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. Note that only 64-bit distributions are currently supported due to 64-bit operations in the Perl code. 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).

Installation

pgBackRest is written in Perl which is included with Debian/Ubuntu by default. The DBD::Pg module must also be installed.

db-master Install the DBD::Pg module

```
sudo apt-get install libdbd-pg-perl
```

Debian/Ubuntu packages for pgBackRest are available, but if they are not provided on your distribution/version it is easy to download the source and install manually.

db-master Download version 1.12 of pgBackRest

```
wget -q -O - \
       https://github.com/pgbackrest/pgbackrest/archive/release/1.12.tar.gz | \
       tar zx -C ~
```

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-master 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
```

The new version can now be installed.

db-master Install pgBackRest

```
sudo cp -r ~/pgbackrest-release-1.12/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 ~/pgbackrest-release-1.12/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
```

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-master Make sure the installation worked

```
sudo -u postgres pgbackrest
```

```
pgBackRest 1.12 - 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.
    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-master 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)
  socket /var/run/postgresql
  port   5432
```

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-master : /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-master : /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-master : /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-master 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-master : /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-master : /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 master cluster.

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

db-master 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) then the archive-timeout option should be increased.

Configure Retention

pgBackRest expires backups based on retention options.

db-master : /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.

Create the Stanza

To create the required stanza data stanza-create must be run on the host where the repository is located. The check command is invoked for PostgreSQL versions >= 9.1 to ensure archiving and backups are also properly configured. For older versions of PostgreSQL , it is recommended that activity be generated by the user if there have been no writes since the last xlog switch and then the check command run manually.

db-master 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 1.12: --db-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --repo-path=/var/lib/pgbackrest --stanza=demo
P00   INFO: switch xlog 000000010000000000000001
P00   INFO: WAL segment 000000010000000000000001 successfully stored in the archive at
   '/var/lib/pgbackrest/archive/demo/9.4-1/0000000100000000/000000010000000000000001-fb8faedead9cfbef
```

```
P00   INFO: successfully created 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.

Note that pg_create_restore_point('pgBackRest Archive Check') and pg_switch_xlog() 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.

db-master Check the configuration

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

```
P00   INFO: check command begin 1.12: --db-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --repo-path=/var/lib/pgbackrest --stanza=demo
P00   INFO: switch xlog 000000010000000000000002
```

```
P00   INFO: WAL segment 000000010000000000000002 successfully stored in the archive at
   '/var/lib/pgbackrest/archive/demo/9.4-1/0000000100000000/000000010000000000000002-35b7cd7d7bfbffc7
```

```
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-master Backup the demo cluster

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

```
P00   INFO: backup command begin 1.12: --db-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --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
   2016-12-13 00:10:04": backup begins after the next regular checkpoint completes
P00   INFO: backup start archive = 000000010000000000000003, lsn = 0/3000028
       [filtered 759 lines of output]
P01   INFO: local process 1 stop for db-1
P01   INFO: backup file /var/lib/postgresql/9.4/demo/base/1/11885 (0B, 100%)
```

```
P00   INFO: full backup size = 19.3MB
```

```
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]
```

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-master Differential backup of the demo cluster

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

```
       [filtered 6 lines of output]
P01   INFO: local process 1 stop for db-1
P01   INFO: backup file /var/lib/postgresql/9.4/demo/backup_label (236B, 100%) checksum
   6ebc1a88d42f30d35289b8190d5519243da8e36f
```

```
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 = 000000010000000000000004, lsn = 0/40000F0
       [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-master Get info for the demo cluster

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

```
stanza: demo
    status: ok

    full backup: 20161213-001009F

        start / stop timestamp: 2016-12-13 00:10:04 / 2016-12-13 00:10:09
        database size: 19.3MB, backup size: 19.3MB
        repository size: 2.2MB, repository backup size: 2.2MB

    diff backup: 20161213-001009F_20161213-001012D

        start / stop timestamp: 2016-12-13 00:10:09 / 2016-12-13 00:10:12
        database size: 19.3MB, backup size: 8.2KB
        repository size: 2.2MB, repository backup size: 344B
        backup reference list: 20161213-001009F
```

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 ' start / stop timestamp 'defines the time period when the backup ran. The' stop timestamp ' 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 ' 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-master 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-master 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-master Remove old files from demo cluster

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

db-master 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-master 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 1.12: --db-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --repo-path=/var/lib/pgbackrest
   --retention-full=2 --stanza=demo --type=incr
P00   INFO: last backup label = 20161213-001009F_20161213-001012D, version = 1.12
```

```
P00   INFO: execute exclusive pg_start_backup() with label "pgBackRest backup started at
   2016-12-13 00:10:22": backup begins after the next regular checkpoint completes
```

```
P00   INFO: backup start archive = 000000020000000000000006, lsn = 0/6000028
P01   INFO: local process 1 start for host db-1
       [filtered 15 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-master : /etc/pgbackrest.conf Enable the start-fast option

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

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

db-master 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 1.12: --db-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --repo-path=/var/lib/pgbackrest
   --retention-full=2 --stanza=demo --start-fast --type=incr
P00   INFO: last backup label = 20161213-001009F_20161213-001026I, version = 1.12
```

```
P00   INFO: execute exclusive pg_start_backup() with label "pgBackRest backup started at
   2016-12-13 00:10:27": backup begins after the requested immediate checkpoint completes
```

```
P00   INFO: backup start archive = 000000020000000000000007, lsn = 0/7000028
P01   INFO: local process 1 start for host db-1
       [filtered 10 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-master Revoke write privileges in the pgBackRest repository and attempt a backup

```
sudo chmod 550 /var/lib/pgbackrest/temp
```

```
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
   2016-12-13 00:10:30": backup begins after the requested immediate checkpoint completes
P00   INFO: backup start archive = 000000020000000000000008, lsn = 0/8000028
```

```
P00  ERROR: [122]: unable to create /var/lib/pgbackrest/temp/demo.tmp: Permission denied
```

```
P00   INFO: backup command end: aborted with exception [122]
```

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

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

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

```
P00   INFO: backup command begin 1.12: --db-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --repo-path=/var/lib/pgbackrest
   --retention-full=2 --stanza=demo --start-fast --type=incr
P00   INFO: last backup label = 20161213-001009F_20161213-001030I, version = 1.12
P00   INFO: execute exclusive pg_start_backup() with label "pgBackRest backup started at
   2016-12-13 00:10:33": backup begins after the requested immediate checkpoint completes
```

```
P00  ERROR: [132]: 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 2016-12-13 00:10:33', 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-master : /etc/pgbackrest.conf Enable the stop-auto option

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

[global]
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-master Perform an incremental backup

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

```
P00   INFO: backup command begin 1.12: --db-path=/var/lib/postgresql/9.4/demo
   --log-level-console=info --log-level-stderr=off --repo-path=/var/lib/pgbackrest
   --retention-full=2 --stanza=demo --start-fast --stop-auto --type=incr
P00   INFO: last backup label = 20161213-001009F_20161213-001030I, version = 1.12
```

```
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
   2016-12-13 00:10:35": backup begins after the requested immediate checkpoint completes
```

```
P00   INFO: backup start archive = 000000020000000000000009, lsn = 0/9000028
P01   INFO: local process 1 start for host db-1
       [filtered 10 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 to make the backup consistent to be archived. This wait time is governed by the 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-master : /etc/pgbackrest.conf Configure retention-full

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

[global]
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-master Perform a full backup

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

```
       [filtered 768 lines of output]
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 1.12: --log-level-console=detail --log-level-stderr=off
   --repo-path=/var/lib/pgbackrest --retention-archive=2 --retention-full=2 --stanza=demo
```

```
P00 DETAIL: archive retention on backup 20161213-001009F, start = 000000010000000000000003
```

```
P00 DETAIL: no archive to remove
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-master Perform a full backup

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

```
       [filtered 768 lines of output]
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 1.12: --log-level-console=info --log-level-stderr=off
   --repo-path=/var/lib/pgbackrest --retention-archive=2 --retention-full=2 --stanza=demo
```

```
P00   INFO: expire full backup set: 20161213-001009F, 20161213-001009F_20161213-001012D,
   20161213-001009F_20161213-001026I, 20161213-001009F_20161213-001030I,
   20161213-001009F_20161213-001039I
```

```
P00   INFO: remove expired backup 20161213-001009F_20161213-001039I
P00   INFO: remove expired backup 20161213-001009F_20161213-001030I
       [filtered 3 lines of output]
```

The 20161213-001009F full backup is expired and archive retention is based on the 20161213-001046F 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-master : /etc/pgbackrest.conf Configure retention-diff

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

[global]
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-master 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-master Perform a differential backup

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

```
       [filtered 12 lines of output]
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 1.12: --log-level-console=info --log-level-stderr=off
   --repo-path=/var/lib/pgbackrest --retention-archive=2 --retention-diff=1 --retention-full=2
   --stanza=demo
```

```
P00   INFO: expire diff backup set: 20161213-001051F_20161213-001055D,
   20161213-001051F_20161213-001059I
```

```
P00   INFO: remove expired backup 20161213-001051F_20161213-001059I
P00   INFO: remove expired backup 20161213-001051F_20161213-001055D
```

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-master : /etc/pgbackrest.conf Configure retention-diff

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

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

db-master Perform differential backup

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

```
       [filtered 9 lines of output]
P00   INFO: execute exclusive pg_stop_backup() and wait for all WAL segments to archive
P00   INFO: backup stop archive = 000000020000000000000012, lsn = 0/120000F0
```

```
P00   INFO: new backup label = 20161213-001051F_20161213-001107D
```

```
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 1.12: --log-level-console=info --log-level-stderr=off
   --repo-path=/var/lib/pgbackrest --retention-archive=2 --retention-diff=2 --retention-full=2
   --stanza=demo
```

db-master 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 1.12: --log-level-console=detail --log-level-stderr=off
   --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 20161213-001046F, start = 0000000200000000000000A, stop =
   0000000200000000000000A
P00 DETAIL: archive retention on backup 20161213-001051F, start = 0000000200000000000000B, stop =
   0000000200000000000000B
```

```
P00 DETAIL: archive retention on backup 20161213-001051F_20161213-001103D, start =
   0000000200000000000000E, stop = 0000000200000000000000E
```

```
P00 DETAIL: archive retention on backup 20161213-001051F_20161213-001107D, start =
   000000020000000000000012
```

```
P00 DETAIL: remove archive: start = 0000000200000000000000C, stop = 0000000200000000000000D
P00 DETAIL: remove archive: start = 0000000200000000000000F, stop = 000000020000000000000011
```

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

The 20161213-001051F_20161213-001103D 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 20161213-001051F_20161213-001103D but before 20161213-001051F_20161213-001107D are removed. WAL segments generated after the new backup 20161213-001051F_20161213-001107D 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-master 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 694 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 84 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 (copied last to ensure aborted restores cannot be started)

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

db-master 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-master 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-master 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-master 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-master 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-master 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-master 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-master 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-master 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-master 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-master 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-master Get the time from PostgreSQL

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

```
2016-12-13 00:11:34.531619+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-master 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-master Stop PostgreSQL , restore the demo cluster to 2016-12-13 00:11:34.531619+00 , and display recovery.conf

```
sudo pg_ctlcluster 9.4 demo stop
```

```
sudo -u postgres pgbackrest --stanza=demo --delta \
       --type=time "--target=2016-12-13 00:11:34.531619+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 = '2016-12-13 00:11:34.531619+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-master 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-master 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 2016-12-13 00:11:30 UTC
LOG:  creating missing WAL directory "pg_xlog/archive_status"
```

```
LOG:  starting point-in-time recovery to 2016-12-13 00:11:34.531619+00
```

```
LOG:  restored log file "00000004.history" from archive
LOG:  restored log file "000000040000000000000017" from archive
       [filtered 2 lines of output]
LOG:  incomplete startup packet
LOG:  restored log file "000000040000000000000018" from archive
```

```
LOG:  recovery stopping before commit of transaction 686, time 2016-12-13 00:11:34.736878+00
```

```
LOG:  redo done at 0/180157F0
```

```
LOG:  last completed transaction was at log time 2016-12-13 00:11:34.368893+00
```

```
LOG:  selected new timeline ID: 5
LOG:  restored log file "00000004.history" from archive
       [filtered 4 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-master 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-master 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=2016-12-13 00:11:34.531619+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-master 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 2016-12-13 00:11:44 UTC
LOG:  creating missing WAL directory "pg_xlog/archive_status"
```

```
LOG:  starting point-in-time recovery to 2016-12-13 00:11:34.531619+00
```

```
LOG:  restored log file "00000005.history" from archive
LOG:  restored log file "000000050000000000000019" from archive
LOG:  redo starts at 0/19000028
```

```
LOG:  consistent recovery state reached at 0/190000F0
```

```
LOG:  incomplete startup packet
LOG:  redo done at 0/190000F0
       [filtered 10 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-master Get backup info for the demo cluster

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

```
stanza: demo
    status: ok

    full backup: 20161213-001046F
        start / stop timestamp: 2016-12-13 00:10:40 / 2016-12-13 00:10:46
        database size: 19.3MB, backup size: 19.3MB
        repository size: 2.2MB, repository backup size: 2.2MB

    full backup: 20161213-001051F
        start / stop timestamp: 2016-12-13 00:10:46 / 2016-12-13 00:10:51
        database size: 19.3MB, backup size: 19.3MB
        repository size: 2.2MB, repository backup size: 2.2MB

    diff backup: 20161213-001051F_20161213-001107D
        start / stop timestamp: 2016-12-13 00:11:04 / 2016-12-13 00:11:07
        database size: 19.3MB, backup size: 8.2KB
        repository size: 2.2MB, repository backup size: 346B
        backup reference list: 20161213-001051F

    incr backup: 20161213-001051F_20161213-001120I
        start / stop timestamp: 2016-12-13 00:11:15 / 2016-12-13 00:11:20
        database size: 31.8MB, backup size: 12.7MB
        repository size: 3.7MB, repository backup size: 1.5MB
        backup reference list: 20161213-001051F
```

```
    diff backup: 20161213-001051F_20161213-001134D
```

```
        start / stop timestamp: 2016-12-13 00:11:29 / 2016-12-13 00:11:34
        database size: 25.7MB, backup size: 6.5MB
        repository size: 3MB, repository backup size: 793.1KB
        backup reference list: 20161213-001051F

    incr backup: 20161213-001051F_20161213-001146I
        start / stop timestamp: 2016-12-13 00:11:43 / 2016-12-13 00:11:46
        database size: 25.6MB, backup size: 1.9MB
        repository size: 3MB, repository backup size: 215.3KB
```

```
        backup reference list: 20161213-001051F, 20161213-001051F_20161213-001134D
```

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

db-master 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=2016-12-13 00:11:34.531619+00" \
       --set=20161213-001051F_20161213-001134D 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-master 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 2016-12-13 00:11:30 UTC
LOG:  creating missing WAL directory "pg_xlog/archive_status"
```

```
LOG:  starting point-in-time recovery to 2016-12-13 00:11:34.531619+00
```

```
LOG:  restored log file "00000004.history" from archive
FATAL:  the database system is starting up
       [filtered 3 lines of output]
LOG:  incomplete startup packet
LOG:  restored log file "000000040000000000000018" from archive
```

```
LOG:  recovery stopping before commit of transaction 686, time 2016-12-13 00:11:34.736878+00
```

```
LOG:  redo done at 0/180157F0
```

```
LOG:  last completed transaction was at log time 2016-12-13 00:11:34.368893+00
```

```
LOG:  restored log file "00000005.history" from archive
LOG:  restored log file "00000006.history" from archive
       [filtered 6 lines of output]
```

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 and Configuration

For this example a new host named backup has been created to store the cluster backups. Follow the instructions in Installation to install pgBackRest and Create the Repository to create the pgBackRest repository. The backup host must also be configured with the db-master host/user and database path. The master database 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-master
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-master : /etc/pgbackrest.conf Configure backup-host / backup-user

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

[global]
backup-host=backup
backup-user=backrest
repo-path=/var/lib/pgbackrest

The repository directory will also be removed from the database host. It will not be used anymore so leaving it around may be confusing later on.

db-master Remove repository now that it will be located on the database server

```
sudo find /var/lib/pgbackrest -delete
```

Commands are run the same as on a single host configuration except that the backup and expire command are run from the backup host and all other commands are run from the database host.

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-master 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-master 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 to improve throughput. WAL segments are temporarily stored in a local queue on the database server, specified by the spool-path option, before being transferred to the repository on the backup server.

The spool directory is created to hold the WAL segments while they are waiting to be (optionally) compressed and transferred.

db-master 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.

db-master : /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
repo-path=/var/lib/pgbackrest
spool-path=/var/spool/pgbackrest

The check command ensures that asynchronous archiving is working.

db-master Check asynchronous archiving

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

```
P00   INFO: check command begin 1.12: --backup-host=backup --backup-user=backrest
   --db-path=/var/lib/postgresql/9.4/demo --log-level-console=info --log-level-stderr=off
   --repo-path=/var/lib/pgbackrest --stanza=demo
P00   INFO: switch xlog 000000080000000000000001D
```

```
P00   INFO: WAL segment 000000080000000000000001D successfully stored in the archive at
   '/var/lib/pgbackrest/archive/demo/9.4-1/0000000800000000/000000080000000000000001D-bc22ef18eca39a16
```

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

Starting and Stopping

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

db-master 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: [137]: remote process terminated on db-master host: 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-master 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-master Start the pgBackRest services

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

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

db-master 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: [137]: remote process terminated on db-master host: stop file exists for stanza demo
```

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

db-master 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 master. The standbys are useful for balancing reads and to provide redundancy in case the master host fails.

Hot Standby

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

A new host named db-standby will be created to run the standby. Follow the instructions in Installation to install pgBackRest , Setup Demo Cluster to setup the demo cluster, and Create the Repository to create the pgBackRest repository on the db-standby host.

pgBackRest configuration is very similar to db-master 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
repo-path=/var/lib/pgbackrest

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:  database system was interrupted; last known up at 2016-12-13 00:12:25 UTC
LOG:  creating missing WAL directory "pg_xlog/archive_status"
```

```
LOG:  entering standby mode
```

```
LOG:  restored log file "00000008.history" from archive
LOG:  incomplete startup packet
LOG:  restored log file "000000080000000000000001C" from archive
LOG:  redo starts at 0/1C000028
LOG:  consistent recovery state reached at 0/1C0000F0
```

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

```
LOG:  restored log file "000000080000000000000001D" from archive
```

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

db-master Create a new table on the master

```
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-master .

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-master Call pg_switch_xlog()

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

```
 pg_switch_xlog |              now
----------------+-------------------------------
 0/1E0199A8     | 2016-12-13 00:12:49.435722+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 | 2016-12-13 00:12:52.197946+00

(1 row)
```

Streaming Replication

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

Streaming replication requires a user with the replication privilege.

db-master 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-master . A reload will be required after modifying the pg_hba.conf file.

db-master Create pg_hba.conf entry for replication user

```
sudo -u postgres sh -c 'echo \
       "host    replication     replicator      172.17.0.4/32           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 master 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=172.17.0.2 port=5432 user=replicator

[global]
backup-host=backup
repo-path=/var/lib/pgbackrest
```

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 \
       "172.17.0.2:*: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=172.17.0.2 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 9 lines of output]
LOG:  restored log file "000000080000000000000001D" from archive
LOG:  restored log file "000000080000000000000001E" from archive
```

```
LOG:  started streaming WAL from primary at 0/1F000000 on timeline 8
```

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

db-master Create a new table on the master

```
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 | 2016-12-13 00:13:00.336186+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 | 2016-12-13 00:13:00.588365+00
(1 row)
```

Backup from a Standby

pgBackRest can perform backups on a standby instead of the master. Standby backups require the db-standby host to be configured and the backup-standby option enabled.

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

```
[demo]
db1-host=db-master
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
repo-path=/var/lib/pgbackrest
retention-full=2
start-fast=y
```

Both the master 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 master. The database hosts can be configured in any order. pgBackRest will automatically determine which is the master 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
   2016-12-13 00:13:01": backup begins after the requested immediate checkpoint completes
P00   INFO: backup start archive = 000000080000000000000020, lsn = 0/20000028

P00   INFO: wait for replay on the standby to reach 0/20000028
P00   INFO: replay on the standby reached 0/200000C8

P01   INFO: local process 1 start for host db-1
P02   INFO: local process 2 start for host db-2

P01   INFO: backup file db-master:/var/lib/postgresql/9.4/demo/global/pg_control (8KB, 0%)
   checksum 26f5b5dacaf3af21c173b2248aca00b288c3c269

P01   INFO: local process 1 stop for db-1

P01   INFO: backup file db-master:/var/lib/postgresql/9.4/demo/backup_label (238B, 0%) checksum
   f515af2e1d494250c2ad55f52485d968bae6306b

P02   INFO: backup file db-standby:/var/lib/postgresql/9.4/demo/base/12139/12007 (392KB, 20%)
   checksum 65d1be267782bcaa474c2d3d821218e150c8fbb8
P02   INFO: backup file db-standby:/var/lib/postgresql/9.4/demo/base/12139/11889 (344KB, 38%)
   checksum 63e4a4ca52f0c904b3ddebd3c2856544dc63ca04
      [filtered 37 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-master host.

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

— 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.

Command 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.

```
default: [INSTALL-PATH]/pgbackrest
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 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
```

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
allowed: 16384-8388608
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
```

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
```

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 master 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
```

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
```

Stanza Option (–stanza) Defines a 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 must will be 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
```

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 MB Option (–archive-max-mb) Limit size of the local asynchronous archive queue when archive-async=y .

After the limit is reached, the following will happen:

1. pgBackRest will notify Postgres that the archive was successfully backed up, then DROP IT.
2. An error will be logged to the console and also to the Postgres log.
3. A stop file will be written in the lock directory and no more archive files will be backed up until it is removed.

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.

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.

To start normal archiving again you'll need to remove the stop file which will be located at ${repo-path}/lock/${stanza}-archive.stop where ${repo-path} is the path set in the general section, and ${stanza} is the backup stanza.

```
example: --archive-max-mb=1024
```

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.

```
default: [INSTALL-PATH]/pgbackrest
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 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
```

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
allowed: 16384-8388608
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
```

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
```

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 master 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
```

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
```

Repository Sync Option (–repo-sync) Sync directories in repository.

Syncs directories when writing to the repository. Not all file systems support directory syncs (e.g., NTFS) so this option allows them to be disabled.

```
default: y
example: --no-repo-sync
```

Spool Path Option (–spool-path) Path where WAL segments are spooled during async archiving.

When asynchronous archiving is enabled pgBackRest needs a local directory to store WAL segments before they are compressed and moved to the repository. Depending on the volume of WAL generated this directory could become very large so be sure to plan accordingly.

The max-archive-mb option can be used to limit the amount of WAL that will be spooled locally.

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

Stanza Option (–stanza) Defines a 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 must will be 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
```

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
```

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.

```
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 or premature expiration in the WAL segment archive by storing the WAL segments directly in the backup. PITR won't be possible without the WAL segment archive and this option also consumes more space.

Even though WAL segments will be restored with the backup, PostgreSQL will ignore them if a recovery.conf file exists and instead use archive_command to fetch WAL segments. Specifying type=none when restoring will not create recovery.conf and force PostgreSQL to use the WAL segments in pg_xlog. This will get the database cluster to a consistent state.

```
default: n
example: --archive-copy
```

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.

```
default: [INSTALL-PATH]/pgbackrest
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 from Standby Option (–backup-standby) Backup from the standby cluster.

Enable backup from standby to reduce load on the master cluster. This option requires that both the master and standby hosts be configured.

```
default: n
example: --backup-standby
```

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
```

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: 1-999999999
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: 1-999999999
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: 1-999999999
example: --retention-full=2
```

General Options

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

Set maximum time, in seconds, to wait for WAL segments to reach the archive. The timeout applies to the check command and to the backup command when waiting for WAL segments required to make the backup consistent to be archived.

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

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
allowed: 16384-8388608
example: --buffer-size=32768
```

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
```

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
```

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
```

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
```

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 directory is copied as-is and archive-check is automatically disabled for the backup.

```
default: y
example: --no-online
```

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 master 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
```

Repository Symlink Creation Option (–repo-link) Create convenience symlinks in repository.

Creates the convenience link latest in the stanza directory and internal tablespace symlinks in each backup directory. The internal tablespace symlinks allow clusters to be brought up manually in-place using filesystem snapshots as long as the backup is not compressed.

This option should be disabled when the repository is located on a filesystem that does not support symlinks. No pgBackRest functionality will be affected, but certain manual operations on the repository may be less convenient.

```
default: y
example: --no-repo-link
```

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
```

Repository Sync Option (–repo-sync) Sync directories in repository.

Syncs directories when writing to the repository. Not all file systems support directory syncs (e.g., NTFS) so this option allows them to be disabled.

```
default: y
example: --no-repo-sync
```

Stanza Option (–stanza) Defines a 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 must will be 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
```

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.

```
default: [INSTALL-PATH]/pgbackrest
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 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.

Note that pg_create_restore_point('pgBackRest Archive Check') and pg_switch_xlog() 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.

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.

```
default: y
example: --no-archive-check
```

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.

```
default: [INSTALL-PATH]/pgbackrest
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 from Standby Option (–backup-standby) Backup from the standby cluster.

Enable backup from standby to reduce load on the master cluster. This option requires that both the master and standby hosts be configured.

```
default: n
example: --backup-standby
```

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
```

General Options

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

Set maximum time, in seconds, to wait for WAL segments to reach the archive. The timeout applies to the check command and to the backup command when waiting for WAL segments required to make the backup consistent to be archived.

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

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
allowed: 16384-8388608
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
```

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
```

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
```

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 directory is copied as-is and archive-check is automatically disabled for the backup.

```
default: y
example: --no-online
```

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

Sets the timeout, in seconds, that the master 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
```

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
```

Stanza Option (–stanza) Defines a 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 must will be 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
```

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.

```
default: [INSTALL-PATH]/pgbackrest
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 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

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
```

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: 1-999999999
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: 1-999999999
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: 1-999999999
example: --retention-full=2
```

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
```

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
```

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
```

Stanza Option (–stanza) Defines a 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 must will be 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
```

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.

```
default: [INSTALL-PATH]/pgbackrest
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
```

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

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.

```
default: [INSTALL-PATH]/pgbackrest
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 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
```

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
allowed: 16384-8388608
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
```

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
```

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

Sets the timeout, in seconds, that the master 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
```

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
```

Stanza Option (–stanza) Defines a 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 must will be 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
```

Restore Command (restore)

This command is generally run manually, but there are instances where it might be automated.

Command 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.

```
default: [INSTALL-PATH]/pgbackrest
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 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
```

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
```

Lock Option (–lock) Create a lock so restores on a stanza cannot run simultaneously.

Locking during restores is enabled by default but can be disabled using –no-lock. Be *very* careful when disabling this option because simultaneous restores to a single path might result in a corrupt cluster.

```
default: y
example: --no-lock
```

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 . 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
allowed: 16384-8388608
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
```

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
```

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 master 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
```

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
```

Stanza Option (–stanza) Defines a 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 must will be 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
```

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)

Must be run during pgBackRest installation on the host where the repository is located. The check command will also be invoked if the PostgreSQL database version is greater than or equal to 9.1.

Command 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.

```
default: [INSTALL-PATH]/pgbackrest
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 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
```

General Options

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

Set maximum time, in seconds, to wait for WAL segments to reach the archive. The timeout applies to the check command and to the backup command when waiting for WAL segments required to make the backup consistent to be archived.

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

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
allowed: 16384-8388608
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
```

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
```

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 master 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
```

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
```

Repository Sync Option (–repo-sync) Sync directories in repository.

Syncs directories when writing to the repository. Not all file systems support directory syncs (e.g., NTFS) so this option allows them to be disabled.

```
default: y
example: --no-repo-sync
```

Stanza Option (–stanza) Defines a 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
```

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.

```
default: [INSTALL-PATH]/pgbackrest
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 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.

Command 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.

```
default: [INSTALL-PATH]/pgbackrest
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 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
```

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
```

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
```

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
```

Stanza Option (–stanza) Defines a 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 must will be 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
```

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.

```
default: [INSTALL-PATH]/pgbackrest
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
```

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

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.

```
default: [INSTALL-PATH]/pgbackrest
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 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
```

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
```

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
```

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
```

Stanza Option (–stanza) Defines a 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 must will be 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
```

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.

```
default: [INSTALL-PATH]/pgbackrest
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
```

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)

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 MB Option (–archive-max-mb)

Limit size of the local asynchronous archive queue when archive-async=y .

After the limit is reached, the following will happen:

1. pgBackRest will notify Postgres that the archive was successfully backed up, then DROP IT.
2. An error will be logged to the console and also to the Postgres log.
3. A stop file will be written in the lock directory and no more archive files will be backed up until it is removed.

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.

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.

To start normal archiving again you'll need to remove the stop file which will be located at ${repo-path}/lock/${stanza}-archive.stop where ${repo-path} is the path set in the general section, and ${stanza} is the backup stanza.

```
example: archive-max-mb=1024
```

Backup Options (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.

```
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 or premature expiration in the WAL segment archive by storing the WAL segments directly in the backup. PITR won't be possible without the WAL segment archive and this option also consumes more space.

Even though WAL segments will be restored with the backup, PostgreSQL will ignore them if a recovery.conf file exists and instead use archive_command to fetch WAL segments. Specifying type=none when restoring will not create recovery.conf and force PostgreSQL to use the WAL segments in pg_xlog. This will get the database cluster to a consistent state.

```
default: n
example: archive-copy=y
```

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.

```
default: [INSTALL-PATH]/pgbackrest
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 from Standby Option (–backup-standby)

Backup from the standby cluster.

Enable backup from standby to reduce load on the master cluster. This option requires that both the master and standby hosts be configured.

```
default: n
example: backup-standby=y
```

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
```

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)

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: 1-999999999
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: 1-999999999
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: 1-999999999
example: retention-full=2
```

General Options (general)

Archive Timeout Option (–archive-timeout)

Archive timeout.

Set maximum time, in seconds, to wait for WAL segments to reach the archive. The timeout applies to the check command and to the backup command when waiting for WAL segments required to make the backup consistent to be archived.

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

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
allowed: 16384-8388608
example: buffer-size=32768
```

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
```

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
```

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
```

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 master 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
```

Repository Symlink Creation Option (–repo-link)

Create convenience symlinks in repository.

Creates the convenience link latest in the stanza directory and internal tablespace symlinks in each backup directory. The internal tablespace symlinks allow clusters to be brought up manually in-place using filesystem snapshots as long as the backup is not compressed.

This option should be disabled when the repository is located on a filesystem that does not support symlinks. No pgBackRest functionality will be affected, but certain manual operations on the repository may be less convenient.

```
default: y
example: repo-link=n
```

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
```

Repository Sync Option (–repo-sync)

Sync directories in repository.

Syncs directories when writing to the repository. Not all file systems support directory syncs (e.g., NTFS) so this option allows them to be disabled.

```
default: y
example: repo-sync=n
```

Spool Path Option (–spool-path)

Path where WAL segments are spooled during async archiving.

When asynchronous archiving is enabled pgBackRest needs a local directory to store WAL segments before they are compressed and moved to the repository. Depending on the volume of WAL generated this directory could become very large so be sure to plan accordingly.

The max-archive-mb option can be used to limit the amount of WAL that will be spooled locally.

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

Log Options (log)

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 must will be 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
```

Restore Options (restore)

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)

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.

```
default: [INSTALL-PATH]/pgbackrest
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 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 interface, naming, or the repository format.

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

v1.12 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.*)

Refactoring:

- Abstracted code to determine which database cluster is the master and which are standbys. (*Contributed by Cynthia Shang.*)
- Improved consistency and flexibility of the protocol layer by using JSON for all messages.
- File copy protocol now accepts a function that can do additional processing on the copy buffers and return a result to the calling process.
- Improved IO->bufferRead to always return requested number of bytes until EOF.
- 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.
- Simplified the result hash of File->manifest() , Db->tablespaceMapGet() , and Db->databaseMapGet() .
- Improved errors returned from child processes by removing redundant error level and code.
- Code cleanup in preparation for improved stanza-create command. (*Contributed by Cynthia Shang.*)
- Improved parameter/result logging in debug/trace functions.

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.

- Allow a source to be included as a section so large documents can be broken up.
- Added section link support to Markdown output.
- Added list support to PDF output.
- Added include option to explicitly build sources (complements the exclude option though both cannot be used in the same invocation).
- Added keyword-add option to add keywords without overriding the default keyword.
- Added debug option to doc.pl to easily add the debug keyword to documentation builds.
- Added pre option to doc.pl to easily add the pre keyword to documentation builds.
- Builds in release.pl now remove all docker containers to get consistent IP address assignments.

Documentation Refactoring:

- Improvements to markdown rendering.
- Remove code dependency on project variable, instead use title param.

Test Suite Bug Fixes:

- Removed erroneous –no-config option in help test module.

Test Suite Refactoring:

- Update control and WAL test files to 9.4 with matching system identifiers. (*Contributed by Cynthia Shang.*)
- Improved exception handling in file unit tests.
- Changed the –no-fork test option to –fork with negation to match all other boolean parameters.
- Various improvements to validation of backup and restore.
- Add more realistic data files to synthetic backup and restore tests.

Supported Stable Releases

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.*)

Refactoring:

- Removed extraneous use lib directives from Perl modules. (*Suggested by Devrim Gündüz.*)

Additional Notes

Documentation Bug Fixes:

- Fixed missing variable replacements.
- Removed hard-coded host names from configuration file paths.

Documentation Features:

- Allow command-line length to be configured using cmd-line-len param.
- Added compact param to allow CSS to be embedded in HTML file.
- Added pretty param to produce HTML with proper indenting.
- Only generate HTML menu when required and don't require index page.
- Assign numbers to sections by default.
- VM mount points are now optional.

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.

Refactoring:

- 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.*)
- Added dev option to doc.pl to easily add the dev keyword to documentation builds.

Test Suite Bug Fixes:

- Fixed missing expect output for help module.

- Fixed broken vm-max option in test.pl .

Test Suite Features:

- Update CentOS/Debian package definitions.

Test Suite Refactoring:

- Regression tests can now be run as any properly-configured user, not just vagrant.
- Miminize TeXLive package list to save time during VM builds.

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.*)

Refactoring:

- Cache file log output until the file is created to create a more complete log.

Additional Notes

Documentation Refactoring:

- Show Process ID in output instead of filtering it out with the timestamp.

Test Suite Refactoring:

- Suppress "dpkg-reconfigure: unable to re-open stdin: No file or directory" warning in Vagrant VM build. (*Contributed by John Harvey.*)
- Show Process ID in expect logs instead of filtering it out with the timestamp.

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 master 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 master 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.

Refactoring:

- Refactor of protocol minions in preparation for the new local minion.
- Remove obsolete thread index variable from File() module.
- Changed temporary file names to consistently use the .pgbackrest.tmp extension even if the destination file is compressed or has an appended checksum.
- Improve ASSERT error handling, safely check eval blocks, and convert $@ to $EVAL_ERROR .

Additional Notes

Documentation Bug Fixes:

- Fixed minor documentation reproducibility issues related to binary paths.

Documentation Features:

- Documentation for archive retention. (*Contributed by Cynthia Shang.*)

Documentation Refactoring:

- Suppress TOC for unsupported versions of pgBackRest .

Test Suite Refactoring:

- New vagrant base box and make uid/gid selection for containers dynamic.

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 master.
- More flexible configuration for databases. Master and standby can both be configured on the backup server and pgBackRest will automatically determine which is the master. This means no configuration changes for backup are required after failing over from a master 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.

Refactoring:

- Simplify protocol creation and identifying which host is local/remote.
- Removed all OP_* function constants that were used only for debugging, not in the protocol, and replaced with ___PACKAGE___ .
- Improvements in Db module: separated out connect() function, allow executeSql() calls that do not return data, and improve error handling.

- 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.

Additional Notes

Documentation Refactoring:

- Improve host tag rendering.

Test Suite Refactoring:

- Refactor db version constants into a separate module.
- Update synthetic backup tests to PostgreSQL 9.4.

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 Refactoring:

- Moved the backlog from the pgBackRest website to the GitHub repository wiki. (*Contributed by Cynthia Shang.*)
- Improved rendering of spaces in code blocks.

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.

Refactoring:

- Enhancements to the protocol layer for improved reliability and error handling.
- 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.
- Exceptions are now passed back from threads as messages when possible rather than raised directly.
- Temp files created during backup are now placed in the same directory as the target file.
- Output lock file name when a lock cannot be acquired to aid in debugging.
- Reduce calls to protocolGet() in backup/restore.
- Suppress banners on SSH protocol connections.
- Improved remote error messages to identify the host where the error was raised.

Additional Notes

Documentation Features:

- Added release.pl to make releases reproducible. For now this only includes building and deploying documentation.
- 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.*)

Documentation Refactoring:

- HTML footer dates are statically created in English in order to be reproducible. (*Suggested by Adrian Vondendriesch.*)

Test Suite Bug Fixes:

- Fixed a version checking issue in test.pl .
- Fixed an issue where multi-threaded tests were not being run when requested.

Test Suite Refactoring:

- Reduce the frequency that certain tests are run to save time in regression.
- Disable control master for older OS versions where it is less stable.

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.

Refactoring:

- The pg_xlogfile_name() function is no longer used to construct WAL filenames from LSNs. While this function is convenient it is not available on a standby. Instead, the archive is searched for the LSN in order to find the timeline. If due to some misadventure the LSN appears on multiple timelines then an error will be thrown, whereas before this condition would have passed unnoticed.

- 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.
- 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.
- Changed version variable to a constant. It had originally been designed to play nice with a specific packaging tool but that tool was never used.

Additional Notes

Documentation Bug Fixes:

- Fixed DTD search path that did not work properly when –doc-path was used.
- Fixed pgBackRest -specific xml that was loaded for non- pgBackRest projects.
- Fixed section names being repeated in the info output when multiple –require options depended on the same sections.
- Fixed pgBackRest config sections being blank in the output when not loaded from cache.

Documentation Features:

- Allow hidden options to be added to a command. This allows certain commands (like apt-get) to be forced during the build without making that a part of the documentation.
- Allow command summaries to be inserted anywhere in the documentation to avoid duplication.
- Allow a static date to be used for documentation to generate reproducible builds. (*Suggested by Adrian Vondendriesch.*)
- Update TeX Live to 2016 version.
- Added documentation for asynchronous archiving to the user guide. (*Contributed by Cynthia Shang.*)

Documentation Refactoring:

- 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 .
- New, consolidated implementation for link rendering.
- PostgreSQL version is now a variable to allow multi-version documentation.

Test Suite Features:

- Obsolete containers are removed by the –vm-force option.

Test Suite Refactoring:

- Major refactor of the test suite to make it more modular and object-oriented. Multiple Docker containers can now be created for a single test to simulate more realistic environments. Tests paths have been renamed for clarity.
- Greatly reduced the quantity of Docker containers built by default. Containers are only built for PostgreSQL versions specified in db-minimal and those required to build documentation. Additional containers can be built with –db-version=all or by specifying a version, e.g. –db-version=9.4 .

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.*)

Refactoring:

- Log directory create and file open now using FileCommon functions which produce more detailed error messages on failure.

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.*)
- Added an execution cache so that documentation can be generated without setting up the full container environment. This is useful for packaging, keeps the documentation consistent for a release, and speeds up generation when no changes are made in the execution list.

Documentation Refactoring:

- 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.*)
- Remove function constants and pass strings directly to logDebugParam(). The function names were only used once so creating constants for them was wasteful.
- Lists can now be used outside of p and text tags for more flexible document structuring.

Test Suite Bug Fixes:

- Replaced overzealous perl -cW check which failed on Perl 5.22 with perl -cw .

Test Suite Features:

- Added Ubuntu 16.04 (Xenial) and Debian 8 (Jessie) to the regression suite.
- Upgraded doc/test VM to Ubuntu 16.04. This will help catch Perl errors in the doc code since it is not run across multiple distributions like the core and test code. It is also to be hoped that a newer kernel will make Docker more stable.

Test Suite Refactoring:

- Test release version against the executable using change-log.xml instead of CHANGELOG.md .

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 .

Unsupported 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.

Additional Notes

Documentation Bug Fixes:

- Fixed an issue where document generation failed because some OSs are not tolerant of having multiple installed versions of PostgreSQL . A separate VM is now created for each version. Also added a sleep after database starts during document generation to ensure the database is running before the next command runs. (*Reported by John Harvey.*)

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.

Additional Notes

Documentation Refactoring:

- Minor styling changes, clarifications and rewording in the user guide.

Test Suite Refactoring:

- The dev branch has been renamed to master and for the time being the master branch has renamed to release, though it will probably be removed at some point – thus ends the gitflow experiment for pgBackRest . It is recommended that any forks get re-forked and clones get re-cloned.

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.*)

Documentation Refactoring:

- The website, markdown, and command-line help are now all generated from the same XML source.

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.

Refactoring:

- Removed dependency on IO::String module.

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.

Refactoring:

- Renamed recovery-setting option and section to recovery-option to be more consistent with pgBackRest naming conventions.
- Code cleanup and refactoring to standardize on patterns that have evolved over time.
- 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.

Test Suite Features:

- Expiration tests are now synthetic rather than based on actual backups. This will allow development of more advanced expiration features.

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.

Refactoring:

- 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.
- Major refactoring of the protocol layer to support future development.

Additional Notes

Documentation Features:

- Split most of README.md out into USERGUIDE.md and CHANGELOG.md because it was becoming unwieldy. Changed most references to “database” in the user guide to “database cluster” for clarity.
- Changed most references to “database” in the user guide to “database cluster” for clarity.

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

Refactoring:

- 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.
- More options for regression tests and improved code to run in a variety of environments.

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.*)

Refactoring:

- Removed pg_backrest_remote and added the functionality to pg_backrest as the remote command.
- Moved version number out of the VERSION file to Version.pm to better support packaging. (*Suggested by Michael Renner.*)
- Replaced IPC::System::Simple and Net::OpenSSH with IPC::Open3 to eliminate CPAN dependency for multiple operating systems.

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.

Refactoring:

- 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.*)

Refactoring:

- 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.

Refactoring:

- Improved threading model by starting threads early and terminating them late.

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.

Refactoring:

- Removed dependency on Moose . It wasn't being used extensively and makes for longer startup times.

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

Refactoring:

- Complete rewrite of BackRest::File module to use a custom protocol for remote operations and Perl native GZIP and SHA operations. Compression is performed in threads rather than forked processes.
- Removed dependency on Storable and replaced with a custom ini file implementation.
- Numerous other changes that can only be identified with a diff.

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.

Refactoring:

- Worked on improving error handling in the File object. This is not complete, but works well enough to find a few errors that have been causing us problems (notably, find is occasionally failing building the archive async manifest when system is under load).

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.