If set to full pgBackRest will keep archive logs for the number of full backups defined by retention-archive . If set to diff (differential) pgBackRest will keep archive logs for the number of full and differential backups defined by retention-archive , meaning if the last backup taken was a full backup, it will be counted as a differential for the purpose of retention. If set to incr (incremental) pgBackRest will keep archive logs for the number of full, differential, and incremental backups defined by retention-archive . It is recommended that this setting not be changed from the default which will only expire WAL in conjunction with expiring full backups.

```
default: full
example: --retention-archive-type=diff
```

Differential Retention Option (–retention-diff) Number of differential backups to retain.

When a differential backup expires, all incremental backups associated with the differential backup will also expire. When not defined all differential backups will be kept until the full backups they depend on expire.

```
allowed: 1-9999999
example: --retention-diff=3
```

Full Retention Option (–retention-full) Number of full backups to retain.

When a full backup expires, all differential and incremental backups associated with the full backup will also expire. When the option is not defined a warning will be issued. If indefinite retention is desired then set the option to the max value.

```
allowed: 1-9999999
example: --retention-full=2
```

General Options

Buffer Size Option (–buffer-size) Buffer size for file operations.

Set the buffer size used for copy, compress, and uncompress functions. A maximum of 3 buffers will be in use at a time per process. An additional maximum of 256K per process may be used for zlib buffers.

```
default: 4194304
example: --buffer-size=32768
```

SSH client command Option (–cmd-ssh) Path to ssh client executable.

Use a specific SSH client when an alternate is desired or the ssh executable is not in $PATH.

```
default: ssh
example: --cmd-ssh=/usr/bin/ssh
```

Config Option (–config) pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: /etc/pgbackrest.conf
example: --config=/var/lib/backrest/pgbackrest.conf
```

Lock Path Option (–lock-path) Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: --lock-path=/backup/db/lock
```

Neutral Umask Option (–neutral-umask) Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640. The lock and log directories set the directory and file mode to 0770 and 0660 respectively.

To use the executing user's umask instead specify neutral-umask=n in the config file or –no-neutral-umask on the command line.

```
default: y
example: --no-neutral-umask
```

Stanza Option (–stanza) Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one Postgres database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

```
example: --stanza=main
```

Log Options

Console Log Level Option (–log-level-console) Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-console=error
```

File Log Level Option (–log-level-file) Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: --log-level-file=debug
```

Std Error Log Level Option (–log-level-stderr) Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr .

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-stderr=error
```

Log Path Option (–log-path) Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=none then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Timestamp Option (–log-timestamp) Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Repository Options

Repository Cipher Passphrase Option (–repo-cipher-pass) Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

```
example: --repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

Repository Cipher Type Option (–repo-cipher-type) Cipher used to encrypt the repository.

The following repository types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

```
default: none
example: --repo-cipher-type=aes-256-cbc
```

Repository Path Option (–repo-path) Repository path where WAL segments and backups stored.

The repository is where pgBackRest stores backup and archives WAL segments.

If you are new to backup then it will be difficult to estimate in advance how much space you'll need. The best thing to do is take some backups then record the size of different types of backups (full/incr/diff) and measure the amount of WAL generated per day. This will give you a general idea of how much space you'll need, though of course requirements will likely change over time as your database evolves.

```
default: /var/lib/pgbackrest
example: --repo-path=/backup/db/backrest
```

S3 Repository Bucket Option (–repo-s3-bucket) S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo , so logs and other AWS generated content can also be stored in the bucket.

```
example: --repo-s3-bucket=db-backup
```

S3 SSL CA File Option (–repo-s3-ca-file) S3 SSL CA File.

Use a CA file other than the system default.

```
example: --repo-s3-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

S3 SSL CA Path Option (–repo-s3-ca-path) S3 SSL CA Path.

Use a CA path other than the system default.

```
example: --repo-s3-ca-path=/etc/pki/tls/certs
```

S3 Repository Endpoint Option (–repo-s3-endpoint) S3 repository endpoint.

The AWS end point should be valid for the selected region.

```
example: --repo-s3-endpoint=s3.amazonaws.com
```

S3 Repository Host Option (–repo-s3-host) S3 repository host.

Connect to a host other than the end point. This is typically used for testing.

```
example: --repo-s3-host=127.0.0.1
```

S3 Repository Access Key Option (–repo-s3-key) S3 repository access key.

AWS key used to access this bucket.

```
example: --repo-s3-key=AKIAIOSFODNN7EXAMPLE
```

S3 Repository Secret Access Key Option (–repo-s3-key-secret) S3 repository secret access key.

AWS secret key used to access this bucket.

```
example: --repo-s3-key-secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

S3 Repository Region Option (–repo-s3-region) S3 repository region.

The AWS region where the bucket was created.

```
example: --repo-s3-region=us-east-1
```

S3 Repository Verify SSL Option (–repo-s3-verify-ssl) Verify S3 server certificate.

Disables verification of the S3 server certificate. This should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo-s3-verify-ssl
```

Repository Type Option (–repo-type) Type of storage used for the repository.

The following repository types are supported:

- cifs - Like posix , but disables links and directory fsyncs
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service

```
default: posix
example: --repo-type=cifs
```

Stanza Options

Database Host Command Option (–db-cmd) pgBackRest exe path on the database host.

Required only if the path to pgbackrest is different on the local and database hosts. If not defined, the database host exe path will be set the same as the local exe path.

```
example: --db-cmd=/usr/lib/backrest/bin/pgbackrest
```

Database Host Configuration Option (–db-config) pgBackRest database host configuration file.

Sets the location of the configuration file on the database host. This is only required if the database host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --db-config=/etc/pgbackrest_db.conf
```

Database Host Option (–db-host) Cluster host for operating remotely via SSH.

Used for backups where the database cluster host is different from the backup host.

```
example: --db-host=db.domain.com
```

Database SSH Port Option (–db-ssh-port) Database server SSH port when db-host is set.

Use this option to specify a non-default SSH port for a database server.

```
example: --db-ssh-port=25
```

Help Command (help)

Three levels of help are provided. If no command is specified then general help will be displayed. If a command is specified then a full description of the command will be displayed along with a list of valid options. If an option is specified in addition to a command then the a full description of the option as it applies to the command will be displayed.

Info Command (info)

The info command operates on a single stanza or all stanzas. Text output is the default and gives a human-readable summary of backups for the stanza(s) requested. This format is subject to change with any release.

For machine-readable output use –output=json . The JSON output contains far more information than the text output, however **this feature is currently experimental so the format may change between versions** .

Command Options

Output Option (–output) Output format.

The following output types are supported:

- text - Human-readable summary of backup information.
- json - Exhaustive machine-readable backup information in JSON format.

```
default: text
example: --output=json
```

General Options

Buffer Size Option (–buffer-size) Buffer size for file operations.

Set the buffer size used for copy, compress, and uncompress functions. A maximum of 3 buffers will be in use at a time per process. An additional maximum of 256K per process may be used for zlib buffers.

```
default: 4194304
example: --buffer-size=32768
```

SSH client command Option (–cmd-ssh) Path to ssh client executable.

Use a specific SSH client when an alternate is desired or the ssh executable is not in $PATH.

```
default: ssh
example: --cmd-ssh=/usr/bin/ssh
```

Compress Level Option (–compress-level) Compression level for stored files.

Sets the zlib level to be used for file compression when compress=y .

```
default: 6
allowed: 0-9
example: --compress-level=9
```

Network Compress Level Option (–compress-level-network) Compression level for network transfer when compress=n .

Sets the zlib level to be used for protocol compression when compress=n and the database cluster is not on the same host as the backup. Protocol compression is used to reduce network traffic but can be disabled by setting compress-level-network=0 . When compress=y the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once. SSH compression is always disabled.

```
default: 3
allowed: 0-9
example: --compress-level-network=1
```

Config Option (–config) pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: /etc/pgbackrest.conf
example: --config=/var/lib/backrest/pgbackrest.conf
```

Lock Path Option (–lock-path) Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: --lock-path=/backup/db/lock
```

Protocol Timeout Option (–protocol-timeout) Protocol timeout.

Sets the timeout, in seconds, that the local or remote process will wait for a new message to be received on the protocol layer. This prevents processes from waiting indefinitely for a message. The protocol-timeout option must be greater than the db-timeout option.

```
default: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

Stanza Option (–stanza) Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one Postgres database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

```
example: --stanza=main
```

Log Options

Console Log Level Option (–log-level-console) Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-console=error
```

File Log Level Option (–log-level-file) Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: --log-level-file=debug
```

Std Error Log Level Option (–log-level-stderr) Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr .

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-stderr=error
```

Log Path Option (–log-path) Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=none then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Timestamp Option (–log-timestamp) Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Repository Options

Backup Host Command Option (–backup-cmd) pgBackRest exe path on the backup host.

Required only if the path to pgbackrest is different on the local and backup hosts. If not defined, the backup host exe path will be set the same as the local exe path.

```
example: --backup-cmd=/usr/lib/backrest/bin/pgbackrest
```

Backup Host Configuration Option (–backup-config) pgBackRest backup host configuration file.

Sets the location of the configuration file on the backup host. This is only required if the backup host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --backup-config=/etc/pgbackrest_backup.conf
```

Backup Host Option (–backup-host) Backup host when operating remotely via SSH.

Make sure that trusted SSH authentication is configured between the db host and the backup host.

When backing up to a locally mounted network filesystem this setting is not required.

```
example: --backup-host=backup.domain.com
```

Backup SSH Port Option (–backup-ssh-port) Backup server SSH port when backup-host is set.

Use this option to specify a non-default SSH port for the backup server.

```
example: --backup-ssh-port=25
```

Backup User Option (–backup-user) Backup host user when backup-host is set.

Defines the user that will be used for operations on the backup server. Preferably this is not the postgres user but rather some other user like backrest . If PostgreSQL runs on the backup server the postgres user can be placed in the backrest group so it has read permissions on the repository without being able to damage the contents accidentally.

```
default: backrest
example: --backup-user=backrest
```

Repository Cipher Passphrase Option (–repo-cipher-pass) Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

```
example: --repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

Repository Cipher Type Option (–repo-cipher-type) Cipher used to encrypt the repository.

The following repository types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

```
default: none
example: --repo-cipher-type=aes-256-cbc
```

Repository Path Option (–repo-path) Repository path where WAL segments and backups stored.

The repository is where pgBackRest stores backup and archives WAL segments.

If you are new to backup then it will be difficult to estimate in advance how much space you'll need. The best thing to do is take some backups then record the size of different types of backups (full/incr/diff) and measure the amount of WAL generated per day. This will give you a general idea of how much space you'll need, though of course requirements will likely change over time as your database evolves.

```
default: /var/lib/pgbackrest
example: --repo-path=/backup/db/backrest
```

S3 Repository Bucket Option (–repo-s3-bucket) S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo , so logs and other AWS generated content can also be stored in the bucket.

```
example: --repo-s3-bucket=db-backup
```

S3 SSL CA File Option (–repo-s3-ca-file) S3 SSL CA File.

Use a CA file other than the system default.

```
example: --repo-s3-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

S3 SSL CA Path Option (–repo-s3-ca-path) S3 SSL CA Path.

Use a CA path other than the system default.

```
example: --repo-s3-ca-path=/etc/pki/tls/certs
```

S3 Repository Endpoint Option (–repo-s3-endpoint) S3 repository endpoint.

The AWS end point should be valid for the selected region.

```
example: --repo-s3-endpoint=s3.amazonaws.com
```

S3 Repository Host Option (–repo-s3-host) S3 repository host.

Connect to a host other than the end point. This is typically used for testing.

```
example: --repo-s3-host=127.0.0.1
```

S3 Repository Access Key Option (–repo-s3-key) S3 repository access key.

AWS key used to access this bucket.

```
example: --repo-s3-key=AKIAIOSFODNN7EXAMPLE
```

S3 Repository Secret Access Key Option (–repo-s3-key-secret) S3 repository secret access key.

AWS secret key used to access this bucket.

```
example: --repo-s3-key-secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

S3 Repository Region Option (–repo-s3-region) S3 repository region.

The AWS region where the bucket was created.

```
example: --repo-s3-region=us-east-1
```

S3 Repository Verify SSL Option (–repo-s3-verify-ssl) Verify S3 server certificate.

Disables verification of the S3 server certificate. This should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo-s3-verify-ssl
```

Repository Type Option (–repo-type) Type of storage used for the repository.

The following repository types are supported:

- cifs - Like posix , but disables links and directory fsyncs
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service

```
default: posix
example: --repo-type=cifs
```

Restore Command (restore)

This command is generally run manually, but there are instances where it might be automated.

Command Options

Include Database Option (–db-include) Restore only specified databases.

This feature allows only selected databases to be restored. Databases not specifically included will be restored as sparse, zeroed files to save space but still allow PostgreSQL to perform recovery. After recovery the databases that were not included will not be accessible but can be removed with the drop database command.

Note that built-in databases (template0 , template1 , and postgres) are always restored.

The –db-include option can be passed multiple times to specify more than one database to include.

```
example: --db-include=db_main
```

Delta Option (–delta) Restore using delta.

By default the PostgreSQL data and tablespace directories are expected to be present but empty. This option performs a delta restore using checksums.

```
default: n
example: --delta
```

Force Option (–force) Force a restore.

By itself this option forces the PostgreSQL data and tablespace paths to be completely overwritten. In combination with –delta a timestamp/size delta will be performed instead of using checksums.

```
default: n
example: --force
```

Link All Option (–link-all) Restore all symlinks.

By default symlinked directories and files are restored as normal directories and files in $PGDATA. This is because it may not be safe to restore symlinks to their original destinations on a system other than where the original backup was performed. This option restores all the symlinks just as they were on the original system where the backup was performed.

```
default: n
example: --link-all
```

Link Map Option (–link-map) Modify the destination of a symlink.

Allows the destination file or path of a symlink to be changed on restore. This is useful for restoring to systems that have a different storage layout than the original system where the backup was generated.

```
example: --link-map=pg_xlog=/data/xlog
```

Recovery Option Option (–recovery-option) Set an option in recovery.conf .

See http://www.postgresql.org/docs/X.X/static/recovery-config.html for details on recovery.conf options (replace X.X with your PostgreSQL version). This option can be used multiple times.

Note: The restore_command option will be automatically generated but can be overridden with this option. Be careful about specifying your own restore_command as pgBackRest is designed to handle this for you. Target Recovery options (recovery_target_name, recovery_target_time, etc.) are generated automatically by pgBackRest and should not be set with this option.

Since pgBackRest does not start PostgreSQL after writing the recovery.conf file, it is always possible to edit/check recovery.conf before manually restarting.

```
example: --recovery-option=primary_conninfo=db.mydomain.com
```

Set Option (–set) Backup set to restore.

The backup set to be restored. latest will restore the latest backup, otherwise provide the name of the backup to restore.

```
default: latest
example: --set=20150131-153358F_20150131-153401I
```

Tablespace Map Option (–tablespace-map) Restore a tablespace into the specified directory.

Moves a tablespace to a new location during the restore. This is useful when tablespace locations are not the same on a replica, or an upgraded system has different mount points.

Since PostgreSQL 9.2 tablespace locations are not stored in pg_tablespace so moving tablespaces can be done with impunity. However, moving a tablespace to the data_directory is not recommended and may cause problems. For more information on moving tablespaces http://www.databasesoup.com/2013/11/moving-tablespaces.html is a good resource.

```
example: --tablespace-map=ts_01=/db/ts_01
```

Map All Tablespaces Option (–tablespace-map-all) Restore all tablespaces into the specified directory.

By default tablespaces are restored into their original locations and while this behavior can be modified by with the tablespace-map open it is sometime preferable to remap all tablespaces to a new directory all at once. This is particularly useful for development or staging systems that may not have the same storage layout as the original system where the backup was generated.

The path specified will be the parent path used to create all the tablespaces in the backup.

```
example: --tablespace-map-all=/data/tablespace
```

Target Option (–target) Recovery target.

Defines the recovery target when –type is name , xid , or time .

```
example: --target=2015-01-30 14:15:11 EST
```

Target Action Option (–target-action) Action to take when recovery target is reached.

The following actions are supported:

- pause - pause when recovery target is reached.
- promote - promote and switch timeline when recovery target is reached.
- shutdown - shutdown server when recovery target is reached.

This option is only supported on PostgreSQL >= 9.5.

```
default: pause
example: --target-action=promote
```

Target Exclusive Option (–target-exclusive) Stop just before the recovery target is reached.

Defines whether recovery to the target would be exclusive (the default is inclusive) and is only valid when –type is time or xid . For example, using –target-exclusive would exclude the contents of transaction 1007 when –type=xid and –target=1007 . See the recovery_target_inclusive option in the PostgreSQL docs for more information.

```
default: n
example: --no-target-exclusive
```

Target Timeline Option (–target-timeline) Recover along a timeline.

See recovery_target_timeline in the PostgreSQL docs for more information.

```
example: --target-timeline=3
```

Type Option (–type) Recovery type.

The following recovery types are supported:

- default - recover to the end of the archive stream.
- immediate - recover only until the database becomes consistent. This option is only supported on PostgreSQL >= 9.4.
- name - recover the restore point specified in –target .
- xid - recover to the transaction id specified in –target .
- time - recover to the time specified in –target .
- preserve - preserve the existing recovery.conf file.
- none - no recovery.conf file is written so PostgreSQL will attempt to achieve consistency using WAL segments present in pg_xlog / pg_wal . Provide the required WAL segments or use the archive-copy setting to include them with the backup.

```
default: default
example: --type=xid
```

General Options

Buffer Size Option (–buffer-size) Buffer size for file operations.

Set the buffer size used for copy, compress, and uncompress functions. A maximum of 3 buffers will be in use at a time per process. An additional maximum of 256K per process may be used for zlib buffers.

```
default: 4194304
example: --buffer-size=32768
```

SSH client command Option (–cmd-ssh) Path to ssh client executable.

Use a specific SSH client when an alternate is desired or the ssh executable is not in $PATH.

```
default: ssh
example: --cmd-ssh=/usr/bin/ssh
```

Compress Option (–compress) Use gzip file compression.

Backup files are compatible with command-line gzip tools.

```
default: y
example: --no-compress
```

Compress Level Option (–compress-level) Compression level for stored files.

Sets the zlib level to be used for file compression when compress=y .

```
default: 6
allowed: 0-9
example: --compress-level=9
```

Network Compress Level Option (–compress-level-network) Compression level for network transfer when compress=n .

Sets the zlib level to be used for protocol compression when compress=n and the database cluster is not on the same host as the backup. Protocol compression is used to reduce network traffic but can be disabled by setting compress-level-network=0 . When compress=y the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once. SSH compression is always disabled.

```
default: 3
allowed: 0-9
example: --compress-level-network=1
```

Config Option (–config) pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: /etc/pgbackrest.conf
example: --config=/var/lib/backrest/pgbackrest.conf
```

Lock Path Option (–lock-path) Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: --lock-path=/backup/db/lock
```

Neutral Umask Option (–neutral-umask) Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640. The lock and log directories set the directory and file mode to 0770 and 0660 respectively.

To use the executing user's umask instead specify neutral-umask=n in the config file or –no-neutral-umask on the command line.

```
default: y
example: --no-neutral-umask
```

Process Maximum Option (–process-max) Max processes to use for compress/transfer.

Each process will perform compression and transfer to make the command run faster, but don't set process-max so high that it impacts database performance.

```
default: 1
allowed: 1-96
example: --process-max=4
```

Protocol Timeout Option (–protocol-timeout) Protocol timeout.

Sets the timeout, in seconds, that the local or remote process will wait for a new message to be received on the protocol layer. This prevents processes from waiting indefinitely for a message. The protocol-timeout option must be greater than the db-timeout option.

```
default: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

Stanza Option (–stanza) Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one Postgres database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

```
example: --stanza=main
```

Log Options

Console Log Level Option (–log-level-console) Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-console=error
```

File Log Level Option (–log-level-file) Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: --log-level-file=debug
```

Std Error Log Level Option (–log-level-stderr) Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr .

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-stderr=error
```

Log Path Option (–log-path) Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=none then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Timestamp Option (–log-timestamp) Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Repository Options

Backup Host Command Option (–backup-cmd) pgBackRest exe path on the backup host.

Required only if the path to pgbackrest is different on the local and backup hosts. If not defined, the backup host exe path will be set the same as the local exe path.

```
example: --backup-cmd=/usr/lib/backrest/bin/pgbackrest
```

Backup Host Configuration Option (–backup-config) pgBackRest backup host configuration file.

Sets the location of the configuration file on the backup host. This is only required if the backup host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --backup-config=/etc/pgbackrest_backup.conf
```

Backup Host Option (–backup-host) Backup host when operating remotely via SSH.

Make sure that trusted SSH authentication is configured between the db host and the backup host.

When backing up to a locally mounted network filesystem this setting is not required.

```
example: --backup-host=backup.domain.com
```

Backup SSH Port Option (–backup-ssh-port) Backup server SSH port when backup-host is set.

Use this option to specify a non-default SSH port for the backup server.

```
example: --backup-ssh-port=25
```

Backup User Option (–backup-user) Backup host user when backup-host is set.

Defines the user that will be used for operations on the backup server. Preferably this is not the postgres user but rather some other user like backrest . If PostgreSQL runs on the backup server the postgres user can be placed in the backrest group so it has read permissions on the repository without being able to damage the contents accidentally.

```
default: backrest
example: --backup-user=backrest
```

Repository Cipher Passphrase Option (–repo-cipher-pass) Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

```
example: --repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

Repository Cipher Type Option (–repo-cipher-type) Cipher used to encrypt the repository.

The following repository types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

```
default: none
example: --repo-cipher-type=aes-256-cbc
```

Repository Path Option (–repo-path) Repository path where WAL segments and backups stored.

The repository is where pgBackRest stores backup and archives WAL segments.

If you are new to backup then it will be difficult to estimate in advance how much space you'll need. The best thing to do is take some backups then record the size of different types of backups (full/incr/diff) and measure the amount of WAL generated per day. This will give you a general idea of how much space you'll need, though of course requirements will likely change over time as your database evolves.

```
default: /var/lib/pgbackrest
example: --repo-path=/backup/db/backrest
```

S3 Repository Bucket Option (–repo-s3-bucket) S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo , so logs and other AWS generated content can also be stored in the bucket.

```
example: --repo-s3-bucket=db-backup
```

S3 SSL CA File Option (–repo-s3-ca-file) S3 SSL CA File.

Use a CA file other than the system default.

```
example: --repo-s3-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

S3 SSL CA Path Option (–repo-s3-ca-path) S3 SSL CA Path.

Use a CA path other than the system default.

```
example: --repo-s3-ca-path=/etc/pki/tls/certs
```

S3 Repository Endpoint Option (–repo-s3-endpoint) S3 repository endpoint.

The AWS end point should be valid for the selected region.

```
example: --repo-s3-endpoint=s3.amazonaws.com
```

S3 Repository Host Option (–repo-s3-host) S3 repository host.

Connect to a host other than the end point. This is typically used for testing.

```
example: --repo-s3-host=127.0.0.1
```

S3 Repository Access Key Option (–repo-s3-key) S3 repository access key.

AWS key used to access this bucket.

```
example: --repo-s3-key=AKIAIOSFODNN7EXAMPLE
```

S3 Repository Secret Access Key Option (–repo-s3-key-secret) S3 repository secret access key.

AWS secret key used to access this bucket.

```
example: --repo-s3-key-secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

S3 Repository Region Option (–repo-s3-region) S3 repository region.

The AWS region where the bucket was created.

```
example: --repo-s3-region=us-east-1
```

S3 Repository Verify SSL Option (–repo-s3-verify-ssl) Verify S3 server certificate.

Disables verification of the S3 server certificate. This should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo-s3-verify-ssl
```

Repository Type Option (–repo-type) Type of storage used for the repository.

The following repository types are supported:

- cifs - Like posix , but disables links and directory fsyncs
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service

```
default: posix
example: --repo-type=cifs
```

Stanza Options

Database Path Option (–db-path) Cluster data directory.

This should be the same as the data_directory setting in postgresql.conf . Even though this value can be read from postgresql.conf or the database cluster it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

The db-path option is tested against the value reported by PostgreSQL on every online backup so it should always be current.

```
example: --db-path=/data/db
```

Stanza Create Command (stanza-create)

The stanza-create command must be run on the host where the repository is located after the stanza has been configured in pgbackrest.conf .

Command Options

Backup from Standby Option (–backup-standby) Backup from the standby cluster.

Enable backup from standby to reduce load on the primary cluster. This option requires that both the primary and standby hosts be configured.

```
default: n
example: --backup-standby
```

Force Option (–force) Force stanza creation.

Caution : Use –force only as a last resort, when all else fails. If data is missing from the repository then the recreated .info files will likely be corrupt.

If the required stanza .info files do not exist in the repository but backups or WAL segments do exist, then this option can be used to force the stanza to be created from the existing data in the repository. This is most likely to be useful after corruption or an incomplete restore of the repository from elsewhere.

```
default: n
example: --no-force
```

Online Option (–online) Create on an online cluster.

Specifying –no-online prevents pgBackRest from connecting to PostgreSQL when creating the stanza.

```
default: y
example: --no-online
```

General Options

Buffer Size Option (–buffer-size) Buffer size for file operations.

Set the buffer size used for copy, compress, and uncompress functions. A maximum of 3 buffers will be in use at a time per process. An additional maximum of 256K per process may be used for zlib buffers.

```
default: 4194304
example: --buffer-size=32768
```

SSH client command Option (–cmd-ssh) Path to ssh client executable.

Use a specific SSH client when an alternate is desired or the ssh executable is not in $PATH.

```
default: ssh
example: --cmd-ssh=/usr/bin/ssh
```

Compress Level Option (–compress-level) Compression level for stored files.

Sets the zlib level to be used for file compression when compress=y .

```
default: 6
allowed: 0-9
example: --compress-level=9
```

Network Compress Level Option (–compress-level-network) Compression level for network transfer when compress=n .

Sets the zlib level to be used for protocol compression when compress=n and the database cluster is not on the same host as the backup. Protocol compression is used to reduce network traffic but can be disabled by setting compress-level-network=0 . When compress=y the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once. SSH compression is always disabled.

```
default: 3
allowed: 0-9
example: --compress-level-network=1
```

Config Option (–config) pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: /etc/pgbackrest.conf
example: --config=/var/lib/backrest/pgbackrest.conf
```

Database Timeout Option (–db-timeout) Database query timeout.

Sets the timeout, in seconds, for queries against the database. This includes the pg_start_backup() and pg_stop_backup() functions which can each take a substantial amount of time. Because of this the timeout should be kept high unless you know that these functions will return quickly (i.e. if you have set startfast=y and you know that the database cluster will not generate many WAL segments during the backup).

```
default: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

Lock Path Option (–lock-path) Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: --lock-path=/backup/db/lock
```

Neutral Umask Option (–neutral-umask) Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640. The lock and log directories set the directory and file mode to 0770 and 0660 respectively.

To use the executing user's umask instead specify neutral-umask=n in the config file or –no-neutral-umask on the command line.

```
default: y
example: --no-neutral-umask
```

Protocol Timeout Option (–protocol-timeout) Protocol timeout.

Sets the timeout, in seconds, that the local or remote process will wait for a new message to be received on the protocol layer. This prevents processes from waiting indefinitely for a message. The protocol-timeout option must be greater than the db-timeout option.

```
default: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

Stanza Option (–stanza) Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one Postgres database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

```
example: --stanza=main
```

Log Options

Console Log Level Option (–log-level-console) Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-console=error
```

File Log Level Option (–log-level-file) Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: --log-level-file=debug
```

Std Error Log Level Option (–log-level-stderr) Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr .

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-stderr=error
```

Log Path Option (–log-path) Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=none then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Timestamp Option (–log-timestamp) Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Repository Options

Backup Host Command Option (–backup-cmd) pgBackRest exe path on the backup host.

Required only if the path to pgbackrest is different on the local and backup hosts. If not defined, the backup host exe path will be set the same as the local exe path.

```
example: --backup-cmd=/usr/lib/backrest/bin/pgbackrest
```

Backup Host Configuration Option (–backup-config) pgBackRest backup host configuration file.

Sets the location of the configuration file on the backup host. This is only required if the backup host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --backup-config=/etc/pgbackrest_backup.conf
```

Backup Host Option (–backup-host) Backup host when operating remotely via SSH.

Make sure that trusted SSH authentication is configured between the db host and the backup host.

When backing up to a locally mounted network filesystem this setting is not required.

```
example: --backup-host=backup.domain.com
```

Backup SSH Port Option (–backup-ssh-port) Backup server SSH port when backup-host is set.

Use this option to specify a non-default SSH port for the backup server.

```
example: --backup-ssh-port=25
```

Backup User Option (–backup-user) Backup host user when backup-host is set.

Defines the user that will be used for operations on the backup server. Preferably this is not the postgres user but rather some other user like backrest . If PostgreSQL runs on the backup server the postgres user can be placed in the backrest group so it has read permissions on the repository without being able to damage the contents accidentally.

```
default: backrest
example: --backup-user=backrest
```

Repository Cipher Passphrase Option (–repo-cipher-pass) Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

```
example: --repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

Repository Cipher Type Option (–repo-cipher-type) Cipher used to encrypt the repository.

The following repository types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

```
default: none
example: --repo-cipher-type=aes-256-cbc
```

Repository Path Option (–repo-path) Repository path where WAL segments and backups stored.

The repository is where pgBackRest stores backup and archives WAL segments.

If you are new to backup then it will be difficult to estimate in advance how much space you'll need. The best thing to do is take some backups then record the size of different types of backups (full/incr/diff) and measure the amount of WAL generated per day. This will give you a general idea of how much space you'll need, though of course requirements will likely change over time as your database evolves.

```
default: /var/lib/pgbackrest
example: --repo-path=/backup/db/backrest
```

S3 Repository Bucket Option (–repo-s3-bucket) S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo , so logs and other AWS generated content can also be stored in the bucket.

```
example: --repo-s3-bucket=db-backup
```

S3 SSL CA File Option (–repo-s3-ca-file) S3 SSL CA File.

Use a CA file other than the system default.

```
example: --repo-s3-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

S3 SSL CA Path Option (–repo-s3-ca-path) S3 SSL CA Path.

Use a CA path other than the system default.

```
example: --repo-s3-ca-path=/etc/pki/tls/certs
```

S3 Repository Endpoint Option (–repo-s3-endpoint) S3 repository endpoint.

The AWS end point should be valid for the selected region.

```
example: --repo-s3-endpoint=s3.amazonaws.com
```

S3 Repository Host Option (–repo-s3-host) S3 repository host.

Connect to a host other than the end point. This is typically used for testing.

```
example: --repo-s3-host=127.0.0.1
```

S3 Repository Access Key Option (–repo-s3-key) S3 repository access key.

AWS key used to access this bucket.

```
example: --repo-s3-key=AKIAIOSFODNN7EXAMPLE
```

S3 Repository Secret Access Key Option (–repo-s3-key-secret) S3 repository secret access key.

AWS secret key used to access this bucket.

```
example: --repo-s3-key-secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

S3 Repository Region Option (–repo-s3-region) S3 repository region.

The AWS region where the bucket was created.

```
example: --repo-s3-region=us-east-1
```

S3 Repository Verify SSL Option (–repo-s3-verify-ssl) Verify S3 server certificate.

Disables verification of the S3 server certificate. This should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo-s3-verify-ssl
```

Repository Type Option (–repo-type) Type of storage used for the repository.

The following repository types are supported:

- cifs - Like posix , but disables links and directory fsyncs
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service

```
default: posix
example: --repo-type=cifs
```

Stanza Options

Database Host Command Option (–db-cmd) pgBackRest exe path on the database host.

Required only if the path to pgbackrest is different on the local and database hosts. If not defined, the database host exe path will be set the same as the local exe path.

```
example: --db-cmd=/usr/lib/backrest/bin/pgbackrest
```

Database Host Configuration Option (–db-config) pgBackRest database host configuration file.

Sets the location of the configuration file on the database host. This is only required if the database host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --db-config=/etc/pgbackrest_db.conf
```

Database Host Option (–db-host) Cluster host for operating remotely via SSH.

Used for backups where the database cluster host is different from the backup host.

```
example: --db-host=db.domain.com
```

Database Path Option (–db-path) Cluster data directory.

This should be the same as the data_directory setting in postgresql.conf . Even though this value can be read from postgresql.conf or the database cluster it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

The db-path option is tested against the value reported by PostgreSQL on every online backup so it should always be current.

```
example: --db-path=/data/db
```

Database Port Option (–db-port) Cluster port.

Port that PostgreSQL is running on. This usually does not need to be specified as most database clusters run on the default port.

```
default: 5432
example: --db-port=6543
```

Database Socket Path Option (–db-socket-path) Cluster unix socket path.

The unix socket directory that was specified when PostgreSQL was started. pgBackRest will automatically look in the standard location for your OS so there usually no need to specify this setting unless the socket directory was explicitly modified with the unix_socket_directory setting in postgressql.conf .

```
example: --db-socket-path=/var/run/postgresql
```

Database SSH Port Option (–db-ssh-port) Database server SSH port when db-host is set.

Use this option to specify a non-default SSH port for a database server.

```
example: --db-ssh-port=25
```

Database User Option (–db-user) Cluster host logon user when db-host is set.

This user will also own the remote pgBackRest process and will initiate connections to PostgreSQL . For this to work correctly the user should be the PostgreSQL database cluster owner which is generally postgres , the default.

```
default: postgres
example: --db-user=db_owner
```

Stanza Delete Command (stanza-delete)

The stanza-delete command removes data in the repository associated with a stanza. Use this command with caution — it will permanently remove all backups and archives from the pgBackRest repository for the specified stanza.

To delete a stanza:

- Shut down the PostgreSQL cluster associated with the stanza (or use –force to override).
- Run the stop command on the backup host (the host where the repository is mounted).
- Run the stanza-delete command on the backup host.

Once the command successfully completes, it is the responsibility of the user to remove the stanza from all pgBackRest configuration files.

Command Options

Force Option (–force) Force stanza delete.

If PostgreSQL is still running for the stanza, then this option can be used to force the stanza to be deleted from the repository.

```
default: n
example: --no-force
```

General Options

Buffer Size Option (–buffer-size) Buffer size for file operations.

Set the buffer size used for copy, compress, and uncompress functions. A maximum of 3 buffers will be in use at a time per process. An additional maximum of 256K per process may be used for zlib buffers.

```
default: 4194304
example: --buffer-size=32768
```

SSH client command Option (–cmd-ssh) Path to ssh client executable.

Use a specific SSH client when an alternate is desired or the ssh executable is not in $PATH.

```
default: ssh
example: --cmd-ssh=/usr/bin/ssh
```

Compress Level Option (–compress-level) Compression level for stored files.

Sets the zlib level to be used for file compression when compress=y .

```
default: 6
allowed: 0-9
example: --compress-level=9
```

Network Compress Level Option (–compress-level-network) Compression level for network transfer when compress=n .

Sets the zlib level to be used for protocol compression when compress=n and the database cluster is not on the same host as the backup. Protocol compression is used to reduce network traffic but can be disabled by setting compress-level-network=0 . When compress=y the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once. SSH compression is always disabled.

```
default: 3
allowed: 0-9
example: --compress-level-network=1
```

Config Option (–config) pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: /etc/pgbackrest.conf
example: --config=/var/lib/backrest/pgbackrest.conf
```

Database Timeout Option (–db-timeout) Database query timeout.

Sets the timeout, in seconds, for queries against the database. This includes the pg_start_backup() and pg_stop_backup() functions which can each take a substantial amount of time. Because of this the timeout should be kept high unless you know that these functions will return quickly (i.e. if you have set startfast=y and you know that the database cluster will not generate many WAL segments during the backup).

```
default: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

Lock Path Option (–lock-path) Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: --lock-path=/backup/db/lock
```

Neutral Umask Option (–neutral-umask) Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640. The lock and log directories set the directory and file mode to 0770 and 0660 respectively.

To use the executing user's umask instead specify neutral-umask=n in the config file or –no-neutral-umask on the command line.

```
default: y
example: --no-neutral-umask
```

Protocol Timeout Option (–protocol-timeout) Protocol timeout.

Sets the timeout, in seconds, that the local or remote process will wait for a new message to be received on the protocol layer. This prevents processes from waiting indefinitely for a message. The protocol-timeout option must be greater than the db-timeout option.

```
default: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

Stanza Option (–stanza) Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one Postgres database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

```
example: --stanza=main
```

Log Options

Console Log Level Option (–log-level-console) Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-console=error
```

File Log Level Option (–log-level-file) Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: --log-level-file=debug
```

Std Error Log Level Option (–log-level-stderr) Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr .

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-stderr=error
```

Log Path Option (–log-path) Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=none then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Timestamp Option (–log-timestamp) Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Repository Options

Repository Cipher Passphrase Option (–repo-cipher-pass) Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

```
example: --repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

Repository Cipher Type Option (–repo-cipher-type) Cipher used to encrypt the repository.

The following repository types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

```
default: none
example: --repo-cipher-type=aes-256-cbc
```

Repository Path Option (–repo-path) Repository path where WAL segments and backups stored.

The repository is where pgBackRest stores backup and archives WAL segments.

If you are new to backup then it will be difficult to estimate in advance how much space you'll need. The best thing to do is take some backups then record the size of different types of backups (full/incr/diff) and measure the amount of WAL generated per day. This will give you a general idea of how much space you'll need, though of course requirements will likely change over time as your database evolves.

```
default: /var/lib/pgbackrest
example: --repo-path=/backup/db/backrest
```

S3 Repository Bucket Option (–repo-s3-bucket) S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo , so logs and other AWS generated content can also be stored in the bucket.

```
example: --repo-s3-bucket=db-backup
```

S3 SSL CA File Option (–repo-s3-ca-file) S3 SSL CA File.

Use a CA file other than the system default.

```
example: --repo-s3-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

S3 SSL CA Path Option (–repo-s3-ca-path) S3 SSL CA Path.

Use a CA path other than the system default.

```
example: --repo-s3-ca-path=/etc/pki/tls/certs
```

S3 Repository Endpoint Option (–repo-s3-endpoint) S3 repository endpoint.

The AWS end point should be valid for the selected region.

```
example: --repo-s3-endpoint=s3.amazonaws.com
```

S3 Repository Host Option (–repo-s3-host) S3 repository host.

Connect to a host other than the end point. This is typically used for testing.

```
example: --repo-s3-host=127.0.0.1
```

S3 Repository Access Key Option (–repo-s3-key) S3 repository access key.

AWS key used to access this bucket.

```
example: --repo-s3-key=AKIAIOSFODNN7EXAMPLE
```

S3 Repository Secret Access Key Option (–repo-s3-key-secret) S3 repository secret access key.

AWS secret key used to access this bucket.

```
example: --repo-s3-key-secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

S3 Repository Region Option (–repo-s3-region) S3 repository region.

The AWS region where the bucket was created.

```
example: --repo-s3-region=us-east-1
```

S3 Repository Verify SSL Option (–repo-s3-verify-ssl) Verify S3 server certificate.

Disables verification of the S3 server certificate. This should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo-s3-verify-ssl
```

Repository Type Option (–repo-type) Type of storage used for the repository.

The following repository types are supported:

- cifs - Like posix , but disables links and directory fsyncs
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service

```
default: posix
example: --repo-type=cifs
```

Stanza Options

Database Host Command Option (–db-cmd) pgBackRest exe path on the database host.

Required only if the path to pgbackrest is different on the local and database hosts. If not defined, the database host exe path will be set the same as the local exe path.

```
example: --db-cmd=/usr/lib/backrest/bin/pgbackrest
```

Database Host Configuration Option (–db-config) pgBackRest database host configuration file.

Sets the location of the configuration file on the database host. This is only required if the database host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --db-config=/etc/pgbackrest_db.conf
```

Database Host Option (–db-host) Cluster host for operating remotely via SSH.

Used for backups where the database cluster host is different from the backup host.

```
example: --db-host=db.domain.com
```

Database Path Option (–db-path) Cluster data directory.

This should be the same as the data_directory setting in postgresql.conf . Even though this value can be read from postgresql.conf or the database cluster it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

The db-path option is tested against the value reported by PostgreSQL on every online backup so it should always be current.

```
example: --db-path=/data/db
```

Database Port Option (–db-port) Cluster port.

Port that PostgreSQL is running on. This usually does not need to be specified as most database clusters run on the default port.

```
default: 5432
example: --db-port=6543
```

Database Socket Path Option (–db-socket-path) Cluster unix socket path.

The unix socket directory that was specified when PostgreSQL was started. pgBackRest will automatically look in the standard location for your OS so there usually no need to specify this setting unless the socket directory was explicitly modified with the unix_socket_directory setting in postgressql.conf .

```
example: --db-socket-path=/var/run/postgresql
```

Database SSH Port Option (–db-ssh-port) Database server SSH port when db-host is set.

Use this option to specify a non-default SSH port for a database server.

```
example: --db-ssh-port=25
```

Database User Option (–db-user) Cluster host logon user when db-host is set.

This user will also own the remote pgBackRest process and will initiate connections to PostgreSQL . For this to work correctly the user should be the PostgreSQL database cluster owner which is generally postgres , the default.

```
default: postgres
example: --db-user=db_owner
```

Stanza Upgrade Command (stanza-upgrade)

Immediately after upgrading PostgreSQL to a newer major version, the db-path for all pgBackRest configurations must be set to the new database location and the stanza-upgrade run on the backup host. If the database is offline use the –no-online option.

Command Options

Backup from Standby Option (–backup-standby) Backup from the standby cluster.

Enable backup from standby to reduce load on the primary cluster. This option requires that both the primary and standby hosts be configured.

```
default: n
example: --backup-standby
```

Online Option (–online) Update an online cluster.

Specifying –no-online prevents pgBackRest from connecting to PostgreSQL when upgrading the stanza.

```
default: y
example: --no-online
```

General Options

Buffer Size Option (–buffer-size) Buffer size for file operations.

Set the buffer size used for copy, compress, and uncompress functions. A maximum of 3 buffers will be in use at a time per process. An additional maximum of 256K per process may be used for zlib buffers.

```
default: 4194304
example: --buffer-size=32768
```

SSH client command Option (–cmd-ssh) Path to ssh client executable.

Use a specific SSH client when an alternate is desired or the ssh executable is not in $PATH.

```
default: ssh
example: --cmd-ssh=/usr/bin/ssh
```

Compress Level Option (–compress-level) Compression level for stored files.

Sets the zlib level to be used for file compression when compress=y .

```
default: 6
allowed: 0-9
example: --compress-level=9
```

Network Compress Level Option (–compress-level-network) Compression level for network transfer when compress=n .

Sets the zlib level to be used for protocol compression when compress=n and the database cluster is not on the same host as the backup. Protocol compression is used to reduce network traffic but can be disabled by setting compress-level-network=0 . When compress=y the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once. SSH compression is always disabled.

```
default: 3
allowed: 0-9
example: --compress-level-network=1
```

Config Option (–config) pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: /etc/pgbackrest.conf
example: --config=/var/lib/backrest/pgbackrest.conf
```

Database Timeout Option (–db-timeout) Database query timeout.

Sets the timeout, in seconds, for queries against the database. This includes the pg_start_backup() and pg_stop_backup() functions which can each take a substantial amount of time. Because of this the timeout should be kept high unless you know that these functions will return quickly (i.e. if you have set startfast=y and you know that the database cluster will not generate many WAL segments during the backup).

```
default: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

Lock Path Option (–lock-path) Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: --lock-path=/backup/db/lock
```

Neutral Umask Option (–neutral-umask) Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640. The lock and log directories set the directory and file mode to 0770 and 0660 respectively.

To use the executing user's umask instead specify neutral-umask=n in the config file or –no-neutral-umask on the command line.

```
default: y
example: --no-neutral-umask
```

Protocol Timeout Option (–protocol-timeout) Protocol timeout.

Sets the timeout, in seconds, that the local or remote process will wait for a new message to be received on the protocol layer. This prevents processes from waiting indefinitely for a message. The protocol-timeout option must be greater than the db-timeout option.

```
default: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

Stanza Option (–stanza) Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one Postgres database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

```
example: --stanza=main
```

Log Options

Console Log Level Option (–log-level-console) Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)

- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-console=error
```

File Log Level Option (–log-level-file) Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: --log-level-file=debug
```

Std Error Log Level Option (–log-level-stderr) Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr .

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-stderr=error
```

Log Path Option (–log-path) Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=none then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Timestamp Option (–log-timestamp) Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Repository Options

Backup Host Command Option (–backup-cmd) pgBackRest exe path on the backup host.

Required only if the path to pgbackrest is different on the local and backup hosts. If not defined, the backup host exe path will be set the same as the local exe path.

```
example: --backup-cmd=/usr/lib/backrest/bin/pgbackrest
```

Backup Host Configuration Option (–backup-config) pgBackRest backup host configuration file.

Sets the location of the configuration file on the backup host. This is only required if the backup host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --backup-config=/etc/pgbackrest_backup.conf
```

Backup Host Option (–backup-host) Backup host when operating remotely via SSH.

Make sure that trusted SSH authentication is configured between the db host and the backup host.

When backing up to a locally mounted network filesystem this setting is not required.

```
example: --backup-host=backup.domain.com
```

Backup SSH Port Option (–backup-ssh-port) Backup server SSH port when backup-host is set.

Use this option to specify a non-default SSH port for the backup server.

```
example: --backup-ssh-port=25
```

Backup User Option (–backup-user) Backup host user when backup-host is set.

Defines the user that will be used for operations on the backup server. Preferably this is not the postgres user but rather some other user like backrest . If PostgreSQL runs on the backup server the postgres user can be placed in the backrest group so it has read permissions on the repository without being able to damage the contents accidentally.

```
default: backrest
example: --backup-user=backrest
```

Repository Cipher Passphrase Option (–repo-cipher-pass) Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

```
example: --repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

Repository Cipher Type Option (–repo-cipher-type) Cipher used to encrypt the repository.

The following repository types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

```
default: none
example: --repo-cipher-type=aes-256-cbc
```

Repository Path Option (–repo-path) Repository path where WAL segments and backups stored.

The repository is where pgBackRest stores backup and archives WAL segments.

If you are new to backup then it will be difficult to estimate in advance how much space you'll need. The best thing to do is take some backups then record the size of different types of backups (full/incr/diff) and measure the amount of WAL generated per day. This will give you a general idea of how much space you'll need, though of course requirements will likely change over time as your database evolves.

```
default: /var/lib/pgbackrest
example: --repo-path=/backup/db/backrest
```

S3 Repository Bucket Option (–repo-s3-bucket) S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo , so logs and other AWS generated content can also be stored in the bucket.

```
example: --repo-s3-bucket=db-backup
```

S3 SSL CA File Option (–repo-s3-ca-file) S3 SSL CA File.

Use a CA file other than the system default.

```
example: --repo-s3-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

S3 SSL CA Path Option (–repo-s3-ca-path) S3 SSL CA Path.

Use a CA path other than the system default.

```
example: --repo-s3-ca-path=/etc/pki/tls/certs
```

S3 Repository Endpoint Option (–repo-s3-endpoint) S3 repository endpoint.

The AWS end point should be valid for the selected region.

```
example: --repo-s3-endpoint=s3.amazonaws.com
```

S3 Repository Host Option (–repo-s3-host) S3 repository host.

Connect to a host other than the end point. This is typically used for testing.

```
example: --repo-s3-host=127.0.0.1
```

S3 Repository Access Key Option (–repo-s3-key) S3 repository access key.

AWS key used to access this bucket.

```
example: --repo-s3-key=AKIAIOSFODNN7EXAMPLE
```

S3 Repository Secret Access Key Option (–repo-s3-key-secret) S3 repository secret access key.

AWS secret key used to access this bucket.

```
example: --repo-s3-key-secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

S3 Repository Region Option (–repo-s3-region) S3 repository region.

The AWS region where the bucket was created.

```
example: --repo-s3-region=us-east-1
```

S3 Repository Verify SSL Option (–repo-s3-verify-ssl) Verify S3 server certificate.

Disables verification of the S3 server certificate. This should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo-s3-verify-ssl
```

Repository Type Option (–repo-type) Type of storage used for the repository.

The following repository types are supported:

- cifs - Like posix , but disables links and directory fsyncs
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service

```
default: posix
example: --repo-type=cifs
```

Stanza Options

Database Host Command Option (–db-cmd) pgBackRest exe path on the database host.

Required only if the path to pgbackrest is different on the local and database hosts. If not defined, the database host exe path will be set the same as the local exe path.

```
example: --db-cmd=/usr/lib/backrest/bin/pgbackrest
```

Database Host Configuration Option (–db-config) pgBackRest database host configuration file.

Sets the location of the configuration file on the database host. This is only required if the database host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --db-config=/etc/pgbackrest_db.conf
```

Database Host Option (–db-host) Cluster host for operating remotely via SSH.

Used for backups where the database cluster host is different from the backup host.

```
example: --db-host=db.domain.com
```

Database Path Option (–db-path) Cluster data directory.

This should be the same as the data_directory setting in postgresql.conf . Even though this value can be read from postgresql.conf or the database cluster it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

The db-path option is tested against the value reported by PostgreSQL on every online backup so it should always be current.

```
example: --db-path=/data/db
```

Database Port Option (–db-port) Cluster port.

Port that PostgreSQL is running on. This usually does not need to be specified as most database clusters run on the default port.

```
default: 5432
example: --db-port=6543
```

Database Socket Path Option (–db-socket-path) Cluster unix socket path.

The unix socket directory that was specified when PostgreSQL was started. pgBackRest will automatically look in the standard location for your OS so there usually no need to specify this setting unless the socket directory was explicitly modified with the unix_socket_directory setting in postgressql.conf .

```
example: --db-socket-path=/var/run/postgresql
```

Database SSH Port Option (–db-ssh-port) Database server SSH port when db-host is set.

Use this option to specify a non-default SSH port for a database server.

```
example: --db-ssh-port=25
```

Database User Option (–db-user) Cluster host logon user when db-host is set.

This user will also own the remote pgBackRest process and will initiate connections to PostgreSQL . For this to work correctly the user should be the PostgreSQL database cluster owner which is generally postgres , the default.

```
default: postgres
example: --db-user=db_owner
```

Start Command (start)

If the pgBackRest processes were previously stopped using the stop command then they can be started again using the start command. Note that this will not immediately start up any pgBackRest processes but they are allowed to run.

General Options

SSH client command Option (–cmd-ssh) Path to ssh client executable.

Use a specific SSH client when an alternate is desired or the ssh executable is not in $PATH.

```
default: ssh
example: --cmd-ssh=/usr/bin/ssh
```

Config Option (–config) pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: /etc/pgbackrest.conf
example: --config=/var/lib/backrest/pgbackrest.conf
```

Lock Path Option (–lock-path) Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: --lock-path=/backup/db/lock
```

Neutral Umask Option (–neutral-umask) Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640. The lock and log directories set the directory and file mode to 0770 and 0660 respectively.

To use the executing user's umask instead specify neutral-umask=n in the config file or –no-neutral-umask on the command line.

```
default: y
example: --no-neutral-umask
```

Stanza Option (–stanza) Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one Postgres database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

```
example: --stanza=main
```

Log Options

Console Log Level Option (–log-level-console) Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-console=error
```

File Log Level Option (–log-level-file) Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: --log-level-file=debug
```

Std Error Log Level Option (–log-level-stderr) Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr .

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-stderr=error
```

Log Path Option (–log-path) Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=none then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Timestamp Option (–log-timestamp) Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Repository Options

Backup Host Command Option (–backup-cmd) pgBackRest exe path on the backup host.

Required only if the path to pgbackrest is different on the local and backup hosts. If not defined, the backup host exe path will be set the same as the local exe path.

```
example: --backup-cmd=/usr/lib/backrest/bin/pgbackrest
```

Backup Host Configuration Option (–backup-config) pgBackRest backup host configuration file.

Sets the location of the configuration file on the backup host. This is only required if the backup host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --backup-config=/etc/pgbackrest_backup.conf
```

Backup Host Option (–backup-host) Backup host when operating remotely via SSH.

Make sure that trusted SSH authentication is configured between the db host and the backup host.

When backing up to a locally mounted network filesystem this setting is not required.

```
example: --backup-host=backup.domain.com
```

Backup SSH Port Option (–backup-ssh-port) Backup server SSH port when backup-host is set.

Use this option to specify a non-default SSH port for the backup server.

```
example: --backup-ssh-port=25
```

Backup User Option (–backup-user) Backup host user when backup-host is set.

Defines the user that will be used for operations on the backup server. Preferably this is not the postgres user but rather some other user like backrest . If PostgreSQL runs on the backup server the postgres user can be placed in the backrest group so it has read permissions on the repository without being able to damage the contents accidentally.

```
default: backrest
example: --backup-user=backrest
```

Repository Cipher Passphrase Option (–repo-cipher-pass) Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

```
example: --repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

Repository Cipher Type Option (–repo-cipher-type) Cipher used to encrypt the repository.

The following repository types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

```
default: none
example: --repo-cipher-type=aes-256-cbc
```

Repository Path Option (–repo-path) Repository path where WAL segments and backups stored.

The repository is where pgBackRest stores backup and archives WAL segments.

If you are new to backup then it will be difficult to estimate in advance how much space you'll need. The best thing to do is take some backups then record the size of different types of backups (full/incr/diff) and measure the amount of WAL generated per day. This will give you a general idea of how much space you'll need, though of course requirements will likely change over time as your database evolves.

```
default: /var/lib/pgbackrest
example: --repo-path=/backup/db/backrest
```

S3 Repository Bucket Option (–repo-s3-bucket) S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo , so logs and other AWS generated content can also be stored in the bucket.

```
example: --repo-s3-bucket=db-backup
```

S3 SSL CA File Option (–repo-s3-ca-file) S3 SSL CA File.

Use a CA file other than the system default.

```
example: --repo-s3-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

S3 SSL CA Path Option (–repo-s3-ca-path) S3 SSL CA Path.

Use a CA path other than the system default.

```
example: --repo-s3-ca-path=/etc/pki/tls/certs
```

S3 Repository Endpoint Option (–repo-s3-endpoint) S3 repository endpoint.

The AWS end point should be valid for the selected region.

```
example: --repo-s3-endpoint=s3.amazonaws.com
```

S3 Repository Host Option (–repo-s3-host) S3 repository host.

Connect to a host other than the end point. This is typically used for testing.

```
example: --repo-s3-host=127.0.0.1
```

S3 Repository Access Key Option (–repo-s3-key) S3 repository access key.

AWS key used to access this bucket.

```
example: --repo-s3-key=AKIAIOSFODNN7EXAMPLE
```

S3 Repository Secret Access Key Option (–repo-s3-key-secret) S3 repository secret access key.

AWS secret key used to access this bucket.

```
example: --repo-s3-key-secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

S3 Repository Region Option (–repo-s3-region) S3 repository region.

The AWS region where the bucket was created.

```
example: --repo-s3-region=us-east-1
```

S3 Repository Verify SSL Option (–repo-s3-verify-ssl) Verify S3 server certificate.

Disables verification of the S3 server certificate. This should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo-s3-verify-ssl
```

Repository Type Option (–repo-type) Type of storage used for the repository.

The following repository types are supported:

- cifs - Like posix , but disables links and directory fsyncs
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service

```
default: posix
example: --repo-type=cifs
```

Stanza Options

Database Host Command Option (–db-cmd) pgBackRest exe path on the database host.

Required only if the path to pgbackrest is different on the local and database hosts. If not defined, the database host exe path will be set the same as the local exe path.

```
example: --db-cmd=/usr/lib/backrest/bin/pgbackrest
```

Database Host Configuration Option (–db-config) pgBackRest database host configuration file.

Sets the location of the configuration file on the database host. This is only required if the database host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --db-config=/etc/pgbackrest_db.conf
```

Database Host Option (–db-host) Cluster host for operating remotely via SSH.

Used for backups where the database cluster host is different from the backup host.

```
example: --db-host=db.domain.com
```

Database SSH Port Option (–db-ssh-port) Database server SSH port when db-host is set.

Use this option to specify a non-default SSH port for a database server.

```
example: --db-ssh-port=25
```

Stop Command (stop)

Does not allow any new pgBackRest processes to run. By default running processes will be allowed to complete successfully. Use the –force option to terminate running processes.

pgBackRest processes will return an error if they are run after the stop command completes.

Command Options

Force Option (–force) Force all pgBackRest processes to stop.

This option will send TERM signals to all running pgBackRest processes to effect a graceful but immediate shutdown. Note that this will also shutdown processes that were initiated on another system but have remotes running on the current system. For instance, if a backup was started on the backup server then running stop –force on the database server will shutdown the backup process on the backup server.

```
default: n
example: --force
```

General Options

SSH client command Option (–cmd-ssh) Path to ssh client executable.

Use a specific SSH client when an alternate is desired or the ssh executable is not in $PATH.

```
default: ssh
example: --cmd-ssh=/usr/bin/ssh
```

Config Option (–config) pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: /etc/pgbackrest.conf
example: --config=/var/lib/backrest/pgbackrest.conf
```

Lock Path Option (–lock-path) Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: --lock-path=/backup/db/lock
```

Neutral Umask Option (–neutral-umask) Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640. The lock and log directories set the directory and file mode to 0770 and 0660 respectively.

To use the executing user's umask instead specify neutral-umask=n in the config file or –no-neutral-umask on the command line.

```
default: y
example: --no-neutral-umask
```

Stanza Option (–stanza) Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one Postgres database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

```
example: --stanza=main
```

Log Options

Console Log Level Option (–log-level-console) Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-console=error
```

File Log Level Option (–log-level-file) Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: --log-level-file=debug
```

Std Error Log Level Option (–log-level-stderr) Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr .

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-stderr=error
```

Log Path Option (–log-path) Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=none then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Timestamp Option (–log-timestamp) Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Repository Options

Backup Host Command Option (–backup-cmd) pgBackRest exe path on the backup host.

Required only if the path to pgbackrest is different on the local and backup hosts. If not defined, the backup host exe path will be set the same as the local exe path.

```
example: --backup-cmd=/usr/lib/backrest/bin/pgbackrest
```

Backup Host Configuration Option (–backup-config) pgBackRest backup host configuration file.

Sets the location of the configuration file on the backup host. This is only required if the backup host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --backup-config=/etc/pgbackrest_backup.conf
```

Backup Host Option (–backup-host) Backup host when operating remotely via SSH.

Make sure that trusted SSH authentication is configured between the db host and the backup host.

When backing up to a locally mounted network filesystem this setting is not required.

```
example: --backup-host=backup.domain.com
```

Backup SSH Port Option (–backup-ssh-port) Backup server SSH port when backup-host is set.

Use this option to specify a non-default SSH port for the backup server.

```
example: --backup-ssh-port=25
```

Backup User Option (–backup-user) Backup host user when backup-host is set.

Defines the user that will be used for operations on the backup server. Preferably this is not the postgres user but rather some other user like backrest . If PostgreSQL runs on the backup server the postgres user can be placed in the backrest group so it has read permissions on the repository without being able to damage the contents accidentally.

```
default: backrest
example: --backup-user=backrest
```

Repository Cipher Passphrase Option (–repo-cipher-pass) Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

```
example: --repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

Repository Cipher Type Option (–repo-cipher-type) Cipher used to encrypt the repository.

The following repository types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

```
default: none
example: --repo-cipher-type=aes-256-cbc
```

Repository Path Option (–repo-path) Repository path where WAL segments and backups stored.

The repository is where pgBackRest stores backup and archives WAL segments.

If you are new to backup then it will be difficult to estimate in advance how much space you'll need. The best thing to do is take some backups then record the size of different types of backups (full/incr/diff) and measure the amount of WAL generated per day. This will give you a general idea of how much space you'll need, though of course requirements will likely change over time as your database evolves.

```
default: /var/lib/pgbackrest
example: --repo-path=/backup/db/backrest
```

S3 Repository Bucket Option (–repo-s3-bucket) S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo , so logs and other AWS generated content can also be stored in the bucket.

```
example: --repo-s3-bucket=db-backup
```

S3 SSL CA File Option (–repo-s3-ca-file) S3 SSL CA File.

Use a CA file other than the system default.

```
example: --repo-s3-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

S3 SSL CA Path Option (–repo-s3-ca-path) S3 SSL CA Path.

Use a CA path other than the system default.

```
example: --repo-s3-ca-path=/etc/pki/tls/certs
```

S3 Repository Endpoint Option (–repo-s3-endpoint) S3 repository endpoint.

The AWS end point should be valid for the selected region.

```
example: --repo-s3-endpoint=s3.amazonaws.com
```

S3 Repository Host Option (–repo-s3-host) S3 repository host.

Connect to a host other than the end point. This is typically used for testing.

```
example: --repo-s3-host=127.0.0.1
```

S3 Repository Access Key Option (–repo-s3-key) S3 repository access key.

AWS key used to access this bucket.

```
example: --repo-s3-key=AKIAIOSFODNN7EXAMPLE
```

S3 Repository Secret Access Key Option (–repo-s3-key-secret) S3 repository secret access key.

AWS secret key used to access this bucket.

```
example: --repo-s3-key-secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

S3 Repository Region Option (–repo-s3-region) S3 repository region.

The AWS region where the bucket was created.

```
example: --repo-s3-region=us-east-1
```

S3 Repository Verify SSL Option (–repo-s3-verify-ssl) Verify S3 server certificate.

Disables verification of the S3 server certificate. This should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo-s3-verify-ssl
```

Repository Type Option (–repo-type) Type of storage used for the repository.

The following repository types are supported:

- cifs - Like posix , but disables links and directory fsyncs
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service

```
default: posix
example: --repo-type=cifs
```

Stanza Options

Database Host Command Option (–db-cmd) pgBackRest exe path on the database host.

Required only if the path to pgbackrest is different on the local and database hosts. If not defined, the database host exe path will be set the same as the local exe path.

```
example: --db-cmd=/usr/lib/backrest/bin/pgbackrest
```

Database Host Configuration Option (–db-config) pgBackRest database host configuration file.

Sets the location of the configuration file on the database host. This is only required if the database host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: --db-config=/etc/pgbackrest_db.conf
```

Database Host Option (–db-host) Cluster host for operating remotely via SSH.

Used for backups where the database cluster host is different from the backup host.

```
example: --db-host=db.domain.com
```

Database SSH Port Option (–db-ssh-port) Database server SSH port when db-host is set.

Use this option to specify a non-default SSH port for a database server.

```
example: --db-ssh-port=25
```

Version Command (version)

Displays installed pgBackRest version.

— title: Configuration Reference draft: false —

Introduction

pgBackRest can be used entirely with command-line parameters but a configuration file is more practical for installations that are complex or set a lot of options. The default location for the configuration file is /etc/pgbackrest.conf .

Archive Options (archive)

The archive section defines parameters when doing async archiving. This means that the archive files will be stored locally, then a background process will pick them and move them to the backup.

Asynchronous Archiving Option (–archive-async)

Archive WAL segments asynchronously.

WAL segments will be copied to the local repo, then a process will be forked to compress the segment and transfer it to the remote repo if configured. Control will be returned to PostgreSQL as soon as the WAL segment is copied locally.

```
default: n
example: archive-async=y
```

Maximum Archive Queue Size Option (–archive-queue-max)

Limit size (in bytes) of the PostgreSQL archive queue.

After the limit is reached, the following will happen:

1. pgBackRest will notify PostgreSQL that the WAL was successfully archived, then **DROP IT** .
2. A warning will be output to the Postgres log.

If this occurs then the archive log stream will be interrupted and PITR will not be possible past that point. A new backup will be required to regain full restore capability.

In asynchronous mode the entire queue will be dropped to prevent spurts of WAL getting through before the queue limit is exceeded again.

The purpose of this feature is to prevent the log volume from filling up at which point Postgres will stop completely. Better to lose the backup than have PostgreSQL go down.

```
example: archive-queue-max=1073741824
```

Archive Timeout Option (–archive-timeout)

Archive timeout.

Set maximum time, in seconds, to wait for each WAL segment to reach the pgBackRest archive repository. The timeout applies to the check and backup commands when waiting for WAL segments required for backup consistency to be archived.

```
default: 60
allowed: 0.1-86400
example: archive-timeout=30
```

Backup Options (backup)

The backup section defines settings related to backup.

Check Archive Option (–archive-check)

Check that WAL segments are present in the archive before backup completes.

Checks that all WAL segments required to make the backup consistent are present in the WAL archive. It's a good idea to leave this as the default unless you are using another method for archiving.

This option must be enabled if archive-copy is enabled.

```
default: y
example: archive-check=n
```

Copy Archive Option (–archive-copy)

Copy WAL segments needed for consistency to the backup.

This slightly paranoid option protects against corruption in the WAL segment archive by storing the WAL segments required for consistency directly in the backup. WAL segments are still stored in the archive so this option will use additional space.

On restore, the WAL segments will be present in pg_xlog/pg_wal and PostgreSQL will use them in preference to calling the restore_command .

The archive-check option must be enabled if archive-copy is enabled.

```
default: n
example: archive-copy=y
```

Backup from Standby Option (–backup-standby)

Backup from the standby cluster.

Enable backup from standby to reduce load on the primary cluster. This option requires that both the primary and standby hosts be configured.

```
default: n
example: backup-standby=y
```

Page Checksums Option (–checksum-page)

Validate data page checksums.

Directs pgBackRest to validate all data page checksums while backing up a cluster. This option will be automatically enabled when the required C library is present and checksums are enabled on the cluster.

Failures in checksum validation will not abort a backup. Rather, warnings will be emitted in the log (and to the console with default settings) and the list of invalid pages will be stored in the backup manifest.

```
example: checksum-page=n
```

Hardlink Option (–hardlink)

Hardlink files between backups.

Enable hard-linking of files in differential and incremental backups to their full backups. This gives the appearance that each backup is a full backup. Be careful, though, because modifying files that are hard-linked can affect all the backups in the set.

```
default: n
example: hardlink=y
```

Manifest Save Threshold Option (–manifest-save-threshold)

Manifest save threshold during backup.

Defines how often the manifest will be saved during a backup (in bytes). Saving the manifest is important because it stores the checksums and allows the resume function to work efficiently. The actual threshold used is 1% of the backup size or manifest-save-threshold , whichever is greater.

```
default: 1073741824
example: manifest-save-threshold=5368709120
```

Resume Option (–resume)

Allow resume of failed backup.

Defines whether the resume feature is enabled. Resume can greatly reduce the amount of time required to run a backup after a previous backup of the same type has failed. It adds complexity, however, so it may be desirable to disable in environments that do not require the feature.

```
default: y
example: resume=n
```

Start Fast Option (–start-fast)

Force a checkpoint to start backup quickly.

Forces a checkpoint (by passing y to the fast parameter of pg_start_backup()) so the backup begins immediately. Otherwise the backup will start after the next regular checkpoint.

This feature only works in PostgreSQL >= 8.4 .

```
default: n
example: start-fast=y
```

Stop Auto Option (–stop-auto)

Stop prior failed backup on new backup.

This will only be done if an exclusive advisory lock can be acquired to demonstrate that the prior failed backup process has really stopped.

This feature relies on pg_is_in_backup() so only works on PostgreSQL >= 9.3 .

The setting is disabled by default because it assumes that pgBackRest is the only process doing exclusive online backups. It depends on an advisory lock that only pgBackRest sets so it may abort other processes that do exclusive online backups. Note that base_backup and pg_dump are safe to use with this setting because they do not call pg_start_backup() so are not exclusive.

```
default: n
example: stop-auto=y
```

Expire Options (expire)

The expire section defines how long backups will be retained. Expiration only occurs when the number of complete backups exceeds the allowed retention. In other words, if retention-full is set to 2, then there must be 3 complete backups before the oldest will be expired. Make sure you always have enough space for retention + 1 backups.

Archive Retention Option (–retention-archive)

Number of backups worth of continuous WAL to retain.

Note that the WAL segments required to make a backup consistent are always retained until the backup is expired regardless of how this option is configured.

If this value is not set, then the archive to expire will default to the retention-full (or retention-diff) value corresponding to the retention-archive-type if set to full (or diff). This will ensure that WAL is only expired for backups that are already expired.

This option must be set if retention-archive-type is set to incr . If disk space is at a premium, then this setting, in conjunction with retention-archive-type , can be used to aggressively expire WAL segments. However, doing so negates the ability to perform PITR from the backups with expired WAL and is therefore **not** recommended.

```
allowed: 1-9999999
example: retention-archive=2
```

Archive Retention Type Option (–retention-archive-type)

Backup type for WAL retention.

If set to full pgBackRest will keep archive logs for the number of full backups defined by retention-archive . If set to diff (differential) pgBackRest will keep archive logs for the number of full and differential backups defined by retention-archive , meaning if the last backup taken was a full backup, it will be counted as a differential for the purpose of retention. If set to incr (incremental) pgBackRest will keep archive logs for the number of full, differential, and incremental backups defined by retention-archive . It is recommended that this setting not be changed from the default which will only expire WAL in conjunction with expiring full backups.

```
default: full
example: retention-archive-type=diff
```

Differential Retention Option (–retention-diff)

Number of differential backups to retain.

When a differential backup expires, all incremental backups associated with the differential backup will also expire. When not defined all differential backups will be kept until the full backups they depend on expire.

```
allowed: 1-9999999
example: retention-diff=3
```

Full Retention Option (–retention-full)

Number of full backups to retain.

When a full backup expires, all differential and incremental backups associated with the full backup will also expire. When the option is not defined a warning will be issued. If indefinite retention is desired then set the option to the max value.

```
allowed: 1-9999999
example: retention-full=2
```

General Options (general)

The general section defines options that are common for many commands.

Buffer Size Option (–buffer-size)

Buffer size for file operations.

Set the buffer size used for copy, compress, and uncompress functions. A maximum of 3 buffers will be in use at a time per process. An additional maximum of 256K per process may be used for zlib buffers.

```
default: 4194304
example: buffer-size=32768
```

SSH client command Option (–cmd-ssh)

Path to ssh client executable.

Use a specific SSH client when an alternate is desired or the ssh executable is not in $PATH.

```
default: ssh
example: cmd-ssh=/usr/bin/ssh
```

Compress Option (–compress)

Use gzip file compression.

Backup files are compatible with command-line gzip tools.

```
default: y
example: compress=n
```

Compress Level Option (–compress-level)

Compression level for stored files.

Sets the zlib level to be used for file compression when compress=y .

```
default: 6
allowed: 0-9
example: compress-level=9
```

Network Compress Level Option (–compress-level-network)

Compression level for network transfer when compress=n .

Sets the zlib level to be used for protocol compression when compress=n and the database cluster is not on the same host as the backup. Protocol compression is used to reduce network traffic but can be disabled by setting compress-level-network=0 . When compress=y the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once. SSH compression is always disabled.

```
default: 3
allowed: 0-9
example: compress-level-network=1
```

Database Timeout Option (–db-timeout)

Database query timeout.

Sets the timeout, in seconds, for queries against the database. This includes the pg_start_backup() and pg_stop_backup() functions which can each take a substantial amount of time. Because of this the timeout should be kept high unless you know that these functions will return quickly (i.e. if you have set startfast=y and you know that the database cluster will not generate many WAL segments during the backup).

```
default: 1800
allowed: 0.1-604800
example: db-timeout=600
```

Lock Path Option (–lock-path)

Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: lock-path=/backup/db/lock
```

Neutral Umask Option (–neutral-umask)

Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640. The lock and log directories set the directory and file mode to 0770 and 0660 respectively.

To use the executing user's umask instead specify neutral-umask=n in the config file or –no-neutral-umask on the command line.

```
default: y
example: neutral-umask=n
```

Process Maximum Option (–process-max)

Max processes to use for compress/transfer.

Each process will perform compression and transfer to make the command run faster, but don't set process-max so high that it impacts database performance.

```
default: 1
allowed: 1-96
example: process-max=4
```

Protocol Timeout Option (–protocol-timeout)

Protocol timeout.

Sets the timeout, in seconds, that the local or remote process will wait for a new message to be received on the protocol layer. This prevents processes from waiting indefinitely for a message. The protocol-timeout option must be greater than the db-timeout option.

```
default: 1830
allowed: 0.1-604800
example: protocol-timeout=630
```

Spool Path Option (–spool-path)

Path where transient data is stored.

This path is used to store acknowledgements from the asynchronous archive-push process. These files are generally very small (zero to a few hundred bytes) so not much space is required.

The data stored in the spool path is not strictly temporary since it can and should survive a reboot. However, loss of the data in the spool path is not a problem. pgBackRest will simply recheck each WAL segment to ensure it is safely archived.

```
default: /var/spool/pgbackrest
example: spool-path=/backup/db/spool
```

Log Options (log)

The log section defines logging-related settings.

IMPORTANT NOTE : Trace-level logging may expose secrets such as keys and passwords. Use with caution!

Console Log Level Option (–log-level-console)

Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: log-level-console=error
```

File Log Level Option (–log-level-file)

Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: log-level-file=debug
```

Std Error Log Level Option (–log-level-stderr)

Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr .

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: log-level-stderr=error
```

Log Path Option (–log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=none then no log path is required.

```
default: /var/log/pgbackrest
example: log-path=/backup/db/log
```

Log Timestamp Option (–log-timestamp)

Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: log-timestamp=n
```

Repository Options (repository)

The repository section defines options used to configure the repository.

Backup Host Command Option (–backup-cmd)

pgBackRest exe path on the backup host.

Required only if the path to pgbackrest is different on the local and backup hosts. If not defined, the backup host exe path will be set the same as the local exe path.

```
example: backup-cmd=/usr/lib/backrest/bin/pgbackrest
```

Backup Host Configuration Option (–backup-config)

pgBackRest backup host configuration file.

Sets the location of the configuration file on the backup host. This is only required if the backup host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: backup-config=/etc/pgbackrest_backup.conf
```

Backup Host Option (–backup-host)

Backup host when operating remotely via SSH.

Make sure that trusted SSH authentication is configured between the db host and the backup host.

When backing up to a locally mounted network filesystem this setting is not required.

```
example: backup-host=backup.domain.com
```

Backup SSH Port Option (–backup-ssh-port)

Backup server SSH port when backup-host is set.

Use this option to specify a non-default SSH port for the backup server.

```
example: backup-ssh-port=25
```

Backup User Option (–backup-user)

Backup host user when backup-host is set.

Defines the user that will be used for operations on the backup server. Preferably this is not the postgres user but rather some other user like backrest . If PostgreSQL runs on the backup server the postgres user can be placed in the backrest group so it has read permissions on the repository without being able to damage the contents accidentally.

```
default: backrest
example: backup-user=backrest
```

Repository Cipher Passphrase Option (–repo-cipher-pass)

Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

```
example: repo-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

Repository Cipher Type Option (–repo-cipher-type)

Cipher used to encrypt the repository.

The following repository types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

```
default: none
example: repo-cipher-type=aes-256-cbc
```

Repository Path Option (–repo-path)

Repository path where WAL segments and backups stored.

The repository is where pgBackRest stores backup and archives WAL segments.

If you are new to backup then it will be difficult to estimate in advance how much space you'll need. The best thing to do is take some backups then record the size of different types of backups (full/incr/diff) and measure the amount of WAL generated per day. This will give you a general idea of how much space you'll need, though of course requirements will likely change over time as your database evolves.

```
default: /var/lib/pgbackrest
example: repo-path=/backup/db/backrest
```

S3 Repository Bucket Option (–repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo , so logs and other AWS generated content can also be stored in the bucket.

```
example: repo-s3-bucket=db-backup
```

S3 SSL CA File Option (–repo-s3-ca-file)

S3 SSL CA File.

Use a CA file other than the system default.

```
example: repo-s3-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

S3 SSL CA Path Option (–repo-s3-ca-path)

S3 SSL CA Path.

Use a CA path other than the system default.

```
example: repo-s3-ca-path=/etc/pki/tls/certs
```

S3 Repository Endpoint Option (–repo-s3-endpoint)

S3 repository endpoint.

The AWS end point should be valid for the selected region.

```
example: repo-s3-endpoint=s3.amazonaws.com
```

S3 Repository Host Option (–repo-s3-host)

S3 repository host.

Connect to a host other than the end point. This is typically used for testing.

```
example: repo-s3-host=127.0.0.1
```

S3 Repository Access Key Option (–repo-s3-key)

S3 repository access key.

AWS key used to access this bucket.

```
example: repo-s3-key=AKIAIOSFODNN7EXAMPLE
```

S3 Repository Secret Access Key Option (–repo-s3-key-secret)

S3 repository secret access key.

AWS secret key used to access this bucket.

```
example: repo-s3-key-secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

S3 Repository Region Option (–repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

```
example: repo-s3-region=us-east-1
```

S3 Repository Verify SSL Option (–repo-s3-verify-ssl)

Verify S3 server certificate.

Disables verification of the S3 server certificate. This should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: repo-s3-verify-ssl=n
```

Repository Type Option (–repo-type)

Type of storage used for the repository.

The following repository types are supported:

- cifs - Like posix , but disables links and directory fsyncs
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service

```
default: posix
example: repo-type=cifs
```

Restore Options (restore)

The restore section defines settings used for restoring backups.

Include Database Option (–db-include)

Restore only specified databases.

This feature allows only selected databases to be restored. Databases not specifically included will be restored as sparse, zeroed files to save space but still allow PostgreSQL to perform recovery. After recovery the databases that were not included will not be accessible but can be removed with the drop database command.

Note that built-in databases (template0 , template1 , and postgres) are always restored.

The –db-include option can be passed multiple times to specify more than one database to include.

```
example: db-include=db_main
```

Link All Option (–link-all)

Restore all symlinks.

By default symlinked directories and files are restored as normal directories and files in $PGDATA. This is because it may not be safe to restore symlinks to their original destinations on a system other than where the original backup was performed. This option restores all the symlinks just as they were on the original system where the backup was performed.

```
default: n
example: link-all=y
```

Link Map Option (–link-map)

Modify the destination of a symlink.

Allows the destination file or path of a symlink to be changed on restore. This is useful for restoring to systems that have a different storage layout than the original system where the backup was generated.

```
example: link-map=pg_xlog=/data/xlog
```

Recovery Option Option (–recovery-option)

Set an option in recovery.conf .

See http://www.postgresql.org/docs/X.X/static/recovery-config.html for details on recovery.conf options (replace X.X with your PostgreSQL version). This option can be used multiple times.

Note: The restore_command option will be automatically generated but can be overridden with this option. Be careful about specifying your own restore_command as pgBackRest is designed to handle this for you. Target Recovery options (recovery_target_name, recovery_target_time, etc.) are generated automatically by pgBackRest and should not be set with this option.

Since pgBackRest does not start PostgreSQL after writing the recovery.conf file, it is always possible to edit/check recovery.conf before manually restarting.

```
example: recovery-option=primary_conninfo=db.mydomain.com
```

Tablespace Map Option (–tablespace-map)

Restore a tablespace into the specified directory.

Moves a tablespace to a new location during the restore. This is useful when tablespace locations are not the same on a replica, or an upgraded system has different mount points.

Since PostgreSQL 9.2 tablespace locations are not stored in pg_tablespace so moving tablespaces can be done with impunity. However, moving a tablespace to the data_directory is not recommended and may cause problems. For more information on moving tablespaces http://www.databasesoup.com/2013/11/moving-tablespaces.html is a good resource.

```
example: tablespace-map=ts_01=/db/ts_01
```

Map All Tablespaces Option (–tablespace-map-all)

Restore all tablespaces into the specified directory.

By default tablespaces are restored into their original locations and while this behavior can be modified by with the tablespace-map open it is sometime preferable to remap all tablespaces to a new directory all at once. This is particularly useful for development or staging systems that may not have the same storage layout as the original system where the backup was generated.

The path specified will be the parent path used to create all the tablespaces in the backup.

```
example: tablespace-map-all=/data/tablespace
```

Stanza Options (stanza)

A stanza defines the backup configuration for a specific PostgreSQL database cluster. The stanza section must define the database cluster path and host/user if the database cluster is remote. Also, any global configuration sections can be overridden to define stanza-specific settings.

Indexing : All db- options can be indexed for configuring standby replicas. For example if a single standby replica is configured then index the db- options as db2- (e.g. db2-host, db2-path, etc). If an index is not specified (e.g. db-host) it will be aliased to db1- (e.g. db1-host).

Database Host Command Option (–db-cmd)

pgBackRest exe path on the database host.

Required only if the path to pgbackrest is different on the local and database hosts. If not defined, the database host exe path will be set the same as the local exe path.

```
example: db-cmd=/usr/lib/backrest/bin/pgbackrest
```

Database Host Configuration Option (–db-config)

pgBackRest database host configuration file.

Sets the location of the configuration file on the database host. This is only required if the database host configuration file is in a different location than the local configuration file.

```
default: /etc/pgbackrest.conf
example: db-config=/etc/pgbackrest_db.conf
```

Database Host Option (–db-host)

Cluster host for operating remotely via SSH.

Used for backups where the database cluster host is different from the backup host.

```
example: db-host=db.domain.com
```

Database Path Option (–db-path)

Cluster data directory.

This should be the same as the data_directory setting in postgresql.conf . Even though this value can be read from postgresql.conf or the database cluster it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

The db-path option is tested against the value reported by PostgreSQL on every online backup so it should always be current.

```
example: db-path=/data/db
```

Database Port Option (–db-port)

Cluster port.

Port that PostgreSQL is running on. This usually does not need to be specified as most database clusters run on the default port.

```
default: 5432
example: db-port=6543
```

Database Socket Path Option (–db-socket-path)

Cluster unix socket path.

The unix socket directory that was specified when PostgreSQL was started. pgBackRest will automatically look in the standard location for your OS so there usually no need to specify this setting unless the socket directory was explicitly modified with the unix_socket_directory setting in postgressql.conf .

```
example: db-socket-path=/var/run/postgresql
```

Database SSH Port Option (–db-ssh-port)

Database server SSH port when db-host is set.

Use this option to specify a non-default SSH port for a database server.

```
example: db-ssh-port=25
```

Database User Option (–db-user)

Cluster host logon user when db-host is set.

This user will also own the remote pgBackRest process and will initiate connections to PostgreSQL . For this to work correctly the user should be the PostgreSQL database cluster owner which is generally postgres , the default.

```
default: postgres
example: db-user=db_owner
```

— title: Releases draft: false —

Introduction

pgBackRest release numbers consist of two parts, major and minor. A major release may break compatibility with the prior major release, for instance the 1.XX releases are not compatible with the 0.XX releases. Minor releases can include bug fixes and features but do not change the repository format and strive to avoid changing options and naming.

The notes for a release may also contain "Additional Notes" but changes in this section are only to documentation or the test suite and have no direct impact the on the pgBackRest codebase.

Current Stable Release

v1.29 Release Notes

Critical Bug Fix for Backup Resume

Released July 5, 2018

IMPORTANT NOTE : This release fixes a critical bug in the backup resume feature. All resumed backups prior to this release should be considered inconsistent. A backup will be resumed after a prior backup fails, unless resume=n has been specified. A resumed backup can be identified by checking the backup log for the message "aborted backup of same type exists, will be cleaned to remove invalid files and resumed" . If the message exists, do not use this backup or any backup in the same set for a restore and check the restore logs to see if a resumed backup was restored. If so, there may be inconsistent data in the cluster.

Bug Fixes:

- Fix critical bug in resume that resulted in inconsistent backups. A regression in v0.82 removed the timestamp comparison when deciding which files from the aborted backup to keep on resume. See note above for more details. (*Reported by David Youatt, Yogesh Sharma, Stephen Frost.*)
- Fix non-compliant ISO-8601 timestamp format in S3 authorization headers. AWS and some gateways were tolerant of space rather than zero-padded hours while others were not. (*Fixed by Andrew Schwartz.*)
- Fix directory syncs running recursively when only the specified directory should be synced. (*Reported by Craig A. James.*)
- Fix –target-action and –recovery-option options being reported as invalid when restoring with –type=immediate . (*Reported by Brad Nicholson.*)
- Fix archive-copy throwing "path not found" error for incr/diff backups. (*Reported by yummyliu, Vitaliy Kukharik.*)
- Fix failure in manifest build when two or more files in PGDATA are linked to the same directory. (*Reported by Vitaliy Kukharik.*)
- Fix delta restore failing when a linked file was missing.
- Fix error in selective restore when only one user database exists in the cluster. (*Fixed by Cynthia Shang. Reported by Nj Baliyan.*)

Improvements:

- Improve the HTTP client to set content-length to 0 when not specified by the server. S3 (and gateways) always set content-length or transfer-encoding but HTTP 1.1 does not require it and proxies (e.g. HAProxy) may not include either. (*Suggested by Adam K. Sumner.*)
- Improve performance of HTTPS client. Buffering now takes the pending bytes on the socket into account (when present) rather than relying entirely on select() . In some instances the final bytes would not be flushed until the connection was closed.
- Improve S3 delete performance. The constant S3_BATCH_MAX had been replaced with a hard-coded value of 2, probably during testing.
- Make backup/restore path sync more efficient. Scanning the entire directory can be very expensive if there are a lot of small tables. The backup manifest contains the path list so use it to perform syncs instead of scanning the backup/restore path. Remove recursive path sync functionality since it is no longer used.

Additional Notes

Documentation Bug Fixes:

- Update docs with 32-bit support and caveats. 32-bit support was added in v1.26 . (*Reported by Viorel Tabara.*)

Documentation Improvements:

- Clarify that S3 buckets must be created by the user. (*Suggested by David Youatt.*)
- Update out-of-date description for the spool-path option.

Stable Releases

v1.28 Release Notes

Stanza Delete

Released February 1, 2018

Bug Fixes:

- Fixed inability to restore a single database contained in a tablespace using –db-include. (*Fixed by Cynthia Shang. Reported by Chiranjeevi Ravilla.*)
- Ensure latest db-id is selected on when matching archive.info to backup.info . This provides correct matching in the event there are system-id and db-version duplicates (e.g. after reverting a pg_upgrade). (*Fixed by Cynthia Shang. Reported by Adam K. Sumner.*)
- Fixed overly chatty error message when reporting an invalid command. (*Reported by Jason O'Donnell.*)

Features:

- Add stanza-delete command to cleanup unused stanzas. (*Contributed by Cynthia Shang. Suggested by Magnus Hagander.*)

Improvements:

- Improve stanza-create command so that it does not error when the stanza already exists. (*Contributed by Cynthia Shang.*)

Additional Notes

Documentation Improvements:

- Update stanza-create –force documentation to urge caution when using. (*Suggested by Jason O'Donnell.*)

v1.27 Release Notes

Bug Fixes and Documentation

Released December 19, 2017

Bug Fixes:

- Fixed an issue that suppressed locality errors for backup and restore . When a backup host is present, backups should only be allowed on the backup host and restores should only be allowed on the database host unless an alternate configuration is created that ignores the remote host. (*Reported by Lardière Sébastien.*)
- Fixed an issue where WAL was not expired on PostgreSQL 10. This was caused by a faulty regex that expected all PostgreSQL major versions to be X.X. (*Reported by Adam Brusselback.*)
- Fixed an issue where the –no-config option was not passed to child processes. This meant the child processes would still read the local config file and possibly cause unexpected behaviors.
- Fixed info command to eliminate "db (prior)" output if no backups or archives exist for a prior version of the cluster. (*Fixed by Cynthia Shang. Reported by Stephen Frost.*)

Additional Notes

Documentation Features:

- Document the relationship between the archive-copy and archive-check options. (*Suggested by Markus Nullmeier.*)
- Improve archive-copy reference documentation.

v1.26 Release Notes

Repository Encryption

Released November 21, 2017

Bug Fixes:

- Fixed an issue that could cause copying large manifests to fail during restore. (*Reported by Craig A. James.*)
- Fixed incorrect WAL offset for 32-bit architectures. (*Fixed by Javier Wilson.*)
- Fixed an issue retrieving WAL for old database versions. After a stanza-upgrade it should still be possible to restore backups from the previous version and perform recovery with archive-get . However, archive-get only checked the most recent db version/id and failed. Also clean up some issues when the same db version/id appears multiple times in the history. (*Fixed by Cynthia Shang. Reported by Clinton Adams.*)
- Fixed an issue with invalid backup groups being set correctly on restore. If the backup cannot map a group to a name it stores the group in the manifest as false then uses either the owner of $PGDATA to set the group during restore or failing that the group of the current user. This logic was not working correctly because the selected group was overwriting the user on restore leaving the group undefined and the user incorrectly set to the group. (*Reported by Jeff McCormick.*)
- Fixed an issue passing parameters to remotes. When more than one db was specified the path, port, and socket path would for db1 were passed no matter which db was actually being addressed. (*Reported by Uspen.*)

Features:

- Repository encryption support. (*Contributed by Cynthia Shang, David Steele.*)

Improvements:

- Disable gzip filter when –compress-level-network=0 . The filter was used with compress level set to 0 which added overhead without any benefit.
- Inflate performance improvement for gzip filter.

Additional Notes

Documentation Features:

- Add template to improve initial information gathered for issue submissions. (*Contributed by Cynthia Shang.*)

Documentation Improvements:

- Clarify usage of the archive-timeout option and describe how it is distinct from the PostgreSQL archive_timeout setting. (*Contributed by Cynthia Shang. Suggested by Keith Fiske.*)

Test Suite Features:

- Automated tests for 32-bit i386/i686 architecture.

v1.25 Release Notes

S3 Performance Improvements

Released October 24, 2017

Bug Fixes:

- Fix custom settings for compress-level option being ignored. (*Reported by Jens Wilke.*)
- Remove error when overlapping timelines are detected. Overlapping timelines are valid in many Point-in-Time-Recovery (PITR) scenarios. (*Reported by blogh.*)
- Fix instances where database-id was not rendered as an integer in JSON info output. (*Fixed by Cynthia Shang. Reported by Jason O'Donnell.*)

Features:

- Improve performance of list requests on S3. Any beginning literal portion of a filter expression is used to generate a search prefix which often helps keep the request small enough to avoid rate limiting. (*Suggested by Mihail Shvein.*)

Additional Notes

Test Suite Features:

- Add I/O performance tests.

v1.24 Release Notes

New Backup Exclusions

Released September 28, 2017

Bug Fixes:

- Fixed an issue where warnings were being emitted in place of lower priority log messages during backup from standby initialization. (*Reported by Uspen.*)
- Fixed an issue where some db-* options (e.g. db-port) were not being passed to remotes. (*Reported by Uspen.*)

Features:

- Exclude contents of pg_snapshots , pg_serial , pg_notify , and pg_dynshmem from backup since they are rebuilt on startup.
- Exclude pg_internal.init files from backup since they are rebuilt on startup.

Improvements:

- Open log file after async process is completely separated from the main process to prevent the main process from also logging to the file. (*Suggested by Jens Wilke.*)

Additional Notes

Documentation Features:

- Add trusted SSH configuration.

Documentation Improvements:

- Rename master to primary in documentation to align with PostgreSQL convention.

v1.23 Release Notes

Multiple Standbys and PostgreSQL 10 Support

Released September 3, 2017

Bug Fixes:

- Fixed an issue that could cause compression to abort on growing files. (*Reported by Jesper St John, Aleksandr Rogozin.*)
- Fixed an issue with keep-alives not being sent to the remote from the local process. (*Reported by William Cox.*)

Features:

- Up to seven standbys can be configured for backup from standby. (*Contributed by Cynthia Shang.*)
- PostgreSQL 10 support.
- Allow content-length (in addition to chunked encoding) when reading XML data to improve compatibility with third-party S3 gateways. (*Suggested by Victor Gdalevich.*)

Improvements:

- Increase HTTP timeout for S3.
- Add HTTP retries to harden against transient S3 network errors.

Additional Notes

Documentation Bug Fixes:

- Fixed document generation to include section summaries on the Configuration page. (*Fixed by Cynthia Shang.*)

v1.22 Release Notes

Fixed S3 Retry

Released August 9, 2017

Bug Fixes:

- Fixed authentication issue in S3 retry.

v1.21 Release Notes

Improved Info Output and SSH Port Option

Released August 8, 2017

Bug Fixes:

- The archive_status directory is now recreated on restore to support PostgreSQL 8.3 which does not recreate it automatically like more recent versions do. (*Reported by Stephen Frost.*)
- Fixed an issue that could cause the empty archive directory for an old PostgreSQL version to be left behind after a stanza-upgrade . (*Fixed by Cynthia Shang.*)

Features:

- Modified the info command (both text and JSON output) to display the archive ID and minimum/maximum WAL currently present in the archive for the current and prior, if any, database cluster version. (*Contributed by Cynthia Shang.*)
- Added –backup-ssh-port and –db-ssh-port options to support non-default SSH ports. (*Contributed by Cynthia Shang.*)

Improvements:

- Retry when S3 returns an internal error (500).

Additional Notes

Documentation Bug Fixes:

- Fix description of –online based on the command context.

Documentation Features:

- Add creation of /etc/pgbackrest.conf to manual installation instructions.

Documentation Improvements:

- Move repository options into a separate section in command/command-line help. (*Suggested by Stephen Frost.*)

v1.20 Release Notes

Critical 8.3/8.4 Bug Fix

Released June 27, 2017

IMPORTANT NOTE : PostgreSQL 8.3 and 8.4 installations utilizing tablespaces should upgrade immediately from any 1.XX release and run a full backup. A bug prevented tablespaces from being backed up on these versions only. PostgreSQL 9.0 is not affected.

Bug Fixes:

- Fixed an issue that prevented tablespaces from being backed up on PostgreSQL 8.4 .
- Fixed missing flag in C library build that resulted in a mismatched binary on 32-bit systems. (*Reported by Adrian Vondendriesch.*)

Features:

- Add s3-repo-ca-path and s3-repo-ca-file options to accommodate systems where CAs are not automatically found by IO::Socket::SSL , i.e. RHEL7 , or to load custom CAs. (*Suggested by Scott Frazer.*)

Additional Notes

Test Suite Features:

- Add documentation builds to CI.

v1.19 Release Notes

S3 Support

Released June 12, 2017

Bug Fixes:

- Fixed the info command so the WAL archive min/max displayed is for the current database version. (*Fixed by Cynthia Shang.*)
- Fixed the backup command so the backup-standby option is reset (and the backup proceeds on the primary) if the standby is not configured and/or reachable. (*Fixed by Cynthia Shang.*)
- Fixed config warnings raised from a remote process causing errors in the master process. (*Fixed by Cynthia Shang.*)

Features:

- Amazon S3 repository support. (*Reviewed by Cynthia Shang.*)

Additional Notes

Documentation Bug Fixes:

- Changed invalid max-archive-mb option in configuration reference to archive-queue-max .
- Fixed missing sudo in installation section. (*Fixed by Lætitia.*)

v1.18 Release Notes

Stanza Upgrade, Refactoring, and Locking Improvements

Released April 12, 2017

Bug Fixes:

- Fixed an issue where read-only operations that used local worker processes (i.e. restore) were creating write locks that could interfere with parallel archive-push . (*Reported by Jens Wilke.*)

Features:

- Added the stanza-upgrade command to provide a mechanism for upgrading a stanza after upgrading to a new major version of PostgreSQL . (*Contributed by Cynthia Shang.*)
- Added validation of pgbackrest.conf to display warnings if options are not valid or are not in the correct section. (*Contributed by Cynthia Shang.*)

Improvements:

- Simplify locking scheme. Now, only the master process will hold write locks (for archive-push and backup commands) and not all local and remote worker processes as before.
- Do not set timestamps of files in the backup directories to match timestamps in the cluster directory. This was originally done to enable backup resume, but that process is now implemented with checksums.
- Improved error message when the restore command detects the presence of postmaster.pid . (*Suggested by Yogesh Sharma.*)
- Renumber return codes between 25 and 125 to avoid PostgreSQL interpreting some as fatal signal exceptions. (*Suggested by Yogesh Sharma.*)

v1.17 Release Notes

Page Checksum Bug Fix

Released March 13, 2017

Bug Fixes:

- Fixed an issue where newly initialized (but unused) pages would cause page checksum warnings. (*Reported by Stephen Frost.*)

v1.16 Release Notes

Page Checksum Improvements, CI, and Package Testing

Released March 2, 2017

Bug Fixes:

- Fixed an issue where tables over 1GB would report page checksum warnings after the first segment. (*Reported by Stephen Frost.*)
- Fixed an issue where databases created with a non-default tablespace would raise bogus warnings about pg_filenode.map and pg_internal.init not being page aligned. (*Reported by blogh.*)

Additional Notes

Test Suite Features:

- Continuous integration using travis-ci .
- Automated builds of Debian packages for all supported distributions.

v1.15 Release Notes

Refactoring and Bug Fixes

Released February 13, 2017

Bug Fixes:

- Fixed a regression introduced in v1.13 that could cause backups to fail if files were removed (e.g. tables dropped) while the manifest was being built. (*Reported by Navid Golpayegani.*)

v1.14 Release Notes

Refactoring and Bug Fixes

Released February 13, 2017

Bug Fixes:

- Fixed an issue where an archive-push error would not be retried and would instead return errors to PostgreSQL indefinitely (unless the .error file was manually deleted). (*Reported by Jens Wilke.*)
- Fixed a race condition in parallel archiving where creation of new paths generated an error when multiple processes attempted to do so at the same time. (*Reported by Jens Wilke.*)

Improvements:

- Improved performance of wal archive min/max provided by the info command. (*Suggested by Jens Wilke.*)

Additional Notes

Documentation Features:

- Updated async archiving documentation to more accurately describe how the new method works and how it differs from the old method. (*Suggested by Jens Wilke.*)

v1.13 Release Notes

Parallel Archiving, Stanza Create, Improved Info and Check

Released February 5, 2017

IMPORTANT NOTE : The new implementation of asynchronous archiving no longer copies WAL to a separate queue. If there is any WAL left over in the old queue after upgrading to 1.13 , it will be abandoned and **not** pushed to the repository.

To prevent this outcome, stop archiving by setting archive_command = false . Next, drain the async queue by running pgbackrest –stanza=[stanza-name] archive-push and wait for the process to complete. Check that the queue in [spool-path]/archive/[stanza-name]/out

is empty. Finally, install 1.13 and restore the original archive_command .

IMPORTANT NOTE : The stanza-create command is not longer optional and must be executed before backup or archiving can be performed on a **new** stanza. Pre-existing stanzas do not require stanza-create to be executed.

Bug Fixes:

- Fixed const assignment giving compiler warning in C library. (*Fixed by Adrian Vondendriesch.*)
- Fixed a few directory syncs that were missed for the –repo-sync option.
- Fixed an issue where a missing user/group on restore could cause an "uninitialized value" error in File->owner() . (*Reported by Leonardo GG Avellar.*)
- Fixed an issue where protocol mismatch errors did not output the expected value.
- Fixed a spurious archive-get log message that indicated an exit code of 1 was an abnormal termination.

Features:

- Improved, multi-process implementation of asynchronous archiving.
- Improved stanza-create command so that it can repair broken repositories in most cases and is robust enough to be made mandatory. (*Contributed by Cynthia Shang.*)
- Improved check command to run on a standby, though only basic checks are done because pg_switch_xlog() cannot be executed on a replica. (*Contributed by Cynthia Shang.*)
- Added archive and backup WAL ranges to the info command.
- Added warning to update pg_tablespace.spclocation when remapping tablespaces in PostgreSQL < 9.2. (*Contributed by blogh.*)
- Remove remote lock requirements for the archive-get , restore , info , and check commands since they are read-only operations. (*Suggested by Michael Vitale.*)

Improvements:

- Log file banner is not output until the first log entry is written. (*Suggested by Jens Wilke.*)
- Reduced the likelihood of torn pages causing a false positive in page checksums by filtering on start backup LSN.
- Remove Intel-specific optimization from C library build flags. (*Contributed by Adrian Vondendriesch.*)
- Remove –lock option. This option was introduced before the lock directory could be located outside the repository and is now obsolete.
- Added –log-timestamp option to allow timestamps to be suppressed in logging. This is primarily used to avoid filters in the automated documentation.
- Return proper error code when unable to convert a relative path to an absolute path. (*Suggested by Yogesh Sharma.*)

Additional Notes

Documentation Features:

- Added documentation to the User Guide for the process-max option. (*Contributed by Cynthia Shang.*)

v1.12 Release Notes

Page Checksums, Configuration, and Bug Fixes

Released December 12, 2016

IMPORTANT NOTE : In prior releases it was possible to specify options on the command-line that were invalid for the current command without getting an error. An error will now be generated for invalid options so it is important to carefully check command-line options in your environment to prevent disruption.

Bug Fixes:

- Fixed an issue where options that were invalid for the specified command could be provided on the command-line without generating an error. The options were ignored and did not cause any change in behavior, but it did lead to some confusion. Invalid options will now generate an error. (*Reported by Nikhilchandra Kulkarni.*)
- Fixed an issue where internal symlinks were not being created for tablespaces in the repository. This issue was only apparent when trying to bring up clusters in-place manually using filesystem snapshots and did not affect normal backup and restore.
- Fixed an issue that prevented errors from being output to the console before the logging system was initialized, i.e. while parsing options. Error codes were still being returned accurately so this would not have made a process look like it succeeded when it did not. (*Reported by Adrian Vondendriesch.*)

- Fixed an issue where the db-port option specified on the backup server would not be properly passed to the remote unless it was from the first configured database. (*Reported by Michael Vitale.*)

Features:

- Added the –checksum-page option to allow pgBackRest to validate page checksums in data files when checksums are enabled on PostgreSQL >= 9.3. Note that this functionality requires a C library which may not initially be available in OS packages. The option will automatically be enabled when the library is present and checksums are enabled on the cluster. (*Suggested by Stephen Frost.*)
- Added the –repo-link option to allow internal symlinks to be suppressed when the repository is located on a filesystem that does not support symlinks. This does not affect any pgBackRest functionality, but the convenience link latest will not be created and neither will internal tablespace symlinks, which will affect the ability to bring up clusters in-place manually using filesystem snapshots.
- Added the –repo-sync option to allow directory syncs in the repository to be disabled for file systems that do not support them, e.g. NTFS.
- Added a predictable log entry to signal that a command has completed successfully. For example a backup ends successfully with: INFO: backup command end: completed successfully . (*Suggested by Jens Wilke.*)

Improvements:

- For simplicity, the pg_control file is now copied with the rest of the files instead of by itself of at the end of the process. The backup command does not require this behavior and the restore copies to a temporary file which is renamed at the end of the restore.

Additional Notes

Documentation Bug Fixes:

- Fixed an issue that suppressed exceptions in PDF builds.
- Fixed regression in section links introduced in v1.10 .

Documentation Features:

- Added Retention to QuickStart section.

v1.11 Release Notes

Bug Fix for Asynchronous Archiving Efficiency

Released November 17, 2016

Bug Fixes:

- Fixed an issue where asynchronous archiving was transferring one file per execution instead of transferring files in batches. This regression was introduced in v1.09 and affected efficiency only, all WAL segments were correctly archived in asynchronous mode. (*Reported by Stephen Frost.*)

v1.10 Release Notes

Stanza Creation and Minor Bug Fixes

Released November 8, 2016

Bug Fixes:

- Fixed an issue where a backup could error if no changes were made to a database between backups and only pg_control changed.
- Fixed an issue where tablespace paths with the same prefix would cause an invalid link error. (*Reported by Nikhilchandra Kulkarni.*)

Features:

- Added the stanza-create command to formalize creation of stanzas in the repository. (*Contributed by Cynthia Shang.*)

Improvements:

- Removed extraneous use lib directives from Perl modules. (*Suggested by Devrim Gündüz.*)

v1.09 Release Notes

9.6 Support, Configurability, and Bug Fixes

Released October 10, 2016

Bug Fixes:

- Fixed the check command to prevent an error message from being logged if the backup directory does not exist. (*Fixed by Cynthia Shang.*)
- Fixed error message to properly display the archive command when an invalid archive command is detected. (*Reported by Jason O'Donnell.*)
- Fixed an issue where the async archiver would not be started if archive-push did not have enough space to queue a new WAL segment. This meant that the queue would never be cleared without manual intervention (such as calling archive-push directly). PostgreSQL now receives errors when there is not enough space to store new WAL segments but the async process will still be started so that space is eventually freed. (*Reported by Jens Wilke.*)
- Fixed a remote timeout that occurred when a local process generated checksums (during resume or restore) but did not copy files, allowing the remote to go idle. (*Reported by Jens Wilke.*)

Features:

- Non-exclusive backups will automatically be used on PostgreSQL 9.6.
- Added the cmd-ssh option to allow the ssh client to be specified. (*Suggested by Jens Wilke.*)
- Added the log-level-stderr option to control whether console log messages are sent to stderr or stdout . By default this is set to warn which represents a change in behavior from previous versions, even though it may be more intuitive. Setting log-level-stderr=off will preserve the old behavior. (*Suggested by Sascha Biberhofer.*)
- Set application_name to "pgBackRest [command]" for database connections. (*Suggested by Jens Wilke.*)
- Check that archive_mode is enabled when archive-check option enabled.

Improvements:

- Clarified error message when unable to acquire pgBackRest advisory lock to make it clear that it is not a PostgreSQL backup lock. (*Suggested by Jens Wilke.*)
- pgBackRest version number included in command start INFO log output.
- Process ID logged for local process start/stop INFO log output.

Additional Notes

Documentation Features:

- Added archive-timeout option documentation to the user guide. (*Contributed by Cynthia Shang.*)

v1.08 Release Notes

Bug Fixes and Log Improvements

Released September 14, 2016

Bug Fixes:

- Fixed an issue where local processes were not disconnecting when complete and could later timeout. (*Reported by Todd Vernick.*)
- Fixed an issue where the protocol layer could timeout while waiting for WAL segments to arrive in the archive. (*Reported by Todd Vernick.*)

Improvements:

- Cache file log output until the file is created to create a more complete log.

v1.07 Release Notes

Thread to Process Conversion and Bug Fixes

Released September 7, 2016

Bug Fixes:

- Fixed an issue where tablespaces were copied from the primary during standby backup.
- Fixed the check command so backup info is checked remotely and not just locally. (*Fixed by Cynthia Shang.*)
- Fixed an issue where retention-archive was not automatically being set when retention-archive-type=diff , resulting in a less aggressive than intended expiration of archive. (*Fixed by Cynthia Shang.*)

Features:

- Converted Perl threads to processes to improve compatibility and performance.
- Exclude contents of $PGDATA/pg_replslot directory so that replication slots on the primary do not become part of the backup.
- The archive-start and archive-stop settings are now filled in backup.manifest even when archive-check=n . (*Suggested by Jens Wilke.*)
- Additional warnings when archive retention settings may not have the intended effect or would allow indefinite retention. (*Contributed by Cynthia Shang.*)
- Experimental support for non-exclusive backups in PostgreSQL 9.6 rc1. Changes to the control/catalog/WAL versions in subsequent release candidates may break compatibility but pgBackRest will be updated with each release to keep pace.

Additional Notes

Documentation Bug Fixes:

- Fixed minor documentation reproducibility issues related to binary paths.

Documentation Features:

- Documentation for archive retention. (*Contributed by Cynthia Shang.*)

v1.06 Release Notes

Backup from Standby and Bug Fixes

Released August 25, 2016

Bug Fixes:

- Fixed an issue where a tablespace link that referenced another link would not produce an error, but instead skip the tablespace entirely. (*Reported by Michael Vitale.*)
- Fixed an issue where options that should not allow multiple values could be specified multiple times in pgbackrest.conf without an error being raised. (*Reported by Michael Vitale.*)
- Fixed an issue where the protocol-timeout option was not automatically increased when the db-timeout option was increased. (*Reported by Todd Vernick.*)

Features:

- Backup from a standby cluster. A connection to the primary cluster is still required to start/stop the backup and copy files that are not replicated, but the vast majority of files are copied from the standby in order to reduce load on the primary.
- More flexible configuration for databases. Master and standby can both be configured on the backup server and pgBackRest will automatically determine which is the primary. This means no configuration changes for backup are required after failing over from a primary to standby when a separate backup server is used.
- Exclude directories during backup that are cleaned, recreated, or zeroed by PostgreSQL at startup. These include pgsql_tmp and pg_stat_tmp . The postgresql.auto.conf.tmp file is now excluded in addition to files that were already excluded: backup_label.old , postmaster.opts , postmaster.pid , recovery.conf , recovery.done .
- Experimental support for non-exclusive backups in PostgreSQL 9.6 beta4. Changes to the control/catalog/WAL versions in subsequent betas may break compatibility but pgBackRest will be updated with each release to keep pace.

Improvements:

- Improve error message for links that reference links in manifest build.
- Added hints to error message when relative paths are detected in archive-push or archive-get .
- Improve backup log messages to indicate which host the files are being copied from.

v1.05 Release Notes

Bug Fix for Tablespace Link Checking

Released August 9, 2016

Bug Fixes:

- Fixed an issue where tablespace paths that had $PGDATA as a substring would be identified as a subdirectories of $PGDATA even when they were not. Also hardened relative path checking a bit. (*Reported by Chris Fort.*)

Additional Notes

Documentation Features:

- Added documentation for scheduling backups with cron. (*Contributed by Cynthia Shang.*)

Documentation Improvements:

- Moved the backlog from the pgBackRest website to the GitHub repository wiki. (*Contributed by Cynthia Shang.*)

v1.04 Release Notes

Various Bug Fixes

Released July 30, 2016

Bug Fixes:

- Fixed an issue an where an extraneous remote was created causing threaded backup/restore to possibly timeout and/or throw a lock conflict. (*Reported by Michael Vitale.*)
- Fixed an issue where db-path was not required for the check command so an assert was raised when it was missing rather than a polite error message. (*Reported by Michael Vitale.*)
- Fixed check command to throw an error when database version/id does not match that of the archive. (*Fixed by Cynthia Shang.*)
- Fixed an issue where a remote could try to start its own remote when the backup-host option was not present in pgbackrest.conf on the database server. (*Reported by Lardière Sébastien.*)
- Fixed an issue where the contents of pg_xlog were being backed up if the directory was symlinked. This didn't cause any issues during restore but was a waste of space.
- Fixed an invalid log() call in lock routines.

Features:

- Experimental support for non-exclusive backups in PostgreSQL 9.6 beta3. Changes to the control/catalog/WAL versions in subsequent betas may break compatibility but pgBackRest will be updated with each release to keep pace.

Improvements:

- Suppress banners on SSH protocol connections.
- Improved remote error messages to identify the host where the error was raised.
- All remote types now take locks. The exceptions date to when the test harness and pgBackRest were running in the same VM and no longer apply.

Additional Notes

Documentation Features:

- Added clarification on why the default for the backrest-user option is backrest . (*Suggested by Michael Vitale.*)
- Updated information about package availability on supported platforms. (*Suggested by Michael Vitale.*)

v1.03 Release Notes

Check Command and Bug Fixes

Released July 2, 2016

Bug Fixes:

- Fixed an issue where keep-alives could be starved out by lots of small files during multi-threaded backup . They were also completely absent from single/multi-threaded backup resume and restore checksumming. (*Reported by Janice Parkinson, Chris Barber.*)
- Fixed an issue where the expire command would refuse to run when explicitly called from the command line if the db-host option was set. This was not an issue when expire was run automatically after a backup (*Reported by Chris Barber.*)
- Fixed an issue where validation was being running on archive_command even when the archive-check option was disabled.

Features:

- Added check command to validate that pgBackRest is configured correctly for archiving and backups. (*Contributed by Cynthia Shang.*)
- Added the protocol-timeout option. Previously protocol-timeout was set as db-timeout + 30 seconds.
- Failure to shutdown remotes at the end of the backup no longer throws an exception. Instead a warning is generated that recommends a higher protocol-timeout .
- Experimental support for non-exclusive backups in PostgreSQL 9.6 beta2. Changes to the control/catalog/WAL versions in subsequent betas may break compatibility but pgBackRest will be updated with each release to keep pace.

Improvements:

- Improved handling of users/groups captured during backup that do not exist on the restore host. Also explicitly handle the case where user/group is not mapped to a name.
- Option handling is now far more strict. Previously it was possible for a command to use an option that was not explicitly assigned to it. This was especially true for the backup-host and db-host options which are used to determine locality.

Additional Notes

Documentation Improvements:

- Allow a static date to be used for documentation to generate reproducible builds. (*Suggested by Adrian Vondendriesch.*)
- Added documentation for asynchronous archiving to the user guide. (*Contributed by Cynthia Shang.*)
- Recommended install location for pgBackRest modules is now /usr/share/perl5 since /usr/lib/perl5 has been removed from the search path in newer versions of Perl.
- Added instructions for removing prior versions of pgBackRest .

v1.02 Release Notes

Bug Fix for Perl 5.22

Released June 2, 2016

Bug Fixes:

- Fix usage of sprintf() due to new constraints in Perl 5.22. Parameters not referenced in the format string are no longer allowed. (*Fixed by Adrian Vondendriesch.*)

Additional Notes

Documentation Bug Fixes:

- Fixed syntax that was not compatible with Perl 5.2X. (*Fixed by Christoph Berg, Adrian Vondendriesch.*)
- Fixed absolute paths that were used for the PDF logo. (*Reported by Adrian Vondendriesch.*)

Documentation Features:

- Release notes are now broken into sections so that bugs, features, and refactors are clearly delineated. An “Additional Notes” section has been added for changes to documentation and the test suite that do not affect the core code.
- Added man page generation. (*Contributed by Adrian Vondendriesch, David Steele.*)
- The change log was the last piece of documentation to be rendered in Markdown only. Wrote a converter so the document can be output by the standard renderers. The change log will now be located on the website and has been renamed to “Releases” . (*Contributed by Cynthia Shang.*)

v1.01 Release Notes

Enhanced Info, Selective Restore, and 9.6 Support

Released May 17, 2016

Features:

- Enhanced text output of info command to include timestamps, sizes, and the reference list for all backups. (*Contributed by Cynthia Shang.*)
- Allow selective restore of databases from a cluster backup. This feature can result in major space and time savings when only specific databases are restored. Unrestored databases will not be accessible but must be manually dropped before they will be removed from the shared catalogue. (*Reviewed by Cynthia Shang, Greg Smith, Stephen Frost. Suggested by Stephen Frost.*)
- Experimental support for non-exclusive backups in PostgreSQL 9.6 beta1. Changes to the control/catalog/WAL versions in subsequent betas may break compatibility but pgBackRest will be updated with each release to keep pace. (*Reviewed by Cynthia Shang.*)

v1.00 Release Notes

New Repository Format and Configuration Scheme, Link Support

Released April 14, 2016

IMPORTANT NOTE : This flag day release breaks compatibility with older versions of pgBackRest . The manifest format, on-disk structure, configuration scheme, and the exe/path names have all changed. You must create a new repository to hold backups for this version of pgBackRest and keep your older repository for a time in case you need to do a restore. Restores from the prior repository will require the prior version of pgBackRest but because of name changes it is possible to have 1.00 and a prior version of pgBackRest installed at the same time. See the notes below for more detailed information on what has changed.

Features:

- Implemented a new configuration scheme which should be far simpler to use. See the User Guide and Configuration Reference for details but for a simple configuration all options can now be placed in the stanza section. Options that are shared between stanzas can be placed in the [global] section. More complex configurations can still make use of command sections though this should be a rare use case. (*Suggested by Michael Renner.*)
- The repo-path option now always refers to the repository where backups and archive are stored, whether local or remote, so the repo-remote-path option has been removed. The new spool-path option can be used to define a location for queueing WAL segments when archiving asynchronously. A local repository is no longer required.
- The default configuration filename is now pgbackrest.conf instead of pg_backrest.conf . This was done for consistency with other naming changes but also to prevent old config files from being loaded accidentally when migrating to 1.00 . (*Suggested by Michael Renner, Stephen Frost.*)
- The default repository name was changed from /var/lib/backup to /var/lib/pgbackrest . (*Suggested by Michael Renner, Stephen Frost.*)
- Lock files are now stored in /tmp/pgbackrest by default. These days /run/pgbackrest is the preferred location but that would require init scripts which are not part of this release. The lock-path option can be used to configure the lock directory.
- Log files are now stored in /var/log/pgbackrest by default and no longer have the date appended so they can be managed with logrotate . The log-path option can be used to configure the lock directory. (*Suggested by Stephen Frost.*)
- Executable filename changed from pg_backrest to pgbackrest . (*Suggested by Michael Renner, Stephen Frost.*)
- All files and directories linked from PGDATA are now included in the backup. By default links will be restored directly into PGDATA as files or directories. The –link-all option can be used to restore all links to their original locations. The –link-map option can be used to remap a link to a new location.
- Removed –tablespace option and replaced with –tablespace-map-all option which should more clearly indicate its function.
- Added detail log level which will output more information than info without being as verbose as debug .

Pre-Stable Releases

v0.92 Release Notes

Command-line Repository Path Fix

Released April 6, 2016

Bug Fixes:

- Fixed an issue where the master process was passing –repo-remote-path instead of –repo-path to the remote and causing the lock files to be created in the default repository directory (/var/lib/backup), generally ending in failure. This was only an issue when –repo-remote-path was defined on the command line rather than in pg_backrest.conf . (*Reported by Jan Wieck.*)

v0.91 Release Notes

Tablespace Bug Fix and Minor Enhancements

Released March 22, 2016

IMPORTANT BUG FIX FOR TABLESPACES : A change to the repository format was accidentally introduced in 0.90 which means the on-disk backup was no longer a valid PostgreSQL cluster when the backup contained tablespaces. This only affected users who directly copied the backups to restore PostgreSQL clusters rather than using the restore command. However, the fix breaks compatibility with older backups that contain tablespaces no matter how they are being restored (pgBackRest will throw errors and refuse to restore). New full backups should be taken immediately after installing version 0.91 for any clusters that contain tablespaces. If older backups need to be restored then use a version of pgBackRest that matches the backup version.

Bug Fixes:

- Fixed repository incompatibility introduced in pgBackRest 0.90. (*Reported by Evan Benoit.*)

Features:

- Copy global/pg_control last during backups.
- Write .info and .manifest files to temp before moving them to their final locations and fsync'ing.
- Rename –no-start-stop option to –no-online .

Additional Notes

Test Suite Features:

- Static source analysis using Perl-Critic, currently passes on gentle.

v0.90 Release Notes

9.5 Support, Various Enhancements, and Minor Bug Fixes

Released February 7, 2016

Bug Fixes:

- Fixed an issue where specifying –no-archive-check would throw a configuration error. (*Reported by Jason O'Donnell.*)
- Fixed an issue where a temp WAL file left over after a well-timed system crash could cause the next archive-push to fail.
- The retention-archive option can now be be safely set to less than backup retention (retention-full or retention-diff) without also specifying archive-copy=n . The WAL required to make the backups that fall outside of archive retention consistent will be preserved in the archive. However, in this case PITR will not be possible for the backups that fall outside of archive retention.

Features:

- When backing up and restoring tablespaces pgBackRest only operates on the subdirectory created for the version of PostgreSQL being run against. Since multiple versions can live in a tablespace (especially during a binary upgrade) this prevents too many files from being copied during a backup and other versions possibly being wiped out during a restore. This only applies to PostgreSQL >= 9.0 — prior versions of PostgreSQL could not share a tablespace directory.
- Generate an error when archive-check=y but archive_command does not execute pg_backrest . (*Contributed by Jason O'Donnell.*)
- Improved error message when repo-path or repo-remote-path does not exist.
- Added checks for –delta and –force restore options to ensure that the destination is a valid $PGDATA directory. pgBackRest will check for the presence of PG_VERSION or backup.manifest (left over from an aborted restore). If neither file is found then –delta and –force will be disabled but the restore will proceed unless there are files in the $PGDATA directory (or any tablespace directories) in which case the operation will be aborted.
- When restore –set=latest (the default) the actual backup restored will be output to the log.
- Support for PostgreSQL 9.5 partial WAL segments and recovery_target_action setting. The archive_mode = 'always' setting is not yet supported.
- Support for recovery_target = 'immediate' recovery setting introduced in PostgreSQL 9.4.
- The following tablespace checks have been added: paths or files in pg_tblspc, relative links in pg_tblspc, tablespaces in $PGDATA. All three will generate errors.

v0.89 Release Notes

Timeout Bug Fix and Restore Read-Only Repositories

Released December 24, 2015

Bug Fixes:

- Fixed an issue where longer-running backups/restores would timeout when remote and threaded. Keepalives are now used to make sure the remote for the main process does not timeout while the thread remotes do all the work. The error message for timeouts was also improved to make debugging easier. (*Reported by Stephen Frost.*)

Features:

- Allow restores to be performed on a read-only repository by using –no-lock and –log-level-file=off . The –no-lock option can only be used with restores.

v0.88 Release Notes

Documentation and Minor Bug Fixes

Released November 22, 2015

Bug Fixes:

- Fixed an issue where the start / stop commands required the –config option. (*Reported by Dmitry Didovicher.*)
- Fixed an issue where log files were being overwritten instead of appended. (*Reported by Stephen Frost, Dmitry Didovicher.*)
- Fixed an issue where backup-user was not optional.

Features:

- Symlinks are no longer created in backup directories in the repository. These symlinks could point virtually anywhere and potentially be dangerous. Symlinks are still recreated during a restore. (*Suggested by Stephen Frost.*)
- Added better messaging for backup expiration. Full and differential backup expirations are logged on a single line along with a list of all dependent backups expired.
- Archive retention is automatically set to full backup retention if not explicitly configured.

Additional Notes

Documentation Features:

- Added documentation in the user guide for delta restores, expiration, dedicated backup hosts, starting and stopping pgBackRest , and replication.

v0.87 Release Notes

Website and User Guide

Released October 28, 2015

Features:

- The backup_label.old and recovery.done files are now excluded from backups.

Additional Notes

Documentation Features:

- Added a new user guide that covers pgBackRest basics and some advanced topics including PITR. Much more to come, but it's a start. (*Contributed by David Steele, Stephen Frost. Reviewed by Michael Renner, Cynthia Shang, Eric Radman, Dmitry Didovicher.*)

v0.85 Release Notes

Start/Stop Commands and Minor Bug Fixes

Released October 8, 2015

Bug Fixes:

- Fixed an issue where an error could be returned after a backup or restore completely successfully.
- Fixed an issue where a resume would fail if temp files were left in the root backup directory when the backup failed. This scenario was likely if the backup process got terminated during the copy phase.

Features:

- Added stop and start commands to prevent pgBackRest processes from running on a system where PostgreSQL is shutdown or the system needs to be quiesced for some other reason.
- Experimental support for PostgreSQL 9.5 beta1. This may break when the control version or WAL magic changes in future versions but will be updated in each pgBackRest release to keep pace. All regression tests pass except for –target-resume tests (this functionality has changed in 9.5) and there is no testing yet for .partial WAL segments.

v0.82 Release Notes

Refactoring, Command-line Help, and Minor Bug Fixes

Released September 14, 2015

Bug Fixes:

- Fixed an issue where resumed compressed backups were not preserving existing files.
- Fixed an issue where resume and incr/diff would not ensure that the prior backup had the same compression and hardlink settings.
- Fixed an issue where a cold backup using –no-start-stop could be started on a running PostgreSQL cluster without –force specified.
- Fixed an issue where a thread could be started even when none were requested.
- Fixed an issue where the pgBackRest version number was not being updated in backup.info and archive.info after an upgrade/downgrade.
- Fixed an issue where the info command was throwing an exception when the repository contained no stanzas. (*Reported by Stephen Frost.*)
- Fixed an issue where the PostgreSQL pg_stop_backup() NOTICEs were being output to stderr . (*Reported by Stephen Frost.*)

Features:

- Experimental support for PostgreSQL 9.5 alpha2. This may break when the control version or WAL magic changes in future versions but will be updated in each pgBackRest release to keep pace. All regression tests pass except for –target-resume tests (this functionality has changed in 9.5) and there is no testing yet for .partial WAL segments.

Improvements:

- Renamed recovery-setting option and section to recovery-option to be more consistent with pgBackRest naming conventions.
- Added dynamic module loading to speed up commands, especially asynchronous archiving.

Additional Notes

Documentation Features:

- Command-line help is now extracted from the same XML source that is used for the other documentation and includes much more detail.

v0.80 Release Notes

DBI Support, Stability, and Convenience Features

Released August 9, 2015

Bug Fixes:

- Fixed an issue that caused the formatted timestamp for both the oldest and newest backups to be reported as the current time by the info command. Only text output was affected – json output reported the correct epoch values. (*Reported by Michael Renner.*)
- Fixed protocol issue that was preventing ssh errors (especially on connection) from being logged.

Features:

- The repository is now created and updated with consistent directory and file modes. By default umask is set to 0000 but this can be disabled with the neutral-umask setting. (*Suggested by Cynthia Shang.*)
- Added the stop-auto option to allow failed backups to automatically be stopped when a new backup starts.
- Added the db-timeout option to limit the amount of time pgBackRest will wait for pg_start_backup() and pg_stop_backup() to return.
- Remove pg_control file at the beginning of the restore and copy it back at the very end. This prevents the possibility that a partial restore can be started by PostgreSQL .
- Added checks to be sure the db-path setting is consistent with db-port by comparing the data_directory as reported by the cluster against the db-path setting and the version as reported by the cluster against the value read from pg_control . The db-socket-path setting is checked to be sure it is an absolute path.
- Experimental support for PostgreSQL 9.5 alpha1. This may break when the control version or WAL magic changes in future versions but will be updated in each pgBackRest release to keep pace. All regression tests pass except for –target-resume tests (this functionality has changed in 9.5) and there is no testing yet for .partial WAL segments.

Improvements:

- Now using Perl DBI and DBD::Pg for connections to PostgreSQL rather than psql . The cmd-psql and cmd-psql-option settings have been removed and replaced with db-port and db-socket-path . Follow the instructions in the Installation Guide to install DBD::Pg on your operating system.

Additional Notes

Test Suite Features:

- Added vagrant test configurations for Ubuntu 14.04 and CentOS 7.

v0.78 Release Notes

Remove CPAN Dependencies, Stability Improvements

Released July 13, 2015

Improvements:

- Removed dependency on CPAN packages for multi-threaded operation. While it might not be a bad idea to update the threads and Thread::Queue packages, it is no longer necessary.
- Modified wait backoff to use a Fibonacci rather than geometric sequence. This will make wait time grow less aggressively while still giving reasonable values.

Additional Notes

Test Suite Features:

- Added vagrant test configurations for Ubuntu 12.04 and CentOS 6.

v0.77 Release Notes

CentOS/RHEL 6 Support and Protocol Improvements

Released June 30, 2015

Features:

- Added file and directory syncs to the File object for additional safety during backup/restore and archiving. (*Suggested by Andres Freund.*)
- Added support for Perl 5.10.1 and OpenSSH 5.3 which are default for CentOS/RHEL 6. (*Suggested by Eric Radman.*)
- Improved error message when backup is run without archive_command set and without –no-archive-check specified. (*Suggested by Eric Radman.*)

v0.75 Release Notes

New Repository Format, Info Command and Experimental 9.5 Support

Released June 14, 2015

IMPORTANT NOTE : This flag day release breaks compatibility with older versions of pgBackRest . The manifest format, on-disk structure, and the binary names have all changed. You must create a new repository to hold backups for this version of pgBackRest and keep your older repository for a time in case you need to do a restore. The pg_backrest.conf file has not changed but you'll need to change any references to pg_backrest.pl in cron (or elsewhere) to pg_backrest (without the .pl extension).

Features:

- Added the info command.
- Logging now uses unbuffered output. This should make log files that are being written by multiple threads less chaotic. (*Suggested by Michael Renner.*)
- Experimental support for PostgreSQL 9.5. This may break when the control version or WAL magic changes but will be updated in each release.

Improvements:

- More efficient file ordering for backup . Files are copied in descending size order so a single thread does not end up copying a large file at the end. This had already been implemented for restore .

v0.70 Release Notes

Stability Improvements for Archiving, Improved Logging and Help

Released June 1, 2015

Bug Fixes:

- Fixed an issue where archive-copy would fail on an incr/diff backup when hardlink=n . In this case the pg_xlog path does not already exist and must be created. (*Reported by Michael Renner.*)
- Fixed an issue in async archiving where archive-push was not properly returning 0 when archive-max-mb was reached and moved the async check after transfer to avoid having to remove the stop file twice. Also added unit tests for this case and improved error messages to make it clearer to the user what went wrong. (*Reported by Michael Renner.*)
- Fixed a locking issue that could allow multiple operations of the same type against a single stanza. This appeared to be benign in terms of data integrity but caused spurious errors while archiving and could lead to errors in backup/restore. (*Reported by Michael Renner.*)

Features:

- Allow duplicate WAL segments to be archived when the checksum matches. This is necessary for some recovery scenarios.
- Allow comments/disabling in pg_backrest.conf using the # character. Only # characters in the forst character of the line are honored. (*Suggested by Michael Renner.*)
- Better logging before pg_start_backup() to make it clear when the backup is waiting on a checkpoint. (*Suggested by Michael Renner.*)
- Various command behavior and logging fixes. (*Reviewed by Michael Renner. Suggested by Michael Renner.*)

Improvements:

- Replaced JSON module with JSON::PP which ships with core Perl.

Additional Notes

Documentation Bug Fixes:

- Various help fixes. (*Reviewed by Michael Renner. Reported by Michael Renner.*)

v0.65 Release Notes

Improved Resume and Restore Logging, Compact Restores

Released May 11, 2015

Bug Fixes:

- Fixed an issue where an absolute path was not written into recovery.conf when the restore was run with a relative path.

Features:

- Better resume support. Resumed files are checked to be sure they have not been modified and the manifest is saved more often to preserve checksums as the backup progresses. More unit tests to verify each resume case.
- Resume is now optional. Use the resume setting or –no-resume from the command line to disable.
- More info messages during restore. Previously, most of the restore messages were debug level so not a lot was output in the log.
- Added tablespace setting to allow tablespaces to be restored into the pg_tblspc path. This produces compact restores that are convenient for development, staging, etc. Currently these restores cannot be backed up as pgBackRest expects only links in the pg_tblspc path.

v0.61 Release Notes

Bug Fix for Uncompressed Remote Destination

Released April 21, 2015

Bug Fixes:

- Fixed a buffering error that could occur on large, highly-compressible files when copying to an uncompressed remote destination. The error was detected in the decompression code and resulted in a failed backup rather than corruption so it should not affect successful backups made with previous versions.

v0.60 Release Notes

Better Version Support and WAL Improvements

Released April 19, 2015

Bug Fixes:

- Pushing duplicate WAL now generates an error. This worked before only if checksums were disabled.

Features:

- Database System IDs are used to make sure that all WAL in an archive matches up. This should help prevent misconfigurations that send WAL from multiple clusters to the same archive.

Additional Notes

Test Suite Features:

- Regression tests working back to PostgreSQL 8.3.

v0.50 Release Notes

Restore and Much More

Released March 25, 2015

Bug Fixes:

- Fixed broken checksums and now they work with normal and resumed backups. Finally realized that checksums and checksum deltas should be functionally separated and this simplified a number of things. Issue #28 has been created for checksum deltas.
- Fixed an issue where a backup could be resumed from an aborted backup that didn't have the same type and prior backup.

Features:

- Added restore functionality.
- All options can now be set on the command-line making pg_backrest.conf optional.
- De/compression is now performed without threads and checksum/size is calculated in stream. That means file checksums are no longer optional.
- Added option –no-start-stop to allow backups when Postgres is shut down. If postmaster.pid is present then –force is required to make the backup run (though if Postgres is running an inconsistent backup will likely be created). This option was added primarily for the purpose of unit testing, but there may be applications in the real world as well.
- Checksum for backup.manifest to detect a corrupted/modified manifest.
- Link latest always points to the last backup. This has been added for convenience and to make restores simpler.

Additional Notes

Test Suite Features:

- More comprehensive unit tests in all areas.

v0.30 Release Notes

Core Restructuring and Unit Tests

Released October 5, 2014

Additional Notes

Documentation Features:

- Added much needed documentation

Test Suite Features:

- Fairly comprehensive unit tests for all the basic operations. More work to be done here for sure, but then there is always more work to be done on unit tests.

v0.19 Release Notes

Improved Error Reporting/Handling

Released May 13, 2014

Bug Fixes:

- Found and squashed a nasty bug where file_copy() was defaulted to ignore errors. There was also an issue in file_exists() that was causing the test to fail when the file actually did exist. Together they could have resulted in a corrupt backup with no errors, though it is very unlikely.

v0.18 Release Notes

Return Soft Error When Archive Missing

Released April 13, 2014

Bug Fixes:

- The archive-get command now returns a 1 when the archive file is missing to differentiate from hard errors (ssh connection failure, file copy error, etc.) This lets PostgreSQL know that that the archive stream has terminated normally. However, this does not take into account possible holes in the archive stream. (*Reported by Stephen Frost.*)

v0.17 Release Notes

Warn When Archive Directories Cannot Be Deleted

Released April 3, 2014

Bug Fixes:

- If an archive directory which should be empty could not be deleted backrest was throwing an error. There's a good fix for that coming, but for the time being it has been changed to a warning so processing can continue. This was impacting backups as sometimes the final archive file would not get pushed if the first archive file had been in a different directory (plus some bad luck).

v0.16 Release Notes

RequestTTY=yes for SSH Sessions

Released April 1, 2014

Bug Fixes:

- Added RequestTTY=yes to ssh sessions. Hoping this will prevent random lockups.

v0.15 Release Notes

Added archive-get

Released March 29, 2014

Features:

- Added archive-get functionality to aid in restores.
- Added option to force a checkpoint when starting the backup, start-fast=y .

v0.11 Release Notes

Minor Fixes

Released March 26, 2014

Bug Fixes:

- Removed master_stderr_discard option on database SSH connections. There have been occasional lockups and they could be related to issues originally seen in the file code. (*Reported by Stephen Frost.*)
- Changed lock file conflicts on backup and expire commands to ERROR . They were set to DEBUG due to a copy-and-paste from the archive locks.

v0.10 Release Notes

Backup and Archiving are Functional

Released March 5, 2014

Features:

- No restore functionality, but the backup directories are consistent PostgreSQL data directories. You'll need to either uncompress the files or turn off compression in the backup. Uncompressed backups on a ZFS (or similar) filesystem are a good option because backups can be restored locally via a snapshot to create logical backups or do spot data recovery.
- Archiving is single-threaded. This has not posed an issue on our multi-terabyte databases with heavy write volume. Recommend a large WAL volume or to use the async option with a large volume nearby.
- Backups are multi-threaded, but the Net::OpenSSH library does not appear to be 100% thread-safe so it will very occasionally lock up on a thread. There is an overall process timeout that resolves this issue by killing the process. Yes, very ugly.
- Checksums are lost on any resumed backup. Only the final backup will record checksum on multiple resumes. Checksums from previous backups are correctly recorded and a full backup will reset everything.
- The backup.manifest is being written as Storable because Config::IniFile does not seem to handle large files well. Would definitely like to save these as human-readable text.

Additional Notes

Documentation Features:

- Absolutely no documentation (outside the code). Well, excepting these release notes.