pgBackRest Releases

pgBackRest User Guide
RHEL
Home
User Guides
Releases
Configuration
Commands
News
FAQ
Metrics
Table of Contents
Introduction
Concepts
Backup
Restore
Write Ahead Log (WAL)
Encryption
Upgrading pgBackRest
Upgrading pgBackRest from v2.x to v2.y
Build
Installation
Quick Start
Setup Demo Cluster
Configure Cluster Stanza

Create the Repository
Configure Archiving
Configure Retention
Configure Repository Encryption
Create the Stanza
Check the Configuration
Performance Tuning
Perform a Backup
Schedule a Backup
Backup Information
Restore a Backup
Monitoring
In PostgreSQL
Backup
File Bundling
Block Incremental
Backup Annotations
Retention
Full Backup Retention
Differential Backup Retention
Archive Retention
Restore
File Ownership
Delta Option
Restore Selected Databases
Point-in-Time Recovery
Delete a Stanza
Multiple Repositories
Azure-Compatible Object Store Support
S3-Compatible Object Store Support
SFTP Support

GCS-Compatible Object Store Support

Target Time for Repository

Dedicated Repository Host

Installation

Configuration

Setup TLS Server

Create and Check Stanza

Perform a Backup

Restore a Backup

Parallel Backup / Restore

Starting and Stopping

Replication

Installation

Hot Standby

Streaming Replication

Multiple Stanzas

Installation

Configuration

Setup Demo Cluster

Create the Stanza and Check Configuration

Asynchronous Archiving

Archive Push

Archive Get

Backup from a Standby

Upgrading PostgreSQL

Introduction

This user guide is intended to be followed sequentially from beginning to end — each section depends on the last. For example, the Restore 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 in this guide are targeted at RHEL and PostgreSQL 14, it should be fairly easy to apply the examples to any Unix distribution and PostgreSQL version. The only OS-specific commands are those to create, start, stop, and drop PostgreSQL clusters. The pgBackRest commands will be the same on any Unix system though the location of the executable may vary. While pgBackRest strives to operate consistently across versions of PostgreSQL, there are subtle differences between versions of PostgreSQL that may show up in this guide when illustrating certain examples, e.g. PostgreSQL path/file names and settings.

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

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

Incremental Backup: pgBackRest copies only those database cluster files that have changed since the last backup (which can be another incremental backup,

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

Restore

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

Write Ahead Log (WAL)

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

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

Encryption

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

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

Upgrading pgBackRest

Upgrading pgBackRest from v2.x to v2.y

Upgrading from v2.x to v2.y is straight-forward. The repository format has not changed, so for most installations it is simply a matter of installing binaries for the new version. It is also possible to downgrade if you have not used new features that are unsupported by the older version.

IMPORTANT:

The local and remote pgBackRest versions must match exactly so they should be upgraded together. If there is a mismatch, WAL archiving and backups will

not function until the versions match. In such a case, the following error will be reported: [ProtocolError] expected value '2.x' for greeting key 'version' but got '2.y'.

Build

Installing pgBackRest from a package is preferable to building from source. See Installation for more information about packages.

When building from source it is best to use a build host rather than building on production. Many of the tools required for the build should generally not be installed in production. pgBackRest consists of a single executable so it is easy to copy to a new host once it is built.

build Download version 2.59.1 of pgBackRest to /build path

```
mkdir -p /build
```

```
curl -fsSL \
       https://github.com/pgbackrest/pgbackrest/releases/download/release%2F2.59.1/pgbackres
       tar zx -C /build
```

build Install build dependencies

```
sudo yum install meson gcc postgresql14-devel openssl-devel libxml2-devel \
       lz4-devel libzstd-devel bzip2-devel libssh2-devel systemd-devel
```

build Configure and compile pgBackRest

```
meson setup /build/pgbackrest /build/pgbackrest-2.59.1
```

```
ninja -C /build/pgbackrest
```

build Optionally run smoke tests to verify pgBackRest was built correctly

```
meson test -C /build/pgbackrest --suite smoke
```

```
ninja: Entering directory `/build/pgbackrest'
ninja: no work to do.
```

```
1/1 smoke OK                12.40s
```

```
Ok:                 1
       [filtered 6 lines of output]
```

Installation

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

Installing pgBackRest from a package is preferable to building from source. When installing from a package the rest of the instructions in this section are generally not required, but it is possible that a package will skip creating one of the directories or apply incorrect permissions. In that case it may be necessary to manually create directories or update permissions.

RHEL packages for pgBackRest are available at yum.postgresql.org.

If packages are not provided for your distribution/version you can build from source and then install manually as shown here.

pg-primary Install dependencies

```
sudo yum install postgresql-libs libssh2
```

pg-primary Copy pgBackRest binary from build host

```
sudo scp build:/build/pgbackrest/src/pgbackrest /usr/bin
```

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

pgBackRest requires log and configuration directories and a configuration file.

pg-primary Create pgBackRest configuration file and directories

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

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

```
sudo mkdir -p /etc/pgbackrest
```

```
sudo mkdir -p /etc/pgbackrest/conf.d
```

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

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

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

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.

pg-primary Make sure the installation worked

```
sudo -u postgres pgbackrest
```

```
pgBackRest 2.59.1 - General help

Usage:
    pgbackrest [options] [command]

Commands:
    annotate        add or modify backup annotation
    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
```

```
    repo-get        get a file from a repository
    repo-ls         list files in a repository
    restore         restore a database cluster
    server          pgBackRest server
    server-ping     ping pgBackRest server
    stanza-create   create the required stanza data
    stanza-delete   delete a stanza
    stanza-upgrade  upgrade a stanza
    start           allow pgBackRest processes to run
    stop            stop pgBackRest processes from running
    verify          verify contents of a repository
    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.

pg-primary Create the demo cluster

```
sudo -u postgres /usr/pgsql-14/bin/initdb \
       -D /var/lib/pgsql/14/data -k -A peer
```

By default RHEL includes the day of the week in the log filename. This makes the user guide a bit more complicated so the log_filename is set to a constant.

pg-primary:/var/lib/pgsql/14/data/postgresql.conf Set log_filename

log_filename = 'postgresql.log'

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 PostgreSQL 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 pg-path is exactly equal to data_directory as reported by PostgreSQL.

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

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

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

[demo]
pg1-path=/var/lib/pgsql/14/data

pgBackRest configuration files follow a Windows INI-like 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, but end-of-line comments following a value on the same line are not supported. Quoting is not supported and whitespace is trimmed from keys and values. Sections will be merged if they appear more than once.

There are multiple ways the pgBackRest configuration files can be loaded:

- config and config-include-path are default: the default config file will be loaded, if it exists, and *.conf files in the default config include path will be appended, if they exist.
- config option is specified: only the specified config file will be loaded and is expected to exist.
- config-include-path is specified: *.conf files in the config include path will be loaded and the path is required to exist. The default config file will be be loaded if it exists. If it is desirable to load only the files in the specified config include path, then the --no-config option can also be passed.
- config and config-include-path are specified: using the user-specified values, the config file will be loaded and *.conf files in the config include path will be appended. The files are expected to exist.
- config-path is specified: this setting will override the base path for the default location of the config file and/or the base path of the default

config-include-path setting unless the config and/or config-include-path option is explicitly set.

Files are concatenated as if they were one big file and each file must be valid individually. This means sections must be specified in each file where they are needed to store a key/value. Order doesn't matter but there is precedence based on sections. The precedence (highest to lowest) is:

- [*stanza*:*command*]
- [*stanza*]
- [global:*command*]
- [global]

NOTE:

--config, --config-include-path and --config-path are command-line only options.

pgBackRest can also be configured using environment variables as described in the command reference.

pg-primary Configure log-path using the environment

```
sudo -u postgres bash -c ' \
       export PGBACKREST_LOG_PATH=/path/set/by/env && \
       pgbackrest --log-level-console=error help backup log-path'
```

```
pgBackRest 2.59.1 - 'backup' command - 'log-path' option help

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that
if log-level-file=off then no log path is required.

current: /path/set/by/env

default: /var/log/pgbackrest
```

Create the Repository

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

It may 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.

pg-primary Create the pgBackRest repository

```
sudo mkdir -p /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.

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

```
[demo]
pg1-path=/var/lib/pgsql/14/data


[global]
repo1-path=/var/lib/pgbackrest
```

Multiple repositories may also be configured. See Multiple Repositories for details.

Configure Archiving

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

pg-primary:/var/lib/pgsql/14/data/postgresql.conf Configure archive settings

```
archive_command = 'pgbackrest --stanza=demo archive-push %p'
archive_mode = on
log_filename = 'postgresql.log'
```

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

pg-primary Restart the demo cluster

```
sudo systemctl restart postgresql-14.service
```

The hot_standby setting is enabled by default and should be left that way on every cluster. Any cluster may be restored as a standby later, e.g. a primary that is rebuilt as a standby after failover, and a standby will not accept read-only connections without it.

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

The archive-push command can be configured with its own options. For example, a lower compression level may be set to speed archiving without affecting the

compression used for backups.

pg-primary:/etc/pgbackrest/pgbackrest.conf Config archive-push to use a lower compression level

```
[demo]
pg1-path=/var/lib/pgsql/14/data

[global]
repo1-path=/var/lib/pgbackrest

[global:archive-push]
compress-level=3
```

This configuration technique can be used for any command and can even target a specific stanza, e.g. demo:archive-push.

Configure Retention

pgBackRest expires backups based on retention options.

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

```
[demo]
pg1-path=/var/lib/pgsql/14/data

[global]
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2

[global:archive-push]
compress-level=3
```

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

Configure Repository Encryption

The repository will be configured with a cipher type and key to demonstrate encryption. Encryption is always performed client-side even if the repository type (e.g. S3 or other object store) supports encryption.

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

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

```
[demo]
pg1-path=/var/lib/pgsql/14/data

[global]
```

repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2

[global:archive-push]
compress-level=3

NOTE:

Encryption settings are placed in the [global] section, above, so the info command can read all stanzas. Without the stanza option the info command reads encryption settings only from the [global] section, so encryption settings configured per stanza require the stanza option to read an encrypted stanza.

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

Create the Stanza

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

pg-primary Create the stanza and check the configuration

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

```
P00   INFO: stanza-create command begin 2.59.1: --exec-id=1187-8aaf6311 --log-level-console=
P00   INFO: stanza-create for stanza 'demo' on repo1
```

```
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 for the specified stanza. It will attempt to check all repositories and databases that are configured for the host on which the command is run. 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 PostgreSQL or repository host. The command may also be run on the standby host, however, since pg_switch_xlog()/pg_switch_wal() cannot be performed on the standby, the command will only test the repository configuration.

Note that pg_create_restore_point('pgBackRest Archive Check') and pg_switch_xlog()/pg_switch_wal() are called to force PostgreSQL to archive a WAL segment.

pg-primary Check the configuration

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

```
P00   INFO: check command begin 2.59.1: --exec-id=1220-8e2ca9fb --log-level-console=info --r
P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)

P00   INFO: WAL segment 000000010000000000000001 successfully archived to '/var/lib/pgbackre

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

Performance Tuning

pgBackRest has a number of performance options that are not enabled by default to maintain backward compatibility in the repository. However, when creating a new repository the following options are recommended. They can also be used on an existing repository with the caveat that older versions of pgBackRest will not be able to read the repository. This incompatibility depends on when the feature was introduced, as noted in the list below.

- compress-type - determines the compression algorithm used by the backup and archive-push commands. The default is gz (Gzip) but zst (Zstandard) is recommended because it is much faster and provides compression similar to gz. zst has been supported by the compress-type option since v2.27. See Compress Type for more details.
- repo-bundle - combines small files during backup to save space and improve the speed of both the backup and restore commands, especially on object stores such as S3. The repo-bundle option was introduced in v2.39. See File Bundling for more details.
- repo-block - stores only the portions of files that have changed rather than the entire file during diff/incr backup. This saves space and increases the speed of the backup. The repo-block option was introduced in v2.46 but at least v2.52.1 is recommended. See Block Incremental for more details.

There are other performance options that are not enabled by default because they require additional configuration or because the default is safe (but not optimal). These options are available in all v2 versions of pgBackRest.

- process-max - determines how many processes will be used for commands. The default is 1, which is almost never the appropriate value. Each command uses process-max differently so refer to each command's documentation for details on usage.
- archive-async - archives WAL files to the repository in batch which greatly increases archiving speed. It is not enabled by default because it requires a spool path to be created. See Asynchronous Archiving for more details.
- backup-standby - performs the backup on a standby rather than the primary to reduce load on the primary. It is not enabled by default because it requires additional configuration and the presence of one or more standby hosts. See Backup from a Standby for more details.

Perform a Backup

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. Generally, it is best to set start-fast=y so that the backup starts immediately. This forces a checkpoint, but since backups are usually run once a day an additional checkpoint should not have a noticeable impact on performance. However, on very busy clusters it may be best to pass --start-fast on the command-line as needed.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure backup fast start

```
[demo]
pg1-path=/var/lib/pgsql/14/data

[global]
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y

[global:archive-push]
compress-level=3
```

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

pg-primary Backup the demo cluster

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

P00   INFO: backup command begin 2.59.1: --exec-id=1313-653dd239 --log-level-console=info --

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

P00   INFO: execute backup start: backup begins after the requested immediate checkpoint com
P00   INFO: backup start archive = 000000010000000000000002, lsn = 0/2000028
       [filtered 3 lines of output]
P00   INFO: check archive for segment(s) 000000010000000000000002:000000010000000000000003
P00   INFO: new backup label = 20260817-044340F

P00   INFO: full backup size = 25.2MB, file total = 951

P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.59.1: --exec-id=1313-653dd239 --log-level-console=info --
```

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.

pg-primary Differential backup of the demo cluster

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

       [filtered 7 lines of output]
P00   INFO: check archive for segment(s) 000000010000000000000004:000000010000000000000005
P00   INFO: new backup label = 20260817-044340F_20260817-044344D

P00   INFO: diff backup size = 9.2KB, file total = 951

P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.59.1: --exec-id=1384-482147d8 --log-level-console=info --
```

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.

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

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.

pg-primary Get info for the demo cluster

```
sudo -u postgres pgbackrest info

stanza: demo
    status: ok
    cipher: aes-256-cbc

    db (current)
```

```
wal archive min/max (14): 000000010000000000000001/000000010000000000000005

full backup: 20260817-044340F

    timestamp start/stop: 2026-08-17 04:43:40+00 / 2026-08-17 04:43:43+00
    wal start/stop: 000000010000000000000002 / 000000010000000000000003
    database size: 25.2MB, database backup size: 25.2MB
    repo1: backup set size: 3.2MB, backup size: 3.2MB

diff backup: 20260817-044340F_20260817-044344D

    timestamp start/stop: 2026-08-17 04:43:44+00 / 2026-08-17 04:43:45+00
    wal start/stop: 000000010000000000000004 / 000000010000000000000005
    database size: 25.2MB, database backup size: 9.2KB
    repo1: backup set size: 3.2MB, backup size: 864B
    backup reference total: 1 full
```

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 and is kept stable unless a bug is found.

To speed up execution, limit the output to only progress information by specifying --detail-level=progress. Note that this skips all checks except for availability of the stanza.

Each stanza has a separate section and it is possible to limit output to a single stanza with the --stanza option. The stanza 'status' gives a brief indication of the stanza's health. If this is 'ok' then pgBackRest is functioning normally. If there are multiple repositories, then a status of 'mixed' indicates that the stanza is not in a healthy state on one or more of the repositories; in this case the state of the stanza will be detailed per repository. For cases in which an error on a repository occurred that is not one of the known error codes, then an error code of 'other' will be used and the full error details will be provided. The 'wal archive min/max' shows the minimum and maximum WAL currently stored in the archive and, in the case of multiple repositories, will be reported across all repositories unless the --repo option is set. Note that there may be gaps due to archive retention policies or other reasons.

The 'backup/expire running' and/or 'restore running' messages will appear beside the 'status' information if any of those commands are currently running on the host. Per-repo progress will be also reported in text output and a 'repo' array will be included in JSON output.

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

The 'timestamp start/stop' defines the time period when the backup ran. The

'timestamp stop' can be used to determine the backup to use when performing Point-In-Time Recovery. More information about Point-In-Time Recovery can be found in the Point-In-Time Recovery section.

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

The 'database size' is the full uncompressed size of the database while 'database backup size' is the amount of data in the database to actually back up (these will be the same for full backups).

The 'repo' indicates in which repository this backup resides. The 'backup set size' includes all the files from this backup and any referenced backups in the repository that are required to restore the database from this backup while '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.

The 'backup reference total' summarizes the list of additional backups that are required to restore this backup. Use the --set option to display the complete reference list.

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.

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

```
sudo systemctl stop postgresql-14.service
```

```
sudo -u postgres rm /var/lib/pgsql/14/data/global/pg_control
```

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

pg-primary Attempt to start the corrupted demo cluster

```
sudo systemctl start postgresql-14.service
```

```
sudo systemctl status postgresql-14.service
```

```
postgresql-14.service - PostgreSQL 14 database server
    Loaded: loaded (/usr/lib/systemd/system/postgresql-14.service, disabled)

    Active: failed (failed)
```

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.

pg-primary Remove old files from demo cluster

```
sudo -u postgres find /var/lib/pgsql/14/data -mindepth 1 -delete
```

pg-primary Restore the demo cluster and start PostgreSQL

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

```
sudo systemctl start postgresql-14.service
```

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.

Monitoring

Monitoring is an important part of any production system. There are many tools available and pgBackRest can be monitored on any of them with a little work.

pgBackRest can output information about the repository in JSON format which includes a list of all backups for each stanza and WAL archive info.

In PostgreSQL

The PostgreSQL COPY command allows pgBackRest info to be loaded into a table. The following example wraps that logic in a function that can be used to perform real-time queries.

pg-primary Load pgBackRest info function for PostgreSQL

```
sudo -u postgres cat \
       /var/lib/pgsql/pgbackrest/doc/example/pgsql-pgbackrest-info.sql
```

```
-- An example of monitoring pgBackRest from within PostgreSQL
--
-- Use copy to export data from the pgBackRest info command into the jsonb
-- type so it can be queried directly by PostgreSQL.

-- Create monitor schema
create schema monitor;

-- Get pgBackRest info in JSON format
create function monitor.pgbackrest_info()
    returns jsonb AS $$
declare
    data jsonb;
begin
    -- Create a temp table to hold the JSON data
    create temp table temp_pgbackrest_data (data text);

    -- Copy data into the table directly from the pgBackRest info command
    copy temp_pgbackrest_data (data)
        from program
```

```
            'pgbackrest --output=json info' (format text);

    select replace(temp_pgbackrest_data.data, E'\n', '\n')::jsonb
      into data
      from temp_pgbackrest_data;

    drop table temp_pgbackrest_data;

    return data;
end $$ language plpgsql;
```

```
sudo -u postgres psql -f \
       /var/lib/pgsql/pgbackrest/doc/example/pgsql-pgbackrest-info.sql
```

Now the monitor.pgbackrest_info() function can be used to determine the last successful backup time and archived WAL for a stanza.

pg-primary Query last successful backup time and archived WAL

```
sudo -u postgres cat \
       /var/lib/pgsql/pgbackrest/doc/example/pgsql-pgbackrest-query.sql
```

```
-- Get last successful backup for each stanza
--
-- Requires the monitor.pgbackrest_info function.
with stanza as
(
    select data->'name' as name,
           data->'backup'->(
               jsonb_array_length(data->'backup') - 1) as last_backup,
           data->'archive'->(
               jsonb_array_length(data->'archive') - 1) as current_archive
      from jsonb_array_elements(monitor.pgbackrest_info()) as data
)
select name,
       to_timestamp(
           (last_backup->'timestamp'->>'stop')::numeric) as last_successful_backup,
       current_archive->>'max' as last_archived_wal
  from stanza;
```

```
sudo -u postgres psql -f \
       /var/lib/pgsql/pgbackrest/doc/example/pgsql-pgbackrest-query.sql
```

```
  name  | last_successful_backup |    last_archived_wal
--------+------------------------+--------------------------
 "demo" | 2026-08-17 04:43:45+00 | 000000010000000000000005
(1 row)
```

Backup

When multiple repositories are configured, pgBackRest will backup to the highest priority repository (e.g. repo1) unless the --repo option is specified.

pgBackRest does not have a built-in scheduler so it's best to run it from cron or some other scheduling mechanism.

See Perform a Backup for more details and examples.

File Bundling

Bundling files together in the repository saves time during the backup and some space in the repository. This is especially pronounced when the repository is stored on an object store such as S3 or file systems with large block sizes. Per-file creation time on object stores is higher and very small files might cost as much to store as larger files.

The file bundling feature is enabled with the repo-bundle option.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure repo1-bundle

```
[demo]
pg1-path=/var/lib/pgsql/14/data

[global]
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y

[global:archive-push]
compress-level=3
```

A full backup without file bundling will have 1000+ files in the backup path, but with bundling the total number of files is greatly reduced. An additional benefit is that zero-length files are not stored (except in the manifest), whereas in a normal backup each zero-length file is stored individually.

pg-primary Perform a full backup

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

pg-primary Check file total

```
sudo -u postgres find /var/lib/pgbackrest/backup/demo/latest/ -type f | wc -l

5
```

The repo-bundle-size and repo-bundle-limit options can be used for tuning, though the defaults should be optimal in most cases.

While file bundling is generally more efficient, the downside is that it is more difficult to manually retrieve files from the repository. It may not be ideal for deduplicated storage since each full backup will arrange files in the bundles differently. Lastly, file bundles cannot be resumed, so be careful not to set repo-bundle-limit too high.

Block Incremental

Block incremental backups save space by only storing the parts of a file that have changed since the prior backup rather than storing the entire file.

The block incremental feature is enabled with the repo-block option and it works best when enabled for all backup types. File bundling must also be enabled.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure repo1-block

```
[demo]
pg1-path=/var/lib/pgsql/14/data


[global]
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y


[global:archive-push]
compress-level=3
```

Backup Annotations

Users can attach informative key/value pairs to the backup. This option may be used multiple times to attach multiple annotations.

pg-primary Perform a full backup with annotations

```
sudo -u postgres pgbackrest --stanza=demo --annotation=source="demo backup" \
       --annotation=key=value --type=full backup
```

Annotations are output by the info command text output when a backup is specified with --set and always appear in the JSON output.

pg-primary Get info for the demo cluster

```
sudo -u postgres pgbackrest --stanza=demo --set=20260817-044358F info
```

```
stanza: demo
    status: ok
    cipher: aes-256-cbc
```

```
    db (current)
        wal archive min/max (14): 000000020000000000000007/000000020000000000000009

        full backup: 20260817-044358F
            timestamp start/stop: 2026-08-17 04:43:58+00 / 2026-08-17 04:44:00+00
            wal start/stop: 000000020000000000000008 / 000000020000000000000009
            lsn start/stop: 0/8000028 / 0/9000050
            database size: 25.2MB, database backup size: 25.2MB
            repo1: backup size: 3.2MB
            database list: postgres (13755)

            annotation(s)

                key: value
                source: demo backup
```

Annotations included with the backup command can be added, modified, or removed afterwards using the annotate command.

pg-primary Change backup annotations

```
sudo -u postgres pgbackrest --stanza=demo --set=20260817-044358F \
       --annotation=key= --annotation=new_key=new_value annotate

sudo -u postgres pgbackrest --stanza=demo --set=20260817-044358F info

stanza: demo
    status: ok
    cipher: aes-256-cbc

    db (current)
        wal archive min/max (14): 000000020000000000000007/000000020000000000000009

        full backup: 20260817-044358F
            timestamp start/stop: 2026-08-17 04:43:58+00 / 2026-08-17 04:44:00+00
            wal start/stop: 000000020000000000000008 / 000000020000000000000009
            lsn start/stop: 0/8000028 / 0/9000050
            database size: 25.2MB, database backup size: 25.2MB
            repo1: backup size: 3.2MB
            database list: postgres (13755)

            annotation(s)

                new_key: new_value
                source: demo backup
```

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.

pgBackRest does full backup rotation based on the retention type which can be a count or a time period. When a count is specified, then expiration is not concerned with when the backups were created but with how many must be retained. Differential backups are count-based but will always be expired when the full backup they depend on is expired. Incremental backups are not expired by retention independently — they are always expired with their related full or differential backup. See sections Full Backup Retention and Differential Backup Retention for details and examples.

Archived WAL is retained by default for backups that have not expired, however, although not recommended, this schedule can be modified per repository with the retention-archive options. See section Archive Retention for details and examples.

The expire command is run automatically after each successful backup and can also be run by the user. When run by the user, expiration will occur as defined by the retention settings for each configured repository. If the --repo option is provided, expiration will occur only on the specified repository. Expiration can also be limited by the user to a specific backup set with the --set option and, unless the --repo option is specified, all repositories will be searched and any matching the set criteria will be expired. It should be noted that the archive retention schedule will be checked and performed any time the expire command is run.

Full Backup Retention

The repo1-retention-full-type determines how the option repo1-retention-full is interpreted; either as the count of full backups to be retained or how many days to retain full backups. New backups must be completed before expiration will occur — that means if repo1-retention-full-type=count and repo1-retention-full=2 then there will be three full backups stored before the oldest one is expired, or if repo1-retention-full-type=time and repo1-retention-full=20 then there must be one full backup that is at least 20 days old before expiration can occur.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure repo1-retention-full

```
[demo]
pg1-path=/var/lib/pgsql/14/data

[global]
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
```

repo1-retention-full=2
start-fast=y

[global:archive-push]
compress-level=3

Backup repo1-retention-full=2 but currently there is only one full backup so the next full backup to run will not expire any full backups.

pg-primary Perform a full backup

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

       [filtered 963 lines of output]
P00   INFO: repo1: remove expired backup 20260817-044356F
P00 DETAIL: repo1: 14-1 archive retention on backup 20260817-044358F, start = 00000002000000

P00   INFO: repo1: 14-1 remove archive, start = 000000020000000000000007, stop = 00000002000

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.

pg-primary Perform a full backup

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

       [filtered 11 lines of output]
P00   INFO: repo1: expire full backup 20260817-044358F
P00   INFO: repo1: remove expired backup 20260817-044358F

P00   INFO: repo1: 14-1 remove archive, start = 000000020000000000000008, stop = 00000002000

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

The 20260817-044340F full backup is expired and archive retention is based on the 20260817-044401F which is now the oldest full backup.

Differential Backup Retention

Set repo1-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.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure repo1-retention-diff

[demo]
pg1-path=/var/lib/pgsql/14/data

```
[global]
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-diff=1
repo1-retention-full=2
start-fast=y

[global:archive-push]
compress-level=3
```

Backup repo1-retention-diff=1 so two differentials will need to be performed before one is expired. An incremental backup is added to demonstrate incremental expiration, which in this case depends on the differential expiration.

pg-primary Perform differential and incremental backups

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

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

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

pg-primary Perform a differential backup

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

       [filtered 10 lines of output]
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.59.1: --exec-id=2616-f4c946a1 --log-level-console=info --

P00   INFO: repo1: expire diff backup set 20260817-044403F_20260817-044405D, 20260817-044403

P00   INFO: repo1: remove expired backup 20260817-044403F_20260817-044406I
P00   INFO: repo1: remove expired backup 20260817-044403F_20260817-044405D
P00   INFO: expire command end: completed successfully
```

Archive Retention

Although pgBackRest automatically removes archived WAL segments when expiring backups (the default expires WAL for full backups based on the repo1-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. To determine what will be expired without actually expiring anything, the dry-run option can be provided on the command line with the expire command.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure repo1-retention-diff

```
[demo]
pg1-path=/var/lib/pgsql/14/data

[global]
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-diff=2
repo1-retention-full=2
start-fast=y

[global:archive-push]
compress-level=3
```

pg-primary Perform differential backup

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

       [filtered 6 lines of output]
P00   INFO: backup stop archive = 000000020000000000000017, lsn = 0/17000050
P00   INFO: check archive for segment(s) 000000020000000000000016:000000020000000000000017

P00   INFO: new backup label = 20260817-044403F_20260817-044409D

P00   INFO: diff backup size = 11.5KB, file total = 951
P00   INFO: backup command end: completed successfully
       [filtered 2 lines of output]
```

pg-primary Expire archive

```
sudo -u postgres pgbackrest --stanza=demo --log-level-console=detail \
       --repo1-retention-archive-type=diff --repo1-retention-archive=1 expire

P00   INFO: expire command begin 2.59.1: --exec-id=2851-7355d46a --log-level-console=detail
P00 DETAIL: repo1: 14-1 archive retention on backup 20260817-044401F, start = 0000000200000
P00 DETAIL: repo1: 14-1 archive retention on backup 20260817-044403F, start = 0000000200000

P00 DETAIL: repo1: 14-1 archive retention on backup 20260817-044403F_20260817-044407D, start

P00 DETAIL: repo1: 14-1 archive retention on backup 20260817-044403F_20260817-044409D, start

P00   INFO: repo1: 14-1 remove archive, start = 00000002000000000000000E, stop = 0000000200
P00   INFO: repo1: 14-1 remove archive, start = 000000020000000000000014, stop = 0000000200
```

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

The 20260817-044403F_20260817-044407D 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 20260817-044403F_20260817-044407D but before 20260817-044403F_20260817-044409D are removed. WAL segments generated after the new backup 20260817-044403F_20260817-044409D 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 command automatically defaults to selecting the latest backup from the first repository where backups exist (see Quick Start - Restore a Backup). The order in which the repositories are checked is dictated by the pgbackrest.conf (e.g. repo1 will be checked before repo2). To select from a specific repository, the --repo option can be passed (e.g. --repo=1). The --set option can be passed if a backup other than the latest is desired.

When PITR of --type=time or --type=lsn is specified, then the target time or target lsn must be specified with the --target option. If a backup is not specified via the --set option, then the configured repositories will be checked, in order, for a backup that contains the requested time or lsn. If no matching backup is found, the latest backup from the first repository containing backups will be used for --type=time while no backup will be selected for --type=lsn. For other types of PITR, e.g. xid, the --set option must be provided if the target is prior to the latest backup. See Point-in-Time Recovery for more details and examples.

Replication slots are not included per recommendation of PostgreSQL. See Backing Up The Data Directory in the PostgreSQL documentation for more information.

The following sections introduce additional restore command features.

File Ownership

If a restore is run as a non-root user (the typical scenario) then all files restored will belong to the user/group executing pgBackRest. If existing files are not owned by the executing user/group then an error will result if the ownership cannot be updated to the executing user/group. In that case the file ownership will need to be updated by a privileged user before the restore can be retried.

If a restore is run as the root user then pgBackRest will attempt to recreate the ownership recorded in the manifest when the backup was made. Only user/group **names** are stored in the manifest so the same names must exist on the restore host for this to work. If the user/group name cannot be found locally

then the user/group of the PostgreSQL data directory will be used and finally root if the data directory user/group cannot be mapped to a name.

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.

pg-primary Stop the demo cluster, perform delta restore

```
sudo systemctl stop postgresql-14.service
```

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

```
       [filtered 2 lines of output]
P00 DETAIL: check '/var/lib/pgsql/14/data' exists
P00 DETAIL: remove 'global/pg_control' so cluster will not start if restore does not complet
```

```
P00   INFO: remove invalid files/links/paths from '/var/lib/pgsql/14/data'
```

```
P00 DETAIL: remove invalid file '/var/lib/pgsql/14/data/backup_label.old'
P00 DETAIL: remove invalid file '/var/lib/pgsql/14/data/base/13755/pg_internal.init'
       [filtered 996 lines of output]
```

pg-primary Restart PostgreSQL

```
sudo systemctl start postgresql-14.service
```

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.

pg-primary Create two test databases

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

```
CREATE DATABASE
```

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

CREATE DATABASE
```

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

pg-primary Create a test table in each database

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

CREATE TABLE
INSERT 0 1

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

CREATE TABLE
INSERT 0 1
```

A fresh backup is run so pgBackRest is aware of the new databases.

pg-primary Perform a backup

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

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.

pg-primary Show space used by test1 database

```
sudo -u postgres du -sh /var/lib/pgsql/14/data/base/32768

8.4M   /var/lib/pgsql/14/data/base/32768
```

If the database to restore is not known, use the info command set option to discover databases that are part of the backup set.

pg-primary Show database list for backup

```
sudo -u postgres pgbackrest --stanza=demo \
       --set=20260817-044403F_20260817-044418I info

       [filtered 12 lines of output]
            repo1: backup size: 2.1MB
            backup reference list: 20260817-044403F, 20260817-044403F_20260817-044409D

            database list: postgres (13755), test1 (32768), test2 (32769)
```

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

WARNING:

Recovery may error unless --type=immediate is specified. This is because after consistency is reached PostgreSQL will flag zeroed pages as errors even for a full-page write. For PostgreSQL 13 the ignore_invalid_pages setting may be used to ignore invalid pages. In this case it is important to check the logs after recovery to ensure that no invalid pages were reported in the selected databases.

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

```
sudo systemctl stop postgresql-14.service
```

```
sudo -u postgres pgbackrest --stanza=demo --delta \
       --db-include=test2 --type=immediate --target-action=promote restore
```

```
sudo systemctl start postgresql-14.service
```

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

pg-primary Demonstrate that the test2 database was recovered

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

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

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

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

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

```
psql: error: connection to server on socket "/run/postgresql/.s.PGSQL.5432" failed: FATAL:
```

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.

pg-primary Show space used by test1 database after recovery

```
sudo -u postgres du -sh /var/lib/pgsql/14/data/base/32768
```

```
8.0K   /var/lib/pgsql/14/data/base/32768
```

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.

pg-primary Drop the test1 database

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

```
DROP DATABASE
```

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

pg-primary List remaining databases

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

```
  oid  |  datname
-------+-----------
     1 | template1
 13754 | template0
 13755 | postgres
 32769 | 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 a backup to a specified lsn, 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.

pg-primary Create a table with very important data

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

```
       [filtered 4 lines of output]
    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.

pg-primary Get the time from PostgreSQL

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

2026-08-17 04:44:30.009846+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.

pg-primary Drop the important table

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

BEGIN
DROP TABLE

COMMITERROR:  relation "important_table" does not exist

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

If the wrong backup is selected for restore then recovery to the required time target will fail. To demonstrate this a new incremental backup is performed where important_table does not exist.

pg-primary Perform an incremental backup

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

sudo -u postgres pgbackrest info

       [filtered 38 lines of output]
            backup reference total: 1 full, 1 diff

        incr backup: 20260817-044403F_20260817-044431I

            timestamp start/stop: 2026-08-17 04:44:31+00 / 2026-08-17 04:44:32+00
            wal start/stop: 000000040000000000000001A / 000000040000000000000001A
       [filtered 2 lines of output]
```

It will not be possible to recover the lost table from this backup since PostgreSQL can only play forward, not backward.

pg-primary Attempt recovery from an incorrect backup

```
sudo systemctl stop postgresql-14.service
```

```
sudo -u postgres pgbackrest --stanza=demo --delta \
       --set=20260817-044403F_20260817-044431I --target-timeline=current \
       --type=time "--target=2026-08-17 04:44:30.009846+00" --target-action=promote restore
```

```
sudo systemctl start postgresql-14.service
```

```
sudo -u postgres cat /var/lib/pgsql/14/data/log/postgresql.log
```

```
       [filtered 11 lines of output]
LOG:  database system is ready to accept read-only connections
LOG:  redo done at 0/1A000100 system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.01
```

```
FATAL:  recovery ended before configured recovery target was reached
```

```
LOG:  startup process (PID 4008) exited with exit code 1
LOG:  terminating any other active server processes
LOG:  database system is shut down
```

A reliable method is to allow pgBackRest to automatically select a backup capable of recovery to the time target, i.e. a backup that ended before the specified time.

NOTE:

pgBackRest cannot automatically select a backup when the restore type is xid or name.

pg-primary Restore the demo cluster to 2026-08-17 04:44:30.009846+00

```
sudo -u postgres pgbackrest --stanza=demo --delta \
       --type=time "--target=2026-08-17 04:44:30.009846+00" \
       --target-action=promote restore
```

```
sudo -u postgres cat /var/lib/pgsql/14/data/postgresql.auto.conf
```

```
       [filtered 9 lines of output]
# Recovery settings generated by pgBackRest restore on 2026-08-17 04:44:37
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
```

```
recovery_target_time = '2026-08-17 04:44:30.009846+00'
```

```
recovery_target_action = 'promote'
```

pgBackRest has generated the recovery settings in postgresql.auto.conf so PostgreSQL can be started immediately. %f is how PostgreSQL specifies the WAL segment it needs and %p is the location where it should be copied. Once PostgreSQL has finished recovery the table will exist again and can be queried.

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

```
sudo systemctl start postgresql-14.service
```

```
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.

pg-primary Examine the PostgreSQL log output

```
sudo -u postgres cat /var/lib/pgsql/14/data/log/postgresql.log

       [filtered 5 lines of output]
LOG:  database system was interrupted; last known up at 2026-08-17 04:44:18 UTC
LOG:  restored log file "00000004.history" from archive

LOG:  starting point-in-time recovery to 2026-08-17 04:44:30.009846+00

LOG:  restored log file "00000004.history" from archive
LOG:  restored log file "000000040000000000000019" from archive
       [filtered 2 lines of output]
LOG:  consistent recovery state reached at 0/19000100
LOG:  database system is ready to accept read-only connections

LOG:  recovery stopping before commit of transaction 743, time 2026-08-17 04:44:31.350414+00

LOG:  redo done at 0/1901E6C0 system usage: CPU: user: 0.00 s, system: 0.01 s, elapsed: 0.01

LOG:  last completed transaction was at log time 2026-08-17 04:44:28.631376+00

LOG:  selected new timeline ID: 5
LOG:  archive recovery complete
LOG:  database system is ready to accept connections
```

Delete a Stanza

The stanza-delete command removes data in the repository associated with a stanza.

WARNING:

Use this command with caution — it will permanently remove all backups and archives from the pgBackRest repository for the specified stanza.

To delete a stanza:

- Shut down the PostgreSQL cluster associated with the stanza (or use --force to override).
- Run the stop command on the host where the stanza-delete command will be run.

- Run the stanza-delete command.

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

A stanza may only be deleted from one repository at a time. To delete the stanza from multiple repositories, repeat the stanza-delete command for each repository while specifying the --repo option.

pg-primary Stop PostgreSQL cluster to be removed

```
sudo systemctl stop postgresql-14.service
```

pg-primary Stop pgBackRest for the stanza

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

```
P00   INFO: stop command begin 2.59.1: --exec-id=4359-ed60b799 --log-level-console=info --no
```

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

pg-primary Delete the stanza from one repository

```
sudo -u postgres pgbackrest --stanza=demo --repo=1 \
       --log-level-console=info stanza-delete
```

```
P00   INFO: stanza-delete command begin 2.59.1: --exec-id=4390-92592378 --log-level-console=
```

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

Multiple Repositories

Multiple repositories may be configured as demonstrated in S3 Support. A potential benefit is the ability to have a local repository for fast restores and a remote repository for redundancy.

Some commands, e.g. stanza-create/stanza-upgrade, will automatically work with all configured repositories while others, e.g. stanza-delete, will require a repository to be specified using the repo option. See the command reference for details on which commands require the repository to be specified.

Note that the repo option is not required when only repo1 is configured in order to maintain backward compatibility. However, the repo option *is* required when a single repo is configured as, e.g. repo2. This is to prevent command breakage if a new repository is added later.

The archive-push command will always push WAL to the archive in all configured repositories. When a repository cannot be reached, WAL will still be pushed to other repositories. However, for this to work effectively, archive-async=y must be enabled; otherwise, the other repositories can only get one WAL segment ahead of the unreachable repository. Also, note that if WAL cannot be pushed to any repository, then PostgreSQL will not remove it from the pg_wal directory, which may cause the volume to run out of space.

Backups need to be scheduled individually for each repository. In many cases this is desirable since backup types and retention will vary by repository. Likewise, restores must specify a repository. It is generally better to specify a repository for restores that has low latency/cost even if that means more recovery time. Only restore testing can determine which repository will be most efficient.

Azure-Compatible Object Store Support

pgBackRest supports locating repositories in Azure-compatible object stores. The container used to store the repository must be created in advance — pgBackRest will not do it automatically. The repository can be located in the container root (/) but it's usually best to place it in a subpath so object store logs or other data can also be stored in the container without conflicts.

WARNING:

Do not enable "hierarchical namespace" as this will cause errors during expire.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure Azure

```
[demo]
pg1-path=/var/lib/pgsql/14/data

[global]
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-diff=2
repo1-retention-full=2
repo2-azure-account=pgbackrest
repo2-azure-container=demo-container
repo2-azure-key=YXpLZXk=
repo2-path=/demo-repo
repo2-retention-full=4
repo2-type=azure
start-fast=y

[global:archive-push]
compress-level=3
```

Shared access signatures may be used by setting the repo2-azure-key-type option to sas and the repo2-azure-key option to the shared access signature token.

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

pg-primary Create the stanza

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

```
P00   INFO: stanza-create command begin 2.59.1: --exec-id=4596-566ff3a8 --log-level-console=
P00   INFO: stanza-create for stanza 'demo' on repo1
P00   INFO: stanza-create for stanza 'demo' on repo2

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

File creation time in Azure is relatively slow so backup/restore performance is improved by enabling file bundling.

pg-primary Backup the demo cluster

```
sudo -u postgres pgbackrest --stanza=demo --repo=2 \
       --log-level-console=info backup

P00   INFO: backup command begin 2.59.1: --exec-id=4629-4d019785 --log-level-console=info --

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

P00   INFO: execute backup start: backup begins after the requested immediate checkpoint com
P00   INFO: backup start archive = 000000050000000000000001B, lsn = 0/1B000028
       [filtered 3 lines of output]
P00   INFO: check archive for segment(s) 000000050000000000000001B:000000050000000000000001B
P00   INFO: new backup label = 20260817-044452F

P00   INFO: full backup size = 33.5MB, file total = 1249

P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.59.1: --exec-id=4629-4d019785 --log-level-console=info --
```

S3-Compatible Object Store Support

pgBackRest supports locating repositories in S3-compatible object stores. The bucket used to store the repository must be created in advance — pgBackRest will not do it automatically. The repository can be located in the bucket root (/) but it's usually best to place it in a subpath so object store logs or other data can also be stored in the bucket without conflicts.

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

```
[demo]
pg1-path=/var/lib/pgsql/14/data


[global]
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-diff=2
repo1-retention-full=2
repo2-azure-account=pgbackrest
repo2-azure-container=demo-container
```

repo2-azure-key=YXpLZXk=
repo2-path=/demo-repo
repo2-retention-full=4
repo2-type=azure
repo3-path=/demo-repo
repo3-retention-full=4
repo3-s3-bucket=demo-bucket
repo3-s3-endpoint=s3.us-east-1.amazonaws.com
repo3-s3-key=accessKey1
repo3-s3-key-secret=verySecretKey1
repo3-s3-region=us-east-1
repo3-type=s3
start-fast=y

[global:archive-push]
compress-level=3

NOTE:

The region and endpoint will need to be configured to where the bucket is located. The values given here are for the us-east-1 region.

A role should be created to run pgBackRest and the bucket permissions should be set as restrictively as possible. If the role is associated with an instance in AWS then pgBackRest will automatically retrieve temporary credentials when repo3-s3-key-type=auto, which means that keys do not need to be explicitly set in /etc/pgbackrest/pgbackrest.conf.

This sample Amazon S3 policy will restrict all reads and writes to the bucket and repository path.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::demo-bucket"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:prefix": [
                        "",
                        "demo-repo"
                    ],
```

```
                    "s3:delimiter": [
                        "/"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::demo-bucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "demo-repo/*"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectTagging",
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::demo-bucket/demo-repo/*"
            ]
        }
    ]
}
```

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

pg-primary Create the stanza

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

```
       [filtered 4 lines of output]
P00   INFO: stanza 'demo' already exists on repo2 and is valid
P00   INFO: stanza-create for stanza 'demo' on repo3
```

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

File creation time in S3 is relatively slow so backup/restore performance is improved by enabling file bundling.

pg-primary Backup the demo cluster

```
sudo -u postgres pgbackrest --stanza=demo --repo=3 \
       --log-level-console=info backup

P00   INFO: backup command begin 2.59.1: --exec-id=5019-cc9e6a36 --log-level-console=info --

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

P00   INFO: execute backup start: backup begins after the requested immediate checkpoint com
P00   INFO: backup start archive = 000000050000000000000001D, lsn = 0/1D000028
       [filtered 3 lines of output]
P00   INFO: check archive for segment(s) 000000050000000000000001D:000000050000000000000001D
P00   INFO: new backup label = 20260817-044510F

P00   INFO: full backup size = 33.5MB, file total = 1249

P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.59.1: --exec-id=5019-cc9e6a36 --log-level-console=info --
```

SFTP Support

pgBackRest supports locating repositories on SFTP hosts. SFTP file transfer is relatively slow so commands benefit by increasing process-max to parallelize file transfer.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure SFTP

```
[demo]
pg1-path=/var/lib/pgsql/14/data


[global]
process-max=4
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-diff=2
repo1-retention-full=2
repo2-azure-account=pgbackrest
repo2-azure-container=demo-container
repo2-azure-key=YXpLZXk=
repo2-path=/demo-repo
repo2-retention-full=4
repo2-type=azure
repo3-path=/demo-repo
```

repo3-retention-full=4
repo3-s3-bucket=demo-bucket
repo3-s3-endpoint=s3.us-east-1.amazonaws.com
repo3-s3-key=accessKey1
repo3-s3-key-secret=verySecretKey1
repo3-s3-region=us-east-1
repo3-type=s3
repo4-bundle=y
repo4-path=/demo-repo
repo4-sftp-host=sftp-server
repo4-sftp-host-key-hash-type=sha1
repo4-sftp-host-user=pgbackrest
repo4-sftp-private-key-file=/var/lib/pgsql/.ssh/id_rsa_sftp
repo4-sftp-public-key-file=/var/lib/pgsql/.ssh/id_rsa_sftp.pub
repo4-type=sftp
start-fast=y

[global:archive-push]
compress-level=3

When utilizing SFTP, if libssh2 is compiled against OpenSSH then repo4-sftp-public-key-file is optional.

pg-primary Generate SSH keypair for SFTP backup

```
sudo -u postgres mkdir -m 750 -p /var/lib/pgsql/.ssh
```

```
sudo -u postgres ssh-keygen -f /var/lib/pgsql/.ssh/id_rsa_sftp \
       -t rsa -b 4096 -N "" -m PEM
```

sftp-server Copy pg-primary SFTP backup public key to sftp-server

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

```
(sudo ssh root@pg-primary cat /var/lib/pgsql/.ssh/id_rsa_sftp.pub) | \
       sudo -u pgbackrest tee -a /home/pgbackrest/.ssh/authorized_keys
```

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

pg-primary Add sftp-server fingerprint to known_hosts file since repo4-sftp-host-key-check-type defaults to “strict”

```
ssh-keyscan -H sftp-server >> /var/lib/pgsql/.ssh/known_hosts 2>/dev/null
```

pg-primary Create the stanza

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

```
       [filtered 6 lines of output]
P00   INFO: stanza 'demo' already exists on repo3 and is valid
P00   INFO: stanza-create for stanza 'demo' on repo4
```

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

pg-primary Backup the demo cluster

```
sudo -u postgres pgbackrest --stanza=demo --repo=4 \
       --log-level-console=info backup

P00   INFO: backup command begin 2.59.1: --exec-id=5320-594a571a --log-level-console=info --
P00   WARN: option 'repo4-retention-full' is not set for 'repo4-retention-full-type=count',
            HINT: to retain full backups indefinitely (without warning), set option 'repo4-r

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

P00   INFO: execute backup start: backup begins after the requested immediate checkpoint com
P00   INFO: backup start archive = 00000005000000000000001E, lsn = 0/1E000028
       [filtered 3 lines of output]
P00   INFO: check archive for segment(s) 00000005000000000000001E:00000005000000000000001F
P00   INFO: new backup label = 20260817-044529F

P00   INFO: full backup size = 33.5MB, file total = 1249

P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.59.1: --exec-id=5320-594a571a --log-level-console=info --
P00   INFO: expire command end: completed successfully
```

GCS-Compatible Object Store Support

pgBackRest supports locating repositories in GCS-compatible object stores. The bucket used to store the repository must be created in advance — pgBackRest will not do it automatically. The repository can be located in the bucket root (/) but it's usually best to place it in a subpath so object store logs or other data can also be stored in the bucket without conflicts.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure GCS

```
[demo]
pg1-path=/var/lib/pgsql/14/data


[global]
process-max=4
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-diff=2
repo1-retention-full=2
repo2-azure-account=pgbackrest
repo2-azure-container=demo-container
repo2-azure-key=YXpLZXk=
repo2-path=/demo-repo
repo2-retention-full=4
repo2-type=azure
```

```
repo3-path=/demo-repo
repo3-retention-full=4
repo3-s3-bucket=demo-bucket
repo3-s3-endpoint=s3.us-east-1.amazonaws.com
repo3-s3-key=accessKey1
repo3-s3-key-secret=verySecretKey1
repo3-s3-region=us-east-1
repo3-type=s3
repo4-bundle=y
repo4-path=/demo-repo
repo4-sftp-host=sftp-server
repo4-sftp-host-key-hash-type=sha1
repo4-sftp-host-user=pgbackrest
repo4-sftp-private-key-file=/var/lib/pgsql/.ssh/id_rsa_sftp
repo4-sftp-public-key-file=/var/lib/pgsql/.ssh/id_rsa_sftp.pub
repo4-type=sftp
repo5-gcs-bucket=demo-bucket
repo5-gcs-key=/etc/pgbackrest/gcs-key.json
repo5-path=/demo-repo
repo5-type=gcs
start-fast=y

[global:archive-push]
compress-level=3
```

When running in GCE set repo5-gcs-key-type=auto to automatically authenticate using the instance service account.

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

File creation time in GCS is relatively slow so backup/restore performance is improved by enabling file bundling.

Target Time for Repository

The target time defines the time that commands use to read a repository on versioned storage. This allows the command to read the repository as it was at a point-in-time in order to recover data that has been deleted or corrupted by user accident or malware.

Versioned storage is supported by S3, GCS, and Azure but is generally not enabled by default. In addition to enabling versioning, it may be useful to enable object locking for S3 and soft delete for GCS or Azure.

When the repo-target-time option is specified then the repo option must also be provided. It is likely that not all repository types will support versioning and in general it makes sense to target a single repository for recovery.

Note that comparisons to the storage timestamp are <= the timestamp provided and milliseconds are truncated from the timestamp when provided.

To demonstrate this feature the demo stanza in the S3 repo is deleted.

pg-primary Delete stanza in S3 repository

```
sudo systemctl stop postgresql-14.service
```

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

```
sudo -u postgres pgbackrest --stanza=demo --repo=3 stanza-delete
```

Once the stanza is deleted the info command will show the repository in an error state.

pg-primary Error on info

```
sudo -u postgres pgbackrest --stanza=demo --repo=3 info
```

```
stanza: demo

    status: error (missing stanza path)
```

However, since the storage is versioned, it is possible to look at the repository at a time before the stanza was deleted. Finding the target time can be tricky depending on the situation, but in this case the time when the stanza was deleted can be determined by checking when backup.info was deleted.

pg-primary List versions of backup.info in the bucket

```
key=demo-repo/backup/demo/backup.info; \
       aws s3api list-object-versions --bucket demo-bucket \
       --prefix $key --output table \
       --query "sort_by([Versions[?Key=='$key'].{Action:'PUT', \
       Modified:LastModified,Object:Key}, \
       DeleteMarkers[?Key=='$key'].{Action:'DELETE', \
       Modified:LastModified,Object:Key}][],&Modified)"
```

```
-------------------------------------------------------------------------------------
|                                  ListObjectVersions                                 |
+--------+-----------------------------------+-----------------------------------+
| Action |             Modified              |               Object              |
+--------+-----------------------------------+-----------------------------------+
|  PUT   |  2026-08-17T04:45:10.531000+00:00  |  demo-repo/backup/demo/backup.info  |
|  PUT   |  2026-08-17T04:45:25.802000+00:00  |  demo-repo/backup/demo/backup.info  |
|  DELETE|  2026-08-17T04:45:36.676000+00:00  |  demo-repo/backup/demo/backup.info  |
+--------+-----------------------------------+-----------------------------------+
```

Now the info command can be run with a target time that will show the repository before it was deleted.

pg-primary Info with target time

```
sudo -u postgres pgbackrest --stanza=demo --repo=3 \
       --repo-target-time="2026-08-17 04:45:26+00" info
```

```
        [filtered 5 lines of output]
         wal archive min/max (14): 0000000500000000000001C/0000000500000000000001D

         full backup: 20260817-044510F

             timestamp start/stop: 2026-08-17 04:45:10+00 / 2026-08-17 04:45:25+00
             wal start/stop: 0000000500000000000001D / 0000000500000000000001D
             repo3: backup set size: 4.2MB, backup size: 4.2MB
```

If the required backup is shown by the info command then it can be restored using the same target time.

pg-primary Restore with target time

```
sudo -u postgres pgbackrest --stanza=demo --repo=3 --delta \
       --repo-target-time="2026-08-17 04:45:26+00" --log-level-console=info restore

P00   INFO: restore command begin 2.59.1: --delta --exec-id=5643-cba10b18 --log-level-consol

P00   INFO: repo3: restore backup set 20260817-044510F, recovery will start at 2026-08-17 04

P00   INFO: remove invalid files/links/paths from '/var/lib/pgsql/14/data'
P00   INFO: write updated /var/lib/pgsql/14/data/postgresql.auto.conf
       [filtered 2 lines of output]

sudo systemctl start postgresql-14.service
```

Dedicated Repository Host

The configuration described in Quickstart is suitable for simple installations but for enterprise configurations it is more typical to have a dedicated repository host where the backups and WAL archive files are stored. 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 repository host.

On PostgreSQL hosts, pg1-path is required to be the path of the local PostgreSQL cluster and no pg1-host should be configured. When configuring a repository host, the pgbackrest configuration file must have the pg-host option configured to connect to the primary and standby (if any) hosts. The repository host has the only pgbackrest configuration that should be aware of more than one PostgreSQL host. Order does not matter, e.g. pg1-path/pg1-host, pg2-path/pg2-host can be primary or standby.

Installation

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

NOTE:

The pgBackRest version installed on the repository host must exactly match the version installed on the PostgreSQL host.

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

NOTE:

When pgBackRest is installed from a package, a logrotate configuration such as /etc/logrotate.d/pgbackrest may be provided that rotates the logs as a specific user via the su directive (e.g. su postgres postgres). Since the files in /var/log/pgbackrest are owned by the user that runs pgBackRest (here pgbackrest), the su directive must be updated to match this user or logrotate will fail with a permission error.

repository Create pgbackrest user

```
sudo groupadd pgbackrest
```

```
sudo adduser -gpgbackrest -n pgbackrest
```

Installing pgBackRest from a package is preferable to building from source. When installing from a package the rest of the instructions in this section are generally not required, but it is possible that a package will skip creating one of the directories or apply incorrect permissions. In that case it may be necessary to manually create directories or update permissions.

RHEL packages for pgBackRest are available at yum.postgresql.org.

If packages are not provided for your distribution/version you can build from source and then install manually as shown here.

repository Install dependencies

```
sudo yum install postgresql-libs libssh2
```

repository Copy pgBackRest binary from build host

```
sudo scp build:/build/pgbackrest/src/pgbackrest /usr/bin
```

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

pgBackRest requires log and configuration directories and a configuration file.

repository Create pgBackRest configuration file and directories

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

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

```
sudo mkdir -p /etc/pgbackrest
```

```
sudo mkdir -p /etc/pgbackrest/conf.d
```

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

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

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

repository Create the pgBackRest repository

```
sudo mkdir -p /var/lib/pgbackrest
```

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

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

Configuration

pgBackRest can use TLS with client certificates to enable communication between the hosts. It is also possible to use SSH, see Setup SSH.

pgBackRest expects client/server certificates to be generated in the same way as PostgreSQL. See Secure TCP/IP Connections with TLS for detailed instructions on generating certificates.

The repository host must be configured with the pg-primary host/user and database path. The primary will be configured as pg1 to allow a standby to be added later.

repository:/etc/pgbackrest/pgbackrest.conf Configure pg1-host/pg1-host-user and pg1-path

```
[demo]
pg1-host=pg-primary
pg1-host-ca-file=/etc/pgbackrest/cert/ca.crt
pg1-host-cert-file=/etc/pgbackrest/cert/client.crt
pg1-host-key-file=/etc/pgbackrest/cert/client.key
pg1-host-type=tls
pg1-path=/var/lib/pgsql/14/data

[global]
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y
tls-server-address=*
tls-server-auth=pgbackrest-client=*
tls-server-ca-file=/etc/pgbackrest/cert/ca.crt
tls-server-cert-file=/etc/pgbackrest/cert/server.crt
tls-server-key-file=/etc/pgbackrest/cert/server.key
```

The database host must be configured with the repository host/user. The default for the repo1-host-user option is pgbackrest. If the postgres user does restores on the repository 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 pgbackrest user.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure repo1-host/repo1-host-user

[demo]
pg1-path=/var/lib/pgsql/14/data

[global]
log-level-file=detail
repo1-host=repository
repo1-host-ca-file=/etc/pgbackrest/cert/ca.crt
repo1-host-cert-file=/etc/pgbackrest/cert/client.crt
repo1-host-key-file=/etc/pgbackrest/cert/client.key
repo1-host-type=tls
tls-server-address=*
tls-server-auth=pgbackrest-client=demo
tls-server-ca-file=/etc/pgbackrest/cert/ca.crt
tls-server-cert-file=/etc/pgbackrest/cert/server.crt
tls-server-key-file=/etc/pgbackrest/cert/server.key

PostgreSQL configuration may be found in the Configure Archiving section.

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

Setup TLS Server

The pgBackRest TLS server must be configured and started on each host.

repository Setup pgBackRest Server

```
sudo cat /etc/systemd/system/pgbackrest.service
```

```
[Unit]
Description=pgBackRest Server
After=network.target
StartLimitIntervalSec=0

[Service]
Type=notify
Restart=always
RestartSec=1
User=pgbackrest
ExecStart=/usr/bin/pgbackrest server
ExecStartPost=/bin/sleep 3
ExecStartPost=/bin/bash -c "[ ! -z $MAINPID ]"
ExecReload=/bin/kill -HUP $MAINPID

[Install]
WantedBy=multi-user.target
```

```
sudo systemctl enable pgbackrest
```

```
sudo systemctl start pgbackrest
```

pg-primary Setup pgBackRest Server

```
sudo cat /etc/systemd/system/pgbackrest.service
```

```
[Unit]
Description=pgBackRest Server
After=network.target
StartLimitIntervalSec=0

[Service]
Type=notify
Restart=always
RestartSec=1
User=postgres
ExecStart=/usr/bin/pgbackrest server
ExecStartPost=/bin/sleep 3
ExecStartPost=/bin/bash -c "[ ! -z $MAINPID ]"
ExecReload=/bin/kill -HUP $MAINPID

[Install]
WantedBy=multi-user.target
```

```
sudo systemctl enable pgbackrest
```

```
sudo systemctl start pgbackrest
```

Create and Check Stanza

Create the stanza in the new repository.

repository Create the stanza

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

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

pg-primary Check the configuration

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

repository Check the configuration

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

Perform a Backup

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

repository Backup the demo cluster

```
sudo -u pgbackrest 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 repository 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.

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

```
sudo systemctl stop postgresql-14.service
```

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

```
sudo systemctl start postgresql-14.service
```

Parallel Backup / Restore

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

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

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

repository Perform a backup with single process

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

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

```
[demo]
pg1-host=pg-primary
pg1-host-ca-file=/etc/pgbackrest/cert/ca.crt
pg1-host-cert-file=/etc/pgbackrest/cert/client.crt
pg1-host-key-file=/etc/pgbackrest/cert/client.key
pg1-host-type=tls
pg1-path=/var/lib/pgsql/14/data

[global]
process-max=3
repo1-path=/var/lib/pgbackrest
```

repo1-retention-full=2
start-fast=y
tls-server-address=*
tls-server-auth=pgbackrest-client=*
tls-server-ca-file=/etc/pgbackrest/cert/ca.crt
tls-server-cert-file=/etc/pgbackrest/cert/server.crt
tls-server-key-file=/etc/pgbackrest/cert/server.key

repository Perform a backup with multiple processes

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

repository Get backup info for the demo cluster

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

```
stanza: demo
    status: ok
    cipher: none

    db (current)
        wal archive min/max (14): 000000070000000000000023/000000070000000000000025

        full backup: 20260817-044625F

            timestamp start/stop: 2026-08-17 04:46:25+00 / 2026-08-17 04:46:29+00

            wal start/stop: 000000070000000000000023 / 000000070000000000000023
            database size: 33.5MB, database backup size: 33.5MB
            repo1: backup set size: 4.2MB, backup size: 4.2MB

        full backup: 20260817-044630F

            timestamp start/stop: 2026-08-17 04:46:30+00 / 2026-08-17 04:46:32+00

            wal start/stop: 000000070000000000000024 / 000000070000000000000025
            database size: 33.5MB, database backup size: 33.5MB
            repo1: backup set size: 4.2MB, backup size: 4.2MB
```

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

Starting and Stopping

If a standby is promoted for testing, or a test cluster is restored from a production backup, then it is a good idea to prevent those clusters from writing to pgBackRest repositories. This can be accomplished with the stop command.

The commands that write and are blocked by stop are: archive-push, backup, expire, stanza-create, and stanza-upgrade. Note that stanza-delete is an exception to this rule (see Delete a Stanza for more details).

pg-primary Stop pgBackRest write commands

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

New pgBackRest write commands will no longer run.

repository Attempt a backup

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

```
P00   WARN: unable to check pg1: [StopError] raised from remote-0 tls protocol on 'pg-primar
```

```
P00  ERROR: [056]: unable to find primary cluster - cannot proceed
            HINT: are all available clusters in recovery?
```

Specify the --force option to terminate any pgBackRest write commands that are currently running. This includes asynchronous archive-get (though it will run again if PostgreSQL requires it). If pgBackRest is already stopped then stopping again will generate a warning.

pg-primary Stop the pgBackRest services again

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

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

Start pgBackRest write commands again with the start command. Write commands that were in progress before the stop will not automatically start again, but they are now allowed to start.

pg-primary Start pgBackRest write commands

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

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

pg-primary Stop pgBackRest write commands for the demo stanza

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

New pgBackRest write commands for the specified stanza will no longer run.

repository Attempt a backup

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

```
P00   WARN: unable to check pg1: [StopError] raised from remote-0 tls protocol on 'pg-primar
```

```
P00  ERROR: [056]: unable to find primary cluster - cannot proceed
            HINT: are all available clusters in recovery?
```

The stanza must also be specified when starting pgBackRest write commands for a single stanza.

pg-primary Start pgBackRest write commands for the demo stanza

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

Replication

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

Installation

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

Installing pgBackRest from a package is preferable to building from source. When installing from a package the rest of the instructions in this section are generally not required, but it is possible that a package will skip creating one of the directories or apply incorrect permissions. In that case it may be necessary to manually create directories or update permissions.

RHEL packages for pgBackRest are available at yum.postgresql.org.

If packages are not provided for your distribution/version you can build from source and then install manually as shown here.

pg-standby Install dependencies

```
sudo yum install postgresql-libs libssh2
```

pg-standby Copy pgBackRest binary from build host

```
sudo scp build:/build/pgbackrest/src/pgbackrest /usr/bin
```

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

pgBackRest requires log and configuration directories and a configuration file.

pg-standby Create pgBackRest configuration file and directories

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

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

```
sudo mkdir -p /etc/pgbackrest
```

```
sudo mkdir -p /etc/pgbackrest/conf.d
```

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

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

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

Hot Standby

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

pgBackRest configuration is very similar to pg-primary except that the standby recovery type will be used to keep the cluster in recovery mode when the end of the WAL stream has been reached.

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

```
[demo]
pg1-path=/var/lib/pgsql/14/data

[global]
log-level-file=detail
repo1-host=repository
repo1-host-ca-file=/etc/pgbackrest/cert/ca.crt
repo1-host-cert-file=/etc/pgbackrest/cert/client.crt
repo1-host-key-file=/etc/pgbackrest/cert/client.key
repo1-host-type=tls
tls-server-address=*
tls-server-auth=pgbackrest-client=demo
tls-server-ca-file=/etc/pgbackrest/cert/ca.crt
tls-server-cert-file=/etc/pgbackrest/cert/server.crt
tls-server-key-file=/etc/pgbackrest/cert/server.key
```

pg-standby Setup pgBackRest Server

```
sudo cat /etc/systemd/system/pgbackrest.service
```

```
[Unit]
Description=pgBackRest Server
After=network.target
StartLimitIntervalSec=0

[Service]
Type=notify
Restart=always
RestartSec=1
User=postgres
ExecStart=/usr/bin/pgbackrest server
ExecStartPost=/bin/sleep 3
ExecStartPost=/bin/bash -c "[ ! -z $MAINPID ]"
ExecReload=/bin/kill -HUP $MAINPID

[Install]
WantedBy=multi-user.target
```

```
sudo systemctl enable pgbackrest
```

```
sudo systemctl start pgbackrest
```

Create the path where PostgreSQL will be restored.

pg-standby Create PostgreSQL path

```
sudo -u postgres mkdir -p -m 700 /var/lib/pgsql/14/data
```

Now the standby can be created with the restore command.

IMPORTANT:

If the cluster is intended to be promoted without becoming the new primary (e.g. for reporting or testing), use --archive-mode=off or set archive_mode=off in postgresql.conf to disable archiving. If archiving is not disabled then the repository may be polluted with WAL that can make restores more difficult.

pg-standby Restore the demo standby cluster

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

```
sudo -u postgres cat /var/lib/pgsql/14/data/postgresql.auto.conf
```

```
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:43:50
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:44:12
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:44:37
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
# Removed by pgBackRest restore on 2026-08-17 04:45:41 # recovery_target_time = '2026-08-17
# Removed by pgBackRest restore on 2026-08-17 04:45:41 # recovery_target_action = 'promote'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:45:41
restore_command = 'pgbackrest --repo=3 --repo-target-time="2026-08-17 04:45:26+00" --stanza=

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:46:21
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:46:54
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
```

The configuration is in case the standby is promoted to a primary.

pg-standby:/var/lib/pgsql/14/data/postgresql.conf Configure PostgreSQL

archive_command = 'pgbackrest --stanza=demo archive-push %p'
archive_mode = on
log_filename = 'postgresql.log'

pg-standby Start PostgreSQL

```
sudo systemctl start postgresql-14.service
```

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.

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

```
sudo -u postgres cat /var/lib/pgsql/14/data/log/postgresql.log

       [filtered 4 lines of output]
LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432"
LOG:  database system was interrupted; last known up at 2026-08-17 04:46:30 UTC

LOG:  entering standby mode

LOG:  restored log file "00000007.history" from archive
LOG:  restored log file "000000070000000000000024" from archive
       [filtered 3 lines of output]
```

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

pg-primary Create a new table on the primary

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

       [filtered 4 lines of output]
    message
----------------

 Important Data

(1 row)
```

And then query the same table on pg-standby.

pg-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 pg-primary.

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

pg-primary Call pg_switch_wal()

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

```
 pg_switch_wal |       current_timestamp
---------------+-------------------------------
 0/2601E7C0    | 2026-08-17 04:47:00.421919+00
(1 row)
```

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

pg-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     |      current_timestamp
----------------+-------------------------------

 Important Data | 2026-08-17 04:47:01.60213+00

(1 row)
```

Check the standby configuration for access to the repository.

pg-standby Check the configuration

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

```
P00   INFO: check command begin 2.59.1: --exec-id=1243-de0d9475 --log-level-console=info --l
P00   INFO: check repo1 (standby)

P00   INFO: switch wal not performed because this is a standby

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

Streaming Replication

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

Streaming replication requires a user with the replication privilege.

pg-primary Create replication user

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

```
CREATE ROLE
```

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

pg-primary Create pg_hba.conf entry for replication user

```
sudo -u postgres sh -c 'echo \
       "host    replication     replicator      172.17.0.8/32           md5" \
       >> /var/lib/pgsql/14/data/pg_hba.conf'

sudo systemctl reload postgresql-14.service
```

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

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

```
[demo]
pg1-path=/var/lib/pgsql/14/data
recovery-option=primary_conninfo=host=172.17.0.6 port=5432 user=replicator

[global]
log-level-file=detail
repo1-host=repository
repo1-host-ca-file=/etc/pgbackrest/cert/ca.crt
repo1-host-cert-file=/etc/pgbackrest/cert/client.crt
repo1-host-key-file=/etc/pgbackrest/cert/client.key
repo1-host-type=tls
tls-server-address=*
tls-server-auth=pgbackrest-client=demo
tls-server-ca-file=/etc/pgbackrest/cert/ca.crt
tls-server-cert-file=/etc/pgbackrest/cert/server.crt
tls-server-key-file=/etc/pgbackrest/cert/server.key
```

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

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

```
sudo -u postgres sh -c 'echo \
       "172.17.0.6:*:replication:replicator:jw8s0F4" \
       >> /var/lib/pgsql/.pgpass'

sudo -u postgres chmod 600 /var/lib/pgsql/.pgpass
```

Now the standby can be created with the restore command.

pg-standby Stop PostgreSQL and restore the demo standby cluster

```
sudo systemctl stop postgresql-14.service

sudo -u postgres pgbackrest --stanza=demo --delta --type=standby restore

sudo -u postgres cat /var/lib/pgsql/14/data/postgresql.auto.conf

# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:43:50
```

```
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:44:12
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:44:37
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
# Removed by pgBackRest restore on 2026-08-17 04:45:41 # recovery_target_time = '2026-08-17
# Removed by pgBackRest restore on 2026-08-17 04:45:41 # recovery_target_action = 'promote'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:45:41
restore_command = 'pgbackrest --repo=3 --repo-target-time="2026-08-17 04:45:26+00" --stanza=

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:46:21
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:47:07
primary_conninfo = 'host=172.17.0.6 port=5432 user=replicator'
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
```

NOTE:

The primary_conninfo setting has been written into the postgresql.auto.conf file because it was configured as a recovery-option in pgbackrest.conf. The --type=preserve option can be used with the restore to leave the existing postgresql.auto.conf file in place if that behavior is preferred.

By default RHEL stores the postgresql.conf file in the PostgreSQL data directory. That means the change made to postgresql.conf was overwritten by the last restore and the hot_standby setting must be enabled again. Other solutions to this problem are to store the postgresql.conf file elsewhere or to enable the hot_standby setting on the pg-primary host where it will be ignored.

pg-standby:/var/lib/pgsql/14/data/postgresql.conf Enable hot_standby

archive_command = 'pgbackrest --stanza=demo archive-push %p'
archive_mode = on
hot_standby = on
log_filename = 'postgresql.log'

pg-standby Start PostgreSQL

```
sudo systemctl start postgresql-14.service
```

The PostgreSQL log will confirm that streaming replication has started.

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

```
sudo -u postgres cat /var/lib/pgsql/14/data/log/postgresql.log
```

```
       [filtered 12 lines of output]
LOG:  database system is ready to accept read-only connections
LOG:  restored log file "000000070000000000000026" from archive

LOG:  started streaming WAL from primary at 0/27000000 on timeline 7
```

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

pg-primary Create a new table on the primary

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

       [filtered 4 lines of output]
    message     |       current_timestamp
----------------+-------------------------------

 Important Data | 2026-08-17 04:47:12.811608+00

(1 row)
```

pg-standby Query table on the standby

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

    message     |       current_timestamp
----------------+-------------------------------

 Important Data | 2026-08-17 04:47:12.996136+00

(1 row)
```

Multiple Stanzas

pgBackRest supports multiple stanzas. The most common usage is sharing a repository host among multiple stanzas.

Installation

A new host named pg-alt is created to run the new primary.

Installing pgBackRest from a package is preferable to building from source. When installing from a package the rest of the instructions in this section are generally not required, but it is possible that a package will skip creating one of the directories or apply incorrect permissions. In that case it may be necessary to manually create directories or update permissions.

RHEL packages for pgBackRest are available at yum.postgresql.org.

If packages are not provided for your distribution/version you can build from source and then install manually as shown here.

pg-alt Install dependencies

```
sudo yum install postgresql-libs libssh2
```

pg-alt Copy pgBackRest binary from build host

```
sudo scp build:/build/pgbackrest/src/pgbackrest /usr/bin
```

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

pgBackRest requires log and configuration directories and a configuration file.

pg-alt Create pgBackRest configuration file and directories

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

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

```
sudo mkdir -p /etc/pgbackrest
```

```
sudo mkdir -p /etc/pgbackrest/conf.d
```

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

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

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

Configuration

pgBackRest configuration is nearly identical to pg-primary except that the demo-alt stanza will be used so backups and archive will be stored in a separate location.

pg-alt:/etc/pgbackrest/pgbackrest.conf Configure pgBackRest on the new primary

```
[demo-alt]
pg1-path=/var/lib/pgsql/14/data

[global]
log-level-file=detail
repo1-host=repository
repo1-host-ca-file=/etc/pgbackrest/cert/ca.crt
repo1-host-cert-file=/etc/pgbackrest/cert/client.crt
repo1-host-key-file=/etc/pgbackrest/cert/client.key
repo1-host-type=tls
tls-server-address=*
tls-server-auth=pgbackrest-client=demo-alt
tls-server-ca-file=/etc/pgbackrest/cert/ca.crt
tls-server-cert-file=/etc/pgbackrest/cert/server.crt
tls-server-key-file=/etc/pgbackrest/cert/server.key
```

repository:/etc/pgbackrest/pgbackrest.conf Configure pg1-host/pg1-host-user and pg1-path

[demo]
pg1-host=pg-primary
pg1-host-ca-file=/etc/pgbackrest/cert/ca.crt
pg1-host-cert-file=/etc/pgbackrest/cert/client.crt
pg1-host-key-file=/etc/pgbackrest/cert/client.key
pg1-host-type=tls
pg1-path=/var/lib/pgsql/14/data

[demo-alt]
pg1-host=pg-alt
pg1-host-ca-file=/etc/pgbackrest/cert/ca.crt
pg1-host-cert-file=/etc/pgbackrest/cert/client.crt
pg1-host-key-file=/etc/pgbackrest/cert/client.key
pg1-host-type=tls
pg1-path=/var/lib/pgsql/14/data

[global]
process-max=3
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y
tls-server-address=*
tls-server-auth=pgbackrest-client=*
tls-server-ca-file=/etc/pgbackrest/cert/ca.crt
tls-server-cert-file=/etc/pgbackrest/cert/server.crt
tls-server-key-file=/etc/pgbackrest/cert/server.key

pg-alt Setup pgBackRest Server

```
sudo cat /etc/systemd/system/pgbackrest.service
```

```
[Unit]
Description=pgBackRest Server
After=network.target
StartLimitIntervalSec=0

[Service]
Type=notify
Restart=always
RestartSec=1
User=postgres
ExecStart=/usr/bin/pgbackrest server
ExecStartPost=/bin/sleep 3
ExecStartPost=/bin/bash -c "[ ! -z $MAINPID ]"
ExecReload=/bin/kill -HUP $MAINPID
```

```
[Install]
WantedBy=multi-user.target
```

```
sudo systemctl enable pgbackrest
```

```
sudo systemctl start pgbackrest
```

Setup Demo Cluster

pg-alt Create the demo cluster

```
sudo -u postgres /usr/pgsql-14/bin/initdb \
       -D /var/lib/pgsql/14/data -k -A peer
```

pg-alt:/var/lib/pgsql/14/data/postgresql.conf Configure PostgreSQL settings

archive_command = 'pgbackrest --stanza=demo-alt archive-push %p'
archive_mode = on
log_filename = 'postgresql.log'

pg-alt Start the demo cluster

```
sudo systemctl restart postgresql-14.service
```

Create the Stanza and Check Configuration

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

pg-alt Create the stanza and check the configuration

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

```
P00   INFO: stanza-create command begin 2.59.1: --exec-id=924-4397c8dc --log-level-console=i
P00   INFO: stanza-create for stanza 'demo-alt' on repo1
```

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

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

```
P00   INFO: check command begin 2.59.1: --exec-id=957-d8d57749 --log-level-console=info --lo
```

```
P00   INFO: check stanza 'demo-alt'
```

```
P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)
```

```
P00   INFO: WAL segment 000000010000000000000001 successfully archived to '/var/lib/pgbackre
```

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

If the check command is run from the repository host then all stanzas will be checked.

repository Check the configuration for all stanzas

```
sudo -u pgbackrest pgbackrest --log-level-console=info check

P00   INFO: check command begin 2.59.1: --exec-id=1369-1c661e91 --log-level-console=info --r

P00   INFO: check stanza 'demo'

P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)

P00   INFO: WAL segment 000000070000000000000027 successfully archived to '/var/lib/pgbackre
P00   INFO: check stanza 'demo-alt'

P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)

P00   INFO: WAL segment 000000010000000000000002 successfully archived to '/var/lib/pgbackre

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

Asynchronous Archiving

Asynchronous archiving is enabled with the archive-async option. This option enables asynchronous operation for both the archive-push and archive-get commands.

A spool path is required. The commands will store transient data here but each command works quite a bit differently so spool path usage is described in detail in each section.

pg-primary Create the spool directory

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

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

pg-standby Create the spool directory

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

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

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

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

```
[demo]
pg1-path=/var/lib/pgsql/14/data


[global]
archive-async=y
```

```
log-level-file=detail
repo1-host=repository
repo1-host-ca-file=/etc/pgbackrest/cert/ca.crt
repo1-host-cert-file=/etc/pgbackrest/cert/client.crt
repo1-host-key-file=/etc/pgbackrest/cert/client.key
repo1-host-type=tls
spool-path=/var/spool/pgbackrest
tls-server-address=*
tls-server-auth=pgbackrest-client=demo
tls-server-ca-file=/etc/pgbackrest/cert/ca.crt
tls-server-cert-file=/etc/pgbackrest/cert/server.crt
tls-server-key-file=/etc/pgbackrest/cert/server.key

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

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

pg-standby:/etc/pgbackrest/pgbackrest.conf Configure the spool path and asynchronous archiving

```
[demo]
pg1-path=/var/lib/pgsql/14/data
recovery-option=primary_conninfo=host=172.17.0.6 port=5432 user=replicator

[global]
archive-async=y
log-level-file=detail
repo1-host=repository
repo1-host-ca-file=/etc/pgbackrest/cert/ca.crt
repo1-host-cert-file=/etc/pgbackrest/cert/client.crt
repo1-host-key-file=/etc/pgbackrest/cert/client.key
repo1-host-type=tls
spool-path=/var/spool/pgbackrest
tls-server-address=*
tls-server-auth=pgbackrest-client=demo
tls-server-ca-file=/etc/pgbackrest/cert/ca.crt
tls-server-cert-file=/etc/pgbackrest/cert/server.crt
tls-server-key-file=/etc/pgbackrest/cert/server.key

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

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

NOTE:

process-max is configured using command sections so that the option is not used by backup and restore. This also allows different values for archive-push and archive-get.

For demonstration purposes streaming replication will be broken to force PostgreSQL to get WAL using the restore_command.

pg-primary Break streaming replication by changing the replication password

```
sudo -u postgres psql -c "alter user replicator password 'bogus'"
```

```
ALTER ROLE
```

pg-standby Restart standby to break connection

```
sudo systemctl restart postgresql-14.service
```

Archive Push

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

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

IMPORTANT:

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

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

pg-primary Test parallel asynchronous archiving

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

```
       select pg_create_restore_point('test async push'); select pg_switch_wal(); \
       select pg_create_restore_point('test async push'); select pg_switch_wal();"

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

P00   INFO: check command begin 2.59.1: --exec-id=6816-7c4848e0 --log-level-console=info --l
P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)

P00   INFO: WAL segment 00000007000000000000002D successfully archived to '/var/lib/pgbackre

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

Now the log file will contain parallel, asynchronous activity.

pg-primary Check results in the log

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

-------------------PROCESS START-------------------
P00   INFO: archive-push:async command begin 2.59.1: [/var/lib/pgsql/14/data/pg_wal] --archi

P00   INFO: push 1 WAL file(s) to archive: 000000070000000000000028
P01 DETAIL: pushed WAL file '000000070000000000000028' to the archive

P00 DETAIL: statistics: {"socket.client":{"total":1},"socket.session":{"total":1},"tls.clien
P00   INFO: archive-push:async command end: completed successfully


-------------------PROCESS START-------------------
P00   INFO: archive-push:async command begin 2.59.1: [/var/lib/pgsql/14/data/pg_wal] --archi

P00   INFO: push 5 WAL file(s) to archive: 000000070000000000000029...000000070000000000000
P01 DETAIL: pushed WAL file '000000070000000000000029' to the archive
P02 DETAIL: pushed WAL file '00000007000000000000002A' to the archive
P01 DETAIL: pushed WAL file '00000007000000000000002B' to the archive
P02 DETAIL: pushed WAL file '00000007000000000000002C' to the archive
P01 DETAIL: pushed WAL file '00000007000000000000002D' to the archive

P00 DETAIL: statistics: {"socket.client":{"total":1},"socket.session":{"total":1},"tls.clien
P00   INFO: archive-push:async command end: completed successfully
```

Archive Get

The asynchronous archive-get command maintains a local queue of WAL to improve throughput. If a WAL segment is not found in the queue it is fetched from the repository along with enough consecutive WAL to fill the queue. The maximum size of the queue is defined by archive-get-queue-max. Whenever the queue is less than half full more WAL will be fetched to fill it.

Asynchronous operation is most useful in environments that generate a lot of WAL or have a high latency connection to the repository storage (i.e., S3 or other object stores). In the case of a high latency connection it may be a good idea to increase process-max.

The [stanza]-archive-get-async.log file can be used to monitor the activity of the asynchronous process.

pg-standby Check results in the log

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

```
-------------------PROCESS START-------------------
P00   INFO: archive-get:async command begin 2.59.1: [000000070000000000000024, 000000070000
P00   INFO: get 8 WAL file(s) from archive: 000000070000000000000024...000000070000000000000

P01 DETAIL: found 000000070000000000000024 in the repo1: 14-1 archive
P02 DETAIL: found 000000070000000000000025 in the repo1: 14-1 archive
P01 DETAIL: found 000000070000000000000026 in the repo1: 14-1 archive
P02 DETAIL: found 000000070000000000000027 in the repo1: 14-1 archive

P00 DETAIL: unable to find 000000070000000000000028 in the archive
P00 DETAIL: statistics: {"socket.client":{"total":1},"socket.session":{"total":1},"tls.clien
       [filtered 24 lines of output]
P00   INFO: archive-get:async command begin 2.59.1: [000000070000000000000028, 000000070000
P00   INFO: get 8 WAL file(s) from archive: 000000070000000000000028...000000070000000000000

P01 DETAIL: found 000000070000000000000028 in the repo1: 14-1 archive
P02 DETAIL: found 000000070000000000000029 in the repo1: 14-1 archive
P01 DETAIL: found 00000007000000000000002A in the repo1: 14-1 archive
P02 DETAIL: found 00000007000000000000002B in the repo1: 14-1 archive
P01 DETAIL: found 00000007000000000000002C in the repo1: 14-1 archive
P02 DETAIL: found 00000007000000000000002D in the repo1: 14-1 archive

P00 DETAIL: unable to find 00000007000000000000002E in the archive
P00 DETAIL: statistics: {"socket.client":{"total":1},"socket.session":{"total":1},"tls.clien
       [filtered 7 lines of output]
```

pg-primary Fix streaming replication by changing the replication password

```
sudo -u postgres psql -c "alter user replicator password 'jw8s0F4'"
```

```
ALTER ROLE
```

Backup from a Standby

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

repository:/etc/pgbackrest/pgbackrest.conf Configure pg2-host/pg2-host-user and pg2-path

```
[demo]
pg1-host=pg-primary
pg1-host-ca-file=/etc/pgbackrest/cert/ca.crt
pg1-host-cert-file=/etc/pgbackrest/cert/client.crt
```

pg1-host-key-file=/etc/pgbackrest/cert/client.key
pg1-host-type=tls
pg1-path=/var/lib/pgsql/14/data
pg2-host=pg-standby
pg2-host-ca-file=/etc/pgbackrest/cert/ca.crt
pg2-host-cert-file=/etc/pgbackrest/cert/client.crt
pg2-host-key-file=/etc/pgbackrest/cert/client.key
pg2-host-type=tls
pg2-path=/var/lib/pgsql/14/data

[demo-alt]
pg1-host=pg-alt
pg1-host-ca-file=/etc/pgbackrest/cert/ca.crt
pg1-host-cert-file=/etc/pgbackrest/cert/client.crt
pg1-host-key-file=/etc/pgbackrest/cert/client.key
pg1-host-type=tls
pg1-path=/var/lib/pgsql/14/data

[global]
backup-standby=y
process-max=3
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y
tls-server-address=*
tls-server-auth=pgbackrest-client=*
tls-server-ca-file=/etc/pgbackrest/cert/ca.crt
tls-server-cert-file=/etc/pgbackrest/cert/server.crt
tls-server-key-file=/etc/pgbackrest/cert/server.key

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

repository Backup the demo cluster from pg2

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

       [filtered 2 lines of output]
P00   INFO: execute backup start: backup begins after the requested immediate checkpoint com
P00   INFO: backup start archive = 000000070000000000000002F, lsn = 0/2F000028

P00   INFO: wait for replay on the standby to reach 0/2F000028
P00   INFO: replay on the standby reached 0/2F000028

P00   INFO: check archive for prior segment 000000070000000000000002E
```

```
P01 DETAIL: backup file pg-primary:/var/lib/pgsql/14/data/log/postgresql.log (10.9KB, 0.42%)
P01 DETAIL: backup file pg-primary:/var/lib/pgsql/14/data/global/pg_control (8KB, 0.73%) che
P01 DETAIL: backup file pg-primary:/var/lib/pgsql/14/data/pg_hba.conf (4.5KB, 0.90%) checksu

P01 DETAIL: match file from prior backup pg-primary:/var/lib/pgsql/14/data/current_logfiles
P01 DETAIL: match file from prior backup pg-primary:/var/lib/pgsql/14/data/pg_logical/replor
       [filtered 1263 lines of output]
```

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

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

Upgrading PostgreSQL

Immediately after upgrading PostgreSQL to a newer major version, the pg-path for all pgBackRest configurations must be set to the new database location and the stanza-upgrade command run. If there is more than one repository configured on the host, the stanza will be upgraded on each. If the database is offline use the --no-online option.

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

pg-primary Stop old cluster

```
sudo systemctl stop postgresql-14.service
```

Stop the old cluster on the standby since it will be restored from the newly upgraded cluster.

pg-standby Stop old cluster

```
sudo systemctl stop postgresql-14.service
```

Create the new cluster and perform upgrade.

pg-primary Create new cluster and perform the upgrade

```
sudo -u postgres /usr/pgsql-15/bin/initdb \
       -D /var/lib/pgsql/15/data -k -A peer

sudo -u postgres sh -c 'cd /var/lib/pgsql && \
       /usr/pgsql-15/bin/pg_upgrade \
       --old-bindir=/usr/pgsql-14/bin \
       --new-bindir=/usr/pgsql-15/bin \
```

```
        --old-datadir=/var/lib/pgsql/14/data \
        --new-datadir=/var/lib/pgsql/15/data \
        --old-options=" -c config_file=/var/lib/pgsql/14/data/postgresql.conf" \
        --new-options=" -c config_file=/var/lib/pgsql/15/data/postgresql.conf"'

        [filtered 41 lines of output]
Checking for extension updates                             ok

Upgrade Complete

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

Configure the new cluster settings and port.

pg-primary:/var/lib/pgsql/15/data/postgresql.conf Configure PostgreSQL

archive_command = 'pgbackrest --stanza=demo archive-push %p'
archive_mode = on
log_filename = 'postgresql.log'

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

pg-primary:/etc/pgbackrest/pgbackrest.conf Upgrade the pg1-path

[demo]
pg1-path=/var/lib/pgsql/15/data

[global]
archive-async=y
log-level-file=detail
repo1-host=repository
repo1-host-ca-file=/etc/pgbackrest/cert/ca.crt
repo1-host-cert-file=/etc/pgbackrest/cert/client.crt
repo1-host-key-file=/etc/pgbackrest/cert/client.key
repo1-host-type=tls
spool-path=/var/spool/pgbackrest
tls-server-address=*
tls-server-auth=pgbackrest-client=demo
tls-server-ca-file=/etc/pgbackrest/cert/ca.crt
tls-server-cert-file=/etc/pgbackrest/cert/server.crt
tls-server-key-file=/etc/pgbackrest/cert/server.key

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

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

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

[demo]
pg1-path=/var/lib/pgsql/15/data
recovery-option=primary_conninfo=host=172.17.0.6 port=5432 user=replicator

[global]
archive-async=y
log-level-file=detail
repo1-host=repository
repo1-host-ca-file=/etc/pgbackrest/cert/ca.crt
repo1-host-cert-file=/etc/pgbackrest/cert/client.crt
repo1-host-key-file=/etc/pgbackrest/cert/client.key
repo1-host-type=tls
spool-path=/var/spool/pgbackrest
tls-server-address=*
tls-server-auth=pgbackrest-client=demo
tls-server-ca-file=/etc/pgbackrest/cert/ca.crt
tls-server-cert-file=/etc/pgbackrest/cert/server.crt
tls-server-key-file=/etc/pgbackrest/cert/server.key

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

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

repository:/etc/pgbackrest/pgbackrest.conf Upgrade pg1-path and pg2-path, disable backup from standby

[demo]
pg1-host=pg-primary
pg1-host-ca-file=/etc/pgbackrest/cert/ca.crt
pg1-host-cert-file=/etc/pgbackrest/cert/client.crt
pg1-host-key-file=/etc/pgbackrest/cert/client.key
pg1-host-type=tls
pg1-path=/var/lib/pgsql/15/data
pg2-host=pg-standby
pg2-host-ca-file=/etc/pgbackrest/cert/ca.crt
pg2-host-cert-file=/etc/pgbackrest/cert/client.crt
pg2-host-key-file=/etc/pgbackrest/cert/client.key
pg2-host-type=tls
pg2-path=/var/lib/pgsql/15/data

[demo-alt]
pg1-host=pg-alt
pg1-host-ca-file=/etc/pgbackrest/cert/ca.crt
pg1-host-cert-file=/etc/pgbackrest/cert/client.crt
pg1-host-key-file=/etc/pgbackrest/cert/client.key

pg1-host-type=tls
pg1-path=/var/lib/pgsql/14/data

[global]
backup-standby=n
process-max=3
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y
tls-server-address=*
tls-server-auth=pgbackrest-client=*
tls-server-ca-file=/etc/pgbackrest/cert/ca.crt
tls-server-cert-file=/etc/pgbackrest/cert/server.crt
tls-server-key-file=/etc/pgbackrest/cert/server.key

pg-primary Copy hba configuration

```
sudo cp /var/lib/pgsql/14/data/pg_hba.conf \
       /var/lib/pgsql/15/data/pg_hba.conf
```

Before starting the new cluster, the stanza-upgrade command must be run.

pg-primary Upgrade the stanza

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

```
P00   INFO: stanza-upgrade command begin 2.59.1: --exec-id=7404-f5b1280d --log-level-console
P00   INFO: stanza-upgrade for stanza 'demo' on repo1
```

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

Start the new cluster and confirm it is successfully installed.

pg-primary Start new cluster

```
sudo systemctl start postgresql-15.service
```

Test configuration using the check command.

pg-primary Check configuration

```
sudo systemctl status postgresql-15.service
```

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

Remove the old cluster.

pg-primary Remove old cluster

```
sudo rm -rf /var/lib/pgsql/14/data
```

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

pg-standby Remove old cluster and create the new cluster

```
sudo rm -rf /var/lib/pgsql/14/data
```

```
sudo -u postgres mkdir -p -m 700 /usr/pgsql-15/bin
```

Run the check on the repository host. The warning regarding the standby being down is expected since the standby cluster is down. Running this command demonstrates that the repository server is aware of the standby and is configured properly for the primary server.

repository Check configuration

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

```
P00   WARN: unable to check pg2: [DbConnectError] raised from remote-0 tls protocol on 'pg-s
                Is the server running locally and accepting connections on that socket?
```

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

repository Run a full backup

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

pg-standby Restore the demo standby cluster

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

pg-standby Start PostgreSQL and check the pgBackRest configuration

```
sudo systemctl start postgresql-15.service
```

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

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

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

```
[demo]
pg1-host=pg-primary
pg1-host-ca-file=/etc/pgbackrest/cert/ca.crt
pg1-host-cert-file=/etc/pgbackrest/cert/client.crt
pg1-host-key-file=/etc/pgbackrest/cert/client.key
pg1-host-type=tls
pg1-path=/var/lib/pgsql/15/data
pg2-host=pg-standby
pg2-host-ca-file=/etc/pgbackrest/cert/ca.crt
pg2-host-cert-file=/etc/pgbackrest/cert/client.crt
pg2-host-key-file=/etc/pgbackrest/cert/client.key
pg2-host-type=tls
pg2-path=/var/lib/pgsql/15/data

[demo-alt]
pg1-host=pg-alt
```

```
pg1-host-ca-file=/etc/pgbackrest/cert/ca.crt
pg1-host-cert-file=/etc/pgbackrest/cert/client.crt
pg1-host-key-file=/etc/pgbackrest/cert/client.key
pg1-host-type=tls
pg1-path=/var/lib/pgsql/14/data

[global]
backup-standby=y
process-max=3
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y
tls-server-address=*
tls-server-auth=pgbackrest-client=*
tls-server-ca-file=/etc/pgbackrest/cert/ca.crt
tls-server-cert-file=/etc/pgbackrest/cert/server.crt
tls-server-key-file=/etc/pgbackrest/cert/server.key
```


pgBackRest User Guide

Debian & Ubuntu

Home

User Guides

Releases

Configuration

Commands

News

FAQ

Metrics

Table of Contents

Introduction

Concepts

Backup

Restore

Write Ahead Log (WAL)

Encryption

Upgrading pgBackRest
Upgrading pgBackRest from v2.x to v2.y
Build
Installation
Quick Start
Setup Demo Cluster
Configure Cluster Stanza
Create the Repository
Configure Archiving
Configure Retention
Configure Repository Encryption
Create the Stanza
Check the Configuration
Performance Tuning
Perform a Backup
Schedule a Backup
Backup Information
Restore a Backup
Monitoring
In PostgreSQL
Using jq
Backup
File Bundling
Block Incremental
Backup Annotations
Retention
Full Backup Retention
Differential Backup Retention
Archive Retention
Restore
File Ownership

Delta Option

Restore Selected Databases

Point-in-Time Recovery

Delete a Stanza

Multiple Repositories

Azure-Compatible Object Store Support

S3-Compatible Object Store Support

SFTP Support

GCS-Compatible Object Store Support

Target Time for Repository

Dedicated Repository Host

Installation

Setup Passwordless SSH

Configuration

Create and Check Stanza

Perform a Backup

Restore a Backup

Parallel Backup / Restore

Starting and Stopping

Replication

Installation

Setup Passwordless SSH

Hot Standby

Streaming Replication

Multiple Stanzas

Installation

Setup Passwordless SSH

Configuration

Setup Demo Cluster

Create the Stanza and Check Configuration

Asynchronous Archiving

Archive Push

Archive Get

Backup from a Standby

Upgrading PostgreSQL

Introduction

This user guide is intended to be followed sequentially from beginning to end — each section depends on the last. For example, the Restore 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 in this guide are targeted at Debian/Ubuntu and PostgreSQL 17, it should be fairly easy to apply the examples to any Unix distribution and PostgreSQL version. The only OS-specific commands are those to create, start, stop, and drop PostgreSQL clusters. The pgBackRest commands will be the same on any Unix system though the location of the executable may vary. While pgBackRest strives to operate consistently across versions of PostgreSQL, there are subtle differences between versions of PostgreSQL that may show up in this guide when illustrating certain examples, e.g. PostgreSQL path/file names and settings.

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

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

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

Restore

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

Write Ahead Log (WAL)

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

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

Encryption

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

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

Upgrading pgBackRest

Upgrading pgBackRest from v2.x to v2.y

Upgrading from v2.x to v2.y is straight-forward. The repository format has not changed, so for most installations it is simply a matter of installing binaries for the new version. It is also possible to downgrade if you have not used new features that are unsupported by the older version.

IMPORTANT:

The local and remote pgBackRest versions must match exactly so they should be upgraded together. If there is a mismatch, WAL archiving and backups will not function until the versions match. In such a case, the following error will be reported: [ProtocolError] expected value '2.x' for greeting key 'version' but got '2.y'.

Build

Installing pgBackRest from a package is preferable to building from source. See Installation for more information about packages.

When building from source it is best to use a build host rather than building on production. Many of the tools required for the build should generally not be installed in production. pgBackRest consists of a single executable so it is easy to copy to a new host once it is built.

build Download version 2.59.1 of pgBackRest to /build path

```
mkdir -p /build
```

```
curl -fsSL \
       https://github.com/pgbackrest/pgbackrest/releases/download/release%2F2.59.1/pgbackres
       tar zx -C /build
```

build Install build dependencies

```
sudo apt-get install python3-distutils meson gcc libpq-dev libssl-dev libxml2-dev \
       pkg-config liblz4-dev libzstd-dev libbz2-dev libz-dev libssh2-1-dev libsystemd-dev
```

build Configure and compile pgBackRest

```
meson setup /build/pgbackrest /build/pgbackrest-2.59.1
```

```
ninja -C /build/pgbackrest
```

build Optionally run smoke tests to verify pgBackRest was built correctly

```
meson test -C /build/pgbackrest --suite smoke
```

```
ninja: Entering directory `/build/pgbackrest'
ninja: no work to do.
```

```
1/1 smoke OK                11.75s

       [filtered 7 lines of output]
```

Installation

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

Installing pgBackRest from a package is preferable to building from source. When installing from a package the rest of the instructions in this section are generally not required, but it is possible that a package will skip creating one of the directories or apply incorrect permissions. In that case it may be necessary to manually create directories or update permissions.

Debian/Ubuntu packages for pgBackRest are available at apt.postgresql.org.

If packages are not provided for your distribution/version you can build from source and then install manually as shown here.

pg-primary Install dependencies

```
sudo apt-get install postgresql-client libxml2 libssh2-1
```

pg-primary Copy pgBackRest binary from build host

```
sudo scp build:/build/pgbackrest/src/pgbackrest /usr/bin

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

pgBackRest requires log and configuration directories and a configuration file.

pg-primary Create pgBackRest configuration file and directories

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

sudo chown postgres:postgres /var/log/pgbackrest

sudo mkdir -p /etc/pgbackrest

sudo mkdir -p /etc/pgbackrest/conf.d

sudo touch /etc/pgbackrest/pgbackrest.conf

sudo chmod 640 /etc/pgbackrest/pgbackrest.conf

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

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.

pg-primary Make sure the installation worked

```
sudo -u postgres pgbackrest
```

```
pgBackRest 2.59.1 - General help

Usage:
    pgbackrest [options] [command]

Commands:
    annotate        add or modify backup annotation
    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
    repo-get        get a file from a repository
    repo-ls         list files in a repository
    restore         restore a database cluster
    server          pgBackRest server
    server-ping     ping pgBackRest server
    stanza-create   create the required stanza data
    stanza-delete   delete a stanza
    stanza-upgrade  upgrade a stanza
    start           allow pgBackRest processes to run
    stop            stop pgBackRest processes from running
    verify          verify contents of a repository
    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.

pg-primary Create the demo cluster

```
sudo -u postgres /usr/lib/postgresql/17/bin/initdb \
       -D /var/lib/postgresql/17/demo -k -A peer
```

```
sudo pg_createcluster 17 demo
```

```
Configuring already existing cluster (configuration: /etc/postgresql/17/demo, data: /var/lib
```

```
Ver Cluster Port Status Owner    Data directory              Log file
17  demo    5432 down   postgres /var/lib/postgresql/17/demo /var/log/postgresql/postgresql-
```

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 PostgreSQL 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 pg-path is exactly equal to data_directory as reported by PostgreSQL.

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

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

[demo]
pg1-path=/var/lib/postgresql/17/demo

pgBackRest configuration files follow a Windows INI-like 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, but end-of-line comments following a value on the same line are not supported. Quoting is not supported and whitespace is trimmed from keys and values. Sections will be merged if they appear more than once.

There are multiple ways the pgBackRest configuration files can be loaded:

- config and config-include-path are default: the default config file will be loaded, if it exists, and *.conf files in the default config include path will be appended, if they exist.

- config option is specified: only the specified config file will be loaded and is expected to exist.
- config-include-path is specified: *.conf files in the config include path will be loaded and the path is required to exist. The default config file will be be loaded if it exists. If it is desirable to load only the files in the specified config include path, then the --no-config option can also be passed.
- config and config-include-path are specified: using the user-specified values, the config file will be loaded and *.conf files in the config include path will be appended. The files are expected to exist.
- config-path is specified: this setting will override the base path for the default location of the config file and/or the base path of the default config-include-path setting unless the config and/or config-include-path option is explicitly set.

Files are concatenated as if they were one big file and each file must be valid individually. This means sections must be specified in each file where they are needed to store a key/value. Order doesn't matter but there is precedence based on sections. The precedence (highest to lowest) is:

- [*stanza*:*command*]
- [*stanza*]
- [global:*command*]
- [global]

NOTE:

--config, --config-include-path and --config-path are command-line only options.

pgBackRest can also be configured using environment variables as described in the command reference.

pg-primary Configure log-path using the environment

```
sudo -u postgres bash -c ' \
       export PGBACKREST_LOG_PATH=/path/set/by/env && \
       pgbackrest --log-level-console=error help backup log-path'

pgBackRest 2.59.1 - 'backup' command - 'log-path' option help

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that
if log-level-file=off then no log path is required.

current: /path/set/by/env

default: /var/log/pgbackrest
```

Create the Repository

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

It may 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.

pg-primary Create the pgBackRest repository

```
sudo mkdir -p /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.

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

[demo]
pg1-path=/var/lib/postgresql/17/demo

[global]
repo1-path=/var/lib/pgbackrest

Multiple repositories may also be configured. See Multiple Repositories for details.

Configure Archiving

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

pg-primary:/etc/postgresql/17/demo/postgresql.conf Configure archive settings

archive_command = 'pgbackrest --stanza=demo archive-push %p'
archive_mode = on

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

pg-primary Restart the demo cluster

```
sudo pg_ctlcluster 17 demo restart
```

The hot_standby setting is enabled by default and should be left that way on every cluster. Any cluster may be restored as a standby later, e.g. a primary

that is rebuilt as a standby after failover, and a standby will not accept read-only connections without it.

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

The archive-push command can be configured with its own options. For example, a lower compression level may be set to speed archiving without affecting the compression used for backups.

pg-primary:/etc/pgbackrest/pgbackrest.conf Config archive-push to use a lower compression level

```
[demo]
pg1-path=/var/lib/postgresql/17/demo

[global]
repo1-path=/var/lib/pgbackrest

[global:archive-push]
compress-level=3
```

This configuration technique can be used for any command and can even target a specific stanza, e.g. demo:archive-push.

Configure Retention

pgBackRest expires backups based on retention options.

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

```
[demo]
pg1-path=/var/lib/postgresql/17/demo

[global]
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2

[global:archive-push]
compress-level=3
```

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

Configure Repository Encryption

The repository will be configured with a cipher type and key to demonstrate encryption. Encryption is always performed client-side even if the repository type (e.g. S3 or other object store) supports encryption.

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

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

```
[demo]
pg1-path=/var/lib/postgresql/17/demo

[global]
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2

[global:archive-push]
compress-level=3
```

NOTE:

Encryption settings are placed in the [global] section, above, so the info command can read all stanzas. Without the stanza option the info command reads encryption settings only from the [global] section, so encryption settings configured per stanza require the stanza option to read an encrypted stanza.

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

Create the Stanza

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

pg-primary Create the stanza and check the configuration

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

P00   INFO: stanza-create command begin 2.59.1: --exec-id=407-5732f46e --log-level-console=i
P00   INFO: stanza-create for stanza 'demo' on repo1

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 for the specified stanza. It will attempt to check all repositories and databases that are configured for the host on which the command is run. 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 PostgreSQL or repository host. The command may also be run on the standby host, however, since pg_switch_xlog()/pg_switch_wal() cannot be performed on the standby, the command will only test the repository configuration.

Note that pg_create_restore_point('pgBackRest Archive Check') and pg_switch_xlog()/pg_switch_wal() are called to force PostgreSQL to archive a WAL segment.

pg-primary Check the configuration

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

```
P00   INFO: check command begin 2.59.1: --exec-id=416-47cf298c --log-level-console=info --nc
P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)
```

```
P00   INFO: WAL segment 000000010000000000000001 successfully archived to '/var/lib/pgbackre
```

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

Performance Tuning

pgBackRest has a number of performance options that are not enabled by default to maintain backward compatibility in the repository. However, when creating a new repository the following options are recommended. They can also be used on an existing repository with the caveat that older versions of pgBackRest will not be able to read the repository. This incompatibility depends on when the feature was introduced, as noted in the list below.

- compress-type - determines the compression algorithm used by the backup and archive-push commands. The default is gz (Gzip) but zst (Zstandard) is recommended because it is much faster and provides compression similar to gz. zst has been supported by the compress-type option since v2.27. See Compress Type for more details.
- repo-bundle - combines small files during backup to save space and improve the speed of both the backup and restore commands, especially on object stores such as S3. The repo-bundle option was introduced in v2.39. See File Bundling for more details.
- repo-block - stores only the portions of files that have changed rather than the entire file during diff/incr backup. This saves space and increases the speed of the backup. The repo-block option was introduced in v2.46 but at least v2.52.1 is recommended. See Block Incremental for more details.

There are other performance options that are not enabled by default because they require additional configuration or because the default is safe (but not optimal). These options are available in all v2 versions of pgBackRest.

- process-max - determines how many processes will be used for commands. The default is 1, which is almost never the appropriate value. Each com-

mand uses process-max differently so refer to each command's documentation for details on usage.
- archive-async - archives WAL files to the repository in batch which greatly increases archiving speed. It is not enabled by default because it requires a spool path to be created. See Asynchronous Archiving for more details.
- backup-standby - performs the backup on a standby rather than the primary to reduce load on the primary. It is not enabled by default because it requires additional configuration and the presence of one or more standby hosts. See Backup from a Standby for more details.

Perform a Backup

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. Generally, it is best to set start-fast=y so that the backup starts immediately. This forces a checkpoint, but since backups are usually run once a day an additional checkpoint should not have a noticeable impact on performance. However, on very busy clusters it may be best to pass --start-fast on the command-line as needed.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure backup fast start

```
[demo]
pg1-path=/var/lib/postgresql/17/demo

[global]
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y

[global:archive-push]
compress-level=3
```

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

pg-primary Backup the demo cluster

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

P00   INFO: backup command begin 2.59.1: --exec-id=443-5b80dea3 --log-level-console=info --r

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

P00   INFO: execute backup start: backup begins after the requested immediate checkpoint com
P00   INFO: backup start archive = 000000010000000000000002, lsn = 0/2000028
       [filtered 3 lines of output]
```

```
P00   INFO: check archive for segment(s) 000000010000000000000002:000000010000000000000003
P00   INFO: new backup label = 20260817-045036F

P00   INFO: full backup size = 22MB, file total = 963

P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.59.1: --exec-id=443-5b80dea3 --log-level-console=info --r
```

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.

pg-primary Differential backup of the demo cluster

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

       [filtered 7 lines of output]
P00   INFO: check archive for segment(s) 000000010000000000000004:000000010000000000000005
P00   INFO: new backup label = 20260817-045036F_20260817-045039D

P00   INFO: diff backup size = 8.3KB, file total = 963

P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.59.1: --exec-id=468-69e2296f --log-level-console=info --r
```

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.

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

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.

pg-primary Get info for the demo cluster

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

```
stanza: demo
    status: ok
    cipher: aes-256-cbc

    db (current)
        wal archive min/max (17): 000000010000000000000001/000000010000000000000005

        full backup: 20260817-045036F

            timestamp start/stop: 2026-08-17 04:50:36+00 / 2026-08-17 04:50:39+00
            wal start/stop: 000000010000000000000002 / 000000010000000000000003
            database size: 22MB, database backup size: 22MB
            repo1: backup set size: 2.9MB, backup size: 2.9MB

        diff backup: 20260817-045036F_20260817-045039D

            timestamp start/stop: 2026-08-17 04:50:39+00 / 2026-08-17 04:50:41+00
            wal start/stop: 000000010000000000000004 / 000000010000000000000005
            database size: 22MB, database backup size: 8.3KB
            repo1: backup set size: 2.9MB, backup size: 464B
            backup reference total: 1 full
```

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 and is kept stable unless a bug is found.

To speed up execution, limit the output to only progress information by specifying --detail-level=progress. Note that this skips all checks except for availability of the stanza.

Each stanza has a separate section and it is possible to limit output to a single stanza with the --stanza option. The stanza 'status' gives a brief indication of the stanza's health. If this is 'ok' then pgBackRest is functioning normally. If there are multiple repositories, then a status of 'mixed' indicates that the stanza is not in a healthy state on one or more of the repositories; in this case the state of the stanza will be detailed per repository. For cases in which an error on a repository occurred that is not one of the known error codes, then an error code of 'other' will be used and the full error details will be provided. The 'wal archive min/max' shows the minimum and maximum WAL currently stored in the archive and, in the case of multiple repositories, will be reported across all

repositories unless the --repo option is set. Note that there may be gaps due to archive retention policies or other reasons.

The 'backup/expire running' and/or 'restore running' messages will appear beside the 'status' information if any of those commands are currently running on the host. Per-repo progress will be also reported in text output and a 'repo' array will be included in JSON output.

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

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

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

The 'database size' is the full uncompressed size of the database while 'database backup size' is the amount of data in the database to actually back up (these will be the same for full backups).

The 'repo' indicates in which repository this backup resides. The 'backup set size' includes all the files from this backup and any referenced backups in the repository that are required to restore the database from this backup while '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.

The 'backup reference total' summarizes the list of additional backups that are required to restore this backup. Use the --set option to display the complete reference list.

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.

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

```
sudo pg_ctlcluster 17 demo stop
```

```
sudo -u postgres rm /var/lib/postgresql/17/demo/global/pg_control
```

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

pg-primary Attempt to start the corrupted demo cluster

```
sudo pg_ctlcluster 17 demo start
```

```
Error: /usr/lib/postgresql/17/bin/pg_ctl /usr/lib/postgresql/17/bin/pg_ctl start -D /var/lib

postgres: could not find the database system

Expected to find it in the directory "/var/lib/postgresql/17/demo",
but could not open file "/var/lib/postgresql/17/demo/global/pg_control": No such file or dir
Examine the log output.
```

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.

pg-primary Remove old files from demo cluster

```
sudo -u postgres find /var/lib/postgresql/17/demo -mindepth 1 -delete
```

pg-primary Restore the demo cluster and start PostgreSQL

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

sudo pg_ctlcluster 17 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.

Monitoring

Monitoring is an important part of any production system. There are many tools available and pgBackRest can be monitored on any of them with a little work.

pgBackRest can output information about the repository in JSON format which includes a list of all backups for each stanza and WAL archive info.

In PostgreSQL

The PostgreSQL COPY command allows pgBackRest info to be loaded into a table. The following example wraps that logic in a function that can be used to perform real-time queries.

pg-primary Load pgBackRest info function for PostgreSQL

```
sudo -u postgres cat \
       /var/lib/postgresql/pgbackrest/doc/example/pgsql-pgbackrest-info.sql

-- An example of monitoring pgBackRest from within PostgreSQL
--
-- Use copy to export data from the pgBackRest info command into the jsonb
-- type so it can be queried directly by PostgreSQL.

-- Create monitor schema
create schema monitor;
```

```
-- Get pgBackRest info in JSON format
create function monitor.pgbackrest_info()
    returns jsonb AS $$
declare
    data jsonb;
begin
    -- Create a temp table to hold the JSON data
    create temp table temp_pgbackrest_data (data text);

    -- Copy data into the table directly from the pgBackRest info command
    copy temp_pgbackrest_data (data)
        from program
            'pgbackrest --output=json info' (format text);

    select replace(temp_pgbackrest_data.data, E'\n', '\n')::jsonb
      into data
      from temp_pgbackrest_data;

    drop table temp_pgbackrest_data;

    return data;
end $$ language plpgsql;
```

```
sudo -u postgres psql -f \
       /var/lib/postgresql/pgbackrest/doc/example/pgsql-pgbackrest-info.sql
```

Now the monitor.pgbackrest_info() function can be used to determine the last successful backup time and archived WAL for a stanza.

pg-primary Query last successful backup time and archived WAL

```
sudo -u postgres cat \
       /var/lib/postgresql/pgbackrest/doc/example/pgsql-pgbackrest-query.sql
```

```
-- Get last successful backup for each stanza
--
-- Requires the monitor.pgbackrest_info function.
with stanza as
(
    select data->'name' as name,
           data->'backup'->(
               jsonb_array_length(data->'backup') - 1) as last_backup,
           data->'archive'->(
               jsonb_array_length(data->'archive') - 1) as current_archive
      from jsonb_array_elements(monitor.pgbackrest_info()) as data
)
select name,
```

```
       to_timestamp(
           (last_backup->'timestamp'->>'stop')::numeric) as last_successful_backup,
       current_archive->>'max' as last_archived_wal
  from stanza;
```

```
sudo -u postgres psql -f \
       /var/lib/postgresql/pgbackrest/doc/example/pgsql-pgbackrest-query.sql
```

```
  name  | last_successful_backup |    last_archived_wal
--------+------------------------+--------------------------
 "demo" | 2026-08-17 04:50:41+00 | 000000010000000000000005
(1 row)
```

Using jq

jq is a command-line utility that can easily extract data from JSON.

pg-primary Install jq utility

```
sudo apt-get install jq
```

Now jq can be used to query the last successful backup time for a stanza.

pg-primary Query last successful backup time

```
sudo -u postgres pgbackrest --output=json --stanza=demo info | \
       jq '.[0] | .backup[-1] | .timestamp.stop'
```

```
1786942241
```

Or the last archived WAL.

pg-primary Query last archived WAL

```
sudo -u postgres pgbackrest --output=json --stanza=demo info | \
       jq '.[0] | .archive[-1] | .max'
```

```
"000000010000000000000005"
```

NOTE:

This syntax requires jq v1.5.

NOTE:

jq may round large numbers such as system identifiers. Test your queries carefully.

Backup

When multiple repositories are configured, pgBackRest will backup to the highest priority repository (e.g. repo1) unless the --repo option is specified.

pgBackRest does not have a built-in scheduler so it's best to run it from cron or some other scheduling mechanism.

See Perform a Backup for more details and examples.

File Bundling

Bundling files together in the repository saves time during the backup and some space in the repository. This is especially pronounced when the repository is stored on an object store such as S3 or file systems with large block sizes. Per-file creation time on object stores is higher and very small files might cost as much to store as larger files.

The file bundling feature is enabled with the repo-bundle option.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure repo1-bundle

```
[demo]
pg1-path=/var/lib/postgresql/17/demo

[global]
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y

[global:archive-push]
compress-level=3
```

A full backup without file bundling will have 1000+ files in the backup path, but with bundling the total number of files is greatly reduced. An additional benefit is that zero-length files are not stored (except in the manifest), whereas in a normal backup each zero-length file is stored individually.

pg-primary Perform a full backup

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

pg-primary Check file total

```
sudo -u postgres find /var/lib/pgbackrest/backup/demo/latest/ -type f | wc -l

5
```

The repo-bundle-size and repo-bundle-limit options can be used for tuning, though the defaults should be optimal in most cases.

While file bundling is generally more efficient, the downside is that it is more difficult to manually retrieve files from the repository. It may not be ideal for deduplicated storage since each full backup will arrange files in the bundles differently. Lastly, file bundles cannot be resumed, so be careful not to set repo-bundle-limit too high.

Block Incremental

Block incremental backups save space by only storing the parts of a file that have changed since the prior backup rather than storing the entire file.

The block incremental feature is enabled with the repo-block option and it works best when enabled for all backup types. File bundling must also be enabled.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure repo1-block

```
[demo]
pg1-path=/var/lib/postgresql/17/demo


[global]
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y


[global:archive-push]
compress-level=3
```

Backup Annotations

Users can attach informative key/value pairs to the backup. This option may be used multiple times to attach multiple annotations.

pg-primary Perform a full backup with annotations

```
sudo -u postgres pgbackrest --stanza=demo --annotation=source="demo backup" \
       --annotation=key=value --type=full backup
```

Annotations are output by the info command text output when a backup is specified with --set and always appear in the JSON output.

pg-primary Get info for the demo cluster

```
sudo -u postgres pgbackrest --stanza=demo --set=20260817-045055F info
```

```
stanza: demo
    status: ok
    cipher: aes-256-cbc

    db (current)
        wal archive min/max (17): 000000020000000000000007/000000020000000000000009

        full backup: 20260817-045055F
            timestamp start/stop: 2026-08-17 04:50:55+00 / 2026-08-17 04:50:56+00
            wal start/stop: 000000020000000000000008 / 000000020000000000000009
            lsn start/stop: 0/8000028 / 0/9000050
```

```
            database size: 22MB, database backup size: 22MB
            repo1: backup size: 2.9MB
            database list: postgres (5)

            annotation(s)

                key: value
                source: demo backup
```

Annotations included with the backup command can be added, modified, or removed afterwards using the annotate command.

pg-primary Change backup annotations

```
sudo -u postgres pgbackrest --stanza=demo --set=20260817-045055F \
       --annotation=key= --annotation=new_key=new_value annotate

sudo -u postgres pgbackrest --stanza=demo --set=20260817-045055F info

stanza: demo
    status: ok
    cipher: aes-256-cbc

    db (current)
        wal archive min/max (17): 000000020000000000000007/000000020000000000000009

        full backup: 20260817-045055F
            timestamp start/stop: 2026-08-17 04:50:55+00 / 2026-08-17 04:50:56+00
            wal start/stop: 000000020000000000000008 / 000000020000000000000009
            lsn start/stop: 0/8000028 / 0/9000050
            database size: 22MB, database backup size: 22MB
            repo1: backup size: 2.9MB
            database list: postgres (5)

            annotation(s)

                new_key: new_value
                source: demo backup
```

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.

pgBackRest does full backup rotation based on the retention type which can be a count or a time period. When a count is specified, then expiration is not concerned with when the backups were created but with how many must be retained. Differential backups are count-based but will always be expired when the full backup they depend on is expired. Incremental backups are not expired by retention independently — they are always expired with their related full or

differential backup. See sections Full Backup Retention and Differential Backup Retention for details and examples.

Archived WAL is retained by default for backups that have not expired, however, although not recommended, this schedule can be modified per repository with the retention-archive options. See section Archive Retention for details and examples.

The expire command is run automatically after each successful backup and can also be run by the user. When run by the user, expiration will occur as defined by the retention settings for each configured repository. If the --repo option is provided, expiration will occur only on the specified repository. Expiration can also be limited by the user to a specific backup set with the --set option and, unless the --repo option is specified, all repositories will be searched and any matching the set criteria will be expired. It should be noted that the archive retention schedule will be checked and performed any time the expire command is run.

Full Backup Retention

The repo1-retention-full-type determines how the option repo1-retention-full is interpreted; either as the count of full backups to be retained or how many days to retain full backups. New backups must be completed before expiration will occur — that means if repo1-retention-full-type=count and repo1-retention-full=2 then there will be three full backups stored before the oldest one is expired, or if repo1-retention-full-type=time and repo1-retention-full=20 then there must be one full backup that is at least 20 days old before expiration can occur.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure repo1-retention-full

```
[demo]
pg1-path=/var/lib/postgresql/17/demo

[global]
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y

[global:archive-push]
compress-level=3
```

Backup repo1-retention-full=2 but currently there is only one full backup so the next full backup to run will not expire any full backups.

pg-primary Perform a full backup

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

       [filtered 975 lines of output]
P00   INFO: repo1: remove expired backup 20260817-045053F
P00 DETAIL: repo1: 17-1 archive retention on backup 20260817-045055F, start = 00000002000000

P00   INFO: repo1: 17-1 remove archive, start = 000000020000000000000007, stop = 00000002000

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.

pg-primary Perform a full backup

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

       [filtered 11 lines of output]
P00   INFO: repo1: expire full backup 20260817-045055F
P00   INFO: repo1: remove expired backup 20260817-045055F

P00   INFO: repo1: 17-1 remove archive, start = 000000020000000000000008, stop = 00000002000

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

The 20260817-045036F full backup is expired and archive retention is based on the 20260817-045058F which is now the oldest full backup.

Differential Backup Retention

Set repo1-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.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure repo1-retention-diff

```
[demo]
pg1-path=/var/lib/postgresql/17/demo


[global]
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-diff=1
repo1-retention-full=2
start-fast=y
```

[global:archive-push]
compress-level=3

Backup repo1-retention-diff=1 so two differentials will need to be performed before one is expired. An incremental backup is added to demonstrate incremental expiration, which in this case depends on the differential expiration.

pg-primary Perform differential and incremental backups

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

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

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

pg-primary Perform a differential backup

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

       [filtered 10 lines of output]
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.59.1: --exec-id=930-d66fce46 --log-level-console=info --r

P00   INFO: repo1: expire diff backup set 20260817-045100F_20260817-045102D, 20260817-04510

P00   INFO: repo1: remove expired backup 20260817-045100F_20260817-045103I
P00   INFO: repo1: remove expired backup 20260817-045100F_20260817-045102D
P00   INFO: expire command end: completed successfully
```

Archive Retention

Although pgBackRest automatically removes archived WAL segments when expiring backups (the default expires WAL for full backups based on the repo1-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. To determine what will be expired without actually expiring anything, the dry-run option can be provided on the command line with the expire command.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure repo1-retention-diff

[demo]
pg1-path=/var/lib/postgresql/17/demo

[global]
repo1-block=y

repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-diff=2
repo1-retention-full=2
start-fast=y

[global:archive-push]
compress-level=3

pg-primary Perform differential backup

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

       [filtered 6 lines of output]
P00   INFO: backup stop archive = 000000020000000000000017, lsn = 0/17000050
P00   INFO: check archive for segment(s) 000000020000000000000016:000000020000000000000017

P00   INFO: new backup label = 20260817-045100F_20260817-045106D

P00   INFO: diff backup size = 8.3KB, file total = 963
P00   INFO: backup command end: completed successfully
       [filtered 2 lines of output]
```

pg-primary Expire archive

```
sudo -u postgres pgbackrest --stanza=demo --log-level-console=detail \
       --repo1-retention-archive-type=diff --repo1-retention-archive=1 expire

P00   INFO: expire command begin 2.59.1: --exec-id=1009-97f280d8 --log-level-console=detail
P00 DETAIL: repo1: 17-1 archive retention on backup 20260817-045058F, start = 0000000200000
P00 DETAIL: repo1: 17-1 archive retention on backup 20260817-045100F, start = 0000000200000

P00 DETAIL: repo1: 17-1 archive retention on backup 20260817-045100F_20260817-045104D, start

P00 DETAIL: repo1: 17-1 archive retention on backup 20260817-045100F_20260817-045106D, start

P00   INFO: repo1: 17-1 remove archive, start = 000000020000000000000000E, stop = 00000002000
P00   INFO: repo1: 17-1 remove archive, start = 000000020000000000000014, stop = 00000002000

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

The 20260817-045100F_20260817-045104D 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 20260817-045100F_20260817-045104D but before 20260817-045100F_20260817-045106D are removed. WAL segments generated after the new backup 20260817-045100F_20260817-045106D 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 command automatically defaults to selecting the latest backup from the first repository where backups exist (see Quick Start - Restore a Backup). The order in which the repositories are checked is dictated by the pgbackrest.conf (e.g. repo1 will be checked before repo2). To select from a specific repository, the --repo option can be passed (e.g. --repo=1). The --set option can be passed if a backup other than the latest is desired.

When PITR of --type=time or --type=lsn is specified, then the target time or target lsn must be specified with the --target option. If a backup is not specified via the --set option, then the configured repositories will be checked, in order, for a backup that contains the requested time or lsn. If no matching backup is found, the latest backup from the first repository containing backups will be used for --type=time while no backup will be selected for --type=lsn. For other types of PITR, e.g. xid, the --set option must be provided if the target is prior to the latest backup. See Point-in-Time Recovery for more details and examples.

Replication slots are not included per recommendation of PostgreSQL. See Backing Up The Data Directory in the PostgreSQL documentation for more information.

The following sections introduce additional restore command features.

File Ownership

If a restore is run as a non-root user (the typical scenario) then all files restored will belong to the user/group executing pgBackRest. If existing files are not owned by the executing user/group then an error will result if the ownership cannot be updated to the executing user/group. In that case the file ownership will need to be updated by a privileged user before the restore can be retried.

If a restore is run as the root user then pgBackRest will attempt to recreate the ownership recorded in the manifest when the backup was made. Only user/group **names** are stored in the manifest so the same names must exist on the restore host for this to work. If the user/group name cannot be found locally then the user/group of the PostgreSQL data directory will be used and finally root if the data directory user/group cannot be mapped to a name.

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.

pg-primary Stop the demo cluster, perform delta restore

```
sudo pg_ctlcluster 17 demo stop
```

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

```
       [filtered 2 lines of output]
P00 DETAIL: check '/var/lib/postgresql/17/demo' exists
P00 DETAIL: remove 'global/pg_control' so cluster will not start if restore does not complet
```

```
P00   INFO: remove invalid files/links/paths from '/var/lib/postgresql/17/demo'
```

```
P00 DETAIL: remove invalid file '/var/lib/postgresql/17/demo/backup_label.old'
P00 DETAIL: remove invalid file '/var/lib/postgresql/17/demo/base/1/pg_internal.init'
       [filtered 769 lines of output]
P01 DETAIL: restore file /var/lib/postgresql/17/demo/base/1/113 - exists and matches backup
P01 DETAIL: restore file /var/lib/postgresql/17/demo/base/1/112 - exists and matches backup
```

```
P01 DETAIL: restore file /var/lib/postgresql/17/demo/PG_VERSION - exists and matches backup
```

```
P01 DETAIL: restore file /var/lib/postgresql/17/demo/base/5/2608_fsm - exists and matches ba
P01 DETAIL: restore file /var/lib/postgresql/17/demo/postgresql.auto.conf - exists and match
       [filtered 233 lines of output]
```

pg-primary Restart PostgreSQL

```
sudo pg_ctlcluster 17 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.

pg-primary Create two test databases

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

```
CREATE DATABASE
```

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

```
CREATE DATABASE
```

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

pg-primary Create a test table in each database

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

```
CREATE TABLE
INSERT 0 1
```

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

```
CREATE TABLE
INSERT 0 1
```

A fresh backup is run so pgBackRest is aware of the new databases.

pg-primary Perform a backup

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

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.

pg-primary Show space used by test1 database

```
sudo -u postgres du -sh /var/lib/postgresql/17/demo/base/32768
```

```
7.4M   /var/lib/postgresql/17/demo/base/32768
```

If the database to restore is not known, use the info command set option to discover databases that are part of the backup set.

pg-primary Show database list for backup

```
sudo -u postgres pgbackrest --stanza=demo \
       --set=20260817-045100F_20260817-045113I info
```

```
       [filtered 12 lines of output]
            repo1: backup size: 1.9MB
            backup reference list: 20260817-045100F, 20260817-045100F_20260817-045106D

            database list: postgres (5), test1 (32768), test2 (32769)
```

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

WARNING:

Recovery may error unless --type=immediate is specified. This is because after consistency is reached PostgreSQL will flag zeroed pages as errors even for a full-page write. For PostgreSQL 13 the ignore_invalid_pages setting may be

used to ignore invalid pages. In this case it is important to check the logs after recovery to ensure that no invalid pages were reported in the selected databases.

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

```
sudo pg_ctlcluster 17 demo stop
```

```
sudo -u postgres pgbackrest --stanza=demo --delta \
       --db-include=test2 --type=immediate --target-action=promote restore
```

```
sudo pg_ctlcluster 17 demo start
```

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

pg-primary Demonstrate that the test2 database was recovered

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

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

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

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

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

```
psql: error: connection to server on socket "/var/run/postgresql/.s.PGSQL.5432" failed: FATA
```

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.

pg-primary Show space used by test1 database after recovery

```
sudo -u postgres du -sh /var/lib/postgresql/17/demo/base/32768
```

```
8.0K   /var/lib/postgresql/17/demo/base/32768
```

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.

pg-primary Drop the test1 database

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

```
DROP DATABASE
```

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

pg-primary List remaining databases

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

```
  oid  |  datname
-------+-----------
     1 | template1
     4 | template0
     5 | postgres

 32769 | 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 a backup to a specified lsn, 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.

pg-primary Create a table with very important data

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

       [filtered 4 lines of output]
    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.

pg-primary Get the time from PostgreSQL

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

```
2026-08-17 04:51:24.23782+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.

pg-primary Drop the important table

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

```
BEGIN
DROP TABLE
```

```
COMMITERROR:  relation "important_table" does not exist
```

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

If the wrong backup is selected for restore then recovery to the required time target will fail. To demonstrate this a new incremental backup is performed where important_table does not exist.

pg-primary Perform an incremental backup

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

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

```
       [filtered 38 lines of output]
            backup reference total: 1 full, 1 diff

        incr backup: 20260817-045100F_20260817-045125I

            timestamp start/stop: 2026-08-17 04:51:25+00 / 2026-08-17 04:51:26+00
            wal start/stop: 00000004000000000000001A / 00000004000000000000001A
       [filtered 2 lines of output]
```

It will not be possible to recover the lost table from this backup since PostgreSQL can only play forward, not backward.

pg-primary Attempt recovery from an incorrect backup

```
sudo pg_ctlcluster 17 demo stop
```

```
sudo -u postgres pgbackrest --stanza=demo --delta \
       --set=20260817-045100F_20260817-045125I --target-timeline=current \
       --type=time "--target=2026-08-17 04:51:24.23782+00" --target-action=promote restore

sudo pg_ctlcluster 17 demo start

       [filtered 13 lines of output]
LOG:  database system is ready to accept read-only connections
LOG:  redo done at 0/1A000120 system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.02

FATAL:  recovery ended before configured recovery target was reached

LOG:  startup process (PID 1402) exited with exit code 1
LOG:  terminating any other active server processes
       [filtered 3 lines of output]
```

A reliable method is to allow pgBackRest to automatically select a backup capable of recovery to the time target, i.e. a backup that ended before the specified time.

NOTE:

pgBackRest cannot automatically select a backup when the restore type is xid or name.

pg-primary Restore the demo cluster to 2026-08-17 04:51:24.23782+00

```
sudo -u postgres pgbackrest --stanza=demo --delta \
       --type=time "--target=2026-08-17 04:51:24.23782+00" \
       --target-action=promote restore

sudo -u postgres cat /var/lib/postgresql/17/demo/postgresql.auto.conf

       [filtered 9 lines of output]
# Recovery settings generated by pgBackRest restore on 2026-08-17 04:51:28
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

recovery_target_time = '2026-08-17 04:51:24.23782+00'

recovery_target_action = 'promote'
```

pgBackRest has generated the recovery settings in postgresql.auto.conf so PostgreSQL can be started immediately. %f is how PostgreSQL specifies the WAL segment it needs and %p is the location where it should be copied. Once PostgreSQL has finished recovery the table will exist again and can be queried.

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

```
sudo pg_ctlcluster 17 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.

pg-primary Examine the PostgreSQL log output

```
sudo -u postgres cat /var/log/postgresql/postgresql-17-demo.log

       [filtered 7 lines of output]
LOG:  restored log file "00000004.history" from archive
LOG:  restored log file "000000040000000000000019" from archive

LOG:  starting point-in-time recovery to 2026-08-17 04:51:24.23782+00

LOG:  restored log file "00000003.history" from archive
LOG:  redo starts at 0/19000028
       [filtered 2 lines of output]
LOG:  database system is ready to accept read-only connections
LOG:  restored log file "00000004000000000000001A" from archive

LOG:  recovery stopping before commit of transaction 748, time 2026-08-17 04:51:25.556252+00

LOG:  redo done at 0/1901CC00 system usage: CPU: user: 0.00 s, system: 0.01 s, elapsed: 0.09

LOG:  last completed transaction was at log time 2026-08-17 04:51:22.942835+00

LOG:  restored log file "000000040000000000000019" from archive
LOG:  selected new timeline ID: 5
       [filtered 5 lines of output]
```

Delete a Stanza

The stanza-delete command removes data in the repository associated with a stanza.

WARNING:

Use this command with caution — it will permanently remove all backups and archives from the pgBackRest repository for the specified stanza.

To delete a stanza:

- Shut down the PostgreSQL cluster associated with the stanza (or use --force to override).
- Run the stop command on the host where the stanza-delete command will be run.
- Run the stanza-delete command.

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

A stanza may only be deleted from one repository at a time. To delete the stanza from multiple repositories, repeat the stanza-delete command for each repository while specifying the --repo option.

pg-primary Stop PostgreSQL cluster to be removed

```
sudo pg_ctlcluster 17 demo stop
```

pg-primary Stop pgBackRest for the stanza

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

```
P00   INFO: stop command begin 2.59.1: --exec-id=1531-7c532db6 --log-level-console=info --nc
```

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

pg-primary Delete the stanza from one repository

```
sudo -u postgres pgbackrest --stanza=demo --repo=1 \
       --log-level-console=info stanza-delete
```

```
P00   INFO: stanza-delete command begin 2.59.1: --exec-id=1538-cfcbf395 --log-level-console=
```

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

Multiple Repositories

Multiple repositories may be configured as demonstrated in S3 Support. A potential benefit is the ability to have a local repository for fast restores and a remote repository for redundancy.

Some commands, e.g. stanza-create/stanza-upgrade, will automatically work with all configured repositories while others, e.g. stanza-delete, will require a repository to be specified using the repo option. See the command reference for details on which commands require the repository to be specified.

Note that the repo option is not required when only repo1 is configured in order to maintain backward compatibility. However, the repo option *is* required when a single repo is configured as, e.g. repo2. This is to prevent command breakage if a new repository is added later.

The archive-push command will always push WAL to the archive in all configured repositories. When a repository cannot be reached, WAL will still be pushed to other repositories. However, for this to work effectively, archive-async=y must be enabled; otherwise, the other repositories can only get one WAL segment ahead of the unreachable repository. Also, note that if WAL cannot be pushed to any repository, then PostgreSQL will not remove it from the pg_wal directory, which may cause the volume to run out of space.

Backups need to be scheduled individually for each repository. In many cases this is desirable since backup types and retention will vary by repository. Likewise, restores must specify a repository. It is generally better to specify a repository for restores that has low latency/cost even if that means more recovery time. Only restore testing can determine which repository will be most efficient.

Azure-Compatible Object Store Support

pgBackRest supports locating repositories in Azure-compatible object stores. The container used to store the repository must be created in advance — pgBackRest will not do it automatically. The repository can be located in the container root (/) but it's usually best to place it in a subpath so object store logs or other data can also be stored in the container without conflicts.

WARNING:

Do not enable "hierarchical namespace" as this will cause errors during expire.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure Azure

```
[demo]
pg1-path=/var/lib/postgresql/17/demo

[global]
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-diff=2
repo1-retention-full=2
repo2-azure-account=pgbackrest
repo2-azure-container=demo-container
repo2-azure-key=YXpLZXk=
repo2-path=/demo-repo
repo2-retention-full=4
repo2-type=azure
start-fast=y

[global:archive-push]
compress-level=3
```

Shared access signatures may be used by setting the repo2-azure-key-type option to sas and the repo2-azure-key option to the shared access signature token.

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

pg-primary Create the stanza

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

P00   INFO: stanza-create command begin 2.59.1: --exec-id=1617-2bccee33 --log-level-console=
P00   INFO: stanza-create for stanza 'demo' on repo1
P00   INFO: stanza-create for stanza 'demo' on repo2

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

File creation time in Azure is relatively slow so backup/restore performance is improved by enabling file bundling.

pg-primary Backup the demo cluster

```
sudo -u postgres pgbackrest --stanza=demo --repo=2 \
       --log-level-console=info backup

P00   INFO: backup command begin 2.59.1: --exec-id=1626-8e5019a5 --log-level-console=info --

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

P00   INFO: execute backup start: backup begins after the requested immediate checkpoint com
P00   INFO: backup start archive = 000000050000000000000001B, lsn = 0/1B000028
       [filtered 3 lines of output]
P00   INFO: check archive for segment(s) 000000050000000000000001B:000000050000000000000001B
P00   INFO: new backup label = 20260817-045139F

P00   INFO: full backup size = 29.2MB, file total = 1265

P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.59.1: --exec-id=1626-8e5019a5 --log-level-console=info --
```

S3-Compatible Object Store Support

pgBackRest supports locating repositories in S3-compatible object stores. The bucket used to store the repository must be created in advance — pgBackRest will not do it automatically. The repository can be located in the bucket root (/) but it's usually best to place it in a subpath so object store logs or other data can also be stored in the bucket without conflicts.

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

```
[demo]
pg1-path=/var/lib/postgresql/17/demo


[global]
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-diff=2
repo1-retention-full=2
repo2-azure-account=pgbackrest
repo2-azure-container=demo-container
repo2-azure-key=YXpLZXk=
repo2-path=/demo-repo
repo2-retention-full=4
repo2-type=azure
repo3-path=/demo-repo
```

repo3-retention-full=4
repo3-s3-bucket=demo-bucket
repo3-s3-endpoint=s3.us-east-1.amazonaws.com
repo3-s3-key=accessKey1
repo3-s3-key-secret=verySecretKey1
repo3-s3-region=us-east-1
repo3-type=s3
start-fast=y

[global:archive-push]
compress-level=3

NOTE:

The region and endpoint will need to be configured to where the bucket is located. The values given here are for the us-east-1 region.

A role should be created to run pgBackRest and the bucket permissions should be set as restrictively as possible. If the role is associated with an instance in AWS then pgBackRest will automatically retrieve temporary credentials when repo3-s3-key-type=auto, which means that keys do not need to be explicitly set in /etc/pgbackrest/pgbackrest.conf.

This sample Amazon S3 policy will restrict all reads and writes to the bucket and repository path.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::demo-bucket"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:prefix": [
                        "",
                        "demo-repo"
                    ],
                    "s3:delimiter": [
                        "/"
                    ]
                }
            }
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::demo-bucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "demo-repo/*"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectTagging",
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::demo-bucket/demo-repo/*"
            ]
        }
    ]
}
```

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

pg-primary Create the stanza

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

```
       [filtered 4 lines of output]
P00   INFO: stanza 'demo' already exists on repo2 and is valid
P00   INFO: stanza-create for stanza 'demo' on repo3

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

File creation time in S3 is relatively slow so backup/restore performance is improved by enabling file bundling.

pg-primary Backup the demo cluster

```
sudo -u postgres pgbackrest --stanza=demo --repo=3 \
       --log-level-console=info backup

P00   INFO: backup command begin 2.59.1: --exec-id=1726-b8012aef --log-level-console=info --

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

P00   INFO: execute backup start: backup begins after the requested immediate checkpoint com
P00   INFO: backup start archive = 00000005000000000000001D, lsn = 0/1D000028
       [filtered 3 lines of output]
P00   INFO: check archive for segment(s) 00000005000000000000001D:00000005000000000000001D
P00   INFO: new backup label = 20260817-045153F

P00   INFO: full backup size = 29.2MB, file total = 1265

P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.59.1: --exec-id=1726-b8012aef --log-level-console=info --
```

SFTP Support

pgBackRest supports locating repositories on SFTP hosts. SFTP file transfer is relatively slow so commands benefit by increasing process-max to parallelize file transfer.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure SFTP

```
[demo]
pg1-path=/var/lib/postgresql/17/demo


[global]
process-max=4
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-diff=2
repo1-retention-full=2
repo2-azure-account=pgbackrest
repo2-azure-container=demo-container
repo2-azure-key=YXpLZXk=
repo2-path=/demo-repo
repo2-retention-full=4
repo2-type=azure
repo3-path=/demo-repo
repo3-retention-full=4
repo3-s3-bucket=demo-bucket
repo3-s3-endpoint=s3.us-east-1.amazonaws.com
repo3-s3-key=accessKey1
repo3-s3-key-secret=verySecretKey1
```

```
repo3-s3-region=us-east-1
repo3-type=s3
repo4-bundle=y
repo4-path=/demo-repo
repo4-sftp-host=sftp-server
repo4-sftp-host-key-hash-type=sha1
repo4-sftp-host-user=pgbackrest
repo4-sftp-private-key-file=/var/lib/postgresql/.ssh/id_rsa_sftp
repo4-sftp-public-key-file=/var/lib/postgresql/.ssh/id_rsa_sftp.pub
repo4-type=sftp
start-fast=y

[global:archive-push]
compress-level=3
```

When utilizing SFTP, if libssh2 is compiled against OpenSSH then repo4-sftp-public-key-file is optional.

pg-primary Generate SSH keypair for SFTP backup

```
sudo -u postgres mkdir -m 750 -p /var/lib/postgresql/.ssh

sudo -u postgres ssh-keygen -f /var/lib/postgresql/.ssh/id_rsa_sftp \
       -t rsa -b 4096 -N "" -m PEM
```

sftp-server Copy pg-primary SFTP backup public key to sftp-server

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

(sudo ssh root@pg-primary cat /var/lib/postgresql/.ssh/id_rsa_sftp.pub) | \
       sudo -u pgbackrest tee -a /home/pgbackrest/.ssh/authorized_keys
```

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

pg-primary Add sftp-server fingerprint to known_hosts file since repo4-sftp-host-key-check-type defaults to “strict”

```
ssh-keyscan -H sftp-server >> /var/lib/postgresql/.ssh/known_hosts 2>/dev/null
```

pg-primary Create the stanza

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

       [filtered 6 lines of output]
P00   INFO: stanza 'demo' already exists on repo3 and is valid
P00   INFO: stanza-create for stanza 'demo' on repo4

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

pg-primary Backup the demo cluster

```
sudo -u postgres pgbackrest --stanza=demo --repo=4 \
       --log-level-console=info backup
```

```
P00   INFO: backup command begin 2.59.1: --exec-id=1815-96aacc2b --log-level-console=info --
P00   WARN: option 'repo4-retention-full' is not set for 'repo4-retention-full-type=count',
            HINT: to retain full backups indefinitely (without warning), set option 'repo4-r

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

P00   INFO: execute backup start: backup begins after the requested immediate checkpoint com
P00   INFO: backup start archive = 000000050000000000000001F, lsn = 0/1F000028
       [filtered 3 lines of output]
P00   INFO: check archive for segment(s) 000000050000000000000001F:000000050000000000000001F
P00   INFO: new backup label = 20260817-045208F

P00   INFO: full backup size = 29.2MB, file total = 1265

P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.59.1: --exec-id=1815-96aacc2b --log-level-console=info --
P00   INFO: expire command end: completed successfully
```

GCS-Compatible Object Store Support

pgBackRest supports locating repositories in GCS-compatible object stores. The bucket used to store the repository must be created in advance — pgBackRest will not do it automatically. The repository can be located in the bucket root (/) but it's usually best to place it in a subpath so object store logs or other data can also be stored in the bucket without conflicts.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure GCS

```
[demo]
pg1-path=/var/lib/postgresql/17/demo


[global]
process-max=4
repo1-block=y
repo1-bundle=y
repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
repo1-cipher-type=aes-256-cbc
repo1-path=/var/lib/pgbackrest
repo1-retention-diff=2
repo1-retention-full=2
repo2-azure-account=pgbackrest
repo2-azure-container=demo-container
repo2-azure-key=YXpLZXk=
repo2-path=/demo-repo
repo2-retention-full=4
repo2-type=azure
repo3-path=/demo-repo
repo3-retention-full=4
repo3-s3-bucket=demo-bucket
repo3-s3-endpoint=s3.us-east-1.amazonaws.com
```

repo3-s3-key=accessKey1
repo3-s3-key-secret=verySecretKey1
repo3-s3-region=us-east-1
repo3-type=s3
repo4-bundle=y
repo4-path=/demo-repo
repo4-sftp-host=sftp-server
repo4-sftp-host-key-hash-type=sha1
repo4-sftp-host-user=pgbackrest
repo4-sftp-private-key-file=/var/lib/postgresql/.ssh/id_rsa_sftp
repo4-sftp-public-key-file=/var/lib/postgresql/.ssh/id_rsa_sftp.pub
repo4-type=sftp
repo5-gcs-bucket=demo-bucket
repo5-gcs-key=/etc/pgbackrest/gcs-key.json
repo5-path=/demo-repo
repo5-type=gcs
start-fast=y

[global:archive-push]
compress-level=3

When running in GCE set repo5-gcs-key-type=auto to automatically authenticate using the instance service account.

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

File creation time in GCS is relatively slow so backup/restore performance is improved by enabling file bundling.

Target Time for Repository

The target time defines the time that commands use to read a repository on versioned storage. This allows the command to read the repository as it was at a point-in-time in order to recover data that has been deleted or corrupted by user accident or malware.

Versioned storage is supported by S3, GCS, and Azure but is generally not enabled by default. In addition to enabling versioning, it may be useful to enable object locking for S3 and soft delete for GCS or Azure.

When the repo-target-time option is specified then the repo option must also be provided. It is likely that not all repository types will support versioning and in general it makes sense to target a single repository for recovery.

Note that comparisons to the storage timestamp are <= the timestamp provided and milliseconds are truncated from the timestamp when provided.

To demonstrate this feature the demo stanza in the S3 repo is deleted.

pg-primary Delete stanza in S3 repository

```
sudo pg_ctlcluster 17 demo stop
```

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

```
sudo -u postgres pgbackrest --stanza=demo --repo=3 stanza-delete
```

Once the stanza is deleted the info command will show the repository in an error state.

pg-primary Error on info

```
sudo -u postgres pgbackrest --stanza=demo --repo=3 info
```

```
stanza: demo
```

```
    status: error (missing stanza path)
```

However, since the storage is versioned, it is possible to look at the repository at a time before the stanza was deleted. Finding the target time can be tricky depending on the situation, but in this case the time when the stanza was deleted can be determined by checking when backup.info was deleted.

pg-primary List versions of backup.info in the bucket

```
key=demo-repo/backup/demo/backup.info; \
       aws s3api list-object-versions --bucket demo-bucket \
       --prefix $key --output table \
       --query "sort_by([Versions[?Key=='$key'].{Action:'PUT', \
       Modified:LastModified,Object:Key}, \
       DeleteMarkers[?Key=='$key'].{Action:'DELETE', \
       Modified:LastModified,Object:Key}][],&Modified)"
```

```
-------------------------------------------------------------------------
|                           ListObjectVersions                          |
+--------+---------------------------+--------------------------------+
| Action |         Modified          |               Object           |
+--------+---------------------------+--------------------------------+
|  PUT   |  2026-08-17T04:51:53.020Z  |  demo-repo/backup/demo/backup.info  |
|  PUT   |  2026-08-17T04:52:04.161Z  |  demo-repo/backup/demo/backup.info  |
|  DELETE|  2026-08-17T04:52:12.845Z  |  demo-repo/backup/demo/backup.info  |
+--------+---------------------------+--------------------------------+
```

Now the info command can be run with a target time that will show the repository before it was deleted.

pg-primary Info with target time

```
sudo -u postgres pgbackrest --stanza=demo --repo=3 \
       --repo-target-time="2026-08-17 04:52:04+00" info
```

```
       [filtered 5 lines of output]
        wal archive min/max (17): 000000050000000000000001C/000000050000000000000001D
```

```
        full backup: 20260817-045153F
```

```
            timestamp start/stop: 2026-08-17 04:51:53+00 / 2026-08-17 04:52:03+00
            wal start/stop: 000000050000000000000001D / 000000050000000000000001D
            repo3: backup set size: 3.8MB, backup size: 3.8MB
```

If the required backup is shown by the info command then it can be restored using the same target time.

pg-primary Restore with target time

```
sudo -u postgres pgbackrest --stanza=demo --repo=3 --delta \
       --repo-target-time="2026-08-17 04:52:04+00" --log-level-console=info restore

P00   INFO: restore command begin 2.59.1: --delta --exec-id=1906-0f3d66ad --log-level-consol

P00   INFO: repo3: restore backup set 20260817-045153F, recovery will start at 2026-08-17 04

P00   INFO: remove invalid files/links/paths from '/var/lib/postgresql/17/demo'
P00   INFO: write updated /var/lib/postgresql/17/demo/postgresql.auto.conf
       [filtered 2 lines of output]

sudo pg_ctlcluster 17 demo start
```

Dedicated Repository Host

The configuration described in Quickstart is suitable for simple installations but for enterprise configurations it is more typical to have a dedicated repository host where the backups and WAL archive files are stored. 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 repository host.

On PostgreSQL hosts, pg1-path is required to be the path of the local PostgreSQL cluster and no pg1-host should be configured. When configuring a repository host, the pgbackrest configuration file must have the pg-host option configured to connect to the primary and standby (if any) hosts. The repository host has the only pgbackrest configuration that should be aware of more than one PostgreSQL host. Order does not matter, e.g. pg1-path/pg1-host, pg2-path/pg2-host can be primary or standby.

Installation

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

NOTE:

The pgBackRest version installed on the repository host must exactly match the version installed on the PostgreSQL host.

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

NOTE:

When pgBackRest is installed from a package, a logrotate configuration such as /etc/logrotate.d/pgbackrest may be provided that rotates the logs as a specific user via the su directive (e.g. su postgres postgres). Since the files in /var/log/pgbackrest are owned by the user that runs pgBackRest (here pgbackrest), the su directive must be updated to match this user or logrotate will fail with a permission error.

repository Create pgbackrest user

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

Installing pgBackRest from a package is preferable to building from source. When installing from a package the rest of the instructions in this section are generally not required, but it is possible that a package will skip creating one of the directories or apply incorrect permissions. In that case it may be necessary to manually create directories or update permissions.

Debian/Ubuntu packages for pgBackRest are available at apt.postgresql.org.

If packages are not provided for your distribution/version you can build from source and then install manually as shown here.

repository Install dependencies

```
sudo apt-get install postgresql-client libxml2 libssh2-1
```

repository Copy pgBackRest binary from build host

```
sudo scp build:/build/pgbackrest/src/pgbackrest /usr/bin
```

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

pgBackRest requires log and configuration directories and a configuration file.

repository Create pgBackRest configuration file and directories

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

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

```
sudo mkdir -p /etc/pgbackrest
```

```
sudo mkdir -p /etc/pgbackrest/conf.d
```

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

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

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

repository Create the pgBackRest repository

```
sudo mkdir -p /var/lib/pgbackrest
```

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

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

Setup Passwordless SSH

pgBackRest can use passwordless SSH to enable communication between the hosts. It is also possible to use TLS, see Setup TLS.

repository Create repository host key pair

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

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

pg-primary Create pg-primary host key pair

```
sudo -u postgres mkdir -m 750 -p /var/lib/postgresql/.ssh
```

```
sudo -u postgres ssh-keygen -f /var/lib/postgresql/.ssh/id_rsa \
       -t rsa -b 4096 -N ""
```

Exchange keys between repository and pg-primary.

repository Copy pg-primary public key to repository

```
(echo -n 'no-agent-forwarding,no-X11-forwarding,no-port-forwarding,' && \
       echo -n 'command="/usr/bin/pgbackrest ${SSH_ORIGINAL_COMMAND#* }" ' && \
       sudo ssh root@pg-primary cat /var/lib/postgresql/.ssh/id_rsa.pub) | \
       sudo -u pgbackrest tee -a /home/pgbackrest/.ssh/authorized_keys
```

pg-primary Copy repository public key to pg-primary

```
(echo -n 'no-agent-forwarding,no-X11-forwarding,no-port-forwarding,' && \
       echo -n 'command="/usr/bin/pgbackrest ${SSH_ORIGINAL_COMMAND#* }" ' && \
       sudo ssh root@repository cat /home/pgbackrest/.ssh/id_rsa.pub) | \
       sudo -u postgres tee -a /var/lib/postgresql/.ssh/authorized_keys
```

Test that connections can be made from repository to pg-primary and vice versa.

repository Test connection from repository to pg-primary

```
sudo -u pgbackrest ssh postgres@pg-primary
```

pg-primary Test connection from pg-primary to repository

```
sudo -u postgres ssh pgbackrest@repository
```

NOTE:

ssh has been configured to only allow pgBackRest to be run via passwordless ssh. This enhances security in the event that one of the service accounts is hijacked.

Configuration

The repository host must be configured with the pg-primary host/user and database path. The primary will be configured as pg1 to allow a standby to be added later.

repository:/etc/pgbackrest/pgbackrest.conf Configure pg1-host/pg1-host-user and pg1-path

```
[demo]
pg1-host=pg-primary
pg1-path=/var/lib/postgresql/17/demo

[global]
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y
```

The database host must be configured with the repository host/user. The default for the repo1-host-user option is pgbackrest. If the postgres user does restores on the repository 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 pgbackrest user.

pg-primary:/etc/pgbackrest/pgbackrest.conf Configure repo1-host/repo1-host-user

```
[demo]
pg1-path=/var/lib/postgresql/17/demo

[global]
log-level-file=detail
repo1-host=repository
```

PostgreSQL configuration may be found in the Configure Archiving section.

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

Create and Check Stanza

Create the stanza in the new repository.

repository Create the stanza

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

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

pg-primary Check the configuration

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

repository Check the configuration

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

Perform a Backup

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

repository Backup the demo cluster

```
sudo -u pgbackrest 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 repository 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.

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

```
sudo pg_ctlcluster 17 demo stop
```

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

```
sudo pg_ctlcluster 17 demo start
```

Parallel Backup / Restore

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

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

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

repository Perform a backup with single process

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

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

```
[demo]
pg1-host=pg-primary
pg1-path=/var/lib/postgresql/17/demo

[global]
```

process-max=3
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y

repository Perform a backup with multiple processes

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

repository Get backup info for the demo cluster

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

```
stanza: demo
    status: ok
    cipher: none

    db (current)
        wal archive min/max (17): 000000070000000000000023/000000070000000000000025

        full backup: 20260817-045249F

            timestamp start/stop: 2026-08-17 04:52:49+00 / 2026-08-17 04:52:53+00

            wal start/stop: 000000070000000000000023 / 000000070000000000000023
            database size: 29.2MB, database backup size: 29.2MB
            repo1: backup set size: 3.8MB, backup size: 3.8MB

        full backup: 20260817-045255F

            timestamp start/stop: 2026-08-17 04:52:55+00 / 2026-08-17 04:52:58+00

            wal start/stop: 000000070000000000000024 / 000000070000000000000025
            database size: 29.2MB, database backup size: 29.2MB
            repo1: backup set size: 3.8MB, backup size: 3.8MB
```

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

Starting and Stopping

If a standby is promoted for testing, or a test cluster is restored from a production backup, then it is a good idea to prevent those clusters from writing to pgBackRest repositories. This can be accomplished with the stop command.

The commands that write and are blocked by stop are: archive-push, backup, expire, stanza-create, and stanza-upgrade. Note that stanza-delete is an exception to this rule (see Delete a Stanza for more details).

pg-primary Stop pgBackRest write commands

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

New pgBackRest write commands will no longer run.

repository Attempt a backup

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

```
P00   WARN: unable to check pg1: [StopError] raised from remote-0 ssh protocol on 'pg-prima
```

```
P00  ERROR: [056]: unable to find primary cluster - cannot proceed
            HINT: are all available clusters in recovery?
```

Specify the --force option to terminate any pgBackRest write commands that are currently running. This includes asynchronous archive-get (though it will run again if PostgreSQL requires it). If pgBackRest is already stopped then stopping again will generate a warning.

pg-primary Stop the pgBackRest services again

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

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

Start pgBackRest write commands again with the start command. Write commands that were in progress before the stop will not automatically start again, but they are now allowed to start.

pg-primary Start pgBackRest write commands

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

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

pg-primary Stop pgBackRest write commands for the demo stanza

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

New pgBackRest write commands for the specified stanza will no longer run.

repository Attempt a backup

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

```
P00   WARN: unable to check pg1: [StopError] raised from remote-0 ssh protocol on 'pg-prima
```

```
P00  ERROR: [056]: unable to find primary cluster - cannot proceed
            HINT: are all available clusters in recovery?
```

The stanza must also be specified when starting pgBackRest write commands for a single stanza.

pg-primary Start pgBackRest write commands for the demo stanza

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

Replication

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

Installation

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

Installing pgBackRest from a package is preferable to building from source. When installing from a package the rest of the instructions in this section are generally not required, but it is possible that a package will skip creating one of the directories or apply incorrect permissions. In that case it may be necessary to manually create directories or update permissions.

Debian/Ubuntu packages for pgBackRest are available at apt.postgresql.org.

If packages are not provided for your distribution/version you can build from source and then install manually as shown here.

pg-standby Install dependencies

```
sudo apt-get install postgresql-client libxml2 libssh2-1
```

pg-standby Copy pgBackRest binary from build host

```
sudo scp build:/build/pgbackrest/src/pgbackrest /usr/bin
```

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

pgBackRest requires log and configuration directories and a configuration file.

pg-standby Create pgBackRest configuration file and directories

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

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

```
sudo mkdir -p /etc/pgbackrest
```

```
sudo mkdir -p /etc/pgbackrest/conf.d
```

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

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

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

Setup Passwordless SSH

pgBackRest can use passwordless SSH to enable communication between the hosts. It is also possible to use TLS, see Setup TLS.

pg-standby Create pg-standby host key pair

```
sudo -u postgres mkdir -m 750 -p /var/lib/postgresql/.ssh
```

```
sudo -u postgres ssh-keygen -f /var/lib/postgresql/.ssh/id_rsa \
       -t rsa -b 4096 -N ""
```

Exchange keys between repository and pg-standby.

repository Copy pg-standby public key to repository

```
(echo -n 'no-agent-forwarding,no-X11-forwarding,no-port-forwarding,' && \
       echo -n 'command="/usr/bin/pgbackrest ${SSH_ORIGINAL_COMMAND#* }" ' && \
       sudo ssh root@pg-standby cat /var/lib/postgresql/.ssh/id_rsa.pub) | \
       sudo -u pgbackrest tee -a /home/pgbackrest/.ssh/authorized_keys
```

pg-standby Copy repository public key to pg-standby

```
(echo -n 'no-agent-forwarding,no-X11-forwarding,no-port-forwarding,' && \
       echo -n 'command="/usr/bin/pgbackrest ${SSH_ORIGINAL_COMMAND#* }" ' && \
       sudo ssh root@repository cat /home/pgbackrest/.ssh/id_rsa.pub) | \
       sudo -u postgres tee -a /var/lib/postgresql/.ssh/authorized_keys
```

Test that connections can be made from repository to pg-standby and vice versa.

repository Test connection from repository to pg-standby

```
sudo -u pgbackrest ssh postgres@pg-standby
```

pg-standby Test connection from pg-standby to repository

```
sudo -u postgres ssh pgbackrest@repository
```

Hot Standby

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

pgBackRest configuration is very similar to pg-primary except that the standby recovery type will be used to keep the cluster in recovery mode when the end of the WAL stream has been reached.

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

```
[demo]
pg1-path=/var/lib/postgresql/17/demo

[global]
log-level-file=detail
repo1-host=repository
```

The demo cluster must be created (even though it will be overwritten on restore) in order to create the PostgreSQL configuration files.

pg-standby Create demo cluster

```
sudo pg_createcluster 17 demo
```

Now the standby can be created with the restore command.

IMPORTANT:

If the cluster is intended to be promoted without becoming the new primary (e.g. for reporting or testing), use --archive-mode=off or set archive_mode=off in postgresql.conf to disable archiving. If archiving is not disabled then the repository may be polluted with WAL that can make restores more difficult.

pg-standby Restore the demo standby cluster

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

```
sudo -u postgres cat /var/lib/postgresql/17/demo/postgresql.auto.conf
```

```
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:50:43
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:51:08
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:51:28
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
# Removed by pgBackRest restore on 2026-08-17 04:52:17 # recovery_target_time = '2026-08-17
# Removed by pgBackRest restore on 2026-08-17 04:52:17 # recovery_target_action = 'promote'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:52:17
restore_command = 'pgbackrest --repo=3 --repo-target-time="2026-08-17 04:52:04+00" --stanza=

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:52:43
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:53:15
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
```

The configuration is in case the standby is promoted to a primary.

pg-standby:/etc/postgresql/17/demo/postgresql.conf Configure PostgreSQL

archive_command = 'pgbackrest --stanza=demo archive-push %p'
archive_mode = on

pg-standby Start PostgreSQL

```
sudo pg_ctlcluster 17 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.

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

```
sudo -u postgres cat /var/log/postgresql/postgresql-17-demo.log

       [filtered 6 lines of output]
LOG:  restored log file "00000007.history" from archive
LOG:  restored log file "000000070000000000000024" from archive

LOG:  entering standby mode

LOG:  redo starts at 0/24000028
LOG:  restored log file "000000070000000000000025" from archive
       [filtered 3 lines of output]
```

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

pg-primary Create a new table on the primary

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

       [filtered 4 lines of output]
    message
----------------

 Important Data

(1 row)
```

And then query the same table on pg-standby.

pg-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 pg-primary.

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

pg-primary Call pg_switch_wal()

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

```
 pg_switch_wal |       current_timestamp
---------------+-------------------------------
 0/260195C8    | 2026-08-17 04:53:22.649899+00
(1 row)
```

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

pg-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     |       current_timestamp
----------------+-------------------------------

 Important Data | 2026-08-17 04:53:24.247724+00

(1 row)
```

Check the standby configuration for access to the repository.

pg-standby Check the configuration

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

```
P00   INFO: check command begin 2.59.1: --exec-id=505-273c787f --log-level-console=info --lo
P00   INFO: check repo1 (standby)

P00   INFO: switch wal not performed because this is a standby

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

Streaming Replication

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

Streaming replication requires a user with the replication privilege.

pg-primary Create replication user

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

```
CREATE ROLE
```

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

pg-primary Create pg_hba.conf entry for replication user

```
sudo -u postgres sh -c 'echo \
       "host    replication     replicator      172.17.0.8/32           md5" \
```

```
       >> /etc/postgresql/17/demo/pg_hba.conf'

sudo pg_ctlcluster 17 demo reload
```

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

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

```
[demo]
pg1-path=/var/lib/postgresql/17/demo
recovery-option=primary_conninfo=host=172.17.0.6 port=5432 user=replicator

[global]
log-level-file=detail
repo1-host=repository
```

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

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

```
sudo -u postgres sh -c 'echo \
       "172.17.0.6:*:replication:replicator:jw8s0F4" \
       >> /var/lib/postgresql/.pgpass'

sudo -u postgres chmod 600 /var/lib/postgresql/.pgpass
```

Now the standby can be created with the restore command.

pg-standby Stop PostgreSQL and restore the demo standby cluster

```
sudo pg_ctlcluster 17 demo stop

sudo -u postgres pgbackrest --stanza=demo --delta --type=standby restore

sudo -u postgres cat /var/lib/postgresql/17/demo/postgresql.auto.conf

# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:50:43
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:51:08
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:51:28
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
# Removed by pgBackRest restore on 2026-08-17 04:52:17 # recovery_target_time = '2026-08-17
# Removed by pgBackRest restore on 2026-08-17 04:52:17 # recovery_target_action = 'promote'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:52:17
```

```
restore_command = 'pgbackrest --repo=3 --repo-target-time="2026-08-17 04:52:04+00" --stanza=

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:52:43
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2026-08-17 04:53:27
primary_conninfo = 'host=172.17.0.6 port=5432 user=replicator'
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
```

NOTE:

The primary_conninfo setting has been written into the postgresql.auto.conf file because it was configured as a recovery-option in pgbackrest.conf. The --type=preserve option can be used with the restore to leave the existing postgresql.auto.conf file in place if that behavior is preferred.

pg-standby Start PostgreSQL

```
sudo pg_ctlcluster 17 demo start
```

The PostgreSQL log will confirm that streaming replication has started.

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

```
sudo -u postgres cat /var/log/postgresql/postgresql-17-demo.log

       [filtered 13 lines of output]
LOG:  consistent recovery state reached at 0/25000088
LOG:  database system is ready to accept read-only connections

LOG:  started streaming WAL from primary at 0/27000000 on timeline 7
```

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

pg-primary Create a new table on the primary

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

       [filtered 4 lines of output]
    message     |       current_timestamp
----------------+-------------------------------

 Important Data | 2026-08-17 04:53:33.817993+00

(1 row)
```

pg-standby Query table on the standby

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

    message     |       current_timestamp
----------------+-------------------------------

 Important Data | 2026-08-17 04:53:34.003849+00

(1 row)
```

Multiple Stanzas

pgBackRest supports multiple stanzas. The most common usage is sharing a repository host among multiple stanzas.

Installation

A new host named pg-alt is created to run the new primary.

Installing pgBackRest from a package is preferable to building from source. When installing from a package the rest of the instructions in this section are generally not required, but it is possible that a package will skip creating one of the directories or apply incorrect permissions. In that case it may be necessary to manually create directories or update permissions.

Debian/Ubuntu packages for pgBackRest are available at apt.postgresql.org.

If packages are not provided for your distribution/version you can build from source and then install manually as shown here.

pg-alt Install dependencies

```
sudo apt-get install postgresql-client libxml2 libssh2-1
```

pg-alt Copy pgBackRest binary from build host

```
sudo scp build:/build/pgbackrest/src/pgbackrest /usr/bin

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

pgBackRest requires log and configuration directories and a configuration file.

pg-alt Create pgBackRest configuration file and directories

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

sudo chown postgres:postgres /var/log/pgbackrest

sudo mkdir -p /etc/pgbackrest

sudo mkdir -p /etc/pgbackrest/conf.d

sudo touch /etc/pgbackrest/pgbackrest.conf

sudo chmod 640 /etc/pgbackrest/pgbackrest.conf

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

Setup Passwordless SSH

pgBackRest can use passwordless SSH to enable communication between the hosts. It is also possible to use TLS, see Setup TLS.

pg-alt Create pg-alt host key pair

```
sudo -u postgres mkdir -m 750 -p /var/lib/postgresql/.ssh
```

```
sudo -u postgres ssh-keygen -f /var/lib/postgresql/.ssh/id_rsa \
       -t rsa -b 4096 -N ""
```

Exchange keys between repository and pg-alt.

repository Copy pg-alt public key to repository

```
(echo -n 'no-agent-forwarding,no-X11-forwarding,no-port-forwarding,' && \
       echo -n 'command="/usr/bin/pgbackrest ${SSH_ORIGINAL_COMMAND#* }" ' && \
       sudo ssh root@pg-alt cat /var/lib/postgresql/.ssh/id_rsa.pub) | \
       sudo -u pgbackrest tee -a /home/pgbackrest/.ssh/authorized_keys
```

pg-alt Copy repository public key to pg-alt

```
(echo -n 'no-agent-forwarding,no-X11-forwarding,no-port-forwarding,' && \
       echo -n 'command="/usr/bin/pgbackrest ${SSH_ORIGINAL_COMMAND#* }" ' && \
       sudo ssh root@repository cat /home/pgbackrest/.ssh/id_rsa.pub) | \
       sudo -u postgres tee -a /var/lib/postgresql/.ssh/authorized_keys
```

Test that connections can be made from repository to pg-alt and vice versa.

repository Test connection from repository to pg-alt

```
sudo -u pgbackrest ssh postgres@pg-alt
```

pg-alt Test connection from pg-alt to repository

```
sudo -u postgres ssh pgbackrest@repository
```

Configuration

pgBackRest configuration is nearly identical to pg-primary except that the demo-alt stanza will be used so backups and archive will be stored in a separate location.

pg-alt:/etc/pgbackrest/pgbackrest.conf Configure pgBackRest on the new primary

```
[demo-alt]
pg1-path=/var/lib/postgresql/17/demo

[global]
log-level-file=detail
repo1-host=repository
```

repository:/etc/pgbackrest/pgbackrest.conf Configure pg1-host/pg1-host-user and pg1-path

[demo]
pg1-host=pg-primary
pg1-path=/var/lib/postgresql/17/demo

[demo-alt]
pg1-host=pg-alt
pg1-path=/var/lib/postgresql/17/demo

[global]
process-max=3
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y

Setup Demo Cluster

pg-alt Create the demo cluster

```
sudo -u postgres /usr/lib/postgresql/17/bin/initdb \
       -D /var/lib/postgresql/17/demo -k -A peer
```

```
sudo pg_createcluster 17 demo
```

```
Configuring already existing cluster (configuration: /etc/postgresql/17/demo, data: /var/lib
Ver Cluster Port Status Owner    Data directory              Log file
17  demo    5432 down   postgres /var/lib/postgresql/17/demo /var/log/postgresql/postgresql-
```

pg-alt:/etc/postgresql/17/demo/postgresql.conf Configure PostgreSQL settings

archive_command = 'pgbackrest --stanza=demo-alt archive-push %p'
archive_mode = on

pg-alt Start the demo cluster

```
sudo pg_ctlcluster 17 demo restart
```

Create the Stanza and Check Configuration

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

pg-alt Create the stanza and check the configuration

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

```
P00   INFO: stanza-create command begin 2.59.1: --exec-id=372-b225fca7 --log-level-console=i
P00   INFO: stanza-create for stanza 'demo-alt' on repo1
```

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

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

P00   INFO: check command begin 2.59.1: --exec-id=382-ccef16c6 --log-level-console=info --lo

P00   INFO: check stanza 'demo-alt'

P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)

P00   INFO: WAL segment 000000010000000000000001 successfully archived to '/var/lib/pgbackre

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

If the check command is run from the repository host then all stanzas will be checked.

repository Check the configuration for all stanzas

```
sudo -u pgbackrest pgbackrest --log-level-console=info check

P00   INFO: check command begin 2.59.1: --exec-id=1249-f7860275 --log-level-console=info --r

P00   INFO: check stanza 'demo'

P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)

P00   INFO: WAL segment 000000070000000000000027 successfully archived to '/var/lib/pgbackre
P00   INFO: check stanza 'demo-alt'

P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)

P00   INFO: WAL segment 000000010000000000000002 successfully archived to '/var/lib/pgbackre

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

Asynchronous Archiving

Asynchronous archiving is enabled with the archive-async option. This option enables asynchronous operation for both the archive-push and archive-get commands.

A spool path is required. The commands will store transient data here but each command works quite a bit differently so spool path usage is described in detail in each section.

pg-primary Create the spool directory

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

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

pg-standby Create the spool directory

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

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

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

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

```
[demo]
pg1-path=/var/lib/postgresql/17/demo

[global]
archive-async=y
log-level-file=detail
repo1-host=repository
spool-path=/var/spool/pgbackrest

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

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

pg-standby:/etc/pgbackrest/pgbackrest.conf Configure the spool path and asynchronous archiving

```
[demo]
pg1-path=/var/lib/postgresql/17/demo
recovery-option=primary_conninfo=host=172.17.0.6 port=5432 user=replicator

[global]
archive-async=y
log-level-file=detail
repo1-host=repository
spool-path=/var/spool/pgbackrest

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

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

NOTE:

process-max is configured using command sections so that the option is not used by backup and restore. This also allows different values for archive-push and archive-get.

For demonstration purposes streaming replication will be broken to force PostgreSQL to get WAL using the restore_command.

pg-primary Break streaming replication by changing the replication password

```
sudo -u postgres psql -c "alter user replicator password 'bogus'"
```

```
ALTER ROLE
```

pg-standby Restart standby to break connection

```
sudo pg_ctlcluster 17 demo restart
```

Archive Push

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

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

IMPORTANT:

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

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

pg-primary Test parallel asynchronous archiving

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

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

```
P00   INFO: check command begin 2.59.1: --exec-id=2530-9a7af218 --log-level-console=info --l
P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)

P00   INFO: WAL segment 000000070000000000000002D successfully archived to '/var/lib/pgbackre

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

Now the log file will contain parallel, asynchronous activity.

pg-primary Check results in the log

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

-------------------PROCESS START-------------------
P00   INFO: archive-push:async command begin 2.59.1: [/var/lib/postgresql/17/demo/pg_wal] --

P00   INFO: push 1 WAL file(s) to archive: 000000070000000000000028
P01 DETAIL: pushed WAL file '000000070000000000000028' to the archive

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


-------------------PROCESS START-------------------
P00   INFO: archive-push:async command begin 2.59.1: [/var/lib/postgresql/17/demo/pg_wal] --

P00   INFO: push 5 WAL file(s) to archive: 000000070000000000000029...00000007000000000000000
P01 DETAIL: pushed WAL file '000000070000000000000029' to the archive
P02 DETAIL: pushed WAL file '00000007000000000000002A' to the archive
P01 DETAIL: pushed WAL file '00000007000000000000002B' to the archive
P02 DETAIL: pushed WAL file '00000007000000000000002C' to the archive
P01 DETAIL: pushed WAL file '00000007000000000000002D' to the archive

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

Archive Get

The asynchronous archive-get command maintains a local queue of WAL to improve throughput. If a WAL segment is not found in the queue it is fetched from the repository along with enough consecutive WAL to fill the queue. The maximum size of the queue is defined by archive-get-queue-max. Whenever the queue is less than half full more WAL will be fetched to fill it.

Asynchronous operation is most useful in environments that generate a lot of WAL or have a high latency connection to the repository storage (i.e., S3 or other object stores). In the case of a high latency connection it may be a good idea to increase process-max.

The [stanza]-archive-get-async.log file can be used to monitor the activity of the asynchronous process.

pg-standby Check results in the log

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

```
-------------------PROCESS START-------------------
P00   INFO: archive-get:async command begin 2.59.1: [000000070000000000000024, 000000070000
P00   INFO: get 8 WAL file(s) from archive: 000000070000000000000024...000000070000000000000

P02 DETAIL: found 000000070000000000000025 in the repo1: 17-1 archive
P01 DETAIL: found 000000070000000000000024 in the repo1: 17-1 archive
P02 DETAIL: found 000000070000000000000026 in the repo1: 17-1 archive
P01 DETAIL: found 000000070000000000000027 in the repo1: 17-1 archive

P00 DETAIL: unable to find 000000070000000000000028 in the archive
P00   INFO: archive-get:async command end: completed successfully
       [filtered 14 lines of output]
P00   INFO: archive-get:async command begin 2.59.1: [000000070000000000000028, 000000070000
P00   INFO: get 8 WAL file(s) from archive: 000000070000000000000028...000000070000000000000

P01 DETAIL: found 000000070000000000000028 in the repo1: 17-1 archive
P02 DETAIL: found 000000070000000000000029 in the repo1: 17-1 archive
P01 DETAIL: found 00000007000000000000002A in the repo1: 17-1 archive
P02 DETAIL: found 00000007000000000000002B in the repo1: 17-1 archive
P02 DETAIL: found 00000007000000000000002D in the repo1: 17-1 archive
P01 DETAIL: found 00000007000000000000002C in the repo1: 17-1 archive

P00 DETAIL: unable to find 00000007000000000000002E in the archive
P00   INFO: archive-get:async command end: completed successfully
       [filtered 9 lines of output]
```

pg-primary Fix streaming replication by changing the replication password

```
sudo -u postgres psql -c "alter user replicator password 'jw8s0F4'"

ALTER ROLE
```

Backup from a Standby

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

repository:/etc/pgbackrest/pgbackrest.conf Configure pg2-host/pg2-host-user and pg2-path

```
[demo]
pg1-host=pg-primary
pg1-path=/var/lib/postgresql/17/demo
pg2-host=pg-standby
pg2-path=/var/lib/postgresql/17/demo


[demo-alt]
pg1-host=pg-alt
pg1-path=/var/lib/postgresql/17/demo
```

[global]
backup-standby=y
process-max=3
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y

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

repository Backup the demo cluster from pg2

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

       [filtered 2 lines of output]
P00   INFO: execute backup start: backup begins after the requested immediate checkpoint com
P00   INFO: backup start archive = 000000070000000000000002F, lsn = 0/2F000028

P00   INFO: wait for replay on the standby to reach 0/2F000028
P00   INFO: replay on the standby reached 0/2F000028

P00   INFO: check archive for prior segment 000000070000000000000002E
P04 DETAIL: backup file pg-standby:/var/lib/postgresql/17/demo/base/5/1247 (120KB, 7.94%) ch

P01 DETAIL: backup file pg-primary:/var/lib/postgresql/17/demo/global/pg_control (8KB, 8.47%

P01 DETAIL: match file from prior backup pg-primary:/var/lib/postgresql/17/demo/pg_logical/m
P02 DETAIL: backup file pg-standby:/var/lib/postgresql/17/demo/base/5/1249 (448KB, 38.10%) c
       [filtered 1277 lines of output]
```

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

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

Upgrading PostgreSQL

Immediately after upgrading PostgreSQL to a newer major version, the pg-path for all pgBackRest configurations must be set to the new database location and the stanza-upgrade command run. If there is more than one repository configured on the host, the stanza will be upgraded on each. If the database is offline use the --no-online option.

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

pg-primary Stop old cluster

```
sudo pg_ctlcluster 17 demo stop
```

Stop the old cluster on the standby since it will be restored from the newly upgraded cluster.

pg-standby Stop old cluster

```
sudo pg_ctlcluster 17 demo stop
```

Create the new cluster and perform upgrade.

pg-primary Create new cluster and perform the upgrade

```
sudo -u postgres /usr/lib/postgresql/18/bin/initdb \
       -D /var/lib/postgresql/18/demo -k -A peer
```

```
sudo pg_createcluster 18 demo
```

```
sudo -u postgres sh -c 'cd /var/lib/postgresql && \
       /usr/lib/postgresql/18/bin/pg_upgrade \
       --old-bindir=/usr/lib/postgresql/17/bin \
       --new-bindir=/usr/lib/postgresql/18/bin \
       --old-datadir=/var/lib/postgresql/17/demo \
       --new-datadir=/var/lib/postgresql/18/demo \
       --old-options=" -c config_file=/etc/postgresql/17/demo/postgresql.conf" \
       --new-options=" -c config_file=/etc/postgresql/18/demo/postgresql.conf"'

       [filtered 44 lines of output]
Checking for extension updates                              ok

Upgrade Complete

----------------
Some statistics are not transferred by pg_upgrade.
       [filtered 4 lines of output]
```

Configure the new cluster settings and port.

pg-primary:/etc/postgresql/18/demo/postgresql.conf Configure PostgreSQL

archive_command = 'pgbackrest --stanza=demo archive-push %p'
archive_mode = on

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

pg-primary:/etc/pgbackrest/pgbackrest.conf Upgrade the pg1-path

```
[demo]
pg1-path=/var/lib/postgresql/18/demo

[global]
archive-async=y
log-level-file=detail
repo1-host=repository
spool-path=/var/spool/pgbackrest

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

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

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

```
[demo]
pg1-path=/var/lib/postgresql/18/demo
recovery-option=primary_conninfo=host=172.17.0.6 port=5432 user=replicator

[global]
archive-async=y
log-level-file=detail
repo1-host=repository
spool-path=/var/spool/pgbackrest

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

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

repository:/etc/pgbackrest/pgbackrest.conf Upgrade pg1-path and pg2-path, disable backup from standby

```
[demo]
pg1-host=pg-primary
pg1-path=/var/lib/postgresql/18/demo
pg2-host=pg-standby
pg2-path=/var/lib/postgresql/18/demo

[demo-alt]
pg1-host=pg-alt
pg1-path=/var/lib/postgresql/17/demo

[global]
backup-standby=n
```

```
process-max=3
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y
```

pg-primary Copy hba configuration

```
sudo cp /etc/postgresql/17/demo/pg_hba.conf \
       /etc/postgresql/18/demo/pg_hba.conf
```

Before starting the new cluster, the stanza-upgrade command must be run.

pg-primary Upgrade the stanza

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

```
P00   INFO: stanza-upgrade command begin 2.59.1: --exec-id=2952-4ae81a5e --log-level-console
P00   INFO: stanza-upgrade for stanza 'demo' on repo1
```

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

Start the new cluster and confirm it is successfully installed.

pg-primary Start new cluster

```
sudo pg_ctlcluster 18 demo start
```

Test configuration using the check command.

pg-primary Check configuration

```
sudo pg_lsclusters
```

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

Remove the old cluster.

pg-primary Remove old cluster

```
sudo pg_dropcluster 17 demo
```

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

pg-standby Remove old cluster and create the new cluster

```
sudo pg_dropcluster 17 demo
```

```
sudo pg_createcluster 18 demo
```

Run the check on the repository host. The warning regarding the standby being down is expected since the standby cluster is down. Running this command demonstrates that the repository server is aware of the standby and is configured properly for the primary server.

repository Check configuration

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

```
P00   WARN: unable to check pg2: [DbConnectError] raised from remote-0 ssh protocol on 'pg-s
                Is the server running locally and accepting connections on that socket?
```

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

repository Run a full backup

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

pg-standby Restore the demo standby cluster

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

pg-standby Start PostgreSQL and check the pgBackRest configuration

```
sudo pg_ctlcluster 18 demo start
```

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

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

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

```
[demo]
pg1-host=pg-primary
pg1-path=/var/lib/postgresql/18/demo
pg2-host=pg-standby
pg2-path=/var/lib/postgresql/18/demo

[demo-alt]
pg1-host=pg-alt
pg1-path=/var/lib/postgresql/17/demo

[global]
backup-standby=y
process-max=3
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y
```

Configuration

News

FAQ

Metrics

Table of Contents

Introduction

Annotate Command (annotate)

Archive Get Command (archive-get)

Archive Push Command (archive-push)

Backup Command (backup)

Check Command (check)

Expire Command (expire)

Help Command (help)

Info Command (info)

Repository Get Command (repo-get)

Repository List Command (repo-ls)

Restore Command (restore)

Server Command (server)

Server Ping Command (server-ping)

Stanza Create Command (stanza-create)

Stanza Delete Command (stanza-delete)

Stanza Upgrade Command (stanza-upgrade)

Start Command (start)

Stop Command (stop)

Verify Command (verify)

Version Command (version)

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 be configured in pgbackrest.conf.

Non-boolean options configured in pgbackrest.conf can be reset to default on the command-line by using the reset- prefix. This feature may be used to restore a backup directly on a repository host. Normally, pgBackRest will error because it can see that the database host is remote and restores cannot be done remotely. By adding --reset-pg1-host on the command-line, pgBackRest will ignore the remote database host and restore locally. It may be necessary to pass a new --pg1-path to force the restore to happen in a specific path, i.e. not the path used on the database host.

The no- prefix may be used to set a boolean option to false on the command-line.

Any option may be set in an environment variable using the PGBACKREST_ prefix and the option name in all caps replacing - with _, e.g. pg1-path becomes PGBACKREST_PG1_PATH and stanza becomes PGBACKREST_STANZA. Boolean options are represented as they would be in a configuration file, e.g. PGBACKREST_COMPRESS="n", and reset-* variants are not allowed. Options that can be specified multiple times on the command-line or in a config file can be represented by separating the values with colons, e.g. PGBACKREST_DB_INCLUDE="db1:db2".

Command-line options override environment options which override config file options.

See Configuration Introduction for information on option types

Annotate Command (annotate)

Annotations included with the backup command can be added, modified, or removed afterwards using the annotate command.

Command Options

Backup Annotation Option (--annotation)

Annotate backup with user-defined key/value pairs.

Users can attach informative key/value pairs to the backup. This option may be used multiple times to attach multiple annotations.

Annotations are output by the info command text output when a backup is specified with --set and always appear in the JSON output.

```
example: --annotation=source="Sunday backup for website database"
```

Set Option (--set)

Backup set to annotate.

The backup set to annotate.

```
example: --set=20150131-153358F_20150131-153401I
```

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

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

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

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: --compress-level-network=1
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

Lock Path Option (--lock-path)

Path where lock files are stored.

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

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

Neutral Umask Option (--neutral-umask)

Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

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

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Stanza Option (--stanza)

Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one PostgreSQL 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
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

`example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL`

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

`example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256`

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

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

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

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

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

- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

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

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

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

The following log levels are supported:

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

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

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

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

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: --log-subprocess
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

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

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

Repository Options

Set Repository Option (--repo)

Set repository.

Set the repository for a command to operate on.

For example, this option may be used to perform a restore from a specific repository, rather than letting pgBackRest choose.

```
allowed: [1, 256]
example: --repo=1
```

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

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

```
example: --repo1-azure-container=pg-backup
```

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: --repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: --repo1-azure-uri-style=path
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

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

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: --repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

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

```
example: --repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: --repo1-gcs-endpoint=localhost
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.
- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: --repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: --repo1-gcs-user-project=my-project
```

Repository Host Option (--repo-host)

Repository host when operating remotely.

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

```
example: --repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: --repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: --repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

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

```
default: pgbackrest
example: --repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

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

It may 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: --repo1-path=/backup/db/backrest
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

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

```
example: --repo1-s3-bucket=pg-backup
```

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

`example: --repo1-s3-endpoint=s3.amazonaws.com`

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys
- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

`default: shared`
`example: --repo1-s3-key-type=auto`

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

`example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7`

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

`example: --repo1-s3-process-cmd=/usr/local/bin/get-credentials --repo1-s3-process-cmd=--role`

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

`example: --repo1-s3-region=us-east-1`

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

```
default: n
example: --no-repo1-s3-requester-pays
```

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

```
example: --repo1-s3-role=authrole
```

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

```
default: s3
example: --repo1-s3-service=s3-outposts
```

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

```
default: sts.amazonaws.com
example: --repo1-s3-sts-host=sts.us-east-1.amazonaws.com
```

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

```
default: host
example: --repo1-s3-uri-style=path
```

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}' ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

```
example: --repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf
```

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.
- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: --repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: --repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: --repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

`example: --repo1-sftp-host-user=pg-backup`

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

`example: --repo1-sftp-known-host=/home/postgres/.ssh/known_hosts`

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

`example: --repo1-sftp-private-key-file=~/.ssh/id_ed25519`

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

`example: --repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub`

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt`

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-path=/etc/pki/tls/certs`

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

```
example: --repo1-storage-host=127.0.0.1
```

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: --repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: --repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies

by storage type, so when searching min/max values use "part size" for AWS S3, "chunk size" for GCS, and "block size" for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]

example: --repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo1-storage-verify-tls
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs
- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: --repo1-type=cifs
```

Archive Get Command (archive-get)

This command is used by PostgreSQL to restore a backup, perform PITR, or as an alternative to streaming for keeping a replica up to date. WAL segments are required for PostgreSQL recovery or to maintain a replica.

When multiple repositories are configured, WAL will be fetched from the repositories in priority order (e.g. repo1, repo2, etc.). In general it is better if faster/cheaper storage has higher priority. If a repository is specified with the --repo option then only that repository will be searched.

The archive-get command is configured and generated by pgBackRest during a restore for use by PostgreSQL. See Point-in-Time Recovery for an example.

Command Options

Asynchronous Archiving Option (--archive-async)

Push/get WAL segments asynchronously.

Enables asynchronous operation for the archive-push and archive-get commands.

Asynchronous operation is more efficient because it can reuse connections and take advantage of parallelism. See the spool-path, archive-get-queue-max, and archive-push-queue-max options for more information.

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

Maximum Archive Get Queue Size Option (--archive-get-queue-max)

Maximum size of the pgBackRest archive-get queue.

Specifies the maximum size of the archive-get queue when archive-async is enabled. The queue is stored in the spool-path and is used to speed providing WAL to PostgreSQL.

```
default: 128MiB
allowed: [0B, 4PiB]
example: --archive-get-queue-max=1GiB
```

Retry Missing WAL Segment Option (--archive-missing-retry)

Retry missing WAL segment

Retry a WAL segment that was previously reported as missing by the archive-get command when in asynchronous mode. This prevents notifications in the spool path from a prior restore from being used and possibly causing a recovery failure if consistency has not been reached.

Disabling this option allows PostgreSQL to more reliably recognize when the end of the WAL in the archive has been reached, which permits it to switch over to streaming from the primary. With retries enabled, a steady stream of

WAL being archived will cause PostgreSQL to continue getting WAL from the archive rather than switch to streaming.

When disabling this option it is important to ensure that the spool path for the stanza is empty. The restore command does this automatically if the spool path is configured at restore time. Otherwise, it is up to the user to ensure the spool path is empty.

```
default: y
example: --no-archive-missing-retry
```

Archive Timeout Option (--archive-timeout)

Archive timeout.

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

```
default: 1m
allowed: [100ms, 1d]
example: --archive-timeout=30
```

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

pgBackRest Command Option (--cmd)

pgBackRest command.

pgBackRest may generate a command string, e.g. when the restore command generates the restore_command setting. The command used to run the pgBackRest process will be used in this case unless the cmd option is provided.

CAUTION:

Wrapping the pgBackRest command may cause unpredictable behavior and is not recommended.

```
default: [path of executed pgbackrest binary]
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

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

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

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: --compress-level-network=1
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

Lock Path Option (--lock-path)

Path where lock files are stored.

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

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

Neutral Umask Option (--neutral-umask)

Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

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, 999]
example: --process-max=4
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

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

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Spool Path Option (--spool-path)

Path where transient data is stored.

This path is used to store data for the asynchronous archive-push and archive-get command.

The asynchronous archive-push command writes acknowledgements into the spool path when it has successfully stored WAL in the archive (and errors on failure) so the foreground process can quickly notify PostgreSQL. Acknowledgement files are very small (zero on success and a few hundred bytes on error).

The asynchronous archive-get command queues WAL in the spool path so it can be provided very quickly when PostgreSQL requests it. Moving files to PostgreSQL is most efficient when the spool path is on the same filesystem as pg_xlog/pg_wal. However, it is not recommended to place the spool path *within* the pg_xlog/pg_wal directory as this may cause issues for PostgreSQL utilities such as pg_rewind.

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

The spool path is intended to be located on a local Posix-compatible filesystem, not a remote filesystem such as NFS or CIFS.

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

Stanza Option (--stanza)

Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one PostgreSQL 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
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL
```

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
```

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

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

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

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

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

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

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

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

The following log levels are supported:

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

- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

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

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

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

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: --log-subprocess
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

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

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

Maintainer Options

Force PostgreSQL Version Option (--pg-version-force)

Force PostgreSQL version.

The specified PostgreSQL version will be used instead of the version automatically detected by reading pg_control or WAL headers. This is mainly useful for PostgreSQL forks or development versions where those values are different from the release version. The version reported by PostgreSQL via ‘server_version_num‘ must match the forced version.

WARNING:

Be cautious when using this option because pg_control and WAL headers will still be read with the expected format for the specified version, i.e. the format from the official open-source version of PostgreSQL. If the fork or development version changes the format of the fields that pgBackRest depends on it will lead

to unexpected behavior. In general, this option will only work as expected if the fork adds all custom struct members *after* the standard PostgreSQL members.

`example: --pg-version-force=15`

Repository Options

Set Repository Option (--repo)

Set repository.

Set the repository for a command to operate on.

For example, this option may be used to perform a restore from a specific repository, rather than letting pgBackRest choose.

```
allowed: [1, 256]
example: --repo=1
```

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

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

`example: --repo1-azure-container=pg-backup`

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: --repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: --repo1-azure-uri-style=path
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

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

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: --repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

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

```
example: --repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: --repo1-gcs-endpoint=localhost
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.
- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: --repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: --repo1-gcs-user-project=my-project
```

Repository Host Option (--repo-host)

Repository host when operating remotely.

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

```
example: --repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: --repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: --repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

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

```
default: pgbackrest
example: --repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

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

It may 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: --repo1-path=/backup/db/backrest
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

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

```
example: --repo1-s3-bucket=pg-backup
```

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

```
example: --repo1-s3-endpoint=s3.amazonaws.com
```

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys
- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

```
default: shared
example: --repo1-s3-key-type=auto
```

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

```
example: --repo1-s3-process-cmd=/usr/local/bin/get-credentials --repo1-s3-process-cmd=--role
```

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

```
example: --repo1-s3-region=us-east-1
```

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

```
default: n
example: --no-repo1-s3-requester-pays
```

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

```
example: --repo1-s3-role=authrole
```

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

```
default: s3
example: --repo1-s3-service=s3-outposts
```

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

```
default: sts.amazonaws.com
example: --repo1-s3-sts-host=sts.us-east-1.amazonaws.com
```

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

```
default: host
example: --repo1-s3-uri-style=path
```

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}' ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

```
example: --repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf
```

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.
- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: --repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: --repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: --repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

`example: --repo1-sftp-host-user=pg-backup`

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

`example: --repo1-sftp-known-host=/home/postgres/.ssh/known_hosts`

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

`example: --repo1-sftp-private-key-file=~/.ssh/id_ed25519`

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

`example: --repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub`

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt`

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-path=/etc/pki/tls/certs`

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

```
example: --repo1-storage-host=127.0.0.1
```

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: --repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: --repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies

by storage type, so when searching min/max values use “part size” for AWS S3, “chunk size” for GCS, and “block size” for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]

example: --repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo1-storage-verify-tls
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Target Time for Repository Option (--repo-target-time)

Target time for repository.

The target time defines the time that commands use to read a repository on versioned storage. This allows the command to read the repository as it was at a point-in-time in order to recover data that has been deleted or corrupted by user accident or malware.

Versioned storage is supported by S3, GCS, and Azure but is generally not enabled by default. In addition to enabling versioning, it may be useful to enable object locking for S3 and soft delete for GCS or Azure.

When the repo-target-time option is specified then the repo option must also be provided. It is likely that not all repository types will support versioning and in general it makes sense to target a single repository for recovery.

Note that comparisons to the storage timestamp are <= the timestamp provided and milliseconds are truncated from the timestamp when provided.

```
example: --repo-target-time=2024-08-08 12:12:12+00
```

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs
- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: --repo1-type=cifs
```

Stanza Options

PostgreSQL Path Option (--pg-path)

PostgreSQL data directory.

This should be the same as the data_directory reported by PostgreSQL. Even though this value can be read from various places, it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

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

```
example: --pg1-path=/data/db
```

Deprecated Name: db-path

Archive Push Command (archive-push)

Accepts a WAL segment from PostgreSQL and archives it in each repository defined by the indexed repo-path option (see the Repository section for information on configuring repositories). The WAL segment may be pushed immediately to the archive or stored locally depending on the value of archive-async. With multiple repositories configured, archive-push will attempt to push to as many repositories as possible.

The archive-push is intended to be configured and called by PostgreSQL. See Configure Archiving for an example.

Command Options

Asynchronous Archiving Option (--archive-async)

Push/get WAL segments asynchronously.

Enables asynchronous operation for the archive-push and archive-get commands.

Asynchronous operation is more efficient because it can reuse connections and take advantage of parallelism. See the spool-path, archive-get-queue-max, and archive-push-queue-max options for more information.

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

Check Archive Option (--archive-check)

Check that WAL segments are in the archive before backup completes.

Checks that all WAL segments required to make the backup consistent are present in the WAL archive. It's a good idea to leave this as the default unless you are using another method for archiving.

This option must be enabled if archive-copy is enabled.

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

Check Archive Mode Option (--archive-mode-check)

Check the PostgreSQL archive_mode setting.

Enabled by default, this option disallows PostgreSQL archive_mode=always.

WAL segments pushed from a standby server might be logically the same as WAL segments pushed from the primary but have different checksums. Disabling archiving from multiple sources is recommended to avoid conflicts.

CAUTION:

If this option is disabled then it is critical to ensure that only one archiver is writing to the repository via the archive-push command.

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

Archive Push Batch Size Option (--archive-push-batch-size)

Maximum amount of WAL to push per asynchronous run.

In asynchronous mode the archive-push process pushes all the WAL segments that are ready in a single run. Since archive-push-queue-max is only checked at the start of each run, a run that processes a very large number of segments can let the queue grow well beyond the limit before it is rechecked.

This option limits the amount of WAL processed per run so the process exits and is spawned again by the next archive-push, which rechecks the queue. Lower values recheck the queue more often at the cost of spawning the asynchronous process more frequently. The value is rounded down to a whole number of WAL segments but at least one segment is always processed.

```
default: 16GiB
allowed: [1MiB, 4PiB]
example: --archive-push-batch-size=1GiB
```

Maximum Archive Push Queue Size Option (--archive-push-queue-max)

Maximum size of the PostgreSQL archive queue.

After the limit is reached, the following will happen:

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

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

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

In asynchronous mode this limit is only checked at the start of each archive-push run, so the queue can grow beyond it within a single run. Reduce archive-push-batch-size to check the queue more frequently.

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

```
allowed: [0B, 4PiB]
example: --archive-push-queue-max=1TiB
```

Deprecated Name: archive-queue-max

Archive Timeout Option (--archive-timeout)

Archive timeout.

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

```
default: 1m
allowed: [100ms, 1d]
example: --archive-timeout=30
```

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root

risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

pgBackRest Command Option (--cmd)

pgBackRest command.

pgBackRest may generate a command string, e.g. when the restore command generates the restore_command setting. The command used to run the pgBackRest process will be used in this case unless the cmd option is provided.

CAUTION:

Wrapping the pgBackRest command may cause unpredictable behavior and is not recommended.

```
default: [path of executed pgbackrest binary]
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

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

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

Compress Option (--compress)

Use file compression.

Backup files are compatible with command-line compression tools.

This option is now deprecated. The compress-type option should be used instead.

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

Compress Level Option (--compress-level)

File compression level.

Sets the level to be used for file compression when compress-type does not equal none or compress=y (deprecated).

```
default (depending on compress-type):
    bz2 - 9
    gz - 6
    lz4 - 1
    zst - 3

allow range (depending on compress-type):
    bz2 - [1, 9]
    gz - [-1, 9]
    lz4 - [-5, 12]
    zst - [-7, 22]

example: --compress-level=9
```

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: --compress-level-network=1
```

Compress Type Option (--compress-type)

File compression type.

The following compression types are supported:

- none - no compression
- bz2 - bzip2 compression format
- gz - gzip compression format
- lz4 - lz4 compression format (not available on all platforms)
- zst - Zstandard compression format (not available on all platforms)

```
default: gz
example: --compress-type=none
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

Lock Path Option (--lock-path)

Path where lock files are stored.

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

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

Neutral Umask Option (--neutral-umask)

Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

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, 999]
example: --process-max=4
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

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

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Spool Path Option (--spool-path)

Path where transient data is stored.

This path is used to store data for the asynchronous archive-push and archive-get command.

The asynchronous archive-push command writes acknowledgements into the spool path when it has successfully stored WAL in the archive (and errors on failure) so the foreground process can quickly notify PostgreSQL. Acknowledgement files are very small (zero on success and a few hundred bytes on error).

The asynchronous archive-get command queues WAL in the spool path so it can be provided very quickly when PostgreSQL requests it. Moving files to PostgreSQL is most efficient when the spool path is on the same filesystem as pg_xlog/pg_wal. However, it is not recommended to place the spool path *within* the pg_xlog/pg_wal directory as this may cause issues for PostgreSQL utilities such as pg_rewind.

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

The spool path is intended to be located on a local Posix-compatible filesystem, not a remote filesystem such as NFS or CIFS.

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

Stanza Option (--stanza)

Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one PostgreSQL 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
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

`example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL`

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

`example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256`

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

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

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

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

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

- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

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

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

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

The following log levels are supported:

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

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

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

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

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: --log-subprocess
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

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

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

Maintainer Options

Check WAL Headers Option (--archive-header-check)

Check PostgreSQL version/id in WAL headers.

Enabled by default, this option checks the WAL header against the PostgreSQL version and system identifier to ensure that the WAL is being copied to the correct stanza. This is in addition to checking pg_control against the stanza and verifying that WAL is being copied from the same PostgreSQL data directory where pg_control is located.

Therefore, disabling this check is fairly safe but should only be done when needed, e.g. if the WAL is encrypted.

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

Force PostgreSQL Version Option (--pg-version-force)

Force PostgreSQL version.

The specified PostgreSQL version will be used instead of the version automatically detected by reading pg_control or WAL headers. This is mainly useful for PostgreSQL forks or development versions where those values are different from the release version. The version reported by PostgreSQL via ‘server_version_num‘ must match the forced version.

WARNING:

Be cautious when using this option because pg_control and WAL headers will still be read with the expected format for the specified version, i.e. the format from the official open-source version of PostgreSQL. If the fork or development version changes the format of the fields that pgBackRest depends on it will lead to unexpected behavior. In general, this option will only work as expected if the fork adds all custom struct members *after* the standard PostgreSQL members.

```
example: --pg-version-force=15
```

Repository Options

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

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

```
example: --repo1-azure-container=pg-backup
```

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: --repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: --repo1-azure-uri-style=path
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

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

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: --repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

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

```
example: --repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: --repo1-gcs-endpoint=localhost
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.
- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: --repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: --repo1-gcs-user-project=my-project
```

Repository Host Option (--repo-host)

Repository host when operating remotely.

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

```
example: --repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: --repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: --repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

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

```
default: pgbackrest
example: --repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

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

It may 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: --repo1-path=/backup/db/backrest
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

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

```
example: --repo1-s3-bucket=pg-backup
```

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

```
example: --repo1-s3-endpoint=s3.amazonaws.com
```

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys
- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

```
default: shared
example: --repo1-s3-key-type=auto
```

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

```
example: --repo1-s3-process-cmd=/usr/local/bin/get-credentials --repo1-s3-process-cmd=--role
```

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

```
example: --repo1-s3-region=us-east-1
```

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

```
default: n
example: --no-repo1-s3-requester-pays
```

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

```
example: --repo1-s3-role=authrole
```

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

```
default: s3
example: --repo1-s3-service=s3-outposts
```

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

```
default: sts.amazonaws.com
example: --repo1-s3-sts-host=sts.us-east-1.amazonaws.com
```

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

```
default: host
example: --repo1-s3-uri-style=path
```

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}' ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

```
example: --repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf
```

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.
- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: --repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: --repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: --repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

```
example: --repo1-sftp-host-user=pg-backup
```

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

```
example: --repo1-sftp-known-host=/home/postgres/.ssh/known_hosts
```

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

```
example: --repo1-sftp-private-key-file=~/.ssh/id_ed25519
```

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

```
example: --repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub
```

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

```
example: --repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

```
example: --repo1-storage-ca-path=/etc/pki/tls/certs
```

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

```
example: --repo1-storage-host=127.0.0.1
```

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: --repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: --repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies by storage type, so when searching min/max values use "part size" for AWS S3, "chunk size" for GCS, and "block size" for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]

example: --repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo1-storage-verify-tls
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs
- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: --repo1-type=cifs
```

Stanza Options

PostgreSQL Path Option (--pg-path)

PostgreSQL data directory.

This should be the same as the data_directory reported by PostgreSQL. Even though this value can be read from various places, it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

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

```
example: --pg1-path=/data/db
```

Deprecated Name: db-path

Backup Command (backup)

When multiple repositories are configured, pgBackRest will backup to the highest priority repository (e.g. repo1) unless the --repo option is specified.

pgBackRest does not have a built-in scheduler so it's best to run it from cron or some other scheduling mechanism.

See Perform a Backup for more details and examples.

Command Options

Backup Annotation Option (--annotation)

Annotate backup with user-defined key/value pairs.

Users can attach informative key/value pairs to the backup. This option may be used multiple times to attach multiple annotations.

Annotations are output by the info command text output when a backup is specified with --set and always appear in the JSON output.

```
example: --annotation=source="Sunday backup for website database"
```

Check Archive Option (--archive-check)

Check that WAL segments are in the archive before backup completes.

Checks that all WAL segments required to make the backup consistent are present in the WAL archive. It's a good idea to leave this as the default unless you are using another method for archiving.

This option must be enabled if archive-copy is enabled.

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

Copy Archive Option (--archive-copy)

Copy WAL segments needed for consistency to the backup.

This slightly paranoid option protects against corruption in the WAL segment archive by storing the WAL segments required for consistency directly in the backup. WAL segments are still stored in the archive so this option will use additional space.

It is best if the archive-push and backup commands have the same compress-type (e.g. lz4) when using this option. Otherwise, the WAL segments will need to be recompressed with the compress-type used by the backup, which can be fairly expensive depending on how much WAL was generated during the backup.

On restore, the WAL segments will be present in pg_xlog/pg_wal and PostgreSQL will use them in preference to calling the restore_command.

The archive-check option must be enabled if archive-copy is enabled.

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

Check Archive Mode Option (--archive-mode-check)

Check the PostgreSQL archive_mode setting.

Enabled by default, this option disallows PostgreSQL archive_mode=always.

WAL segments pushed from a standby server might be logically the same as WAL segments pushed from the primary but have different checksums. Disabling archiving from multiple sources is recommended to avoid conflicts.

CAUTION:

If this option is disabled then it is critical to ensure that only one archiver is writing to the repository via the archive-push command.

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

Archive Timeout Option (--archive-timeout)

Archive timeout.

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

```
default: 1m
allowed: [100ms, 1d]
example: --archive-timeout=30
```

Backup from Standby Option (--backup-standby)

Backup from the standby cluster.

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

The following modes are supported:

- y - Standby is required for backup.
- prefer - Backup from standby if available otherwise backup from primary.
- n - Backup from primary only.

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

Page Checksums Option (--checksum-page)

Validate data page checksums.

Directs pgBackRest to validate all data page checksums while backing up a cluster. This option is automatically enabled when data page 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
```

Path/File Exclusions Option (--exclude)

Exclude paths/files from the backup.

All exclusions are relative to $PGDATA. If the exclusion ends with / then only files in the specified directory will be excluded, e.g. --exclude=junk/ will exclude

all files in the $PGDATA/junk directory but include the directory itself. If the exclusion does not end with / then the file may match the exclusion exactly or match with / appended to the exclusion, e.g. --exclude=junk will exclude the $PGDATA/junk directory and all the files it contains.

Be careful using this feature -- it is very easy to exclude something critical that will make the backup inconsistent. Be sure to test your restores!

All excluded files will be logged at info level along with the exclusion rule. Be sure to audit the list of excluded files to ensure nothing unexpected is being excluded.

NOTE:

Exclusions are not honored on delta restores. Any files/directories that were excluded by the backup will be *removed* on delta restore.

This option should not be used to exclude PostgreSQL logs from a backup. Logs can be moved out of the PGDATA directory using the PostgreSQL log_directory setting, which has the benefit of allowing logs to be preserved after a restore.

Multiple exclusions may be specified on the command-line or in a configuration file.

```
example: --exclude=junk/
```

Expire Auto Option (--expire-auto)

Automatically run the expire command after a successful backup.

The setting is enabled by default. Use caution when disabling this option as doing so will result in retaining all backups and archives indefinitely, which could cause your repository to run out of space. The expire command will need to be run regularly to prevent this from happening.

When expire is run automatically after a successful backup it uses the configuration of the backup command, so options set only in an expire command section (e.g. [global:expire]) are not applied. To apply expire-specific configuration, disable this option and run the expire command separately.

```
default: y
example: --expire-auto
```

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
```

Manifest Save Threshold Option (--manifest-save-threshold)

Manifest save threshold during backup.

Defines how often the manifest will be saved during a backup. 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: 1GiB
allowed: [1B, 1TiB]
example: --manifest-save-threshold=8GiB
```

Online Option (--online)

Perform an online backup.

Specifying --no-online prevents pgBackRest from running the backup start/stop functions on the database cluster. In order for this to work PostgreSQL should be shut down and pgBackRest will generate an error if it is not.

The purpose of this option is to allow offline backups. The pg_xlog/pg_wal directory is copied as-is and archive-check is automatically disabled for the backup.

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

Resume Option (--resume)

Allow resume of failed backup.

Defines whether the resume feature is enabled. Resume can greatly reduce the amount of time required to run a backup after a previous backup of the same type has failed. It adds complexity, however, so it may be desirable to disable in environments that do not require the feature.

```
default: y
example: --no-resume
```

Start Fast Option (--start-fast)

Force a checkpoint to start backup quickly.

Forces a checkpoint (by passing y to the fast parameter of the backup start function) so the backup begins immediately. Otherwise the backup will start after the next regular checkpoint.

```
default: n
example: --start-fast
```

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
```

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

pgBackRest Command Option (--cmd)

pgBackRest command.

pgBackRest may generate a command string, e.g. when the restore command generates the restore_command setting. The command used to run the pgBackRest process will be used in this case unless the cmd option is provided.

CAUTION:

Wrapping the pgBackRest command may cause unpredictable behavior and is not recommended.

```
default: [path of executed pgbackrest binary]
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

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

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

Compress Option (--compress)

Use file compression.

Backup files are compatible with command-line compression tools.

This option is now deprecated. The compress-type option should be used instead.

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

Compress Level Option (--compress-level)

File compression level.

Sets the level to be used for file compression when compress-type does not equal none or compress=y (deprecated).

```
default (depending on compress-type):
    bz2 - 9
    gz - 6
    lz4 - 1
    zst - 3

allow range (depending on compress-type):
    bz2 - [1, 9]
    gz - [-1, 9]
    lz4 - [-5, 12]
    zst - [-7, 22]

example: --compress-level=9
```

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: --compress-level-network=1
```

Compress Type Option (--compress-type)

File compression type.

The following compression types are supported:

- none - no compression
- bz2 - bzip2 compression format
- gz - gzip compression format
- lz4 - lz4 compression format (not available on all platforms)
- zst - Zstandard compression format (not available on all platforms)

```
default: gz
example: --compress-type=none
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

Database Timeout Option (--db-timeout)

Database query timeout.

Sets the timeout, in seconds, for queries against the database. This includes the backup start/stop 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 start-fast=y and you know that the database cluster will not generate many WAL segments during the backup).

NOTE:

The db-timeout option must be less than the protocol-timeout option.

```
default: 30m
allowed: [100ms, 7d]
example: --db-timeout=600
```

Delta Option (--delta)

Restore or backup using checksums.

During a restore, by default the PostgreSQL data and tablespace directories are expected to be present but empty. This option performs a delta restore using checksums.

During a backup, this option will use checksums instead of the timestamps to determine if files will be copied.

```
default: n
example: --delta
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

Lock Path Option (--lock-path)

Path where lock files are stored.

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

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

Neutral Umask Option (--neutral-umask)

Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

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, 999]
example: --process-max=4
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

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

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Stanza Option (--stanza)

Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one PostgreSQL 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
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL
```

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
```

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

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

- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

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

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

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

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

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

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

The following log levels are supported:

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

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

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

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

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: --log-subprocess
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

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

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

Maintainer Options

Page Header Check Option (--page-header-check)

Check PostgreSQL page headers.

Enabled by default, this option adds page header checks.

Disabling this option should be avoided except when necessary, e.g. if pages are encrypted.

```
default: y
example: --no-page-header-check
```

Force PostgreSQL Version Option (--pg-version-force)

Force PostgreSQL version.

The specified PostgreSQL version will be used instead of the version automatically detected by reading pg_control or WAL headers. This is mainly useful for PostgreSQL forks or development versions where those values are different from the release version. The version reported by PostgreSQL via ‘server_version_num‘ must match the forced version.

WARNING:

Be cautious when using this option because pg_control and WAL headers will still be read with the expected format for the specified version, i.e. the format from the official open-source version of PostgreSQL. If the fork or development version changes the format of the fields that pgBackRest depends on it will lead to unexpected behavior. In general, this option will only work as expected if the fork adds all custom struct members *after* the standard PostgreSQL members.

```
example: --pg-version-force=15
```

Repository Options

Set Repository Option (--repo)

Set repository.

Set the repository for a command to operate on.

For example, this option may be used to perform a restore from a specific repository, rather than letting pgBackRest choose.

```
allowed: [1, 256]
example: --repo=1
```

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

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

```
example: --repo1-azure-container=pg-backup
```

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: --repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: --repo1-azure-uri-style=path
```

Block Incremental Backup Option (--repo-block)

Enable block incremental backup.

Block incremental allows for more granular backups by splitting files into blocks that can be backed up independently. This saves space in the repository and

can improve delta restore performance because individual blocks can be fetched without reading the entire file from the repository.

NOTE:

The repo-bundle option must be enabled before repo-block can be enabled.

The block size for a file is determined based on the file size and age. Generally, older/larger files will get larger block sizes. If a file is old enough, it will not be backed up using block incremental.

Block incremental is most efficient when enabled for all backup types, including full. This makes the full a bit larger but subsequent differential and incremental backups can make use of the block maps generated by the full backup to save space.

```
default: n
example: --repo1-block
```

Repository Bundles Option (--repo-bundle)

Bundle files in repository.

Bundle (combine) smaller files to reduce the total number of files written to the repository. Writing fewer files is generally more efficient, especially on object stores such as S3. In addition, zero-length files are not stored (except in the manifest), which saves time and space.

```
default: n
example: --repo1-bundle
```

Repository Bundle Limit Option (--repo-bundle-limit)

Limit for file bundles.

Size limit for files that will be included in bundles. Files larger than this size will be stored separately.

Bundled files cannot be reused when a backup is resumed, so this option controls the files that can be resumed, i.e. higher values result in fewer resumable files.

```
default: 2MiB
allowed: [8KiB, 1PiB]
example: --repo1-bundle-limit=10MiB
```

Repository Bundle Size Option (--repo-bundle-size)

Target size for file bundles.

Defines the target size for files that will be added to a single bundle. The uncompressed bundle size may be as large as repo-bundle-size + repo-bundle-limit, so do not set this option to the maximum size that your file system allows.

In general, it is not a good idea to set this option too high because retries will need to redo the entire bundle.

```
default: 20MiB
allowed: [1MiB, 1PiB]
example: --repo1-bundle-size=10MiB
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

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

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: --repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

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

```
example: --repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: --repo1-gcs-endpoint=localhost
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.
- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: --repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: --repo1-gcs-user-project=my-project
```

Repository Hardlink Option (--repo-hardlink)

Hardlink files between backups in the repository.

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 at the file-system level. Be careful, though, because modifying files that are hard-linked can affect all the backups in the set.

```
default: n
example: --repo1-hardlink
```

Deprecated Name: hardlink

Repository Host Option (--repo-host)

Repository host when operating remotely.

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

```
example: --repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: --repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: --repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

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

```
default: pgbackrest
example: --repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

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

It may 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: --repo1-path=/backup/db/backrest
```

Archive Retention Option (--repo-retention-archive)

Number of backups worth of continuous WAL to retain.

NOTE:

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 and repo-retention-full-type is count (default), then the archive to expire will default to the repo-retention-full (or repo-retention-diff) value corresponding to the repo-retention-archive-type if set to full (or diff). This will ensure that WAL is only expired for backups that are already expired. If repo-retention-full-type is time, then this value will default to removing archives that are earlier than the oldest full backup retained after satisfying the repo-retention-full setting.

This option must be set if repo-retention-archive-type is set to incr. If disk space is at a premium, then this setting, in conjunction with repo-retention-archive-type, can be used to aggressively expire WAL segments. However, doing so negates the ability to perform PITR from the backups with expired WAL and is therefore **not** recommended.

```
allowed: [1, 9999999]
example: --repo1-retention-archive=2
```

Deprecated Name: retention-archive

Archive Retention Type Option (--repo-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 repo-retention-archive. If set to diff (differential) pgBackRest will keep archive logs for the number of full and differential backups defined by repo-retention-archive, meaning if the last backup taken was a full backup, it will be counted as a differential for the purpose of repo-retention. If set to incr (incremental) pgBackRest will keep archive logs for the number of full, differential, and incremental backups defined by repo-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: --repo1-retention-archive-type=diff
```

Deprecated Name: retention-archive-type

Differential Retention Option (--repo-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.

Note that full backups are included in the count of differential backups for the purpose of expiration. This slightly reduces the number of differential backups that need to be retained in most cases.

```
allowed: [1, 9999999]
example: --repo1-retention-diff=3
```

Deprecated Name: retention-diff

Full Retention Option (--repo-retention-full)

Full backup retention count/time.

When a full backup expires, all differential and incremental backups associated with the full backup will also expire. When the option is not defined a warning will be issued. If indefinite retention is desired then set the option to the max value.

```
allowed: [1, 9999999]
example: --repo1-retention-full=2
```

Deprecated Name: retention-full

Full Retention Type Option (--repo-retention-full-type)

Retention type for full backups.

Determines whether the repo-retention-full setting represents a time period (days) or count of full backups to keep.

If set to time then full backups older than repo-retention-full will be removed from the repository if there is at least one other backup that is equal to or greater than the repo-retention-full setting. For example, if repo-retention-full is 30 (days) and there are 2 full backups: one 25 days old and one 35 days old, no full backups will be expired because expiring the 35 day old backup would leave only the 25 day old backup, which would violate the 30 day retention policy of having at least one backup 30 days old before an older one can be expired. Archived WAL older than the oldest full backup remaining will be automatically expired unless repo-retention-archive-type and repo-retention-archive are explicitly set.

If set to count then full backups that exceed repo-retention-full will be expired. For example, if repo-retention-full is 4 and a fifth full backup is completed, then the oldest full backup will be expired to keep the count at 4.

Note that a backup must be successfully completed before it will be considered for retention. For example, if repo-retention-full-type is count and repo-

retention-full is 2, then there must be 3 complete full backups before the oldest will be expired.

```
default: count
example: --repo1-retention-full-type=time
```

Backup History Retention Option (--repo-retention-history)

Days of backup history manifests to retain.

A copy of the backup manifest is stored in the backup.history path when a backup completes. By default these files are never expired since they are useful for data mining, e.g. measuring backup and WAL growth over time.

Set repo-retention-history to define the number of days of backup history manifests to retain. Unexpired backups are always kept in the backup history. Specify repo-retention-history=0 to retain the backup history only for unexpired backups.

When a full backup history manifest is expired, all differential and incremental backup history manifests associated with the full backup also expire.

```
allowed: [0, 9999999]
example: --repo1-retention-history=365
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

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

```
example: --repo1-s3-bucket=pg-backup
```

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

```
example: --repo1-s3-endpoint=s3.amazonaws.com
```

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys

- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

```
default: shared
example: --repo1-s3-key-type=auto
```

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

```
example: --repo1-s3-process-cmd=/usr/local/bin/get-credentials --repo1-s3-process-cmd=--role
```

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

```
example: --repo1-s3-region=us-east-1
```

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

```
default: n
example: --no-repo1-s3-requester-pays
```

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

```
example: --repo1-s3-role=authrole
```

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

```
default: s3
example: --repo1-s3-service=s3-outposts
```

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

```
default: sts.amazonaws.com
example: --repo1-s3-sts-host=sts.us-east-1.amazonaws.com
```

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

```
default: host
example: --repo1-s3-uri-style=path
```

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}' ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

```
example: --repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf
```

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.
- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: --repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: --repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: --repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

```
example: --repo1-sftp-host-user=pg-backup
```

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file

to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

`example: --repo1-sftp-known-host=/home/postgres/.ssh/known_hosts`

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

`example: --repo1-sftp-private-key-file=~/.ssh/id_ed25519`

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

`example: --repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub`

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt`

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-path=/etc/pki/tls/certs`

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

`example: --repo1-storage-host=127.0.0.1`

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: --repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: --repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies by storage type, so when searching min/max values use “part size” for AWS S3, “chunk size” for GCS, and “block size” for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
```

```
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]

example: --repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo1-storage-verify-tls
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Repository Symlink Option (--repo-symlink)

Create symlinks within the repository.

Enable creation of the lastest and tablespace symlinks. These symlinks are most useful when using snapshots to do in-place recovery in the repository, which is an uncommon use case.

While this feature is likely not useful for the vast majority of users it remains on by default for legacy purposes. However, it may be useful to disable symlinks for Posix-like storage that does not support them.

```
default: y
example: --no-repo1-symlink
```

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs
- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: --repo1-type=cifs
```

Stanza Options

PostgreSQL Database Option (--pg-database)

PostgreSQL database.

The database name used when connecting to PostgreSQL. The default is usually best but some installations may not contain this database.

Note that for legacy reasons the setting of the PGDATABASE environment variable will be ignored.

```
default: postgres
example: --pg1-database=backupdb
```

PostgreSQL Host Option (--pg-host)

PostgreSQL host for operating remotely.

Used for backups where the PostgreSQL host is different from the repository host.

```
example: --pg1-host=db.domain.com
```

Deprecated Name: db-host

PostgreSQL Host Certificate Authority File Option (--pg-host-ca-file)

PostgreSQL host certificate authority file.

Use a CA file other than the system default for connecting to the PostgreSQL host.

```
example: --pg1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

PostgreSQL Host Certificate Authority Path Option (--pg-host-ca-path)

PostgreSQL host certificate authority path.

Use a CA path other than the system default for connecting to the PostgreSQL host.

```
example: --pg1-host-ca-path=/etc/pki/tls/certs
```

PostgreSQL Host Certificate File Option (--pg-host-cert-file)

PostgreSQL host certificate file.

Sent to PostgreSQL host to prove client identity.

```
example: --pg1-host-cert-file=/path/to/client.crt
```

PostgreSQL Host Command Option (--pg-host-cmd)

PostgreSQL host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and PostgreSQL hosts. If not defined, the PostgreSQL host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --pg1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: db-cmd

PostgreSQL Host Configuration Option (--pg-host-config)

pgBackRest database host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --pg1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: db-config

PostgreSQL Host Configuration Include Path Option (--pg-host-config-include-path)

pgBackRest database host configuration include path.

Sets the location of the configuration include path on the PostgreSQL host. This is only required if the PostgreSQL host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --pg1-host-config-include-path=/conf/pgbackrest/conf.d
```

PostgreSQL Host Configuration Path Option (--pg-host-config-path)

pgBackRest database host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --pg1-host-config-path=/conf/pgbackrest
```

PostgreSQL Host Key File Option (--pg-host-key-file)

PostgreSQL host key file.

Proves client certificate was sent by owner.

```
example: --pg1-host-key-file=/path/to/client.key
```

PostgreSQL Host Port Option (--pg-host-port)

PostgreSQL host port when pg-host is set.

Use this option to specify a non-default port for the PostgreSQL host protocol.

NOTE:

When pg-host-type=ssh there is no default for pg-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on pg-host-type):
    tls - 8432

allowed: [0, 65535]
example: --pg1-host-port=25
```

Deprecated Name: db-ssh-port

PostgreSQL Host Protocol Type Option (--pg-host-type)

PostgreSQL host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --pg1-host-type=tls
```

PostgreSQL Host User Option (--pg-host-user)

PostgreSQL host logon user when pg-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: --pg1-host-user=db_owner
```

Deprecated Name: db-user

PostgreSQL Path Option (--pg-path)

PostgreSQL data directory.

This should be the same as the data_directory reported by PostgreSQL. Even though this value can be read from various places, it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

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

```
example: --pg1-path=/data/db
```

Deprecated Name: db-path

PostgreSQL Port Option (--pg-port)

PostgreSQL port.

Port that PostgreSQL is running on. This usually does not need to be specified as most PostgreSQL clusters run on the default port.

```
default: 5432
allowed: [0, 65535]
example: --pg1-port=6543
```

Deprecated Name: db-port

PostgreSQL Socket Path Option (--pg-socket-path)

PostgreSQL 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 is usually no need to specify this setting unless the socket directory was explicitly modified with the unix_socket_directories setting in postgresql.conf.

```
example: --pg1-socket-path=/var/run/postgresql
```

Deprecated Name: db-socket-path

PostgreSQL Database User Option (--pg-user)

PostgreSQL database user.

The database user name used when connecting to PostgreSQL. If not specified pgBackRest will connect with the local OS user or PGUSER.

```
example: --pg1-user=backupuser
```

Check Command (check)

The check command validates that pgBackRest and the archive_command setting are configured correctly for archiving and backups for the specified stanza. It will attempt to check all repositories and databases that are configured for the host on which the command is run. 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 PostgreSQL or repository host. The command may also be run on the standby host, however, since pg_switch_xlog()/pg_switch_wal() cannot be performed on the standby, the command will only test the repository configuration.

Note that pg_create_restore_point('pgBackRest Archive Check') and pg_switch_xlog()/pg_switch_wal() are called to force PostgreSQL to archive a WAL segment.

Command Options

Check Archive Option (--archive-check)

Check that WAL segments are in the archive before backup completes.

Checks that all WAL segments required to make the backup consistent are present in the WAL archive. It's a good idea to leave this as the default unless you are using another method for archiving.

This option must be enabled if archive-copy is enabled.

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

Check Archive Mode Option (--archive-mode-check)

Check the PostgreSQL archive_mode setting.

Enabled by default, this option disallows PostgreSQL archive_mode=always.

WAL segments pushed from a standby server might be logically the same as WAL segments pushed from the primary but have different checksums. Disabling archiving from multiple sources is recommended to avoid conflicts.

CAUTION:

If this option is disabled then it is critical to ensure that only one archiver is writing to the repository via the archive-push command.

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

Archive Timeout Option (--archive-timeout)

Archive timeout.

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

```
default: 1m
allowed: [100ms, 1d]
example: --archive-timeout=30
```

Backup from Standby Option (--backup-standby)

Backup from the standby cluster.

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

The following modes are supported:

- y - Standby is required for backup.
- prefer - Backup from standby if available otherwise backup from primary.
- n - Backup from primary only.

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

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

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

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

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: --compress-level-network=1
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

Database Timeout Option (--db-timeout)

Database query timeout.

Sets the timeout, in seconds, for queries against the database. This includes the backup start/stop 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 start-fast=y and you know that the database cluster will not generate many WAL segments during the backup).

NOTE:

The db-timeout option must be less than the protocol-timeout option.

```
default: 30m
allowed: [100ms, 7d]
example: --db-timeout=600
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

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.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

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

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Stanza Option (--stanza)

Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one PostgreSQL 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
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL
```

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
```

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

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

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

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

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

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

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

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

The following log levels are supported:

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

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

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

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

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: --log-subprocess
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

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

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

Maintainer Options

Force PostgreSQL Version Option (--pg-version-force)

Force PostgreSQL version.

The specified PostgreSQL version will be used instead of the version automatically detected by reading pg_control or WAL headers. This is mainly useful for PostgreSQL forks or development versions where those values are different from the release version. The version reported by PostgreSQL via ‘server_version_num‘ must match the forced version.

WARNING:

Be cautious when using this option because pg_control and WAL headers will still be read with the expected format for the specified version, i.e. the format from the official open-source version of PostgreSQL. If the fork or development version changes the format of the fields that pgBackRest depends on it will lead to unexpected behavior. In general, this option will only work as expected if the fork adds all custom struct members *after* the standard PostgreSQL members.

```
example: --pg-version-force=15
```

Repository Options

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

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

```
example: --repo1-azure-container=pg-backup
```

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: --repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: --repo1-azure-uri-style=path
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

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

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: --repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

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

```
example: --repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: --repo1-gcs-endpoint=localhost
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.
- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: --repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: --repo1-gcs-user-project=my-project
```

Repository Host Option (--repo-host)

Repository host when operating remotely.

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

```
example: --repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: --repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: --repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

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

```
default: pgbackrest
example: --repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

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

It may 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: --repo1-path=/backup/db/backrest
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

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

```
example: --repo1-s3-bucket=pg-backup
```

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

```
example: --repo1-s3-endpoint=s3.amazonaws.com
```

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys
- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

```
default: shared
example: --repo1-s3-key-type=auto
```

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

```
example: --repo1-s3-process-cmd=/usr/local/bin/get-credentials --repo1-s3-process-cmd=--role
```

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

```
example: --repo1-s3-region=us-east-1
```

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

```
default: n
example: --no-repo1-s3-requester-pays
```

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

```
example: --repo1-s3-role=authrole
```

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

```
default: s3
example: --repo1-s3-service=s3-outposts
```

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

```
default: sts.amazonaws.com
example: --repo1-s3-sts-host=sts.us-east-1.amazonaws.com
```

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

```
default: host
example: --repo1-s3-uri-style=path
```

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}' ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

```
example: --repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf
```

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.
- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: --repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: --repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: --repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

```
example: --repo1-sftp-host-user=pg-backup
```

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

```
example: --repo1-sftp-known-host=/home/postgres/.ssh/known_hosts
```

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

```
example: --repo1-sftp-private-key-file=~/.ssh/id_ed25519
```

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

```
example: --repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub
```

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

```
example: --repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

```
example: --repo1-storage-ca-path=/etc/pki/tls/certs
```

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

```
example: --repo1-storage-host=127.0.0.1
```

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: --repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: --repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies by storage type, so when searching min/max values use "part size" for AWS S3, "chunk size" for GCS, and "block size" for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]

example: --repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo1-storage-verify-tls
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs
- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: --repo1-type=cifs
```

Stanza Options

PostgreSQL Database Option (--pg-database)

PostgreSQL database.

The database name used when connecting to PostgreSQL. The default is usually best but some installations may not contain this database.

Note that for legacy reasons the setting of the PGDATABASE environment variable will be ignored.

```
default: postgres
example: --pg1-database=backupdb
```

PostgreSQL Host Option (--pg-host)

PostgreSQL host for operating remotely.

Used for backups where the PostgreSQL host is different from the repository host.

```
example: --pg1-host=db.domain.com
```

Deprecated Name: db-host

PostgreSQL Host Certificate Authority File Option (--pg-host-ca-file)

PostgreSQL host certificate authority file.

Use a CA file other than the system default for connecting to the PostgreSQL host.

```
example: --pg1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

PostgreSQL Host Certificate Authority Path Option (--pg-host-ca-path)

PostgreSQL host certificate authority path.

Use a CA path other than the system default for connecting to the PostgreSQL host.

```
example: --pg1-host-ca-path=/etc/pki/tls/certs
```

PostgreSQL Host Certificate File Option (--pg-host-cert-file)

PostgreSQL host certificate file.

Sent to PostgreSQL host to prove client identity.

```
example: --pg1-host-cert-file=/path/to/client.crt
```

PostgreSQL Host Command Option (--pg-host-cmd)

PostgreSQL host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and PostgreSQL hosts. If not defined, the PostgreSQL host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --pg1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: db-cmd

PostgreSQL Host Configuration Option (--pg-host-config)

pgBackRest database host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --pg1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: db-config

PostgreSQL Host Configuration Include Path Option (--pg-host-config-include-path)

pgBackRest database host configuration include path.

Sets the location of the configuration include path on the PostgreSQL host. This is only required if the PostgreSQL host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --pg1-host-config-include-path=/conf/pgbackrest/conf.d
```

PostgreSQL Host Configuration Path Option (--pg-host-config-path)

pgBackRest database host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --pg1-host-config-path=/conf/pgbackrest
```

PostgreSQL Host Key File Option (--pg-host-key-file)

PostgreSQL host key file.

Proves client certificate was sent by owner.

```
example: --pg1-host-key-file=/path/to/client.key
```

PostgreSQL Host Port Option (--pg-host-port)

PostgreSQL host port when pg-host is set.

Use this option to specify a non-default port for the PostgreSQL host protocol.

NOTE:

When pg-host-type=ssh there is no default for pg-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on pg-host-type):
    tls - 8432

allowed: [0, 65535]
example: --pg1-host-port=25
```

Deprecated Name: db-ssh-port

PostgreSQL Host Protocol Type Option (--pg-host-type)

PostgreSQL host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --pg1-host-type=tls
```

PostgreSQL Host User Option (--pg-host-user)

PostgreSQL host logon user when pg-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: --pg1-host-user=db_owner
```

Deprecated Name: db-user

PostgreSQL Path Option (--pg-path)

PostgreSQL data directory.

This should be the same as the data_directory reported by PostgreSQL. Even though this value can be read from various places, it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

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

```
example: --pg1-path=/data/db
```

Deprecated Name: db-path

PostgreSQL Port Option (--pg-port)

PostgreSQL port.

Port that PostgreSQL is running on. This usually does not need to be specified as most PostgreSQL clusters run on the default port.

```
default: 5432
allowed: [0, 65535]
example: --pg1-port=6543
```

Deprecated Name: db-port

PostgreSQL Socket Path Option (--pg-socket-path)

PostgreSQL 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 is usually no need to specify this setting unless the socket directory was explicitly modified with the unix_socket_directories setting in postgresql.conf.

```
example: --pg1-socket-path=/var/run/postgresql
```

Deprecated Name: db-socket-path

PostgreSQL Database User Option (--pg-user)

PostgreSQL database user.

The database user name used when connecting to PostgreSQL. If not specified pgBackRest will connect with the local OS user or PGUSER.

```
example: --pg1-user=backupuser
```

Expire Command (expire)

pgBackRest does full backup rotation based on the retention type which can be a count or a time period. When a count is specified, then expiration is not concerned with when the backups were created but with how many must be retained. Differential backups are count-based but will always be expired when the full backup they depend on is expired. Incremental backups are not expired by retention independently — they are always expired with their related full or differential backup. See sections Full Backup Retention and Differential Backup Retention for details and examples.

Archived WAL is retained by default for backups that have not expired, however, although not recommended, this schedule can be modified per repository with the retention-archive options. See section Archive Retention for details and examples.

The expire command is run automatically after each successful backup and can also be run by the user. When run by the user, expiration will occur as defined by the retention settings for each configured repository. If the --repo option is provided, expiration will occur only on the specified repository. Expiration can also be limited by the user to a specific backup set with the --set option and, unless the --repo option is specified, all repositories will be searched and any matching the set criteria will be expired. It should be noted that the archive retention schedule will be checked and performed any time the expire command is run.

Command Options

Expire Archive Before Option (--archive-expire-before)

Remove WAL archive earlier than the specified WAL segment.

Remove WAL segments earlier than the specified segment, but only those that are not required by a retained backup, that is, segments older than the earliest backup set to retain, or any segment when the repository contains no backup. The specified segment and everything after it are kept. This mirrors the value passed by PostgreSQL to archive_cleanup_command as %r.

This option is primarily intended for archive-only repositories where no backups are stored. It can also be used to reclaim WAL archive space before the first backup when full retention has not yet been met to trigger archive expiration.

```
example: --archive-expire-before=000000010000000000000010
```

Oldest Option (--oldest)

Expire the oldest eligible backup set.

Expire the oldest full backup set that can be removed (meaning at least one newer full backup remains). This is equivalent to manually decrementing retention by one, but computed automatically. All backups related to the expired full backup set (differential and incremental) are also expired.

When used, archive retention is also temporarily adjusted so WAL for the expired backups can be removed in the same run.

If time-based full retention is configured (using --repo-retention-full-type=time) then --oldest uses count-based expiration for this execution.

WARNING:

This option cannot be combined with --set.

```
default: n
example: --oldest
```

Set Option (--set)

Backup set to expire.

The specified backup set (i.e. the backup label provided and all of its dependent backups, if any) will be expired regardless of backup retention rules except that at least one full backup must remain in the repository.

WARNING:

Use this option with extreme caution — it will permanently remove all backups and archives not required to make a backup consistent from the pgBackRest repository for the specified backup set. This process may negate the ability to perform PITR. If --repo-retention-full and/or --repo-retention-archive options are configured, then it is recommended that you override these options by setting their values to the maximum while performing ad hoc expiration in order to prevent an unintended expiration of archives.

```
example: --set=20150131-153358F_20150131-153401I
```

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

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

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

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: --compress-level-network=1
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

Dry Run Option (--dry-run)

Execute a dry-run for the command.

The --dry-run option is a command-line only option and can be passed when it is desirable to determine what modifications will be made by the command without the command actually making any modifications.

```
default: n
example: --dry-run
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

Lock Path Option (--lock-path)

Path where lock files are stored.

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

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

Neutral Umask Option (--neutral-umask)

Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

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

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Stanza Option (--stanza)

Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one PostgreSQL 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
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

`example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL`

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

`example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256`

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

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

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

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

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

- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

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

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

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

The following log levels are supported:

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

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

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

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

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: --log-subprocess
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

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

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

Repository Options

Set Repository Option (--repo)

Set repository.

Set the repository for a command to operate on.

For example, this option may be used to perform a restore from a specific repository, rather than letting pgBackRest choose.

```
allowed: [1, 256]
example: --repo=1
```

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

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

```
example: --repo1-azure-container=pg-backup
```

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: --repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: --repo1-azure-uri-style=path
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

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

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: --repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

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

```
example: --repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: --repo1-gcs-endpoint=localhost
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.
- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: --repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: --repo1-gcs-user-project=my-project
```

Repository Host Option (--repo-host)

Repository host when operating remotely.

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

```
example: --repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: --repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: --repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

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

```
default: pgbackrest
example: --repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

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

It may 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: --repo1-path=/backup/db/backrest
```

Archive Retention Option (--repo-retention-archive)

Number of backups worth of continuous WAL to retain.

NOTE:

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 and repo-retention-full-type is count (default), then the archive to expire will default to the repo-retention-full (or repo-retention-diff) value corresponding to the repo-retention-archive-type if set to full (or diff). This will ensure that WAL is only expired for backups that are already expired. If repo-retention-full-type is time, then this value will default to removing

archives that are earlier than the oldest full backup retained after satisfying the repo-retention-full setting.

This option must be set if repo-retention-archive-type is set to incr. If disk space is at a premium, then this setting, in conjunction with repo-retention-archive-type, can be used to aggressively expire WAL segments. However, doing so negates the ability to perform PITR from the backups with expired WAL and is therefore **not** recommended.

```
allowed: [1, 9999999]
example: --repo1-retention-archive=2
```

Deprecated Name: retention-archive

Archive Retention Type Option (--repo-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 repo-retention-archive. If set to diff (differential) pgBackRest will keep archive logs for the number of full and differential backups defined by repo-retention-archive, meaning if the last backup taken was a full backup, it will be counted as a differential for the purpose of repo-retention. If set to incr (incremental) pgBackRest will keep archive logs for the number of full, differential, and incremental backups defined by repo-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: --repo1-retention-archive-type=diff
```

Deprecated Name: retention-archive-type

Differential Retention Option (--repo-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.

Note that full backups are included in the count of differential backups for the purpose of expiration. This slightly reduces the number of differential backups that need to be retained in most cases.

```
allowed: [1, 9999999]
example: --repo1-retention-diff=3
```

Deprecated Name: retention-diff

Full Retention Option (--repo-retention-full)

Full backup retention count/time.

When a full backup expires, all differential and incremental backups associated with the full backup will also expire. When the option is not defined a warning will be issued. If indefinite retention is desired then set the option to the max value.

```
allowed: [1, 9999999]
example: --repo1-retention-full=2
```

Deprecated Name: retention-full

Full Retention Type Option (--repo-retention-full-type)

Retention type for full backups.

Determines whether the repo-retention-full setting represents a time period (days) or count of full backups to keep.

If set to time then full backups older than repo-retention-full will be removed from the repository if there is at least one other backup that is equal to or greater than the repo-retention-full setting. For example, if repo-retention-full is 30 (days) and there are 2 full backups: one 25 days old and one 35 days old, no full backups will be expired because expiring the 35 day old backup would leave only the 25 day old backup, which would violate the 30 day retention policy of having at least one backup 30 days old before an older one can be expired. Archived WAL older than the oldest full backup remaining will be automatically expired unless repo-retention-archive-type and repo-retention-archive are explicitly set.

If set to count then full backups that exceed repo-retention-full will be expired. For example, if repo-retention-full is 4 and a fifth full backup is completed, then the oldest full backup will be expired to keep the count at 4.

Note that a backup must be successfully completed before it will be considered for retention. For example, if repo-retention-full-type is count and repo-retention-full is 2, then there must be 3 complete full backups before the oldest will be expired.

```
default: count
example: --repo1-retention-full-type=time
```

Backup History Retention Option (--repo-retention-history)

Days of backup history manifests to retain.

A copy of the backup manifest is stored in the backup.history path when a backup completes. By default these files are never expired since they are useful for data mining, e.g. measuring backup and WAL growth over time.

Set repo-retention-history to define the number of days of backup history manifests to retain. Unexpired backups are always kept in the backup history. Specify repo-retention-history=0 to retain the backup history only for unexpired backups.

When a full backup history manifest is expired, all differential and incremental backup history manifests associated with the full backup also expire.

```
allowed: [0, 9999999]
example: --repo1-retention-history=365
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

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

```
example: --repo1-s3-bucket=pg-backup
```

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

```
example: --repo1-s3-endpoint=s3.amazonaws.com
```

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys
- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

```
default: shared
example: --repo1-s3-key-type=auto
```

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

`example: --repo1-s3-process-cmd=/usr/local/bin/get-credentials --repo1-s3-process-cmd=--role`

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

`example: --repo1-s3-region=us-east-1`

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

`default: n`
`example: --no-repo1-s3-requester-pays`

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

`example: --repo1-s3-role=authrole`

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

`default: s3`
`example: --repo1-s3-service=s3-outposts`

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

`default: sts.amazonaws.com`
`example: --repo1-s3-sts-host=sts.us-east-1.amazonaws.com`

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

```
default: host
example: --repo1-s3-uri-style=path
```

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}' ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

```
example: --repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf
```

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.
- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: --repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: --repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: --repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

```
example: --repo1-sftp-host-user=pg-backup
```

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

```
example: --repo1-sftp-known-host=/home/postgres/.ssh/known_hosts
```

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

```
example: --repo1-sftp-private-key-file=~/.ssh/id_ed25519
```

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

```
example: --repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub
```

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

```
example: --repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

```
example: --repo1-storage-ca-path=/etc/pki/tls/certs
```

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

```
example: --repo1-storage-host=127.0.0.1
```

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: --repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: --repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies by storage type, so when searching min/max values use "part size" for AWS S3, "chunk size" for GCS, and "block size" for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]

example: --repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo1-storage-verify-tls
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Repository Symlink Option (--repo-symlink)

Create symlinks within the repository.

Enable creation of the lastest and tablespace symlinks. These symlinks are most useful when using snapshots to do in-place recovery in the repository, which is an uncommon use case.

While this feature is likely not useful for the vast majority of users it remains on by default for legacy purposes. However, it may be useful to disable symlinks for Posix-like storage that does not support them.

```
default: y
example: --no-repo1-symlink
```

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs
- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: --repo1-type=cifs
```

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 (e.g. pgbackrest help backup) 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 (e.g. pgbackrest help backup type) then a full description of the option as it applies to the command will be displayed.

Command Options

Display Help Option (--help)

Display help.

Displays help even if the help command is not specified and overrides the --version option.

```
default: n
example: --help
```

Display Version Option (--version)

Display version.

Displays version even if the version or help command is not specified.

```
default: n
example: --version
```

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 and is kept stable unless a bug is found.

To speed up execution, limit the output to only progress information by specifying --detail-level=progress. Note that this skips all checks except for availability of the stanza.

Each stanza has a separate section and it is possible to limit output to a single stanza with the --stanza option. The stanza 'status' gives a brief indication of the stanza's health. If this is 'ok' then pgBackRest is functioning normally. If there are multiple repositories, then a status of 'mixed' indicates that the stanza is not in a healthy state on one or more of the repositories; in this case the state of the stanza will be detailed per repository. For cases in which an error on a repository occurred that is not one of the known error codes, then an error code of 'other' will be used and the full error details will be provided. The 'wal archive min/max' shows the minimum and maximum WAL currently stored in the archive and, in the case of multiple repositories, will be reported across all repositories unless the --repo option is set. Note that there may be gaps due to archive retention policies or other reasons.

The 'backup/expire running' and/or 'restore running' messages will appear beside the 'status' information if any of those commands are currently running on the host. Per-repo progress will be also reported in text output and a 'repo' array will be included in JSON output.

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

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

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

The 'database size' is the full uncompressed size of the database while 'database backup size' is the amount of data in the database to actually back up (these will be the same for full backups).

The 'repo' indicates in which repository this backup resides. The 'backup set size' includes all the files from this backup and any referenced backups in the repository that are required to restore the database from this backup while '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.

The 'backup reference total' summarizes the list of additional backups that are required to restore this backup. Use the --set option to display the complete reference list.

Command Options

Detail level Option (--detail-level)

Output detail level.

The following levels are supported:

- progress - Output only the current backup/expire progress. This level cannot be used with the --set option.
- full - Output full info.

```
default: full
example: --detail-level=progress
```

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
```

Set Option (--set)

Backup set to detail.

Details include a complete list of additional backups that are required to restore this backup, a list of databases (with OIDs) in the backup set (excluding template databases), tablespaces (with OIDs) with the destination where they will be restored by default, and symlinks with the destination where they will be restored when --link-all is specified.

```
example: --set=20150131-153358F_20150131-153401I
```

Type Option (--type)

Filter on backup type.

Filter the output using one of the following backup types:

- full - Output only full backups.
- diff - Output only differential backups.
- incr - Output only incremental backups.

`example: --type=full`

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

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

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

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: --compress-level-network=1
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
```

```
example: --io-timeout=120
```

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

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

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Stanza Option (--stanza)

Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one PostgreSQL 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
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

`example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL`

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

`example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256`

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

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

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

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors

- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

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

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

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

The following log levels are supported:

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

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

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

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

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: --log-subprocess
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

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

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

Repository Options

Set Repository Option (--repo)

Set repository.

Set the repository for a command to operate on.

For example, this option may be used to perform a restore from a specific repository, rather than letting pgBackRest choose.

```
allowed: [1, 256]
example: --repo=1
```

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

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

```
example: --repo1-azure-container=pg-backup
```

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: --repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: --repo1-azure-uri-style=path
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

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

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: --repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

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

```
example: --repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: --repo1-gcs-endpoint=localhost
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.
- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: --repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: --repo1-gcs-user-project=my-project
```

Repository Host Option (--repo-host)

Repository host when operating remotely.

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

```
example: --repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: --repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: --repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

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

```
default: pgbackrest
example: --repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

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

It may 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: --repo1-path=/backup/db/backrest
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

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

```
example: --repo1-s3-bucket=pg-backup
```

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

`example: --repo1-s3-endpoint=s3.amazonaws.com`

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys
- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

`default: shared`
`example: --repo1-s3-key-type=auto`

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

`example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7`

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

`example: --repo1-s3-process-cmd=/usr/local/bin/get-credentials --repo1-s3-process-cmd=--role`

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

`example: --repo1-s3-region=us-east-1`

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

```
default: n
example: --no-repo1-s3-requester-pays
```

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

```
example: --repo1-s3-role=authrole
```

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

```
default: s3
example: --repo1-s3-service=s3-outposts
```

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

```
default: sts.amazonaws.com
example: --repo1-s3-sts-host=sts.us-east-1.amazonaws.com
```

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

```
default: host
example: --repo1-s3-uri-style=path
```

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}' ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

```
example: --repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf
```

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.
- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: --repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: --repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: --repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

`example: --repo1-sftp-host-user=pg-backup`

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

`example: --repo1-sftp-known-host=/home/postgres/.ssh/known_hosts`

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

`example: --repo1-sftp-private-key-file=~/.ssh/id_ed25519`

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

`example: --repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub`

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt`

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-path=/etc/pki/tls/certs`

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

```
example: --repo1-storage-host=127.0.0.1
```

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: --repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: --repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies

by storage type, so when searching min/max values use “part size” for AWS S3, “chunk size” for GCS, and “block size” for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]

example: --repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo1-storage-verify-tls
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Target Time for Repository Option (--repo-target-time)

Target time for repository.

The target time defines the time that commands use to read a repository on versioned storage. This allows the command to read the repository as it was at a point-in-time in order to recover data that has been deleted or corrupted by user accident or malware.

Versioned storage is supported by S3, GCS, and Azure but is generally not enabled by default. In addition to enabling versioning, it may be useful to enable object locking for S3 and soft delete for GCS or Azure.

When the repo-target-time option is specified then the repo option must also be provided. It is likely that not all repository types will support versioning and in general it makes sense to target a single repository for recovery.

Note that comparisons to the storage timestamp are <= the timestamp provided and milliseconds are truncated from the timestamp when provided.

```
example: --repo-target-time=2024-08-08 12:12:12+00
```

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs
- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: --repo1-type=cifs
```

Repository Get Command (repo-get)

Similar to the unix cat command but works on any supported repository type. This command requires a fully qualified file name and is primarily for administration, investigation, and testing. It is not a required part of a normal pgBackRest setup.

If the repository is encrypted then repo-get will automatically decrypt the file. Files are not automatically decompressed but the output can be piped through the appropriate decompression command, e.g. gzip -d.

If more than one repository is configured, the command will default to the highest priority repository (e.g. repo1) unless the --repo option is specified.

Command Options

Ignore Missing Option (--ignore-missing)

Ignore missing source file.

Exit with 1 if the source file is missing but don't throw an error.

```
default: n
example: --ignore-missing
```

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

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

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

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: --compress-level-network=1
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

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.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

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

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Raw Data Option (--raw)

Do not transform data.

Do not transform (i.e, encrypt, decompress, etc.) data for the current command.

```
default: n
example: --raw
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL
```

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
```

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

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

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

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

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

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

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

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

The following log levels are supported:

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

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

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

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

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: --log-subprocess
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

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

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

Repository Options

Set Repository Option (--repo)

Set repository.

Set the repository for a command to operate on.

For example, this option may be used to perform a restore from a specific repository, rather than letting pgBackRest choose.

```
allowed: [1, 256]
example: --repo=1
```

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

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

```
example: --repo1-azure-container=pg-backup
```

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: --repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: --repo1-azure-uri-style=path
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

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

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: --repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

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

```
example: --repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: --repo1-gcs-endpoint=localhost
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.
- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: --repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: --repo1-gcs-user-project=my-project
```

Repository Host Option (--repo-host)

Repository host when operating remotely.

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

```
example: --repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: --repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: --repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

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

```
default: pgbackrest
example: --repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

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

It may 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: --repo1-path=/backup/db/backrest
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

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

```
example: --repo1-s3-bucket=pg-backup
```

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

```
example: --repo1-s3-endpoint=s3.amazonaws.com
```

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys
- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

```
default: shared
example: --repo1-s3-key-type=auto
```

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

```
example: --repo1-s3-process-cmd=/usr/local/bin/get-credentials --repo1-s3-process-cmd=--role
```

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

```
example: --repo1-s3-region=us-east-1
```

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

```
default: n
example: --no-repo1-s3-requester-pays
```

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

```
example: --repo1-s3-role=authrole
```

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

```
default: s3
example: --repo1-s3-service=s3-outposts
```

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

```
default: sts.amazonaws.com
example: --repo1-s3-sts-host=sts.us-east-1.amazonaws.com
```

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

```
default: host
example: --repo1-s3-uri-style=path
```

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}' ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

```
example: --repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf
```

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.
- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: --repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: --repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: --repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

```
example: --repo1-sftp-host-user=pg-backup
```

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

```
example: --repo1-sftp-known-host=/home/postgres/.ssh/known_hosts
```

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

```
example: --repo1-sftp-private-key-file=~/.ssh/id_ed25519
```

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

```
example: --repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub
```

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

```
example: --repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

```
example: --repo1-storage-ca-path=/etc/pki/tls/certs
```

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

```
example: --repo1-storage-host=127.0.0.1
```

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: --repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: --repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies by storage type, so when searching min/max values use "part size" for AWS S3, "chunk size" for GCS, and "block size" for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]

example: --repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo1-storage-verify-tls
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Target Time for Repository Option (--repo-target-time)

Target time for repository.

The target time defines the time that commands use to read a repository on versioned storage. This allows the command to read the repository as it was at a point-in-time in order to recover data that has been deleted or corrupted by user accident or malware.

Versioned storage is supported by S3, GCS, and Azure but is generally not enabled by default. In addition to enabling versioning, it may be useful to enable object locking for S3 and soft delete for GCS or Azure.

When the repo-target-time option is specified then the repo option must also be provided. It is likely that not all repository types will support versioning and in general it makes sense to target a single repository for recovery.

Note that comparisons to the storage timestamp are <= the timestamp provided and milliseconds are truncated from the timestamp when provided.

```
example: --repo-target-time=2024-08-08 12:12:12+00
```

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs
- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: --repo1-type=cifs
```

Repository List Command (repo-ls)

Similar to the unix ls command but works on any supported repository type. This command accepts a path, absolute or relative to the repository path defined by the --repo-path option, and is primarily for administration, investigation, and testing. It is not a required part of a normal pgBackRest setup.

The default text output prints one file name per line. JSON output is available by specifying --output=json.

If more than one repository is configured, the command will default to the highest priority repository (e.g. repo1) unless the --repo option is specified.

Command Options

Filter Output Option (--filter)

Filter output with a regular expression.

The filter is applied against the file/path names before they are output.

```
example: --filter="(F|D|I)$"
```

Output Option (--output)

Output format.

The following output types are supported:

- text - Simple list with one file/link/path name on each line.
- json - Detailed file/link/path information in JSON format.

In JSON format the available fields are:

- name - file/link/path name (and partial path when recursing).
- type - file, path, or link.
- size - size in bytes (files only).
- time - time last modified (files only).
- destination - link destination (links only).

```
default: text
example: --output=json
```

Recurse Subpaths Option (--recurse)

Include all subpaths in output.

All subpaths and their files will be included in the output.

```
default: n
example: --recurse
```

Sort Output Option (--sort)

Sort output ascending, descending, or none.

The following sort types are supported:

- asc - sort ascending.
- desc - sort descending.
- none - no sorting.

```
default: asc
example: --sort=desc
```

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

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

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

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: --compress-level-network=1
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

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.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

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

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL
```

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
```

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

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

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

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

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

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

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

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

The following log levels are supported:

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

- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

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

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

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

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: --log-subprocess
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

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

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

Repository Options

Set Repository Option (--repo)

Set repository.

Set the repository for a command to operate on.

For example, this option may be used to perform a restore from a specific repository, rather than letting pgBackRest choose.

```
allowed: [1, 256]
example: --repo=1
```

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

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

```
example: --repo1-azure-container=pg-backup
```

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: --repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: --repo1-azure-uri-style=path
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

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

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: --repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

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

```
example: --repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: --repo1-gcs-endpoint=localhost
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.
- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: --repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: --repo1-gcs-user-project=my-project
```

Repository Host Option (--repo-host)

Repository host when operating remotely.

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

```
example: --repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: --repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: --repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

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

```
default: pgbackrest
example: --repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

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

It may 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: --repo1-path=/backup/db/backrest
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

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

```
example: --repo1-s3-bucket=pg-backup
```

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

```
example: --repo1-s3-endpoint=s3.amazonaws.com
```

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys
- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

```
default: shared
example: --repo1-s3-key-type=auto
```

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

```
example: --repo1-s3-process-cmd=/usr/local/bin/get-credentials --repo1-s3-process-cmd=--role
```

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

```
example: --repo1-s3-region=us-east-1
```

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

```
default: n
example: --no-repo1-s3-requester-pays
```

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

```
example: --repo1-s3-role=authrole
```

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

```
default: s3
example: --repo1-s3-service=s3-outposts
```

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

```
default: sts.amazonaws.com
example: --repo1-s3-sts-host=sts.us-east-1.amazonaws.com
```

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

```
default: host
example: --repo1-s3-uri-style=path
```

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}' ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

```
example: --repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf
```

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.

- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: --repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: --repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: --repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

```
example: --repo1-sftp-host-user=pg-backup
```

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

```
example: --repo1-sftp-known-host=/home/postgres/.ssh/known_hosts
```

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

`example: --repo1-sftp-private-key-file=~/.ssh/id_ed25519`

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

`example: --repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub`

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt`

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-path=/etc/pki/tls/certs`

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

`example: --repo1-storage-host=127.0.0.1`

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: --repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: --repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies by storage type, so when searching min/max values use “part size” for AWS S3, “chunk size” for GCS, and “block size” for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]

example: --repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo1-storage-verify-tls
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Target Time for Repository Option (--repo-target-time)

Target time for repository.

The target time defines the time that commands use to read a repository on versioned storage. This allows the command to read the repository as it was at a point-in-time in order to recover data that has been deleted or corrupted by user accident or malware.

Versioned storage is supported by S3, GCS, and Azure but is generally not enabled by default. In addition to enabling versioning, it may be useful to enable object locking for S3 and soft delete for GCS or Azure.

When the repo-target-time option is specified then the repo option must also be provided. It is likely that not all repository types will support versioning and in general it makes sense to target a single repository for recovery.

Note that comparisons to the storage timestamp are <= the timestamp provided and milliseconds are truncated from the timestamp when provided.

```
example: --repo-target-time=2024-08-08 12:12:12+00
```

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs
- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: --repo1-type=cifs
```

Restore Command (restore)

The restore command automatically defaults to selecting the latest backup from the first repository where backups exist (see Quick Start - Restore a Backup). The order in which the repositories are checked is dictated by the pgbackrest.conf (e.g. repo1 will be checked before repo2). To select from a specific repository, the --repo option can be passed (e.g. --repo=1). The --set option can be passed if a backup other than the latest is desired.

When PITR of --type=time or --type=lsn is specified, then the target time or target lsn must be specified with the --target option. If a backup is not specified via the --set option, then the configured repositories will be checked, in order, for a backup that contains the requested time or lsn. If no matching backup is found, the latest backup from the first repository containing backups will be used for --type=time while no backup will be selected for --type=lsn. For other types of PITR, e.g. xid, the --set option must be provided if the target is prior to the latest backup. See Point-in-Time Recovery for more details and examples.

Replication slots are not included per recommendation of PostgreSQL. See Backing Up The Data Directory in the PostgreSQL documentation for more information.

Command Options

Archive Mode Option (--archive-mode)

Preserve or disable archiving on restored cluster.

This option allows archiving to be preserved or disabled on a restored cluster. This is useful when the cluster must be promoted to do some work but is not intended to become the new primary. In this case it is not a good idea to push WAL from the cluster into the repository.

The following modes are supported:

- off - disable archiving by setting archive_mode=off.
- preserve - preserve current archive_mode setting.

NOTE: This option is not available on PostgreSQL < 12.

```
default: preserve
example: --archive-mode=off
```

Exclude Database Option (--db-exclude)

Restore excluding the specified databases.

Databases excluded will be restored as sparse, zeroed files to save space but still allow PostgreSQL to perform recovery. After recovery, those databases will not be accessible but can be removed with the drop database command. The --db-exclude option can be passed multiple times to specify more than one database to exclude.

When used in combination with the --db-include option, --db-exclude will only apply to standard system databases (template0, template1, and postgres).

```
example: --db-exclude=db_main
```

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:

built-in databases (template0, template1, and postgres) are always restored unless specifically excluded.

The --db-include option can be passed multiple times to specify more than one database to include.

See Restore Selected Databases for additional information and caveats.

```
example: --db-include=db_main
```

Force Option (--force)

Force a restore.

By itself this option forces the PostgreSQL data and tablespace paths to be completely overwritten. In combination with --delta a timestamp/size delta will be performed instead of using checksums.

```
default: n
example: --force
```

Link All Option (--link-all)

Restore all symlinks.

By default symlinked directories and files are restored as normal directories and files in $PGDATA. This is because it may not be safe to restore symlinks to their original destinations on a system other than where the original backup was performed. This option restores all the symlinks just as they were on the original system where the backup was performed.

```
default: n
example: --link-all
```

Link Map Option (--link-map)

Modify the destination of a symlink.

Allows the destination file or path of a symlink to be changed on restore. This is useful for restoring to systems that have a different storage layout than the original system where the backup was generated.

```
example: --link-map=pg_xlog=/data/xlog
```

Recovery Option Option (--recovery-option)

Set an option in postgresql.auto.conf or recovery.conf.

See Server Configuration for details on postgresql.auto.conf or recovery.conf options (be sure to select your PostgreSQL version). This option can be used multiple times.

For PostgreSQL >= 12, options will be written into postgresql.auto.conf. For all other versions, options will be written into recovery.conf.

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 postgresql.auto.conf or recovery.conf file, it is always possible to edit/check postgresql.auto.conf or 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.

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.

Tablespaces are restored into their original locations by default. This behavior can be modified for each tablespace with the tablespace-map option, but it is sometimes 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.

CAUTION:

Tablespaces created after the backup started will not be mapped. Make a new backup after a tablespace is created if tablespace mapping is required.

```
example: --tablespace-map-all=/data/tablespace
```

Target Option (--target)

Recovery target.

Defines the recovery target when --type is lsn, name, xid, or time. If the target is prior to the latest backup and --type is not time or lsn, then use the --set option to specify the backup set.

```
example: --target=2015-01-30 14:15:11 EST
```

Target Action Option (--target-action)

Action to take when recovery target is reached.

When hot_standby=on, the default since PostgreSQL 10, this option consistently controls what the cluster does when the target is reached or there is no more WAL in the archive.

When hot_standby=off in PostgreSQL >= 12, pause acts like shutdown. When hot_standby=off in PostgreSQL < 12, pause acts like promote.

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. (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 lsn, 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.
- lsn - recover to the LSN (Log Sequence Number) specified in --target. This option is only supported on PostgreSQL >= 10.
- 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 postgresql.auto.conf or recovery.conf file.
- standby - add standby_mode=on to the postgresql.auto.conf or recovery.conf file so cluster will start in standby mode.
- none - no postgresql.auto.conf or recovery.conf file is written so PostgreSQL will attempt to achieve consistency using WAL segments present in pg_xlog/pg_wal. Provide the required WAL segments or use the archive-copy setting to include them with the backup.

WARNING:

Recovery type=none should be avoided because the timeline will not be incremented at the end of recovery. This can lead to, for example, PostgreSQL attempting to archive duplicate WAL, which will be rejected, and may cause the disk to fill up and result in a PostgreSQL panic. In addition, tools like pg_rewind may not work correctly or may cause corruption.

Note that the default restore type for offline backups is none since Point-in-Time-Recovery is not possible if wal_level=minimal. If type is set explicitly then it will be honored since Point-in-Time-Recovery is possible from offline backups as long as wal_level > minimal.

```
default: default
```

```
example: --type=xid
```

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: y
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

pgBackRest Command Option (--cmd)

pgBackRest command.

pgBackRest may generate a command string, e.g. when the restore command generates the restore_command setting. The command used to run the pgBackRest process will be used in this case unless the cmd option is provided.

CAUTION:

Wrapping the pgBackRest command may cause unpredictable behavior and is not recommended.

```
default: [path of executed pgbackrest binary]
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

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

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

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: --compress-level-network=1
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

Delta Option (--delta)

Restore or backup using checksums.

During a restore, by default the PostgreSQL data and tablespace directories are expected to be present but empty. This option performs a delta restore using checksums.

During a backup, this option will use checksums instead of the timestamps to determine if files will be copied.

```
default: n
example: --delta
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

Lock Path Option (--lock-path)

Path where lock files are stored.

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

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

Neutral Umask Option (--neutral-umask)

Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

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, 999]
example: --process-max=4
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

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

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Stanza Option (--stanza)

Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one PostgreSQL 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
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL
```

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
```

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

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

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

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

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

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

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

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

The following log levels are supported:

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

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

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

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

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: --log-subprocess
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

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

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

Maintainer Options

Force PostgreSQL Version Option (--pg-version-force)

Force PostgreSQL version.

The specified PostgreSQL version will be used instead of the version automatically detected by reading pg_control or WAL headers. This is mainly useful for PostgreSQL forks or development versions where those values are different from the release version. The version reported by PostgreSQL via 'server_version_num' must match the forced version.

WARNING:

Be cautious when using this option because pg_control and WAL headers will still be read with the expected format for the specified version, i.e. the format from the official open-source version of PostgreSQL. If the fork or development version changes the format of the fields that pgBackRest depends on it will lead to unexpected behavior. In general, this option will only work as expected if the fork adds all custom struct members *after* the standard PostgreSQL members.

```
example: --pg-version-force=15
```

Repository Options

Set Repository Option (--repo)

Set repository.

Set the repository for a command to operate on.

For example, this option may be used to perform a restore from a specific repository, rather than letting pgBackRest choose.

```
allowed: [1, 256]
example: --repo=1
```

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

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

```
example: --repo1-azure-container=pg-backup
```

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: --repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: --repo1-azure-uri-style=path
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

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

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: --repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

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

```
example: --repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: --repo1-gcs-endpoint=localhost
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.

- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: --repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: --repo1-gcs-user-project=my-project
```

Repository Host Option (--repo-host)

Repository host when operating remotely.

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

```
example: --repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: --repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: --repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

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

```
default: pgbackrest
example: --repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

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

It may 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: --repo1-path=/backup/db/backrest
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

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

```
example: --repo1-s3-bucket=pg-backup
```

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

```
example: --repo1-s3-endpoint=s3.amazonaws.com
```

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys
- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

```
default: shared
example: --repo1-s3-key-type=auto
```

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

`example: --repo1-s3-process-cmd=/usr/local/bin/get-credentials --repo1-s3-process-cmd=--role`

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

`example: --repo1-s3-region=us-east-1`

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

`default: n`
`example: --no-repo1-s3-requester-pays`

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

`example: --repo1-s3-role=authrole`

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

`default: s3`
`example: --repo1-s3-service=s3-outposts`

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

`default: sts.amazonaws.com`
`example: --repo1-s3-sts-host=sts.us-east-1.amazonaws.com`

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

```
default: host
example: --repo1-s3-uri-style=path
```

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}' ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

```
example: --repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf
```

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.
- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: --repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: --repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: --repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

```
example: --repo1-sftp-host-user=pg-backup
```

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

```
example: --repo1-sftp-known-host=/home/postgres/.ssh/known_hosts
```

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

```
example: --repo1-sftp-private-key-file=~/.ssh/id_ed25519
```

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

```
example: --repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub
```

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

```
example: --repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

```
example: --repo1-storage-ca-path=/etc/pki/tls/certs
```

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

```
example: --repo1-storage-host=127.0.0.1
```

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: --repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: --repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies by storage type, so when searching min/max values use "part size" for AWS S3, "chunk size" for GCS, and "block size" for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]

example: --repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo1-storage-verify-tls
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Target Time for Repository Option (--repo-target-time)

Target time for repository.

The target time defines the time that commands use to read a repository on versioned storage. This allows the command to read the repository as it was at a point-in-time in order to recover data that has been deleted or corrupted by user accident or malware.

Versioned storage is supported by S3, GCS, and Azure but is generally not enabled by default. In addition to enabling versioning, it may be useful to enable object locking for S3 and soft delete for GCS or Azure.

When the repo-target-time option is specified then the repo option must also be provided. It is likely that not all repository types will support versioning and in general it makes sense to target a single repository for recovery.

Note that comparisons to the storage timestamp are <= the timestamp provided and milliseconds are truncated from the timestamp when provided.

```
example: --repo-target-time=2024-08-08 12:12:12+00
```

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs
- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: --repo1-type=cifs
```

Stanza Options

PostgreSQL Path Option (--pg-path)

PostgreSQL data directory.

This should be the same as the data_directory reported by PostgreSQL. Even though this value can be read from various places, it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

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

```
example: --pg1-path=/data/db
```

Deprecated Name: db-path

Server Command (server)

The pgBackRest server allows access to remote hosts without using the SSH protocol.

Command Options

TLS Server Address Option (--tls-server-address)

TLS server address.

IP address the server will listen on for client requests.

```
default: localhost
example: --tls-server-address=*
```

TLS Server Authorized Clients Option (--tls-server-auth)

TLS server authorized clients.

Clients are authorized on the server by verifying their certificate and checking their certificate CN (Common Name) against a list on the server configured with the tls-server-auth option.

A client CN can be authorized for as many stanzas as needed by providing a comma-separated list to the tls-server-auth option or for all stanzas by specifying tls-server-auth=client-cn=*. Wildcards may not be specified for the client CN.

```
example: --tls-server-auth=client-cn=stanza1,stanza2
```

TLS Server Certificate Authorities Option (--tls-server-ca-file)

TLS server certificate authorities.

Checks that client certificates are signed by a trusted certificate authority.

```
example: --tls-server-ca-file=/path/to/server.ca
```

TLS Server Certificate Option (--tls-server-cert-file)

TLS server certificate file.

Sent to the client to show the server identity.

```
example: --tls-server-cert-file=/path/to/server.crt
```

TLS Server Key Option (--tls-server-key-file)

TLS server key file.

Proves server certificate was sent by the owner.

```
example: --tls-server-key-file=/path/to/server.key
```

TLS Server Port Option (--tls-server-port)

TLS server port.

Port the server will listen on for client requests.

```
default: 8432
allowed: [1, 65535]
example: --tls-server-port=8000
```

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

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

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

`example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL`

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

`example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256`

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

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

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

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)

- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

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

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

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

The following log levels are supported:

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

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

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

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

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

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

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

Server Ping Command (server-ping)

Ping a pgBackRest TLS server to ensure it is accepting connections. This serves as an aliveness check only since no authentication is attempted.

If no host is specified on the command-line then the tls-server-host option will be used.

Command Options

TLS Server Address Option (--tls-server-address)

TLS server address.

IP address the server will listen on for client requests.

```
default: localhost
example: --tls-server-address=*
```

TLS Server Port Option (--tls-server-port)

TLS server port.

Port the server will listen on for client requests.

```
default: 8432
allowed: [1, 65535]
example: --tls-server-port=8000
```

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL
```

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
```

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

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

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

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

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

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

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

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

The following log levels are supported:

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

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

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

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

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

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

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

Stanza Create Command (stanza-create)

The stanza-create command must be run after the stanza has been configured in pgbackrest.conf. If there is more than one repository configured, the stanza will be created on each. Stanzas that have already been created will be skipped so it is always safe to run stanza-create, even when a new repository has been configured.

See Create the Stanza for more information and an example.

Command Options

Online Option (--online)

Create on an online cluster.

Specifying --no-online prevents pgBackRest from connecting to PostgreSQL when creating the stanza.

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

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

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

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

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: --compress-level-network=1
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

Database Timeout Option (--db-timeout)

Database query timeout.

Sets the timeout, in seconds, for queries against the database. This includes the backup start/stop 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 start-fast=y and you know that the database cluster will not generate many WAL segments during the backup).

NOTE:

The db-timeout option must be less than the protocol-timeout option.

```
default: 30m
allowed: [100ms, 7d]
example: --db-timeout=600
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

Lock Path Option (--lock-path)

Path where lock files are stored.

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

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

Neutral Umask Option (--neutral-umask)

Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

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

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Stanza Option (--stanza)

Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one PostgreSQL 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
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL
```

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
```

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

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

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

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

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

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

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

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

The following log levels are supported:

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

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

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

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

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: --log-subprocess
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

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

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

Maintainer Options

Force PostgreSQL Version Option (--pg-version-force)

Force PostgreSQL version.

The specified PostgreSQL version will be used instead of the version automatically detected by reading pg_control or WAL headers. This is mainly useful for PostgreSQL forks or development versions where those values are different from the release version. The version reported by PostgreSQL via 'server_version_num' must match the forced version.

WARNING:

Be cautious when using this option because pg_control and WAL headers will still be read with the expected format for the specified version, i.e. the format from the official open-source version of PostgreSQL. If the fork or development version changes the format of the fields that pgBackRest depends on it will lead to unexpected behavior. In general, this option will only work as expected if the fork adds all custom struct members *after* the standard PostgreSQL members.

`example: --pg-version-force=15`

Repository Options

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

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

`example: --repo1-azure-container=pg-backup`

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: --repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: --repo1-azure-uri-style=path
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

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

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: --repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

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

```
example: --repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: --repo1-gcs-endpoint=localhost
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.
- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: --repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: --repo1-gcs-user-project=my-project
```

Repository Host Option (--repo-host)

Repository host when operating remotely.

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

```
example: --repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: --repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: --repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

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

```
default: pgbackrest
example: --repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

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

It may 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: --repo1-path=/backup/db/backrest
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

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

```
example: --repo1-s3-bucket=pg-backup
```

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

```
example: --repo1-s3-endpoint=s3.amazonaws.com
```

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys
- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

```
default: shared
example: --repo1-s3-key-type=auto
```

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

```
example: --repo1-s3-process-cmd=/usr/local/bin/get-credentials --repo1-s3-process-cmd=--role
```

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

```
example: --repo1-s3-region=us-east-1
```

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

```
default: n
example: --no-repo1-s3-requester-pays
```

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

```
example: --repo1-s3-role=authrole
```

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

```
default: s3
example: --repo1-s3-service=s3-outposts
```

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

```
default: sts.amazonaws.com
example: --repo1-s3-sts-host=sts.us-east-1.amazonaws.com
```

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

```
default: host
example: --repo1-s3-uri-style=path
```

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}' ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

```
example: --repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf
```

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.
- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: --repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: --repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: --repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

`example: --repo1-sftp-host-user=pg-backup`

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

`example: --repo1-sftp-known-host=/home/postgres/.ssh/known_hosts`

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

`example: --repo1-sftp-private-key-file=~/.ssh/id_ed25519`

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

`example: --repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub`

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt`

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-path=/etc/pki/tls/certs`

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

```
example: --repo1-storage-host=127.0.0.1
```

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: --repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: --repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies

by storage type, so when searching min/max values use “part size” for AWS S3, “chunk size” for GCS, and “block size” for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]

example: --repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo1-storage-verify-tls
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs
- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: --repo1-type=cifs
```

Stanza Options

PostgreSQL Database Option (--pg-database)

PostgreSQL database.

The database name used when connecting to PostgreSQL. The default is usually best but some installations may not contain this database.

Note that for legacy reasons the setting of the PGDATABASE environment variable will be ignored.

```
default: postgres
example: --pg1-database=backupdb
```

PostgreSQL Host Option (--pg-host)

PostgreSQL host for operating remotely.

Used for backups where the PostgreSQL host is different from the repository host.

```
example: --pg1-host=db.domain.com
```

Deprecated Name: db-host

PostgreSQL Host Certificate Authority File Option (--pg-host-ca-file)

PostgreSQL host certificate authority file.

Use a CA file other than the system default for connecting to the PostgreSQL host.

```
example: --pg1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

PostgreSQL Host Certificate Authority Path Option (--pg-host-ca-path)

PostgreSQL host certificate authority path.

Use a CA path other than the system default for connecting to the PostgreSQL host.

```
example: --pg1-host-ca-path=/etc/pki/tls/certs
```

PostgreSQL Host Certificate File Option (--pg-host-cert-file)

PostgreSQL host certificate file.

Sent to PostgreSQL host to prove client identity.

```
example: --pg1-host-cert-file=/path/to/client.crt
```

PostgreSQL Host Command Option (--pg-host-cmd)

PostgreSQL host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and PostgreSQL hosts. If not defined, the PostgreSQL host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --pg1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: db-cmd

PostgreSQL Host Configuration Option (--pg-host-config)

pgBackRest database host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --pg1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: db-config

PostgreSQL Host Configuration Include Path Option (--pg-host-config-include-path)

pgBackRest database host configuration include path.

Sets the location of the configuration include path on the PostgreSQL host. This is only required if the PostgreSQL host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --pg1-host-config-include-path=/conf/pgbackrest/conf.d
```

PostgreSQL Host Configuration Path Option (--pg-host-config-path)

pgBackRest database host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --pg1-host-config-path=/conf/pgbackrest
```

PostgreSQL Host Key File Option (--pg-host-key-file)

PostgreSQL host key file.

Proves client certificate was sent by owner.

```
example: --pg1-host-key-file=/path/to/client.key
```

PostgreSQL Host Port Option (--pg-host-port)

PostgreSQL host port when pg-host is set.

Use this option to specify a non-default port for the PostgreSQL host protocol.

NOTE:

When pg-host-type=ssh there is no default for pg-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on pg-host-type):
    tls - 8432

allowed: [0, 65535]
example: --pg1-host-port=25
```

Deprecated Name: db-ssh-port

PostgreSQL Host Protocol Type Option (--pg-host-type)

PostgreSQL host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --pg1-host-type=tls
```

PostgreSQL Host User Option (--pg-host-user)

PostgreSQL host logon user when pg-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: --pg1-host-user=db_owner
```

Deprecated Name: db-user

PostgreSQL Path Option (--pg-path)

PostgreSQL data directory.

This should be the same as the data_directory reported by PostgreSQL. Even though this value can be read from various places, it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

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

```
example: --pg1-path=/data/db
```

Deprecated Name: db-path

PostgreSQL Port Option (--pg-port)

PostgreSQL port.

Port that PostgreSQL is running on. This usually does not need to be specified as most PostgreSQL clusters run on the default port.

```
default: 5432
allowed: [0, 65535]
example: --pg1-port=6543
```

Deprecated Name: db-port

PostgreSQL Socket Path Option (--pg-socket-path)

PostgreSQL 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 is usually no need to specify this setting unless the socket directory was explicitly modified with the unix_socket_directories setting in postgresql.conf.

```
example: --pg1-socket-path=/var/run/postgresql
```

Deprecated Name: db-socket-path

PostgreSQL Database User Option (--pg-user)

PostgreSQL database user.

The database user name used when connecting to PostgreSQL. If not specified pgBackRest will connect with the local OS user or PGUSER.

```
example: --pg1-user=backupuser
```

Stanza Delete Command (stanza-delete)

The stanza-delete command removes data in the repository associated with a stanza.

WARNING:

Use this command with caution — it will permanently remove all backups and archives from the pgBackRest repository for the specified stanza.

To delete a stanza:

- Shut down the PostgreSQL cluster associated with the stanza (or use --force to override).
- Run the stop command on the host where the stanza-delete command will be run.
- Run the stanza-delete command.

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

A stanza may only be deleted from one repository at a time. To delete the stanza from multiple repositories, repeat the stanza-delete command for each repository while specifying the --repo option.

Command Options

Force Option (--force)

Force stanza delete.

If PostgreSQL is still running for the stanza, then this option can be used to force the stanza to be deleted from the repository.

```
default: n
example: --no-force
```

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

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

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

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: --compress-level-network=1
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

Database Timeout Option (--db-timeout)

Database query timeout.

Sets the timeout, in seconds, for queries against the database. This includes the backup start/stop 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 start-fast=y and you know that the database cluster will not generate many WAL segments during the backup).

NOTE:

The db-timeout option must be less than the protocol-timeout option.

```
default: 30m
allowed: [100ms, 7d]
example: --db-timeout=600
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

Lock Path Option (--lock-path)

Path where lock files are stored.

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

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

Neutral Umask Option (--neutral-umask)

Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

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

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Stanza Option (--stanza)

Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one PostgreSQL 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
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL
```

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

`example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256`

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

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

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

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

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

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

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

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

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors

- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

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

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

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

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: --log-subprocess
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

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

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

Maintainer Options

Force PostgreSQL Version Option (--pg-version-force)

Force PostgreSQL version.

The specified PostgreSQL version will be used instead of the version automatically detected by reading pg_control or WAL headers. This is mainly useful for PostgreSQL forks or development versions where those values are different from the release version. The version reported by PostgreSQL via ‘server_version_num‘ must match the forced version.

WARNING:

Be cautious when using this option because pg_control and WAL headers will still be read with the expected format for the specified version, i.e. the format from the official open-source version of PostgreSQL. If the fork or development version changes the format of the fields that pgBackRest depends on it will lead to unexpected behavior. In general, this option will only work as expected if the fork adds all custom struct members *after* the standard PostgreSQL members.

`example: --pg-version-force=15`

Repository Options

Set Repository Option (--repo)

Set repository.

Set the repository for a command to operate on.

For example, this option may be used to perform a restore from a specific repository, rather than letting pgBackRest choose.

```
allowed: [1, 256]
example: --repo=1
```

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

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

`example: --repo1-azure-container=pg-backup`

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: --repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: --repo1-azure-uri-style=path
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

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

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: --repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

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

```
example: --repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: --repo1-gcs-endpoint=localhost
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.
- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: --repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: --repo1-gcs-user-project=my-project
```

Repository Host Option (--repo-host)

Repository host when operating remotely.

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

```
example: --repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: --repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: --repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

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

```
default: pgbackrest
example: --repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

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

It may 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: --repo1-path=/backup/db/backrest
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

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

`example: --repo1-s3-bucket=pg-backup`

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

`example: --repo1-s3-endpoint=s3.amazonaws.com`

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys
- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

```
default: shared
example: --repo1-s3-key-type=auto
```

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

`example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7`

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

`example: --repo1-s3-process-cmd=/usr/local/bin/get-credentials --repo1-s3-process-cmd=--role`

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

```
example: --repo1-s3-region=us-east-1
```

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

```
default: n
example: --no-repo1-s3-requester-pays
```

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

```
example: --repo1-s3-role=authrole
```

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

```
default: s3
example: --repo1-s3-service=s3-outposts
```

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

```
default: sts.amazonaws.com
example: --repo1-s3-sts-host=sts.us-east-1.amazonaws.com
```

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

```
default: host
example: --repo1-s3-uri-style=path
```

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}' ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

```
example: --repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf
```

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.
- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: --repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: --repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: --repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

```
example: --repo1-sftp-host-user=pg-backup
```

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

```
example: --repo1-sftp-known-host=/home/postgres/.ssh/known_hosts
```

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

```
example: --repo1-sftp-private-key-file=~/.ssh/id_ed25519
```

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

```
example: --repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub
```

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

```
example: --repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

```
example: --repo1-storage-ca-path=/etc/pki/tls/certs
```

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

```
example: --repo1-storage-host=127.0.0.1
```

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: --repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: --repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies by storage type, so when searching min/max values use "part size" for AWS S3, "chunk size" for GCS, and "block size" for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]

example: --repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo1-storage-verify-tls
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs

- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: --repo1-type=cifs
```

Stanza Options

PostgreSQL Database Option (--pg-database)

PostgreSQL database.

The database name used when connecting to PostgreSQL. The default is usually best but some installations may not contain this database.

Note that for legacy reasons the setting of the PGDATABASE environment variable will be ignored.

```
default: postgres
example: --pg1-database=backupdb
```

PostgreSQL Host Option (--pg-host)

PostgreSQL host for operating remotely.

Used for backups where the PostgreSQL host is different from the repository host.

```
example: --pg1-host=db.domain.com
```

Deprecated Name: db-host

PostgreSQL Host Certificate Authority File Option (--pg-host-ca-file)

PostgreSQL host certificate authority file.

Use a CA file other than the system default for connecting to the PostgreSQL host.

```
example: --pg1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

PostgreSQL Host Certificate Authority Path Option (--pg-host-ca-path)

PostgreSQL host certificate authority path.

Use a CA path other than the system default for connecting to the PostgreSQL host.

```
example: --pg1-host-ca-path=/etc/pki/tls/certs
```

PostgreSQL Host Certificate File Option (--pg-host-cert-file)

PostgreSQL host certificate file.

Sent to PostgreSQL host to prove client identity.

```
example: --pg1-host-cert-file=/path/to/client.crt
```

PostgreSQL Host Command Option (--pg-host-cmd)

PostgreSQL host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and PostgreSQL hosts. If not defined, the PostgreSQL host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --pg1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: db-cmd

PostgreSQL Host Configuration Option (--pg-host-config)

pgBackRest database host configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --pg1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: db-config

PostgreSQL Host Configuration Include Path Option (--pg-host-config-include-path)

pgBackRest database host configuration include path.

Sets the location of the configuration include path on the PostgreSQL host. This is only required if the PostgreSQL host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --pg1-host-config-include-path=/conf/pgbackrest/conf.d
```

PostgreSQL Host Configuration Path Option (--pg-host-config-path)

pgBackRest database host configuration path.

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

```
default: CFGOPTDEF_CONFIG_PATH
example: --pg1-host-config-path=/conf/pgbackrest
```

PostgreSQL Host Key File Option (--pg-host-key-file)

PostgreSQL host key file.

Proves client certificate was sent by owner.

```
example: --pg1-host-key-file=/path/to/client.key
```

PostgreSQL Host Port Option (--pg-host-port)

PostgreSQL host port when pg-host is set.

Use this option to specify a non-default port for the PostgreSQL host protocol.

NOTE:

When pg-host-type=ssh there is no default for pg-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on pg-host-type):
    tls - 8432

allowed: [0, 65535]
example: --pg1-host-port=25
```

Deprecated Name: db-ssh-port

PostgreSQL Host Protocol Type Option (--pg-host-type)

PostgreSQL host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --pg1-host-type=tls
```

PostgreSQL Host User Option (--pg-host-user)

PostgreSQL host logon user when pg-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: --pg1-host-user=db_owner
```

Deprecated Name: db-user

PostgreSQL Path Option (--pg-path)

PostgreSQL data directory.

This should be the same as the data_directory reported by PostgreSQL. Even though this value can be read from various places, it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

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

```
example: --pg1-path=/data/db
```

Deprecated Name: db-path

PostgreSQL Port Option (--pg-port)

PostgreSQL port.

Port that PostgreSQL is running on. This usually does not need to be specified as most PostgreSQL clusters run on the default port.

```
default: 5432
allowed: [0, 65535]
example: --pg1-port=6543
```

Deprecated Name: db-port

PostgreSQL Socket Path Option (--pg-socket-path)

PostgreSQL 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 is usually no need to specify this setting unless the socket directory was explicitly modified with the unix_socket_directories setting in postgresql.conf.

```
example: --pg1-socket-path=/var/run/postgresql
```

Deprecated Name: db-socket-path

PostgreSQL Database User Option (--pg-user)

PostgreSQL database user.

The database user name used when connecting to PostgreSQL. If not specified pgBackRest will connect with the local OS user or PGUSER.

```
example: --pg1-user=backupuser
```

Stanza Upgrade Command (stanza-upgrade)

Immediately after upgrading PostgreSQL to a newer major version, the pg-path for all pgBackRest configurations must be set to the new database location and the stanza-upgrade command run. If there is more than one repository configured on the host, the stanza will be upgraded on each. If the database is offline use the --no-online option.

Command Options

Online Option (--online)

Update an online cluster.

Specifying --no-online prevents pgBackRest from connecting to PostgreSQL when upgrading the stanza.

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

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

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

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

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: --compress-level-network=1
```

Config Option (--config)

pgBackRest configuration file.

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

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

Database Timeout Option (--db-timeout)

Database query timeout.

Sets the timeout, in seconds, for queries against the database. This includes the backup start/stop 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 start-fast=y and you know that the database cluster will not generate many WAL segments during the backup).

NOTE:

The db-timeout option must be less than the protocol-timeout option.

```
default: 30m
allowed: [100ms, 7d]
example: --db-timeout=600
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

Lock Path Option (--lock-path)

Path where lock files are stored.

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

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

Neutral Umask Option (--neutral-umask)

Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

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

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Stanza Option (--stanza)

Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one PostgreSQL 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
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL
```

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

`example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256`

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

`default: warn`
`example: --log-level-console=error`

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

`default: info`
`example: --log-level-file=debug`

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors

- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: off
example: --log-level-stderr=error
```

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: --log-subprocess
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Maintainer Options

Force PostgreSQL Version Option (--pg-version-force)

Force PostgreSQL version.

The specified PostgreSQL version will be used instead of the version automatically detected by reading pg_control or WAL headers. This is mainly useful for PostgreSQL forks or development versions where those values are different from the release version. The version reported by PostgreSQL via ‘server_version_num‘ must match the forced version.

WARNING:

Be cautious when using this option because pg_control and WAL headers will still be read with the expected format for the specified version, i.e. the format from the official open-source version of PostgreSQL. If the fork or development version changes the format of the fields that pgBackRest depends on it will lead

to unexpected behavior. In general, this option will only work as expected if the fork adds all custom struct members *after* the standard PostgreSQL members.

`example: --pg-version-force=15`

Repository Options

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

pgBackRest repositories can be stored in the container root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo, so logs and other Azure-generated content can also be stored in the container.

`example: --repo1-azure-container=pg-backup`

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: --repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: --repo1-azure-uri-style=path
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: --repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo, so logs and other GCS-generated content can also be stored in the bucket.

```
example: --repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: --repo1-gcs-endpoint=localhost
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.
- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: --repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: --repo1-gcs-user-project=my-project
```

Repository Host Option (--repo-host)

Repository host when operating remotely.

When backing up and archiving to a locally mounted filesystem this setting is not required.

```
example: --repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: --repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

Sets the location of the configuration file on the repository host. This is only required if the repository host configuration file is in a different location than the local configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

Sets the location of the configuration path on the repository host. This is only required if the repository host configuration path is in a different location than the local configuration path.

```
default: CFGOPTDEF_CONFIG_PATH
example: --repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: --repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

Defines the user that will be used for operations on the repository host. Preferably this is not the postgres user but rather some other user like pgbackrest. If PostgreSQL runs on the repository host the postgres user can be placed in the pgbackrest group so it has read permissions on the repository without being able to damage the contents accidentally.

```
default: pgbackrest
example: --repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

The repository is where pgBackRest stores backups and archives WAL segments.

It may 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: --repo1-path=/backup/db/backrest
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo, so logs and other AWS generated content can also be stored in the bucket.

```
example: --repo1-s3-bucket=pg-backup
```

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

```
example: --repo1-s3-endpoint=s3.amazonaws.com
```

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys
- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

```
default: shared
example: --repo1-s3-key-type=auto
```

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

```
example: --repo1-s3-process-cmd=/usr/local/bin/get-credentials --repo1-s3-process-cmd=--role
```

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

```
example: --repo1-s3-region=us-east-1
```

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

```
default: n
example: --no-repo1-s3-requester-pays
```

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

```
example: --repo1-s3-role=authrole
```

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

```
default: s3
example: --repo1-s3-service=s3-outposts
```

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

```
default: sts.amazonaws.com
example: --repo1-s3-sts-host=sts.us-east-1.amazonaws.com
```

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

```
default: host
example: --repo1-s3-uri-style=path
```

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}'

ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

```
example: --repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf
```

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.
- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: --repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: --repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: --repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

```
example: --repo1-sftp-host-user=pg-backup
```

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

`example: --repo1-sftp-known-host=/home/postgres/.ssh/known_hosts`

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

`example: --repo1-sftp-private-key-file=~/.ssh/id_ed25519`

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

`example: --repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub`

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt`

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-path=/etc/pki/tls/certs`

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

```
example: --repo1-storage-host=127.0.0.1
```

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: --repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: --repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies by storage type, so when searching min/max values use “part size” for AWS S3, “chunk size” for GCS, and “block size” for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]

example: --repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo1-storage-verify-tls
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs
- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: --repo1-type=cifs
```

Stanza Options

PostgreSQL Database Option (--pg-database)

PostgreSQL database.

The database name used when connecting to PostgreSQL. The default is usually best but some installations may not contain this database.

Note that for legacy reasons the setting of the PGDATABASE environment variable will be ignored.

```
default: postgres
example: --pg1-database=backupdb
```

PostgreSQL Host Option (--pg-host)

PostgreSQL host for operating remotely.

Used for backups where the PostgreSQL host is different from the repository host.

```
example: --pg1-host=db.domain.com
```

Deprecated Name: db-host

PostgreSQL Host Certificate Authority File Option (--pg-host-ca-file)

PostgreSQL host certificate authority file.

Use a CA file other than the system default for connecting to the PostgreSQL host.

```
example: --pg1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

PostgreSQL Host Certificate Authority Path Option (--pg-host-ca-path)

PostgreSQL host certificate authority path.

Use a CA path other than the system default for connecting to the PostgreSQL host.

```
example: --pg1-host-ca-path=/etc/pki/tls/certs
```

PostgreSQL Host Certificate File Option (--pg-host-cert-file)

PostgreSQL host certificate file.

Sent to PostgreSQL host to prove client identity.

```
example: --pg1-host-cert-file=/path/to/client.crt
```

PostgreSQL Host Command Option (--pg-host-cmd)

PostgreSQL host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and PostgreSQL hosts. If not defined, the PostgreSQL host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --pg1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: db-cmd

PostgreSQL Host Configuration Option (--pg-host-config)

pgBackRest database host configuration file.

Sets the location of the configuration file on the PostgreSQL host. This is only required if the PostgreSQL host configuration file is in a different location than the local configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --pg1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: db-config

PostgreSQL Host Configuration Include Path Option (--pg-host-config-include-path)

pgBackRest database host configuration include path.

Sets the location of the configuration include path on the PostgreSQL host. This is only required if the PostgreSQL host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --pg1-host-config-include-path=/conf/pgbackrest/conf.d
```

PostgreSQL Host Configuration Path Option (--pg-host-config-path)

pgBackRest database host configuration path.

Sets the location of the configuration path on the PostgreSQL host. This is only required if the PostgreSQL host configuration path is in a different location than the local configuration path.

```
default: CFGOPTDEF_CONFIG_PATH
example: --pg1-host-config-path=/conf/pgbackrest
```

PostgreSQL Host Key File Option (--pg-host-key-file)

PostgreSQL host key file.

Proves client certificate was sent by owner.

```
example: --pg1-host-key-file=/path/to/client.key
```

PostgreSQL Host Port Option (--pg-host-port)

PostgreSQL host port when pg-host is set.

Use this option to specify a non-default port for the PostgreSQL host protocol.

NOTE:

When pg-host-type=ssh there is no default for pg-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on pg-host-type):
    tls - 8432
```

```
allowed: [0, 65535]
example: --pg1-host-port=25
```

Deprecated Name: db-ssh-port

PostgreSQL Host Protocol Type Option (--pg-host-type)

PostgreSQL host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --pg1-host-type=tls
```

PostgreSQL Host User Option (--pg-host-user)

PostgreSQL host logon user when pg-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: --pg1-host-user=db_owner
```

Deprecated Name: db-user

PostgreSQL Path Option (--pg-path)

PostgreSQL data directory.

This should be the same as the data_directory reported by PostgreSQL. Even though this value can be read from various places, it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

The pg-path option is tested against the value reported by PostgreSQL on every online backup so it should always be current.

```
example: --pg1-path=/data/db
```

Deprecated Name: db-path

PostgreSQL Port Option (--pg-port)

PostgreSQL port.

Port that PostgreSQL is running on. This usually does not need to be specified as most PostgreSQL clusters run on the default port.

```
default: 5432
allowed: [0, 65535]
example: --pg1-port=6543
```

Deprecated Name: db-port

PostgreSQL Socket Path Option (--pg-socket-path)

PostgreSQL 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 is usually no need to specify this setting unless the socket directory was explicitly modified with the unix_socket_directories setting in postgresql.conf.

```
example: --pg1-socket-path=/var/run/postgresql
```

Deprecated Name: db-socket-path

PostgreSQL Database User Option (--pg-user)

PostgreSQL database user.

The database user name used when connecting to PostgreSQL. If not specified pgBackRest will connect with the local OS user or PGUSER.

```
example: --pg1-user=backupuser
```

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. See Starting and Stopping for more information and examples.

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Config Option (--config)

pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

Lock Path Option (--lock-path)

Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: --lock-path=/backup/db/lock
```

Neutral Umask Option (--neutral-umask)

Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

Stanza Option (--stanza)

Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one PostgreSQL database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

```
example: --stanza=main
```

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: --log-level-console=error
```

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors

- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: --log-level-file=debug
```

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: off
example: --log-level-stderr=error
```

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

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. See Starting and Stopping for more information and examples.

Command Options

Force Option (--force)

Force all pgBackRest processes to stop.

This option will send TERM signals to all running pgBackRest processes to effect a graceful but immediate shutdown. Note that this will also shutdown processes that were initiated on another system but have remotes running on the current system. For instance, if a backup was started on the backup server then running stop --force on the database server will shutdown the backup process on the backup server.

```
default: n
example: --force
```

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Config Option (--config)

pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

Lock Path Option (--lock-path)

Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: --lock-path=/backup/db/lock
```

Neutral Umask Option (--neutral-umask)

Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

Stanza Option (--stanza)

Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one PostgreSQL database cluster and therefore one stanza, whereas backup servers will have a stanza for every database cluster that needs to be backed up.

It is tempting to name the stanza after the primary cluster but a better name describes the databases contained in the cluster. Because the stanza name will be used for the primary and all replicas it is more appropriate to choose a name that describes the actual function of the cluster, such as app or dw, rather than the local cluster name, such as main or prod.

`example: --stanza=main`

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

`default: warn`
`example: --log-level-console=error`

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

`default: info`
`example: --log-level-file=debug`

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: off
example: --log-level-stderr=error
```

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Verify Command (verify)

Verify determines if the backups and archives in a repository are valid.

Command Options

Output Option (--output)

Output type.

The following output types are supported:

- none - No verify output.
- text - Output verify information to stdout.

```
default: none
example: --output=text
```

Set Option (--set)

Backup set to verify.

Verify all database and archive files associated with the specified backup set.

```
example: --set=20150131-153358F_20150131-153401I
```

Verbose Option (--verbose)

Verbose output.

Verbose defaults to false, providing a minimal response with important information about errors in the repository. Specifying true provides more information about what was successfully verified.

```
default: n
example: --verbose
```

General Options

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: --allow-root
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: --buffer-size=2MiB
```

pgBackRest Command Option (--cmd)

pgBackRest command.

pgBackRest may generate a command string, e.g. when the restore command generates the restore_command setting. The command used to run the pgBackRest process will be used in this case unless the cmd option is provided.

CAUTION:

Wrapping the pgBackRest command may cause unpredictable behavior and is not recommended.

```
default: [path of executed pgbackrest binary]
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

Use a specific SSH client command when an alternate is desired or the ssh command is not in $PATH.

```
default: ssh
example: --cmd-ssh=/usr/bin/ssh
```

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: --compress-level-network=1
```

Config Option (--config)

pgBackRest configuration file.

Use this option to specify a different configuration file than the default.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --config=/conf/pgbackrest/pgbackrest.conf
```

Config Include Path Option (--config-include-path)

Path to additional pgBackRest configuration files.

Configuration files existing in the specified location with extension .conf will be concatenated with the pgBackRest configuration file, resulting in one configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --config-include-path=/conf/pgbackrest/conf.d
```

Config Path Option (--config-path)

Base path of pgBackRest configuration files.

This setting is used to override the default base path setting for the --config and --config-include-path options unless they are explicitly set on the command-line.

For example, passing only --config-path=/conf/pgbackrest results in the --config default being set to /conf/pgbackrest/pgbackrest.conf and the --config-include-path default being set to /conf/pgbackrest/conf.d.

```
default: CFGOPTDEF_CONFIG_PATH
example: --config-path=/conf/pgbackrest
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
example: --io-timeout=120
```

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.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: --priority=19
```

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, 999]
example: --process-max=4
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

Sets the timeout, in seconds, that the local or remote process will wait for a new message to be received on the protocol layer. This prevents processes from waiting indefinitely for a message.

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: --protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

Stanza Option (--stanza)

Defines the stanza.

A stanza is the configuration for a PostgreSQL database cluster that defines where it is located, how it will be backed up, archiving options, etc. Most db servers will only have one PostgreSQL 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
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: --tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: --tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: --tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: --tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL
```

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

`example: --tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256`

Log Options

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

`default: warn`
`example: --log-level-console=error`

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

`default: info`
`example: --log-level-file=debug`

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors

- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: off
example: --log-level-stderr=error
```

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

```
default: /var/log/pgbackrest
example: --log-path=/backup/db/log
```

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: --log-subprocess
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: --no-log-timestamp
```

Maintainer Options

Force PostgreSQL Version Option (--pg-version-force)

Force PostgreSQL version.

The specified PostgreSQL version will be used instead of the version automatically detected by reading pg_control or WAL headers. This is mainly useful for PostgreSQL forks or development versions where those values are different from the release version. The version reported by PostgreSQL via 'server_version_num' must match the forced version.

WARNING:

Be cautious when using this option because pg_control and WAL headers will still be read with the expected format for the specified version, i.e. the format from the official open-source version of PostgreSQL. If the fork or development version changes the format of the fields that pgBackRest depends on it will lead

to unexpected behavior. In general, this option will only work as expected if the fork adds all custom struct members *after* the standard PostgreSQL members.

`example: --pg-version-force=15`

Repository Options

Set Repository Option (--repo)

Set repository.

Set the repository for a command to operate on.

For example, this option may be used to perform a restore from a specific repository, rather than letting pgBackRest choose.

```
allowed: [1, 256]
example: --repo=1
```

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

pgBackRest repositories can be stored in the container root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo, so logs and other Azure-generated content can also be stored in the container.

`example: --repo1-azure-container=pg-backup`

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: --repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: --repo1-azure-uri-style=path
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: --repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo, so logs and other GCS-generated content can also be stored in the bucket.

```
example: --repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: --repo1-gcs-endpoint=localhost
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.
- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: --repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: --repo1-gcs-user-project=my-project
```

Repository Host Option (--repo-host)

Repository host when operating remotely.

When backing up and archiving to a locally mounted filesystem this setting is not required.

```
example: --repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: --repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: --repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

Sets the location of the configuration file on the repository host. This is only required if the repository host configuration file is in a different location than the local configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: --repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: --repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

Sets the location of the configuration path on the repository host. This is only required if the repository host configuration path is in a different location than the local configuration path.

```
default: CFGOPTDEF_CONFIG_PATH
example: --repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: --repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: --repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

Defines the user that will be used for operations on the repository host. Preferably this is not the postgres user but rather some other user like pgbackrest. If PostgreSQL runs on the repository host the postgres user can be placed in the pgbackrest group so it has read permissions on the repository without being able to damage the contents accidentally.

```
default: pgbackrest
example: --repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

The repository is where pgBackRest stores backups and archives WAL segments.

It may 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: --repo1-path=/backup/db/backrest
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo, so logs and other AWS generated content can also be stored in the bucket.

```
example: --repo1-s3-bucket=pg-backup
```

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

```
example: --repo1-s3-endpoint=s3.amazonaws.com
```

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys
- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

```
default: shared
example: --repo1-s3-key-type=auto
```

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

```
example: --repo1-s3-process-cmd=/usr/local/bin/get-credentials --repo1-s3-process-cmd=--role
```

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

```
example: --repo1-s3-region=us-east-1
```

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

```
default: n
example: --no-repo1-s3-requester-pays
```

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

```
example: --repo1-s3-role=authrole
```

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

```
default: s3
example: --repo1-s3-service=s3-outposts
```

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

```
default: sts.amazonaws.com
example: --repo1-s3-sts-host=sts.us-east-1.amazonaws.com
```

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

```
default: host
example: --repo1-s3-uri-style=path
```

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}' ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

```
example: --repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf
```

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.
- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: --repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: --repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: --repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

`example: --repo1-sftp-host-user=pg-backup`

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

`example: --repo1-sftp-known-host=/home/postgres/.ssh/known_hosts`

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

`example: --repo1-sftp-private-key-file=~/.ssh/id_ed25519`

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

`example: --repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub`

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt`

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

`example: --repo1-storage-ca-path=/etc/pki/tls/certs`

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

```
example: --repo1-storage-host=127.0.0.1
```

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: --repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: --repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies

by storage type, so when searching min/max values use “part size” for AWS S3, “chunk size” for GCS, and “block size” for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]

example: --repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: --no-repo1-storage-verify-tls
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Target Time for Repository Option (--repo-target-time)

Target time for repository.

The target time defines the time that commands use to read a repository on versioned storage. This allows the command to read the repository as it was at a point-in-time in order to recover data that has been deleted or corrupted by user accident or malware.

Versioned storage is supported by S3, GCS, and Azure but is generally not enabled by default. In addition to enabling versioning, it may be useful to enable object locking for S3 and soft delete for GCS or Azure.

When the repo-target-time option is specified then the repo option must also be provided. It is likely that not all repository types will support versioning and in general it makes sense to target a single repository for recovery.

Note that comparisons to the storage timestamp are <= the timestamp provided and milliseconds are truncated from the timestamp when provided.

```
example: --repo-target-time=2024-08-08 12:12:12+00
```

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs
- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: --repo1-type=cifs
```

Version Command (version)

Displays installed pgBackRest version.

Command Options

Output Option (--output)

Output type.

The following output types are supported:

- text - Display the installed pgBackRest version as text.
- num - Display the installed pgBackRest version as an integer.

```
default: text
example: --output=num
```

Metrics
Table of Contents
Introduction
Archive Options
Asynchronous Archiving Option (--archive-async)
Maximum Archive Get Queue Size Option (--archive-get-queue-max)
Retry Missing WAL Segment Option (--archive-missing-retry)
Archive Push Batch Size Option (--archive-push-batch-size)
Maximum Archive Push Queue Size Option (--archive-push-queue-max)
Archive Timeout Option (--archive-timeout)
Backup Options
Backup Annotation Option (--annotation)
Check Archive Option (--archive-check)
Copy Archive Option (--archive-copy)
Check Archive Mode Option (--archive-mode-check)
Backup from Standby Option (--backup-standby)
Page Checksums Option (--checksum-page)
Path/File Exclusions Option (--exclude)
Expire Auto Option (--expire-auto)
Manifest Save Threshold Option (--manifest-save-threshold)
Resume Option (--resume)
Start Fast Option (--start-fast)
General Options
Allow Run as Root Option (--allow-root)
Buffer Size Option (--buffer-size)
pgBackRest Command Option (--cmd)
SSH Client Command Option (--cmd-ssh)
Compress Option (--compress)
Compress Level Option (--compress-level)
Network Compress Level Option (--compress-level-network)
Compress Type Option (--compress-type)

Database Timeout Option (--db-timeout)

Delta Option (--delta)

I/O Timeout Option (--io-timeout)

Lock Path Option (--lock-path)

Neutral Umask Option (--neutral-umask)

Set Process Priority Option (--priority)

Process Maximum Option (--process-max)

Protocol Timeout Option (--protocol-timeout)

Keep Alive Option (--sck-keep-alive)

Spool Path Option (--spool-path)

Keep Alive Count Option (--tcp-keep-alive-count)

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep Alive Interval Option (--tcp-keep-alive-interval)

TLSv1.2 cipher suites Option (--tls-cipher-12)

TLSv1.3 cipher suites Option (--tls-cipher-13)

Log Options

Console Log Level Option (--log-level-console)

File Log Level Option (--log-level-file)

Std Error Log Level Option (--log-level-stderr)

Log Path Option (--log-path)

Log Subprocesses Option (--log-subprocess)

Log Timestamp Option (--log-timestamp)

Maintainer Options

Check WAL Headers Option (--archive-header-check)

Page Header Check Option (--page-header-check)

Force PostgreSQL Version Option (--pg-version-force)

Repository Options

Azure Repository Account Option (--repo-azure-account)

Azure Repository Container Option (--repo-azure-container)

Azure Repository Endpoint Option (--repo-azure-endpoint)

Azure Repository Key Option (--repo-azure-key)

Azure Repository Key Type Option (--repo-azure-key-type)

Azure Repository URI Style Option (--repo-azure-uri-style)

Block Incremental Backup Option (--repo-block)

Repository Bundles Option (--repo-bundle)

Repository Bundle Limit Option (--repo-bundle-limit)

Repository Bundle Size Option (--repo-bundle-size)

Repository Cipher Passphrase Option (--repo-cipher-pass)

Repository Cipher Type Option (--repo-cipher-type)

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS Repository Key Option (--repo-gcs-key)

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS Repository Project ID Option (--repo-gcs-user-project)

Repository Hardlink Option (--repo-hardlink)

Repository Host Option (--repo-host)

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository Host Certificate File Option (--repo-host-cert-file)

Repository Host Command Option (--repo-host-cmd)

Repository Host Configuration Option (--repo-host-config)

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

Repository Host Configuration Path Option (--repo-host-config-path)

Repository Host Key File Option (--repo-host-key-file)

Repository Host Port Option (--repo-host-port)

Repository Host Protocol Type Option (--repo-host-type)

Repository Host User Option (--repo-host-user)

Repository Path Option (--repo-path)

Archive Retention Option (--repo-retention-archive)

Archive Retention Type Option (--repo-retention-archive-type)

Differential Retention Option (--repo-retention-diff)

Full Retention Option (--repo-retention-full)

Full Retention Type Option (--repo-retention-full-type)

Backup History Retention Option (--repo-retention-history)

S3 Repository Bucket Option (--repo-s3-bucket)

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 Repository Access Key Option (--repo-s3-key)

S3 Repository Secret Access Key Option (--repo-s3-key-secret)

S3 Repository Key Type Option (--repo-s3-key-type)

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 Repository Region Option (--repo-s3-region)

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 Repository Role Option (--repo-s3-role)

S3 Repository Service Option (--repo-s3-service)

S3 Repository SSE Customer Key Option (--repo-s3-sse-customer-key)

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 Repository Security Token Option (--repo-s3-token)

S3 Repository URI Style Option (--repo-s3-uri-style)

SFTP Repository Host Option (--repo-sftp-host)

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP Repository Private Key Passphrase Option (--repo-sftp-private-key-passphrase)

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

Repository Storage CA File Option (--repo-storage-ca-file)

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository Storage Host Option (--repo-storage-host)

Repository Storage Port Option (--repo-storage-port)

Repository Storage Tag Option (--repo-storage-tag)

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository Symlink Option (--repo-symlink)

Target Time for Repository Option (--repo-target-time)

Repository Type Option (--repo-type)

Restore Options

Archive Mode Option (--archive-mode)

Exclude Database Option (--db-exclude)

Include Database Option (--db-include)

Link All Option (--link-all)

Link Map Option (--link-map)

Recovery Option Option (--recovery-option)

Tablespace Map Option (--tablespace-map)

Map All Tablespaces Option (--tablespace-map-all)

Server Options

TLS Server Address Option (--tls-server-address)

TLS Server Authorized Clients Option (--tls-server-auth)

TLS Server Certificate Authorities Option (--tls-server-ca-file)

TLS Server Certificate Option (--tls-server-cert-file)

TLS Server Key Option (--tls-server-key-file)

TLS Server Port Option (--tls-server-port)

Stanza Options

PostgreSQL Database Option (--pg-database)

PostgreSQL Host Option (--pg-host)

PostgreSQL Host Certificate Authority File Option (--pg-host-ca-file)

PostgreSQL Host Certificate Authority Path Option (--pg-host-ca-path)

PostgreSQL Host Certificate File Option (--pg-host-cert-file)

PostgreSQL Host Command Option (--pg-host-cmd)

PostgreSQL Host Configuration Option (--pg-host-config)

PostgreSQL Host Configuration Include Path Option (--pg-host-config-include-path)

PostgreSQL Host Configuration Path Option (--pg-host-config-path)

PostgreSQL Host Key File Option (--pg-host-key-file)

PostgreSQL Host Port Option (--pg-host-port)

PostgreSQL Host Protocol Type Option (--pg-host-type)

PostgreSQL Host User Option (--pg-host-user)

PostgreSQL Path Option (--pg-path)

PostgreSQL Port Option (--pg-port)

PostgreSQL Socket Path Option (--pg-socket-path)

PostgreSQL Database User Option (--pg-user)

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/pgbackrest.conf. If no file exists in that location then the old default of /etc/pgbackrest.conf will be checked.

The following option types are used:

String: A text string, commonly an identifier, password, etc.

Command line example: --stanza=demo
Configuration file example: repo1-cipher-pass=zWaf6XtpjIVZC5444yXB...

Path: Used to uniquely identify a location in a directory structure. Paths must begin with /, double // is not allowed, and no ending / is expected.

Command line example: --repo1-path=/var/lib/pgbackrest
Configuration file example: repo1-path=/var/lib/pgbackrest

Boolean: Enables or disables the option. Only y/n are valid argument values.

Command line examples: --start-fast, --no-start-fast, --start-fast=y, --start-fast=n
Configuration file examples: start-fast=y, start-fast=n

Integer: Used for ports, retention/retry counts, parallel processes allowed, etc.

Command line example: --compress-level=3
Configuration file example: pg1-port=5432

Size: Used for buffer sizes, disk usage, etc. Size can be specified in bytes (default) or KiB, MiB, GiB, TiB, or PiB where the multiplier is a power of 1024. For example, the case-insensitive value 5GiB (or 5GB, 5g) can be used instead of 5368709120. Fractional values such as 2.5GiB are not allowed, use 2560MiB instead.

Command line example: --archive-get-queue-max=1GiB
Configuration file example: buffer-size=2MiB

Time: Time in seconds.

Command line example: --io-timeout=90
Configuration file example: db-timeout=600

List: Option may be provided multiple times.

Command line example: --db-exclude=db1 --db-exclude=db2 --db-exclude=db5
Configuration file example, each on its own line: db-exclude=db1 db-exclude=db2 db-exclude=db5

Key/Value: Option may be provided multiple times in the form key=value.

Command line example: --tablespace-map=ts_01=/db/ts_01 --tablespace-map=ts_02=/db/ts_02
Configuration file example, each on its own line: tablespace-map=ts_01=/db/ts_01 tablespace-map=ts_02=/db/ts_02

Archive Options

The archive section defines options for the archive-push and archive-get commands.

Asynchronous Archiving Option (--archive-async)

Push/get WAL segments asynchronously.

Enables asynchronous operation for the archive-push and archive-get commands.

Asynchronous operation is more efficient because it can reuse connections and take advantage of parallelism. See the spool-path, archive-get-queue-max, and archive-push-queue-max options for more information.

```
default: n
example: archive-async=y
```

Maximum Archive Get Queue Size Option (--archive-get-queue-max)

Maximum size of the pgBackRest archive-get queue.

Specifies the maximum size of the archive-get queue when archive-async is enabled. The queue is stored in the spool-path and is used to speed providing WAL to PostgreSQL.

```
default: 128MiB
allowed: [0B, 4PiB]
```

```
example: archive-get-queue-max=1GiB
```

Retry Missing WAL Segment Option (--archive-missing-retry)

Retry missing WAL segment

Retry a WAL segment that was previously reported as missing by the archive-get command when in asynchronous mode. This prevents notifications in the spool path from a prior restore from being used and possibly causing a recovery failure if consistency has not been reached.

Disabling this option allows PostgreSQL to more reliably recognize when the end of the WAL in the archive has been reached, which permits it to switch over to streaming from the primary. With retries enabled, a steady stream of WAL being archived will cause PostgreSQL to continue getting WAL from the archive rather than switch to streaming.

When disabling this option it is important to ensure that the spool path for the stanza is empty. The restore command does this automatically if the spool path is configured at restore time. Otherwise, it is up to the user to ensure the spool path is empty.

```
default: y
example: archive-missing-retry=n
```

Archive Push Batch Size Option (--archive-push-batch-size)

Maximum amount of WAL to push per asynchronous run.

In asynchronous mode the archive-push process pushes all the WAL segments that are ready in a single run. Since archive-push-queue-max is only checked at the start of each run, a run that processes a very large number of segments can let the queue grow well beyond the limit before it is rechecked.

This option limits the amount of WAL processed per run so the process exits and is spawned again by the next archive-push, which rechecks the queue. Lower values recheck the queue more often at the cost of spawning the asynchronous process more frequently. The value is rounded down to a whole number of WAL segments but at least one segment is always processed.

```
default: 16GiB
allowed: [1MiB, 4PiB]
example: archive-push-batch-size=1GiB
```

Maximum Archive Push Queue Size Option (--archive-push-queue-max)

Maximum size of the PostgreSQL archive queue.

After the limit is reached, the following will happen:

- pgBackRest will notify PostgreSQL that the WAL was successfully archived, then **DROP IT**.
- A warning will be output to the PostgreSQL log.

If this occurs then the archive log stream will be interrupted and PITR will not be possible past that point. A new backup will be required to regain full restore capability.

In asynchronous mode the entire queue will be dropped to prevent spurts of WAL getting through before the queue limit is exceeded again.

In asynchronous mode this limit is only checked at the start of each archive-push run, so the queue can grow beyond it within a single run. Reduce archive-push-batch-size to check the queue more frequently.

The purpose of this feature is to prevent the log volume from filling up at which point PostgreSQL will stop completely. Better to lose the backup than have PostgreSQL go down.

```
allowed: [0B, 4PiB]
example: archive-push-queue-max=1TiB
```

Deprecated Name: archive-queue-max

Archive Timeout Option (--archive-timeout)

Archive timeout.

Set maximum time, in seconds, to wait for each WAL segment to reach the pgBackRest archive repository. The timeout applies to the check and backup commands when waiting for WAL segments required for backup consistency to be archived.

```
default: 1m
allowed: [100ms, 1d]
example: archive-timeout=30
```

Backup Options

The backup section defines settings related to backup.

Backup Annotation Option (--annotation)

Annotate backup with user-defined key/value pairs.

Users can attach informative key/value pairs to the backup. This option may be used multiple times to attach multiple annotations.

Annotations are output by the info command text output when a backup is specified with --set and always appear in the JSON output.

```
example: annotation=source="Sunday backup for website database"
```

Check Archive Option (--archive-check)

Check that WAL segments are in the archive before backup completes.

Checks that all WAL segments required to make the backup consistent are present in the WAL archive. It's a good idea to leave this as the default unless you are using another method for archiving.

This option must be enabled if archive-copy is enabled.

```
default: y
example: archive-check=n
```

Copy Archive Option (--archive-copy)

Copy WAL segments needed for consistency to the backup.

This slightly paranoid option protects against corruption in the WAL segment archive by storing the WAL segments required for consistency directly in the backup. WAL segments are still stored in the archive so this option will use additional space.

It is best if the archive-push and backup commands have the same compress-type (e.g. lz4) when using this option. Otherwise, the WAL segments will need to be recompressed with the compress-type used by the backup, which can be fairly expensive depending on how much WAL was generated during the backup.

On restore, the WAL segments will be present in pg_xlog/pg_wal and PostgreSQL will use them in preference to calling the restore_command.

The archive-check option must be enabled if archive-copy is enabled.

```
default: n
example: archive-copy=y
```

Check Archive Mode Option (--archive-mode-check)

Check the PostgreSQL archive_mode setting.

Enabled by default, this option disallows PostgreSQL archive_mode=always.

WAL segments pushed from a standby server might be logically the same as WAL segments pushed from the primary but have different checksums. Disabling archiving from multiple sources is recommended to avoid conflicts.

CAUTION:

If this option is disabled then it is critical to ensure that only one archiver is writing to the repository via the archive-push command.

```
default: y
example: archive-mode-check=n
```

Backup from Standby Option (--backup-standby)

Backup from the standby cluster.

Enable backup from standby to reduce load on the primary cluster. This option requires that both the primary and standby hosts be configured.

The following modes are supported:

- y - Standby is required for backup.
- prefer - Backup from standby if available otherwise backup from primary.
- n - Backup from primary only.

```
default: n
example: backup-standby=y
```

Page Checksums Option (--checksum-page)

Validate data page checksums.

Directs pgBackRest to validate all data page checksums while backing up a cluster. This option is automatically enabled when data page 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
```

Path/File Exclusions Option (--exclude)

Exclude paths/files from the backup.

All exclusions are relative to $PGDATA. If the exclusion ends with / then only files in the specified directory will be excluded, e.g. --exclude=junk/ will exclude all files in the $PGDATA/junk directory but include the directory itself. If the exclusion does not end with / then the file may match the exclusion exactly or match with / appended to the exclusion, e.g. --exclude=junk will exclude the $PGDATA/junk directory and all the files it contains.

Be careful using this feature -- it is very easy to exclude something critical that will make the backup inconsistent. Be sure to test your restores!

All excluded files will be logged at info level along with the exclusion rule. Be sure to audit the list of excluded files to ensure nothing unexpected is being excluded.

NOTE:

Exclusions are not honored on delta restores. Any files/directories that were excluded by the backup will be *removed* on delta restore.

This option should not be used to exclude PostgreSQL logs from a backup. Logs can be moved out of the PGDATA directory using the PostgreSQL log_directory setting, which has the benefit of allowing logs to be preserved after a restore.

Multiple exclusions may be specified on the command-line or in a configuration file.

```
example: exclude=junk/
```

Expire Auto Option (--expire-auto)

Automatically run the expire command after a successful backup.

The setting is enabled by default. Use caution when disabling this option as doing so will result in retaining all backups and archives indefinitely, which could cause your repository to run out of space. The expire command will need to be run regularly to prevent this from happening.

When expire is run automatically after a successful backup it uses the configuration of the backup command, so options set only in an expire command section (e.g. [global:expire]) are not applied. To apply expire-specific configuration, disable this option and run the expire command separately.

```
default: y
example: expire-auto=y
```

Manifest Save Threshold Option (--manifest-save-threshold)

Manifest save threshold during backup.

Defines how often the manifest will be saved during a backup. 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: 1GiB
allowed: [1B, 1TiB]
example: manifest-save-threshold=8GiB
```

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 the backup start function) so the backup begins immediately. Otherwise the backup will start after the next regular checkpoint.

```
default: n
example: start-fast=y
```

General Options

The general section defines options that are common for many commands.

Allow Run as Root Option (--allow-root)

Allow the command to run as the root user.

By default only the restore command may be run as the root user since it is designed to carefully manage file ownership. Running other commands as root risks creating files (e.g. in the repository) that are owned by root and therefore inaccessible to the PostgreSQL user, causing later commands to fail.

Enable this option to run a command as root anyway. However, it is far better to run pgBackRest as the user that owns the repository and PostgreSQL cluster.

```
default: n
example: allow-root=y
```

Buffer Size Option (--buffer-size)

Buffer size for I/O operations.

Buffer size used for copy, compress, encrypt, and other operations. The number of buffers used depends on options and each operation may use additional memory, e.g. gz compression may use an additional 256KiB of memory.

Allowed values are 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB, 2MiB, 4MiB, 8MiB, and 16MiB.

```
default: 1MiB
example: buffer-size=2MiB
```

pgBackRest Command Option (--cmd)

pgBackRest command.

pgBackRest may generate a command string, e.g. when the restore command generates the restore_command setting. The command used to run the pgBackRest process will be used in this case unless the cmd option is provided.

CAUTION:

Wrapping the pgBackRest command may cause unpredictable behavior and is not recommended.

```
default: [path of executed pgbackrest binary]
example: cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

SSH Client Command Option (--cmd-ssh)

SSH client command.

Use a specific SSH client command when an alternate is desired or the ssh command is not in $PATH.

```
default: ssh
example: cmd-ssh=/usr/bin/ssh
```

Compress Option (--compress)

Use file compression.

Backup files are compatible with command-line compression tools.

This option is now deprecated. The compress-type option should be used instead.

```
default: y
example: compress=n
```

Compress Level Option (--compress-level)

File compression level.

Sets the level to be used for file compression when compress-type does not equal none or compress=y (deprecated).

```
default (depending on compress-type):
    bz2 - 9
    gz - 6
    lz4 - 1
    zst - 3

allow range (depending on compress-type):
    bz2 - [1, 9]
    gz - [-1, 9]
    lz4 - [-5, 12]
    zst - [-7, 22]

example: compress-level=9
```

Network Compress Level Option (--compress-level-network)

Network compression level.

Sets the network compression level when compress-type=none and the command is not run on the same host as the repository. Compression is used to reduce network traffic. When compress-type does not equal none the compress-level-network setting is ignored and compress-level is used instead so that the file is only compressed once.

```
default: 1
allowed: [-5, 12]
example: compress-level-network=1
```

Compress Type Option (--compress-type)

File compression type.

The following compression types are supported:

- none - no compression
- bz2 - bzip2 compression format
- gz - gzip compression format
- lz4 - lz4 compression format (not available on all platforms)
- zst - Zstandard compression format (not available on all platforms)

```
default: gz
example: compress-type=none
```

Database Timeout Option (--db-timeout)

Database query timeout.

Sets the timeout, in seconds, for queries against the database. This includes the backup start/stop 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 start-fast=y and you know that the database cluster will not generate many WAL segments during the backup).

NOTE:

The db-timeout option must be less than the protocol-timeout option.

```
default: 30m
allowed: [100ms, 7d]
example: db-timeout=600
```

Delta Option (--delta)

Restore or backup using checksums.

During a restore, by default the PostgreSQL data and tablespace directories are expected to be present but empty. This option performs a delta restore using checksums.

During a backup, this option will use checksums instead of the timestamps to determine if files will be copied.

```
default: n
example: delta=y
```

I/O Timeout Option (--io-timeout)

I/O timeout.

Timeout, in seconds, used for connections and read/write operations.

Note that the entire read/write operation does not need to complete within this timeout but *some* progress must be made, even if it is only a single byte.

```
default: 1m
allowed: [100ms, 1h]
```

```
example: io-timeout=120
```

Lock Path Option (--lock-path)

Path where lock files are stored.

The lock path provides a location for pgBackRest to create lock files to prevent conflicting operations from being run concurrently.

```
default: /tmp/pgbackrest
example: lock-path=/backup/db/lock
```

Neutral Umask Option (--neutral-umask)

Use a neutral umask.

Sets the umask to 0000 so modes in the repository are created in a sensible way. The default directory mode is 0750 and default file mode is 0640.

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
```

Set Process Priority Option (--priority)

Set process priority.

Defines how much priority (i.e. niceness) will be given to the process by the kernel scheduler. Positive values decrease priority and negative values increase priority. In most case processes do not have permission to increase their priority.

```
allowed: [-20, 19]
example: priority=19
```

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, 999]
example: process-max=4
```

Protocol Timeout Option (--protocol-timeout)

Protocol timeout.

Sets the timeout, in seconds, that the local or remote process will wait for a new message to be received on the protocol layer. This prevents processes from waiting indefinitely for a message.

NOTE:

The protocol-timeout option must be greater than the db-timeout option.

```
default: 31m
allowed: [100ms, 7d]
example: protocol-timeout=630
```

Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: sck-keep-alive=n
```

Spool Path Option (--spool-path)

Path where transient data is stored.

This path is used to store data for the asynchronous archive-push and archive-get command.

The asynchronous archive-push command writes acknowledgements into the spool path when it has successfully stored WAL in the archive (and errors on failure) so the foreground process can quickly notify PostgreSQL. Acknowledgement files are very small (zero on success and a few hundred bytes on error).

The asynchronous archive-get command queues WAL in the spool path so it can be provided very quickly when PostgreSQL requests it. Moving files to PostgreSQL is most efficient when the spool path is on the same filesystem as pg_xlog/pg_wal. However, it is not recommended to place the spool path *within* the pg_xlog/pg_wal directory as this may cause issues for PostgreSQL utilities such as pg_rewind.

The data stored in the spool path is not strictly temporary since it can and should survive a reboot. However, loss of the data in the spool path is not a problem. pgBackRest will simply recheck each WAL segment to ensure it is safely archived for archive-push and rebuild the queue for archive-get.

The spool path is intended to be located on a local Posix-compatible filesystem, not a remote filesystem such as NFS or CIFS.

```
default: /var/spool/pgbackrest
example: spool-path=/backup/db/spool
```

Keep Alive Count Option (--tcp-keep-alive-count)

Keep-alive count.

Specifies the number of TCP keep-alive messages that can be lost before the connection is considered dead.

This option is available on systems that support the TCP_KEEPCNT socket option.

```
allowed: [1, 32]
example: tcp-keep-alive-count=3
```

Keep Alive Idle Option (--tcp-keep-alive-idle)

Keep-alive idle time.

Specifies the amount of time (in seconds) with no network activity after which the operating system should send a TCP keep-alive message.

This option is available on systems that support the TCP_KEEPIDLE socket option.

```
allowed: [1, 3600]
example: tcp-keep-alive-idle=60
```

Keep Alive Interval Option (--tcp-keep-alive-interval)

Keep-alive interval time.

Specifies the amount of time (in seconds) after which a TCP keep-alive message that has not been acknowledged should be retransmitted.

This option is available on systems that support the TCP_KEEPINTVL socket option.

```
allowed: [1, 900]
example: tcp-keep-alive-interval=30
```

TLSv1.2 cipher suites Option (--tls-cipher-12)

Allowed TLSv1.2 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. The example is reasonable choice unless you have specific security requirements. If unset (the default), the default of the underlying OpenSSL library applies.

```
example: tls-cipher-12=HIGH:MEDIUM:+3DES:!aNULL
```

TLSv1.3 cipher suites Option (--tls-cipher-13)

Allowed TLSv1.3 cipher suites.

All TLS connections between the pgBackRest client and server are encrypted. By default, connections to objects stores (e.g. S3) are also encrypted.

NOTE:

The absolute minimum security level for any transport connection is TLSv1.2.

The accepted cipher suites can be adjusted if need arises. If unset (the default), the default of the underlying OpenSSL library applies.

`example: tls-cipher-13=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256`

Log Options

The log section defines logging-related settings.

CAUTION:

Trace-level logging may expose secrets such as keys and passwords. Use with caution!

Console Log Level Option (--log-level-console)

Level for console logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: warn
example: log-level-console=error
```

File Log Level Option (--log-level-file)

Level for file logging.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: info
example: log-level-file=debug
```

Std Error Log Level Option (--log-level-stderr)

Level for stderr logging.

Specifies which log levels will output to stderr rather than stdout (specified by log-level-console). The timestamp and process will not be output to stderr.

The following log levels are supported:

- off - No logging at all (not recommended)
- error - Log only errors
- warn - Log warnings and errors
- info - Log info, warnings, and errors
- detail - Log detail, info, warnings, and errors
- debug - Log debug, detail, info, warnings, and errors
- trace - Log trace (very verbose debugging), debug, info, warnings, and errors

```
default: off
example: log-level-stderr=error
```

Log Path Option (--log-path)

Path where log files are stored.

The log path provides a location for pgBackRest to store log files. Note that if log-level-file=off then no log path is required.

```
default: /var/log/pgbackrest
example: log-path=/backup/db/log
```

Log Subprocesses Option (--log-subprocess)

Enable logging in subprocesses.

Enable file logging for any subprocesses created by this process using the log level specified by log-level-file.

```
default: n
example: log-subprocess=y
```

Log Timestamp Option (--log-timestamp)

Enable timestamp in logging.

Enables the timestamp in console and file logging. This option is disabled in special situations such as generating documentation.

```
default: y
example: log-timestamp=n
```

Maintainer Options

Maintainer options are intended to support PostgreSQL forks. The proper settings should be determined by the fork maintainer and then communicated to users of the fork.

WARNING:

Improper use of these options may lead to unexpected behavior or data corruption.

It is the responsibility of the fork maintainer to test pgBackRest with the required options. pgBackRest does not guarantee compatibility with any fork.

Check WAL Headers Option (--archive-header-check)

Check PostgreSQL version/id in WAL headers.

Enabled by default, this option checks the WAL header against the PostgreSQL version and system identifier to ensure that the WAL is being copied to the correct stanza. This is in addition to checking pg_control against the stanza and verifying that WAL is being copied from the same PostgreSQL data directory where pg_control is located.

Therefore, disabling this check is fairly safe but should only be done when needed, e.g. if the WAL is encrypted.

```
default: y
example: archive-header-check=n
```

Page Header Check Option (--page-header-check)

Check PostgreSQL page headers.

Enabled by default, this option adds page header checks.

Disabling this option should be avoided except when necessary, e.g. if pages are encrypted.

```
default: y
example: page-header-check=n
```

Force PostgreSQL Version Option (--pg-version-force)

Force PostgreSQL version.

The specified PostgreSQL version will be used instead of the version automatically detected by reading pg_control or WAL headers. This is mainly useful for PostgreSQL forks or development versions where those values are different from the release version. The version reported by PostgreSQL via 'server_version_num' must match the forced version.

WARNING:

Be cautious when using this option because pg_control and WAL headers will still be read with the expected format for the specified version, i.e. the format from the official open-source version of PostgreSQL. If the fork or development version changes the format of the fields that pgBackRest depends on it will lead to unexpected behavior. In general, this option will only work as expected if the fork adds all custom struct members *after* the standard PostgreSQL members.

```
example: pg-version-force=15
```

Repository Options

The repository section defines options used to configure the repository.

Indexing: All repo- options are indexed to allow for configuring multiple repositories. For example, a single repository is configured with the repo1-path, repo1-host, etc. options. If there is more than one repository configured and the --repo option is not specified for a command, the repositories will be acted upon in highest priority order (e.g. repo1 then repo2).

The repo-retention-* options define how long backups will be retained. Expiration only occurs when the count of complete backups exceeds the allowed retention. In other words, if repo1-retention-full-type is set to count (default) and repo1-retention-full is set to 2, then there must be 3 complete backups before the oldest will be expired. If repo1-retention-full-type is set to time then repo1-retention-full represents days so there must be at least that many days worth of full backups before expiration can occur. Make sure you always have enough space for retention + 1 backups.

Azure Repository Account Option (--repo-azure-account)

Azure repository account.

Azure account used to store the repository.

`example: repo1-azure-account=pg-backup`

Azure Repository Container Option (--repo-azure-container)

Azure repository container.

Azure container used to store the repository.

pgBackRest repositories can be stored in the container root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo, so logs and other Azure-generated content can also be stored in the container.

`example: repo1-azure-container=pg-backup`

Azure Repository Endpoint Option (--repo-azure-endpoint)

Azure repository endpoint.

Endpoint used to connect to the blob service. The default is generally correct unless using Azure Government.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

```
default: blob.core.windows.net
example: repo1-azure-endpoint=blob.core.usgovcloudapi.net
```

Azure Repository Key Option (--repo-azure-key)

Azure repository key.

A shared key or shared access signature depending on the repo-azure-key-type option.

```
example: repo1-azure-key=T+9+aov82qNhrcXSNGZCzm9mjd4d75/oxxOr6r1JVpgTLA==
```

Azure Repository Key Type Option (--repo-azure-key-type)

Azure repository key type.

The following types are supported for authorization:

- shared - Shared key
- sas - Shared access signature
- auto - Automatically authorize using Azure managed identities

```
default: shared
example: repo1-azure-key-type=sas
```

Azure Repository URI Style Option (--repo-azure-uri-style)

Azure URI Style.

The following URI styles are supported:

- host - Connect to account.endpoint host.
- path - Connect to endpoint host and prepend account to URIs.

```
default: host
example: repo1-azure-uri-style=path
```

Block Incremental Backup Option (--repo-block)

Enable block incremental backup.

Block incremental allows for more granular backups by splitting files into blocks that can be backed up independently. This saves space in the repository and can improve delta restore performance because individual blocks can be fetched without reading the entire file from the repository.

NOTE:

The repo-bundle option must be enabled before repo-block can be enabled.

The block size for a file is determined based on the file size and age. Generally, older/larger files will get larger block sizes. If a file is old enough, it will not be backed up using block incremental.

Block incremental is most efficient when enabled for all backup types, including full. This makes the full a bit larger but subsequent differential and incremental backups can make use of the block maps generated by the full backup to save space.

```
default: n
example: repo1-block=y
```

Repository Bundles Option (--repo-bundle)

Bundle files in repository.

Bundle (combine) smaller files to reduce the total number of files written to the repository. Writing fewer files is generally more efficient, especially on object stores such as S3. In addition, zero-length files are not stored (except in the manifest), which saves time and space.

```
default: n
example: repo1-bundle=y
```

Repository Bundle Limit Option (--repo-bundle-limit)

Limit for file bundles.

Size limit for files that will be included in bundles. Files larger than this size will be stored separately.

Bundled files cannot be reused when a backup is resumed, so this option controls the files that can be resumed, i.e. higher values result in fewer resumable files.

```
default: 2MiB
allowed: [8KiB, 1PiB]
example: repo1-bundle-limit=10MiB
```

Repository Bundle Size Option (--repo-bundle-size)

Target size for file bundles.

Defines the target size for files that will be added to a single bundle. The uncompressed bundle size may be as large as repo-bundle-size + repo-bundle-limit, so do not set this option to the maximum size that your file system allows.

In general, it is not a good idea to set this option too high because retries will need to redo the entire bundle.

```
default: 20MiB
allowed: [1MiB, 1PiB]
example: repo1-bundle-size=10MiB
```

Repository Cipher Passphrase Option (--repo-cipher-pass)

Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

NOTE:

When run without the stanza option the info command reads encryption settings only from the global section. If encryption settings are configured per stanza, run the info command with the stanza option to read an encrypted stanza.

```
example: repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

Repository Cipher Type Option (--repo-cipher-type)

Cipher used to encrypt the repository.

The following cipher types are supported:

- none - The repository is not encrypted
- aes-256-cbc - Advanced Encryption Standard with 256 bit key length

Note that encryption is always performed client-side even if the repository type (e.g. S3) supports encryption.

```
default: none
example: repo1-cipher-type=aes-256-cbc
```

GCS Repository Bucket Option (--repo-gcs-bucket)

GCS repository bucket.

GCS bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo, so logs and other GCS-generated content can also be stored in the bucket.

```
example: repo1-gcs-bucket=/pg-backup
```

GCS Repository Endpoint Option (--repo-gcs-endpoint)

GCS repository endpoint.

Endpoint used to connect to the storage service. May be updated to use a local GCS server or alternate endpoint.

```
default: storage.googleapis.com
example: repo1-gcs-endpoint=localhost
```

GCS Repository Key Option (--repo-gcs-key)

GCS repository key.

A token or service key file depending on the repo-gcs-key-type option.

```
example: repo1-gcs-key=/etc/pgbackrest/gcs-key.json
```

GCS Repository Key Type Option (--repo-gcs-key-type)

GCS repository key type.

The following types are supported for authorization:

- auto - Authorize using the instance service account.
- service - Service account from locally stored key.
- token - For local testing, e.g. fakegcs.

When repo-gcs-key-type=service the credentials will be reloaded when the authentication token is renewed.

```
default: service
example: repo1-gcs-key-type=auto
```

GCS Repository Project ID Option (--repo-gcs-user-project)

GCS project ID.

GCS project ID used to determine request billing.

```
example: repo1-gcs-user-project=my-project
```

Repository Hardlink Option (--repo-hardlink)

Hardlink files between backups in the repository.

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 at the file-system level. Be careful, though, because modifying files that are hard-linked can affect all the backups in the set.

```
default: n
example: repo1-hardlink=y
```

Deprecated Name: hardlink

Repository Host Option (--repo-host)

Repository host when operating remotely.

When backing up and archiving to a locally mounted filesystem this setting is not required.

```
example: repo1-host=repo1.domain.com
```

Deprecated Name: backup-host

Repository Host Certificate Authority File Option (--repo-host-ca-file)

Repository host certificate authority file.

Use a CA file other than the system default for connecting to the repository host.

```
example: repo1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

Repository Host Certificate Authority Path Option (--repo-host-ca-path)

Repository host certificate authority path.

Use a CA path other than the system default for connecting to the repository host.

```
example: repo1-host-ca-path=/etc/pki/tls/certs
```

Repository Host Certificate File Option (--repo-host-cert-file)

Repository host certificate file.

Sent to repository host to prove client identity.

```
example: repo1-host-cert-file=/path/to/client.crt
```

Repository Host Command Option (--repo-host-cmd)

Repository host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and repository hosts. If not defined, the repository host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

Repository Host Configuration Option (--repo-host-config)

pgBackRest repository host configuration file.

Sets the location of the configuration file on the repository host. This is only required if the repository host configuration file is in a different location than the local configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: repo1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: backup-config

Repository Host Configuration Include Path Option (--repo-host-config-include-path)

pgBackRest repository host configuration include path.

Sets the location of the configuration include path on the repository host. This is only required if the repository host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: repo1-host-config-include-path=/conf/pgbackrest/conf.d
```

Repository Host Configuration Path Option (--repo-host-config-path)

pgBackRest repository host configuration path.

Sets the location of the configuration path on the repository host. This is only required if the repository host configuration path is in a different location than the local configuration path.

```
default: CFGOPTDEF_CONFIG_PATH
example: repo1-host-config-path=/conf/pgbackrest
```

Repository Host Key File Option (--repo-host-key-file)

Repository host key file.

Proves client certificate was sent by owner.

```
example: repo1-host-key-file=/path/to/client.key
```

Repository Host Port Option (--repo-host-port)

Repository host port when repo-host is set.

Use this option to specify a non-default port for the repository host protocol.

NOTE:

When repo-host-type=ssh there is no default for repo-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on repo-host-type):
    tls - 8432

allowed: [0, 65535]
example: repo1-host-port=25
```

Deprecated Name: backup-ssh-port

Repository Host Protocol Type Option (--repo-host-type)

Repository host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: repo1-host-type=tls
```

Repository Host User Option (--repo-host-user)

Repository host user when repo-host is set.

Defines the user that will be used for operations on the repository host. Preferably this is not the postgres user but rather some other user like pgbackrest. If PostgreSQL runs on the repository host the postgres user can be placed in the pgbackrest group so it has read permissions on the repository without being able to damage the contents accidentally.

```
default: pgbackrest
example: repo1-host-user=repo-user
```

Deprecated Name: backup-user

Repository Path Option (--repo-path)

Path where backups and archive are stored.

The repository is where pgBackRest stores backups and archives WAL segments.

It may 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: repo1-path=/backup/db/backrest
```

Archive Retention Option (--repo-retention-archive)

Number of backups worth of continuous WAL to retain.

NOTE:

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 and repo-retention-full-type is count (default), then the archive to expire will default to the repo-retention-full (or repo-retention-diff) value corresponding to the repo-retention-archive-type if set to full (or diff). This will ensure that WAL is only expired for backups that are already expired. If repo-retention-full-type is time, then this value will default to removing archives that are earlier than the oldest full backup retained after satisfying the repo-retention-full setting.

This option must be set if repo-retention-archive-type is set to incr. If disk space is at a premium, then this setting, in conjunction with repo-retention-archive-type, can be used to aggressively expire WAL segments. However, doing so negates the ability to perform PITR from the backups with expired WAL and is therefore **not** recommended.

```
allowed: [1, 9999999]
example: repo1-retention-archive=2
```

Deprecated Name: retention-archive

Archive Retention Type Option (--repo-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 repo-retention-archive. If set to diff (differential) pgBackRest will keep archive logs for the number of full and differential backups defined by repo-retention-archive, meaning if the last backup taken was a full backup, it will be counted as a differential for the purpose of repo-retention. If set to incr (incremental) pgBackRest will keep archive logs for the number of full, differential, and incremental backups defined by repo-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: repo1-retention-archive-type=diff
```

Deprecated Name: retention-archive-type

Differential Retention Option (--repo-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.

Note that full backups are included in the count of differential backups for the purpose of expiration. This slightly reduces the number of differential backups that need to be retained in most cases.

```
allowed: [1, 9999999]
example: repo1-retention-diff=3
```

Deprecated Name: retention-diff

Full Retention Option (--repo-retention-full)

Full backup retention count/time.

When a full backup expires, all differential and incremental backups associated with the full backup will also expire. When the option is not defined a warning will be issued. If indefinite retention is desired then set the option to the max value.

```
allowed: [1, 9999999]
example: repo1-retention-full=2
```

Deprecated Name: retention-full

Full Retention Type Option (--repo-retention-full-type)

Retention type for full backups.

Determines whether the repo-retention-full setting represents a time period (days) or count of full backups to keep.

If set to time then full backups older than repo-retention-full will be removed from the repository if there is at least one other backup that is equal to or greater than the repo-retention-full setting. For example, if repo-retention-full is 30 (days) and there are 2 full backups: one 25 days old and one 35 days old, no full backups will be expired because expiring the 35 day old backup would leave only the 25 day old backup, which would violate the 30 day retention policy of having at least one backup 30 days old before an older one can be expired. Archived WAL older than the oldest full backup remaining will be automatically expired unless repo-retention-archive-type and repo-retention-archive are explicitly set.

If set to count then full backups that exceed repo-retention-full will be expired. For example, if repo-retention-full is 4 and a fifth full backup is completed, then the oldest full backup will be expired to keep the count at 4.

Note that a backup must be successfully completed before it will be considered for retention. For example, if repo-retention-full-type is count and repo-retention-full is 2, then there must be 3 complete full backups before the oldest will be expired.

```
default: count
example: repo1-retention-full-type=time
```

Backup History Retention Option (--repo-retention-history)

Days of backup history manifests to retain.

A copy of the backup manifest is stored in the backup.history path when a backup completes. By default these files are never expired since they are useful for data mining, e.g. measuring backup and WAL growth over time.

Set repo-retention-history to define the number of days of backup history manifests to retain. Unexpired backups are always kept in the backup history. Specify repo-retention-history=0 to retain the backup history only for unexpired backups.

When a full backup history manifest is expired, all differential and incremental backup history manifests associated with the full backup also expire.

```
allowed: [0, 9999999]
example: repo1-retention-history=365
```

S3 Repository Bucket Option (--repo-s3-bucket)

S3 repository bucket.

S3 bucket used to store the repository.

pgBackRest repositories can be stored in the bucket root by setting repo-path=/ but it is usually best to specify a prefix, such as /repo, so logs and other AWS generated content can also be stored in the bucket.

```
example: repo1-s3-bucket=pg-backup
```

S3 Repository Endpoint Option (--repo-s3-endpoint)

S3 repository endpoint.

The AWS endpoint should be valid for the selected region.

For custom/test configurations the repo-storage-ca-file, repo-storage-ca-path, repo-storage-host, repo-storage-port, and repo-storage-verify-tls options may be useful.

```
example: repo1-s3-endpoint=s3.amazonaws.com
```

S3 Repository Access Key Option (--repo-s3-key)

S3 repository access key.

AWS key used to access this bucket.

```
example: repo1-s3-key=AKIAIOSFODNN7EXAMPLE
```

S3 Repository Secret Access Key Option (--repo-s3-key-secret)

S3 repository secret access key.

AWS secret key used to access this bucket.

```
example: repo1-s3-key-secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

S3 Repository Key Type Option (--repo-s3-key-type)

S3 repository key type.

The following types are supported:

- shared - Shared keys
- auto - Automatically retrieve temporary credentials
- web-id - Automatically retrieve web identity credentials
- pod-id - Automatically retrieve EKS pod identity credentials
- process - Retrieve credentials by executing a process

```
default: shared
example: repo1-s3-key-type=auto
```

S3 Repository KMS Key ID Option (--repo-s3-kms-key-id)

S3 repository KMS key.

Enables S3 server-side encryption using the specified AWS key management service key.

```
example: repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

S3 Authentication Process Command Option (--repo-s3-process-cmd)

S3 authentication process command.

Command (and optional arguments) to execute for retrieving temporary S3 credentials. The first list entry is the command and the remaining entries are passed as parameters.

The process must output JSON containing AccessKeyId, SecretAccessKey, SessionToken, and Expiration fields. Credentials will be automatically refreshed before the expiration time. See Process Credential Provider for format details.

```
example: repo1-s3-process-cmd=/usr/local/bin/get-credentials
example: repo1-s3-process-cmd=--role
example: repo1-s3-process-cmd=my-role
```

S3 Repository Region Option (--repo-s3-region)

S3 repository region.

The AWS region where the bucket was created.

```
example: repo1-s3-region=us-east-1
```

S3 Repository Requestor Pays Option (--repo-s3-requester-pays)

S3 repository requester pays.

Enables S3 requester pays.

```
default: n
example: repo1-s3-requester-pays=n
```

S3 Repository Role Option (--repo-s3-role)

S3 repository role.

The AWS role name (not the full ARN) used to retrieve temporary credentials when repo-s3-key-type=auto.

```
example: repo1-s3-role=authrole
```

S3 Repository Service Option (--repo-s3-service)

S3 signing service.

The S3 signing service used in SigV4 authentication. Defaults to s3 for standard S3 endpoints. Set to s3-outposts when using an S3 Outposts endpoint.

```
default: s3
example: repo1-s3-service=s3-outposts
```

S3 Repository SSE Customer Key Option (--repo-s3-sse-customer-key)

S3 repository SSE customer key.

Enables S3 server-side encryption using the specified customer key.

```
example: repo1-s3-sse-customer-key=bceb4f13-6939-4be3-910d-df54dee817b7
```

S3 Repository STS Endpoint Option (--repo-s3-sts-host)

S3 repository STS endpoint.

The STS endpoint used to retrieve temporary credentials when repo-s3-key-type=web-id is configured. Set to a regional endpoint (e.g. sts.us-east-1.amazonaws.com) to use regional STS, which may be required for GovCloud, China regions, or to reduce latency.

```
default: sts.amazonaws.com
example: repo1-s3-sts-host=sts.us-east-1.amazonaws.com
```

S3 Repository Security Token Option (--repo-s3-token)

S3 repository security token.

AWS security token used with temporary credentials.

`example: repo1-s3-token=AQoDYXdzEPT//////////wEXAMPLEtc764bNrC9SAPBSM22 ...`

S3 Repository URI Style Option (--repo-s3-uri-style)

S3 URI Style.

The following URI styles are supported:

- host - Connect to bucket.endpoint host.
- path - Connect to endpoint host and prepend bucket to URIs.

`default: host`
`example: repo1-s3-uri-style=path`

SFTP Repository Host Option (--repo-sftp-host)

SFTP repository host.

The SFTP host containing the repository.

`example: repo1-sftp-host=sftprepo.domain`

SFTP Repository Host Fingerprint Option (--repo-sftp-host-fingerprint)

SFTP repository host fingerprint.

SFTP repository host fingerprint generation should match the repo-sftp-host-key-hash-type. Generate the fingerprint via awk '{print $2}' ssh_host_xxx_key.pub | base64 -d | (md5sum or sha1sum) -b. The ssh host keys are normally found in the /etc/ssh directory.

`example: repo1-sftp-host-fingerprint=f84e172dfead7aeeeae6c1fdfb5aa8cf`

SFTP Host Key Check Type Option (--repo-sftp-host-key-check-type)

SFTP host key check type.

The following SFTP host key check types are supported:

- strict - pgBackRest will never automatically add host keys to the ~/.ssh/known_hosts file, and refuses to connect to hosts whose host key has changed or is not found in the known hosts files. This option forces the user to manually add all new hosts.
- accept-new - pgBackRest will automatically add new host keys to the user's known hosts file, but will not permit connections to hosts with changed host keys.
- fingerprint - pgBackRest will check the host key against the fingerprint specified by the repo-sftp-host-fingerprint option.
- none - no host key checking will be performed.

```
default: strict
example: repo1-sftp-host-key-check-type=accept-new
```

SFTP Repository Host Key Hash Type Option (--repo-sftp-host-key-hash-type)

SFTP repository host key hash type.

SFTP repository host key hash type. Declares the hash type to be used to compute the digest of the remote system's host key on SSH startup. Newer versions of libssh2 support sha256 in addition to md5 and sha1.

```
example: repo1-sftp-host-key-hash-type=sha256
```

SFTP Repository Host Port Option (--repo-sftp-host-port)

SFTP repository host port.

SFTP repository host port.

```
default: 22
allowed: [1, 65535]
example: repo1-sftp-host-port=22
```

SFTP Repository Host User Option (--repo-sftp-host-user)

SFTP repository host user.

User on the host used to store the repository.

```
example: repo1-sftp-host-user=pg-backup
```

SFTP Known Hosts File Option (--repo-sftp-known-host)

SFTP known hosts file.

A known hosts file to search for an SFTP host match during authentication. When unspecified, pgBackRest will default to searching ~/.ssh/known_hosts, ~/.ssh/known_hosts2, /etc/ssh/ssh_known_hosts, and /etc/ssh/ssh_known_hosts2. If configured with one or more file paths, pgBackRest will search those for a match. File paths must be full or leading tilde paths. The repo-sftp-known-host option can be passed multiple times to specify more than one known hosts file to search. To utilize known hosts file checking repo-sftp-host-fingerprint must not be specified. See also repo-sftp-host-check-type option.

```
example: repo1-sftp-known-host=/home/postgres/.ssh/known_hosts
```

SFTP Repository Private Key File Option (--repo-sftp-private-key-file)

SFTP private key file.

SFTP private key file used for authentication.

```
example: repo1-sftp-private-key-file=~/.ssh/id_ed25519
```

SFTP Repository Private Key Passphrase Option (--repo-sftp-private-key-passphrase)

SFTP private key passphrase.

Passphrase used to access the private key. This is an optional feature when creating an SSH public/private key pair.

`example: repo1-sftp-private-key-passphrase=BeSureToGenerateAndUseASecurePassphrase`

SFTP Repository Public Key File Option (--repo-sftp-public-key-file)

SFTP public key file.

SFTP public key file used for authentication. Optional if compiled against OpenSSL, required if compiled against a different library.

`example: repo1-sftp-public-key-file=~/.ssh/id_ed25519.pub`

Repository Storage CA File Option (--repo-storage-ca-file)

Repository storage CA file.

Use a CA file other than the system default for storage (e.g. S3, Azure) certificates.

`example: repo1-storage-ca-file=/etc/pki/tls/certs/ca-bundle.crt`

Deprecated Names: repo-azure-ca-file, repo-s3-ca-file

Repository Storage TLS CA Path Option (--repo-storage-ca-path)

Repository storage CA path.

Use a CA path other than the system default for storage (e.g. S3, Azure) certificates.

`example: repo1-storage-ca-path=/etc/pki/tls/certs`

Deprecated Names: repo-azure-ca-path, repo-s3-ca-path

Repository Storage Host Option (--repo-storage-host)

Repository storage host.

Connect to a host other than the storage (e.g. S3, Azure) endpoint. This is typically used for testing.

`example: repo1-storage-host=127.0.0.1`

Deprecated Names: repo-azure-host, repo-s3-host

Repository Storage Port Option (--repo-storage-port)

Repository storage port.

Port to use when connecting to the storage (e.g. S3, Azure) endpoint (or host if specified).

```
default: 443
allowed: [1, 65535]
example: repo1-storage-port=9000
```

Deprecated Names: repo-azure-port, repo-s3-port

Repository Storage Tag Option (--repo-storage-tag)

Repository storage tag(s).

Specify tags that will be added to objects when the repository is an object store (e.g. S3). The option can be repeated to add multiple tags.

There is no provision in pgBackRest to modify these tags so be sure to set them correctly before running stanza-create to ensure uniform tags across the entire repository.

```
example: repo1-storage-tag=key1=value1
```

Repository Storage Upload Chunk Size Option (--repo-storage-upload-chunk-size)

Repository storage upload chunk size.

Object stores such as S3 allow files to be uploaded in chunks when the file is too large to be stored in memory. Even if the file can be stored in memory, it is more memory efficient to limit the amount of memory used for uploads.

A larger chunk size will generally lead to better performance because it will minimize upload requests and allow more files to be uploaded in a single request rather than in chunks. The disadvantage is that memory usage will be higher and because the chunk buffer must be allocated per process, larger process-max values will lead to more memory being consumed overall.

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB. Terminology for chunk size varies by storage type, so when searching min/max values use “part size” for AWS S3, “chunk size” for GCS, and “block size” for Azure.

If a file is larger than 1GiB (the maximum size PostgreSQL will create by default) then the chunk size will be increased incrementally up to the maximum allowed in order to complete the file upload.

```
default (depending on repo-type):
    azure - 4MiB
    gcs - 4MiB
    s3 - 5MiB

allow range (depending on repo-type):
    azure - [4MiB, 1GiB]
    gcs - [4MiB, 1GiB]
    s3 - [5MiB, 1GiB]
```

```
example: repo1-storage-upload-chunk-size=16MiB
```

Repository Storage Certificate Verify Option (--repo-storage-verify-tls)

Repository storage certificate verify.

This option provides the ability to enable/disable verification of the storage (e.g. S3, Azure) server TLS certificate. Disabling should only be used for testing or other scenarios where a certificate has been self-signed.

```
default: y
example: repo1-storage-verify-tls=n
```

Deprecated Names: repo-azure-verify-tls, repo-s3-verify-ssl, repo-s3-verify-tls

Repository Symlink Option (--repo-symlink)

Create symlinks within the repository.

Enable creation of the lastest and tablespace symlinks. These symlinks are most useful when using snapshots to do in-place recovery in the repository, which is an uncommon use case.

While this feature is likely not useful for the vast majority of users it remains on by default for legacy purposes. However, it may be useful to disable symlinks for Posix-like storage that does not support them.

```
default: y
example: repo1-symlink=n
```

Target Time for Repository Option (--repo-target-time)

Target time for repository.

The target time defines the time that commands use to read a repository on versioned storage. This allows the command to read the repository as it was at a point-in-time in order to recover data that has been deleted or corrupted by user accident or malware.

Versioned storage is supported by S3, GCS, and Azure but is generally not enabled by default. In addition to enabling versioning, it may be useful to enable object locking for S3 and soft delete for GCS or Azure.

When the repo-target-time option is specified then the repo option must also be provided. It is likely that not all repository types will support versioning and in general it makes sense to target a single repository for recovery.

Note that comparisons to the storage timestamp are <= the timestamp provided and milliseconds are truncated from the timestamp when provided.

```
example: repo-target-time=2024-08-08 12:12:12+00
```

Repository Type Option (--repo-type)

Type of storage used for the repository.

The following repository types are supported:

- azure - Azure Blob Storage Service
- cifs - Like posix, but disables links and directory fsyncs
- gcs - Google Cloud Storage
- posix - Posix-compliant file systems
- s3 - AWS Simple Storage Service
- sftp - Secure File Transfer Protocol

When an NFS mount is used as a posix repository, the same rules apply to pgBackRest as described in the PostgreSQL documentation: Creating a Database Cluster - File Systems.

```
default: posix
example: repo1-type=cifs
```

Restore Options

The restore section defines settings used for restoring backups.

Archive Mode Option (--archive-mode)

Preserve or disable archiving on restored cluster.

This option allows archiving to be preserved or disabled on a restored cluster. This is useful when the cluster must be promoted to do some work but is not intended to become the new primary. In this case it is not a good idea to push WAL from the cluster into the repository.

The following modes are supported:

- off - disable archiving by setting archive_mode=off.
- preserve - preserve current archive_mode setting.

NOTE: This option is not available on PostgreSQL < 12.

```
default: preserve
example: archive-mode=off
```

Exclude Database Option (--db-exclude)

Restore excluding the specified databases.

Databases excluded will be restored as sparse, zeroed files to save space but still allow PostgreSQL to perform recovery. After recovery, those databases will not be accessible but can be removed with the drop database command. The --db-exclude option can be passed multiple times to specify more than one database to exclude.

When used in combination with the --db-include option, --db-exclude will only apply to standard system databases (template0, template1, and postgres).

```
example: db-exclude=db_main
```

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:

built-in databases (template0, template1, and postgres) are always restored unless specifically excluded.

The --db-include option can be passed multiple times to specify more than one database to include.

See Restore Selected Databases for additional information and caveats.

```
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 postgresql.auto.conf or recovery.conf.

See Server Configuration for details on postgresql.auto.conf or recovery.conf options (be sure to select your PostgreSQL version). This option can be used multiple times.

For PostgreSQL >= 12, options will be written into postgresql.auto.conf. For all other versions, options will be written into recovery.conf.

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 postgresql.auto.conf or recovery.conf file, it is always possible to edit/check postgresql.auto.conf or 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.

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.

Tablespaces are restored into their original locations by default. This behavior can be modified for each tablespace with the tablespace-map option, but it is sometimes 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.

CAUTION:

Tablespaces created after the backup started will not be mapped. Make a new backup after a tablespace is created if tablespace mapping is required.

```
example: tablespace-map-all=/data/tablespace
```

Server Options

The server section defines options used for configuring the TLS server.

TLS Server Address Option (--tls-server-address)

TLS server address.

IP address the server will listen on for client requests.

```
default: localhost
example: tls-server-address=*
```

TLS Server Authorized Clients Option (--tls-server-auth)

TLS server authorized clients.

Clients are authorized on the server by verifying their certificate and checking their certificate CN (Common Name) against a list on the server configured with the tls-server-auth option.

A client CN can be authorized for as many stanzas as needed by providing a comma-separated list to the tls-server-auth option or for all stanzas by specifying tls-server-auth=client-cn=*. Wildcards may not be specified for the client CN.

```
example: tls-server-auth=client-cn=stanza1,stanza2
```

TLS Server Certificate Authorities Option (--tls-server-ca-file)

TLS server certificate authorities.

Checks that client certificates are signed by a trusted certificate authority.

```
example: tls-server-ca-file=/path/to/server.ca
```

TLS Server Certificate Option (--tls-server-cert-file)

TLS server certificate file.

Sent to the client to show the server identity.

```
example: tls-server-cert-file=/path/to/server.crt
```

TLS Server Key Option (--tls-server-key-file)

TLS server key file.

Proves server certificate was sent by the owner.

```
example: tls-server-key-file=/path/to/server.key
```

TLS Server Port Option (--tls-server-port)

TLS server port.

Port the server will listen on for client requests.

```
default: 8432
allowed: [1, 65535]
example: tls-server-port=8000
```

Stanza Options

A stanza defines the backup configuration for a specific PostgreSQL database cluster. The stanza section must define the database cluster path and host/user if the database cluster is remote. Also, any global configuration sections can be overridden to define stanza-specific settings.

Indexing: All pg- options are indexed to allow for configuring multiple PostgreSQL hosts. For example, a single primary is configured with the pg1-path, pg1-port, etc. options. If a standby is configured then index the pg- options on the repository host as pg2- (e.g. pg2-host, pg2-path, etc).

PostgreSQL Database Option (--pg-database)

PostgreSQL database.

The database name used when connecting to PostgreSQL. The default is usually best but some installations may not contain this database.

Note that for legacy reasons the setting of the PGDATABASE environment variable will be ignored.

```
default: postgres
example: pg1-database=backupdb
```

PostgreSQL Host Option (--pg-host)

PostgreSQL host for operating remotely.

Used for backups where the PostgreSQL host is different from the repository host.

```
example: pg1-host=db.domain.com
```

Deprecated Name: db-host

PostgreSQL Host Certificate Authority File Option (--pg-host-ca-file)

PostgreSQL host certificate authority file.

Use a CA file other than the system default for connecting to the PostgreSQL host.

```
example: pg1-host-ca-file=/etc/pki/tls/certs/ca-bundle.crt
```

PostgreSQL Host Certificate Authority Path Option (--pg-host-ca-path)

PostgreSQL host certificate authority path.

Use a CA path other than the system default for connecting to the PostgreSQL host.

```
example: pg1-host-ca-path=/etc/pki/tls/certs
```

PostgreSQL Host Certificate File Option (--pg-host-cert-file)

PostgreSQL host certificate file.

Sent to PostgreSQL host to prove client identity.

```
example: pg1-host-cert-file=/path/to/client.crt
```

PostgreSQL Host Command Option (--pg-host-cmd)

PostgreSQL host pgBackRest command.

Required only if the path to the pgBackRest command is different on the local and PostgreSQL hosts. If not defined, the PostgreSQL host command will be set the same as the local command.

```
default: [path of executed pgbackrest binary]
example: pg1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: db-cmd

PostgreSQL Host Configuration Option (--pg-host-config)

pgBackRest database host configuration file.

Sets the location of the configuration file on the PostgreSQL host. This is only required if the PostgreSQL host configuration file is in a different location than the local configuration file.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_FILE
example: pg1-host-config=/conf/pgbackrest/pgbackrest.conf
```

Deprecated Name: db-config

PostgreSQL Host Configuration Include Path Option (--pg-host-config-include-path)

pgBackRest database host configuration include path.

Sets the location of the configuration include path on the PostgreSQL host. This is only required if the PostgreSQL host configuration include path is in a different location than the local configuration include path.

```
default: CFGOPTDEF_CONFIG_PATH "/" PROJECT_CONFIG_INCLUDE_PATH
example: pg1-host-config-include-path=/conf/pgbackrest/conf.d
```

PostgreSQL Host Configuration Path Option (--pg-host-config-path)

pgBackRest database host configuration path.

Sets the location of the configuration path on the PostgreSQL host. This is only required if the PostgreSQL host configuration path is in a different location than the local configuration path.

```
default: CFGOPTDEF_CONFIG_PATH
example: pg1-host-config-path=/conf/pgbackrest
```

PostgreSQL Host Key File Option (--pg-host-key-file)

PostgreSQL host key file.

Proves client certificate was sent by owner.

```
example: pg1-host-key-file=/path/to/client.key
```

PostgreSQL Host Port Option (--pg-host-port)

PostgreSQL host port when pg-host is set.

Use this option to specify a non-default port for the PostgreSQL host protocol.

NOTE:

When pg-host-type=ssh there is no default for pg-host-port. In this case the port will be whatever is configured for the command specified by cmd-ssh.

```
default (depending on pg-host-type):
    tls - 8432

allowed: [0, 65535]
example: pg1-host-port=25
```

Deprecated Name: db-ssh-port

PostgreSQL Host Protocol Type Option (--pg-host-type)

PostgreSQL host protocol type.

The following protocol types are supported:

- ssh - Secure Shell.
- tls - pgBackRest TLS server.

```
default: ssh
example: pg1-host-type=tls
```

PostgreSQL Host User Option (--pg-host-user)

PostgreSQL host logon user when pg-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: pg1-host-user=db_owner
```

Deprecated Name: db-user

PostgreSQL Path Option (--pg-path)

PostgreSQL data directory.

This should be the same as the data_directory reported by PostgreSQL. Even though this value can be read from various places, it is prudent to set it in case those resources are not available during a restore or offline backup scenario.

The pg-path option is tested against the value reported by PostgreSQL on every online backup so it should always be current.

```
example: pg1-path=/data/db
```

Deprecated Name: db-path

PostgreSQL Port Option (--pg-port)

PostgreSQL port.

Port that PostgreSQL is running on. This usually does not need to be specified as most PostgreSQL clusters run on the default port.

```
default: 5432
allowed: [0, 65535]
example: pg1-port=6543
```

Deprecated Name: db-port

PostgreSQL Socket Path Option (--pg-socket-path)

PostgreSQL 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 is usually no need to specify this setting unless the socket directory was explicitly modified with the unix_socket_directories setting in postgresql.conf.

```
example: pg1-socket-path=/var/run/postgresql
```

Deprecated Name: db-socket-path

PostgreSQL Database User Option (--pg-user)

PostgreSQL database user.

The database user name used when connecting to PostgreSQL. If not specified pgBackRest will connect with the local OS user or PGUSER.

```
example: pg1-user=backupuser
```

Table of Contents

Introduction

Current Stable Release

v2.59.1 Release Notes

Stable Releases

Pre-Stable Releases

Introduction

pgBackRest release numbers consist of two parts, major and minor. A major release *may* break compatibility with the prior major release, but v2 releases are fully compatible with v1 repositories and will accept all v1 options. Minor releases can include bug fixes and features but do not change the repository format and strive to avoid changing options and naming.

Documentation for the v1 release can be found here.

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 on the pgBackRest codebase.

Current Stable Release

v2.59.1 Release Notes

PostgreSQL 19beta3 Support

Released August 17, 2026

Bug Fixes:

- Fix hang when the chunk buffer is smaller than the input buffer. (*Reviewed by Andrew Pogrebnoi, Douglas J Hunley. Reported by crajac66.*)
- Compare a new backup label only against its own full backup set. (*Reviewed by Douglas J Hunley. Reported by Anton Glushakov.*)
- Fix source archives generated by GitHub missing files required to build. (*Reported by aardvarkzed.*)

Features:

- PostgreSQL 19beta3 support. (*Contributed by Lardière Sébastien. Reviewed by David Steele.*)

Documentation Improvements:

- Document exception for dots in S3 bucket names when repo-s3-uri-style=path.
- Remove explicit hot_standby configuration from user guide.

Stable Releases

v2.59.0 Release Notes

PostgreSQL 19beta2 Support

Released July 20, 2026

NOTE TO PACKAGERS: A new distribution tarball is available which simplifies the build process by providing pregenerated HTML documentation, man page, and code. The tarball is attached to each release as an asset and is named pgbackrest-{version}.tar.gz. See the README.md in the tarball for details. Please use the new distribution tarball to avoid the additional build tooling required to generate code in future releases. See New Distribution Tarball for more information.

NOTE TO PACKAGERS: There is a new optional dependency on libsystemd.

IMPORTANT NOTE: Starting with this release, only the restore command may be run as root by default. Use allow-root to run other commands as root (though this is not recommended).

Bug Fixes:

- Fix expire removing a backup in progress on another host. (*Reviewed by Douglas J Hunley. Reported by Dmitrii.*)
- Fix archive-push-queue-max not enforced when a WAL file errors. (*Reviewed by Stefan Fercot. Reported by vut12.*)
- Fix async archive-get failure during recovery across a timeline switch. (*Reviewed by Douglas J Hunley. Reported by Jimmy Yih.*)
- Fix potential buffer overrun in error module. (*Fixed by Christophe Pettus. Reviewed by David Steele.*)
- Fix use-after-free of parallel protocol client. (*Reviewed by Douglas J Hunley. Reported by Georgy Shelkovy.*)
- Fix heap overflow when a pg option is set without pg1. (*Reviewed by Douglas J Hunley. Reported by Naveen.*)

Features:

- PostgreSQL 19beta2 support. (*Reviewed by Stefan Fercot.*)
- Add archive-expire-before option to clean up WAL archive. (*Contributed by Stefan Fercot. Reviewed by David Steele.*)
- Add batch delete for Azure storage. (*Reviewed by Douglas J Hunley, Crispy.*)
- Add support for S3 Outposts. (*Contributed by Shiva Kumar Ambigi. Reviewed by David Steele, Roberto Mello.*)
- Add S3 process authentication. (*Reviewed by Andrew Charlton. Suggested by Andrew Charlton.*)

Improvements:

- Reconnect SFTP storage after the server drops an idle connection. (*Reviewed by Stefan Fercot. Suggested by mustafa0x, tisserat.*)
- Add user/group caching for faster manifest build. (*Contributed by Gunnar Lindholm. Reviewed by David Steele.*)
- Add per-repo backup progress to info command output. (*Contributed by Will Morland. Reviewed by David Steele, Stefan Fercot.*)
- Add backup.info checks to verify command. (*Contributed by Denis Garsh. Reviewed by David Steele, Douglas J Hunley.*)
- Allow the S3 STS endpoint to be configured. (*Contributed by Simon Gratton. Reviewed by David Steele.*)
- Add archive-push-batch-size to limit WAL pushed per asynchronous run. (*Reviewed by Stefan Fercot.*)
- Exit async archive-push on first error. (*Reviewed by Stefan Fercot. Suggested by Lardière Sébastien.*)
- Harden HTTP chunked response parsing. (*Contributed by Shubham. Reviewed by David Steele.*)
- Error when running as root unless allow-root is enabled. (*Reviewed by Douglas J Hunley.*)
- Fewer binary searches when building manifest. (*Contributed by Gunnar Lindholm. Reviewed by David Steele.*)
- Improve seek performance during block incremental delta restore. (*Reviewed by David Christensen, Douglas J Hunley.*)
- Bundle backup files in order rather than scanning for a better fit. (*Reviewed by Stefan Fercot.*)
- Report the underlying error when a query fails to complete. (*Reviewed by Andrew Pogrebnoi. Suggested by Marco Fontana.*)
- Report the actual error when an S3 credential request fails. (*Reviewed by Douglas J Hunley. Suggested by Mitchell Grice.*)
- Report archive-push spool path errors to the PostgreSQL log. (*Reviewed by Douglas J Hunley. Suggested by Don Seiler.*)
- Hint that pgBackRest may be out of date when a version is unsupported.
- Add systemd notify integration. (*Contributed by Andrew Jackson. Reviewed by David Steele.*)
- Suppress unused parameter errors in meson compiler probes. (*Contributed by Jörg Plate. Reviewed by David Steele.*)
- C11 is now the minimum C standard. (*Reviewed by Mohammad Ali Nazir Kosar.*)

Documentation Improvements:

- Make stanza option internal for the repo-* commands. (*Reviewed by Stefan Fercot.*)
- Document that info reads encryption settings from global section. (*Reviewed by Stefan Fercot. Suggested by Ron Johnson.*)
- Document that expire-auto uses the backup command configuration. (*Re-*

viewed by Stefan Fercot. Suggested by Lardière Sébastien.)
- Document adjusting the logrotate su directive for a dedicated user. (*Reviewed by Douglas J Hunley. Suggested by lkanbus.*)
- Document that end-of-line comments are not supported in config files. (*Reviewed by Douglas J Hunley. Suggested by Alex Richman.*)

Test Suite Improvements:

- Fix Alpine group conflicts in CI containers. (*Contributed by Artur Zakirov. Reviewed by David Steele.*)

v2.58.0 Release Notes

Object Storage Improvements

Released January 19, 2026

IMPORTANT NOTE: The minimum values for the repo-storage-upload-chunk-size option have increased. They now represent the minimum allowed by the vendors.

Bug Fixes:

- Fix deadlock due to logging in signal handler. (*Fixed by Maxim Michkov. Reviewed by David Steele.*)

Features:

- HTTP support for S3, GCS, and Azure. (*Contributed by Will Morland. Reviewed by David Steele.*)
- Allow expiration of oldest full backup regardless of current retention. (*Contributed by Stefan Fercot. Reviewed by David Steele. Suggested by Ron Johnson.*)
- Support for Azure managed identities. (*Contributed by Moiz Ibrar, Matthew Mols. Reviewed by David Steele.*)
- ***Experimental*** support for S3 EKS pod identity. (*Contributed by Pierre BOUTELOUP. Reviewed by David Steele.*)
- Allow configuration of TLS cipher suites. (*Contributed by Gunnar "Nick" Bluth. Reviewed by David Steele.*)
- Allow process priority to be set. (*Reviewed by Douglas J Hunley.*)

Improvements:

- Allow dots in S3 bucket names when using path-style URIs. (*Contributed by Joakim Hindersson. Reviewed by David Steele.*)
- Require TLS >= 1.2 unless verification is disabled. (*Reviewed by Douglas J Hunley, Gunnar "Nick" Bluth.*)
- Dynamically size S3/GCS/Azure chunks for large uploads. (*Reviewed by Douglas J Hunley. Suggested by Timothée Peignier.*)
- Optimize S3/GCS/Azure chunk size for small files. (*Reviewed by Douglas J Hunley.*)
- Remove support for PostgreSQL 9.5. (*Reviewed by Douglas J Hunley.*)

- Improve logging of default for options with an unresolved dependency. (*Reviewed by Stefan Fercot.*)

Documentation Improvements:

- Remove explicit max_wal_senders/wal_level configuration from user guide. (*Suggested by Jamie Nguyen.*)
- Clarify that bundling is useful for filesystems with large block sizes. (*Suggested by Ron Johnson.*)

v2.57.0 Release Notes

Suppress Repository Symlinks

Released October 18, 2025

Bug Fixes:

- Unnest HTTP/TLS/socket timeouts. (*Reviewed by David Christensen.*)
- Fix possible segfault in page checksum error message. (*Fixed by Zsolt Parragi. Reviewed by David Steele.*)

Features:

- Add repo-symlink option to suppress creation of repository symlinks. (*Reviewed by Douglas J Hunley. Suggested by Ron Johnson.*)

Improvements:

- Add HTTP retries for 408 and 429 errors. (*Reviewed by David Christensen.*)

v2.56.0 Release Notes

Progress Info Improvements

Released July 21, 2025

Bug Fixes:

- Fix issue with adhoc expiration when no backups in a repository. (*Reviewed by Stefan Fercot. Reported by Anup Gupta.*)

Features:

- Add restore progress to info command output. (*Contributed by Denis Garsh, Maxim Michkov. Reviewed by David Steele.*)
- Add progress-only detail level for info command output. (*Contributed by Denis Garsh. Reviewed by David Steele, Stefan Fercot.*)

Improvements:

- Retry failed reads on object stores. (*Reviewed by David Christensen.*)
- Fix defaults in command-line help. (*Reviewed by David Christensen, Chris Bandy.*)

Documentation Improvements:

- Describe discrete option values in a list where appropriate. (*Contributed by Anton Kurochkin. Reviewed by David Steele.*)
- Fix "less than" in help output for archive-mode option. (*Contributed by Anton Kurochkin. Reviewed by David Steele.*)

v2.55.1 Release Notes

Bug Fixes

Released May 5, 2025

Bug Fixes:

- Revert "calculate content-md5 on S3 only when required". (*Reviewed by David Christensen. Reported by Frank Brendel.*)
- Fix lower bounds checking for option keys. (*Reviewed by David Christensen, Wolfgang Walther. Reported by Wolfgang Walther.*)

v2.55.0 Release Notes

Verification Improvements and PostgreSQL 18 Support

Released April 21, 2025

Bug Fixes:

- Fix block incremental restore issue on non-default repository. (*Reviewed by David Christensen, Aleksander Łukasz. Reported by Aleksander Łukasz.*)
- Do not set recovery_target_timeline=current for PostgreSQL < 12. (*Reviewed by Stefan Fercot.*)
- Fix expire archive range logging. (*Reviewed by Stefan Fercot. Reported by Aleš Zelený.*)
- Fix error reporting for queries with no results. (*Reviewed by Stefan Fercot. Reported by Susantha Bathige.*)

Features:

- Verify recovery target timeline. (*Reviewed by Stefan Fercot.*)
- Allow verification of a specified backup. (*Contributed by Maxim Michkov. Reviewed by David Steele.*)
- Add support for S3/GCS requester pays. (*Contributed by Timothée Peignier. Reviewed by David Steele.*)
- PostgreSQL 18 support. (*Reviewed by Stefan Fercot.*)
- Allow connections to PostgreSQL on abstract domain sockets. (*Reviewed by Chris Bandy. Suggested by Chris Bandy.*)
- Add numeric output to version command. (*Contributed by Stefan Fercot. Reviewed by David Steele.*)

Improvements:

- Allow backup command to operate on remote repositories. (*Reviewed by Stefan Fercot.*)
- Use lz4 for protocol compression. (*Reviewed by Stefan Fercot.*)
- Calculate content-md5 on S3 only when required. (*Reviewed by David Christensen.*)
- Warn when a value for a multi-key option is overwritten. (*Reviewed by David Christensen, Stefan Fercot.*)
- Add detail logging for expired archive path. (*Contributed by Stefan Fercot. Reviewed by David Steele.*)
- Remove support for PostgreSQL 9.4. (*Reviewed by Stefan Fercot.*)
- Remove autoconf/make build. (*Reviewed by David Christensen.*)

Documentation Improvements:

- Fix documentation for specifying multiple stanzas with tls-server-auth. (*Reviewed by David Christensen, Stefan Fercot. Suggested by Terry MacAndrew.*)
- Clarify incremental backup expiration. (*Reviewed by Stefan Fercot.*)
- Clarify requirement for local/remote pgBackRest versions to match. (*Contributed by Greg Clough. Reviewed by David Steele.*)
- Add FAQ about exporting self-contained cluster. (*Contributed by Stefan Fercot. Reviewed by David Steele.*)
- Caveat --tablespace-map-all regarding tablespace creation. (*Reviewed by Stefan Fercot, Christophe Courtois. Suggested by Christophe Courtois.*)
- Clarify behavior of --repo-retention-full-type. (*Reviewed by Antoine Beaupré. Suggested by Antoine Beaupré.*)
- Change --process-max recommendation for object stores to --repo-bundle. (*Reviewed by Stefan Fercot.*)
- Update unix_socket_directory to unix_socket_directories. (*Contributed by hyunkyu han. Reviewed by David Steele.*)
- Recommend not placing spool-path within pg_xlog/pg_wal. (*Reviewed by Martín Marqués, Don Seiler. Suggested by Martín Marqués.*)

v2.54.2 Release Notes

Bug Fix

Released January 20, 2025

Bug Fixes:

- Fix issue after disabling bundling with block incremental enabled. (*Reviewed by David Christensen.*)

Documentation Improvements:

- Clarify behavior of multiple configuration files. (*Reviewed by Paul Bierly. Suggested by Paul Bierly.*)

v2.54.1 Release Notes

Bug Fix

Released December 16, 2024

Bug Fixes:

- Fix issue with version/help commands attempting to load pgbackrest.conf. (*Reviewed by Stefan Fercot. Reported by Bradford Boyle, Julian.*)

Test Suite Improvements:

- Stabilize async archiving in integration tests. (*Contributed by Viktor Kurilko. Reviewed by David Steele.*)

v2.54.0 Release Notes

Target Time for Versioned Storage

Released October 21, 2024

NOTE TO PACKAGERS: This is last feature release to support the autoconf/make build. Please migrate to meson if you have not already done so. 2.54.X patch releases (if any) will continue to support autoconf/make.

Bug Fixes:

- Fix PostgreSQL query performance for large datasets. (*Fixed by Thibault Vincent, David Steele. Reviewed by David Christensen, Antoine Millet. Reported by Antoine Millet.*)

Features:

- Allow repositories on versioned storage to be read at a target time. (*Reviewed by Stefan Fercot, David Christensen.*)
- Allow requested standby backup to proceed with no standby. (*Reviewed by Stefan Fercot.*)

Improvements:

- Summarize backup reference list for info command text output. (*Contributed by Stefan Fercot. Reviewed by David Steele.*)
- Refresh web-id token for each S3 authentication. (*Contributed by Brent Graveland. Reviewed by David Steele.*)
- Correctly display current values for indexed options in help. (*Reviewed by David Christensen.*)
- Save backup.info only when contents have changed. (*Reviewed by Stefan Fercot.*)
- Remove limitation on reading files in parallel during restore. (*Reviewed by David Christensen.*)
- Improve SFTP error messages. (*Contributed by Reid Thompson. Reviewed by David Steele.*)

Documentation Features:

- Add performance tuning section to user guide. (*Reviewed by Stefan Fercot.*)

Documentation Improvements:

- Clarify source for data_directory. (*Contributed by Stefan Fercot. Reviewed by David Steele. Suggested by Matthias.*)
- Better logic for deciding when a summary should be lower-cased. (*Suggested by Daniel Westermann.*)

v2.53.1 Release Notes

PostgreSQL 17 Support

Released August 19, 2024

Bug Fixes:

- Fix permissions when restore run as root user. (*Reviewed by Stefan Fercot. Reported by Will M.*)
- Fix segfault on delayed connection errors. (*Reviewed by David Christensen. Reported by Anton Glushakov.*)
- Skip local repository duplicate check for SFTP. (*Fixed by Reid Thompson. Reviewed by David Steele. Reported by Anton Kurochkin.*)

Improvements:

- PostgreSQL 17 support.

v2.53 Release Notes

Concurrent Backups

Released July 22, 2024

IMPORTANT NOTE: The log-level-stderr option default has been changed from warn to off. This makes it easier to capture errors when only redirecting stdout. To preserve the prior behavior set log-level-stderr=warn.

NOTE TO PACKAGERS: The lz4 library is now required by the meson build.

NOTE TO PACKAGERS: Compiler support for ___builtin_clzl() and ___builtin_bswap64() is now required by the meson build.

Bug Fixes:

- Fix SFTP renaming failure when file already exists. (*Fixed by Reid Thompson. Reviewed by David Steele. Reported by ahmed112212.*)

Features:

- Allow backups to run concurrently on different repositories. (*Reviewed by Reid Thompson, Stefan Fercot.*)

- Support IP-based SANs for TLS certificate validation. (*Contributed by David Christensen. Reviewed by David Steele.*)

Improvements:

- Default log-level-stderr option to off. (*Reviewed by Greg Sabino Mullane, Stefan Fercot.*)
- Allow alternative WAL segment sizes for PostgreSQL 10. (*Contributed by Viktor Kurilko. Reviewed by David Steele.*)
- Add hint to check SFTP authorization log. (*Contributed by Vitalii Zurian. Reviewed by Reid Thompson, David Steele.*)

Documentation Improvements:

- Clarify archive-push multi-repo behavior. (*Reviewed by Stefan Fercot.*)

v2.52.1 Release Notes

Bug Fix

Released June 25, 2024

Bug Fixes:

- Fix issue with files larger on the replica than on the primary. (*Reviewed by Stefan Fercot. Reported by Nicolas Lassimonne.*)

v2.52 Release Notes

PostgreSQL 17beta1 Support

Released May 27, 2024

NOTE TO PACKAGERS: The build system for pgBackRest is now meson. The autoconf/make build will not receive any new features and will be removed after a few releases.

Features:

- Add GCS batch delete support. (*Reviewed by Reid Thompson.*)
- S3 SSE-C encryption support. (*Reviewed by Tim Jones. Suggested by Tim Jones.*)
- PostgreSQL 17beta1 support. (*Reviewed by Stefan Fercot.*)

Improvements:

- Allow explicit disabling of optional dependencies in meson builds. (*Contributed by Michael Schout. Reviewed by David Steele.*)
- Dynamically find python in meson build. (*Contributed by Michael Schout. Reviewed by David Steele.*)
- Tag pgbackrest build target in meson as installable. (*Contributed by Bradford Boyle. Reviewed by David Steele.*)

Documentation Improvements:

- Update start/stop documentation to reflect actual functionality. (*Reviewed by Stefan Fercot.*)

v2.51 Release Notes

Meson Build System

Released March 25, 2024

Bug Fixes:

- Skip zero-length files for block incremental delta restore. (*Reviewed by Sebastian Krause, René Højbjerg Larsen. Reported by Sebastian Krause.*)
- Fix performance regression in storage list. (*Reviewed by Stephen Frost. Reported by Maksym Boguk.*)
- Fix progress logging when file size changes during backup. (*Reviewed by Stephen Frost. Reported by samkingno.*)

Improvements:

- Improved support for dual stack connections. (*Reviewed by Stephen Frost. Suggested by Timothée Peignier.*)
- Make meson the primary build system. (*Reviewed by Stephen Frost.*)
- Detect files that have not changed during non-delta incremental backup. (*Reviewed by Stephen Frost.*)
- Prevent invalid recovery when backup_label removed. (*Reviewed by Stephen Frost.*)
- Improve archive-push WAL segment queue handling. (*Reviewed by Stephen Frost.*)
- Limit resume functionality to full backups. (*Reviewed by Stephen Frost, Stefan Fercot.*)
- Update resume functionality for block incremental. (*Reviewed by Stephen Frost.*)
- Allow --version and --help for version and help. (*Reviewed by Greg Sabino Mullane. Suggested by Greg Sabino Mullane.*)
- Add detailed backtrace to autoconf/make build. (*Reviewed by Stephen Frost.*)

Documentation Improvements:

- Update references to recovery.conf. (*Reviewed by Stefan Fercot. Suggested by Stephen Frost.*)

v2.50 Release Notes

Performance Improvements and Bug Fixes

Released January 22, 2024

Bug Fixes:

- Fix short read in block incremental restore. (*Reviewed by Stephen Frost, Brent Graveland. Reported by Adol Rodriguez, Brent Graveland.*)

- Fix overflow suppressing backup progress in info output. (*Fixed by Robert Donovan. Reviewed by Joe Wildish.*)

Improvements:

- Preserve partial files during block incremental delta restore. (*Reviewed by Stephen Frost.*)
- Add support for alternate compile-time page sizes. (*Contributed by Viktor Kurilko. Reviewed by David Steele.*)
- Skip files truncated during backup when bundling. (*Contributed by Georgy Shelkovy. Reviewed by David Steele.*)
- Improve SFTP storage error messages. (*Contributed by Reid Thompson. Reviewed by David Steele.*)

v2.49 Release Notes

Remove PostgreSQL 9.3 Support

Released November 27, 2023

Bug Fixes:

- Fix regression in retries. (*Reviewed by Stephen Frost. Reported by Norman Adkins, Tanel Suurhans, Jordan English, Timothée Peignier.*)
- Fix recursive path remove in SFTP storage driver. (*Fixed by Reid Thompson. Reviewed by Stephen Frost. Reported by Luc.*)

Improvements:

- Remove support for PostgreSQL 9.3. (*Reviewed by Stephen Frost.*)

Documentation Features:

- Document maintainer options. (*Reviewed by Stefan Fercot.*)
- Update point-in-time recovery documentation for PostgreSQL >= 13.

Test Suite Improvements:

- Allow config/load unit test to run without libssh2 installed. (*Contributed by Reid Thompson. Reviewed by David Steele. Suggested by Wu Ning.*)

v2.48 Release Notes

Repository Storage Tags

Released September 25, 2023

Bug Fixes:

- Fix issue restoring block incremental without a block list. (*Reviewed by Stephen Frost, Burak Yurdakul. Reported by Burak Yurdakul.*)

Features:

- Add --repo-storage-tag option to create object tags. (*Reviewed by Stephen Frost, Stefan Fercot, Timothée Peignier.*)

- Add known hosts checking for SFTP storage driver. (*Contributed by Reid Thompson. Reviewed by Stephen Frost, David Steele.*)
- Support for dual stack connections. (*Reviewed by Stephen Frost.*)
- Add backup size completed/total to info command JSON output. (*Contributed by Stefan Fercot. Reviewed by David Steele.*)

Improvements:

- Multi-stanza check command. (*Reviewed by Stephen Frost.*)
- Retry reads of pg_control until checksum is valid. (*Reviewed by Stefan Fercot, Stephen Frost.*)
- Optimize WAL segment check after successful backup. (*Reviewed by Stephen Frost.*)
- Improve GCS multi-part performance. (*Reviewed by Reid Thompson.*)
- Allow archive-get command to run when stanza is stopped. (*Reviewed by Tom Swartz, David Christensen, Reid Thompson.*)
- Accept leading tilde in paths for SFTP public/private keys. (*Contributed by Reid Thompson. Reviewed by David Steele.*)
- Reload GCS credentials before renewing authentication token. (*Reviewed by Stephen Frost. Suggested by Daniel Farina.*)

Documentation Bug Fixes:

- Fix configuration reference example for the tls-server-address option. (*Fixed by Hartmut Goebel. Reviewed by David Steele.*)
- Fix command reference example for the filter option.

Test Suite Improvements:

- Allow storage/sftp unit test to run without libssh2 installed. (*Contributed by Reid Thompson. Reviewed by David Steele. Suggested by Wu Ning.*)

v2.47 Release Notes

Performance Improvements and Bug Fixes

Released July 24, 2023

Bug Fixes:

- Preserve block incremental info in manifest during delta backup. (*Reviewed by Stephen Frost. Reported by Francisco Miguel Biete Banon.*)
- Fix block incremental file names in verify command. (*Reviewed by Reid Thompson. Reported by Francisco Miguel Biete Banon.*)
- Fix spurious automatic delta backup on backup from standby. (*Reviewed by Stephen Frost. Reported by krmozejko, Don Seiler.*)
- Skip recovery.signal for PostgreSQL >= 12 when recovery type=none. (*Reviewed by Stefan Fercot. Reported by T.Anastacio.*)
- Fix unique label generation for diff/incr backup. (*Fixed by Andrey Sokolov. Reviewed by David Steele.*)

- Fix time-based archive expiration when no backups are expired. (*Reviewed by Stefan Fercot.*)

Improvements:

- Improve performance of SFTP storage driver. (*Contributed by Stephen Frost, Reid Thompson. Reviewed by David Steele.*)
- Add timezone offset to info command date/time output. (*Reviewed by Stefan Fercot, Philip Hurst. Suggested by Philip Hurst.*)
- Centralize error handling for unsupported features. (*Reviewed by Stefan Fercot.*)

Documentation Improvements:

- Clarify preference to install from packages in the user guide. (*Reviewed by Stefan Fercot. Suggested by dr-kd.*)

v2.46 Release Notes

Block Incremental Backup and SFTP Storage

Released May 22, 2023

Features:

- Block incremental backup. (*Reviewed by John Morris, Stephen Frost, Stefan Fercot.*)
- SFTP support for repository storage. (*Contributed by Reid Thompson. Reviewed by Stephen Frost, David Steele.*)
- PostgreSQL 16 support. (*Reviewed by Stefan Fercot.*)

Improvements:

- Allow page header checks to be skipped. (*Reviewed by David Christensen. Suggested by David Christensen.*)
- Avoid chown() on recovery files during restore. (*Reviewed by Stefan Fercot, Marcelo Henrique Neppel. Suggested by Marcelo Henrique Neppel.*)
- Add error retry detail for HTTP retries.

Documentation Improvements:

- Add warning about using recovery type=none. (*Reviewed by Stefan Fercot.*)
- Add note about running stanza-create on already-created repositories.

v2.45 Release Notes

Block Incremental Backup (BETA)

Released March 20, 2023

Bug Fixes:

- Skip writing recovery.signal by default for restores of offline backups. (*Reviewed by Stefan Fercot. Reported by Marcel Borger.*)

Features:

- Block incremental backup (BETA). (*Reviewed by John Morris, Stephen Frost, Stefan Fercot.*)

Improvements:

- Keep only one all-default group index. (*Reviewed by Stefan Fercot.*)

Documentation Improvements:

- Add explicit instructions for upgrading between 2.x versions. (*Contributed by Christophe Courtois. Reviewed by David Steele.*)
- Remove references to SSH made obsolete when TLS was introduced.

v2.44 Release Notes

Remove PostgreSQL 9.0/9.1/9.2 Support

Released January 30, 2023

Improvements:

- Remove support for PostgreSQL 9.0/9.1/9.2. (*Reviewed by Stefan Fercot.*)
- Restore errors when no backup matches the current version of PostgreSQL. (*Contributed by Stefan Fercot. Reviewed by David Steele. Suggested by Soulou.*)
- Add compress-level range checking for each compress-type. (*Reviewed by Stefan Fercot. Suggested by gkleen, ViperRu.*)

Documentation Improvements:

- Add warning about enabling "hierarchical namespace" on Azure storage. (*Reviewed by Stefan Fercot. Suggested by Vojtech Galda, Pluggi, asjonos.*)
- Add replacement for linefeeds in monitoring example. (*Reviewed by Stefan Fercot. Suggested by rudonx, gmustdie, Ivan Shelestov.*)
- Clarify target-action behavior on various PostgreSQL versions. (*Contributed by Chris Bandy. Reviewed by David Steele, Anton Kurochkin, Stefan Fercot. Suggested by Anton Kurochkin, Chris Bandy.*)
- Updates and clarifications to index page. (*Reviewed by Stefan Fercot.*)
- Add dark mode to the website. (*Suggested by Stephen Frost.*)

v2.43 Release Notes

Bug Fix

Released November 28, 2022

Bug Fixes:

- Fix missing reference in diff/incr backup. (*Reviewed by Stefan Fercot. Reported by Marcel Borger, ulfedf, jaymefSO.*)

Improvements:

- Add hint when an option is specified without an index. (*Reviewed by Stefan Fercot.*)

v2.42 Release Notes

Bug Fixes

Released November 22, 2022

Bug Fixes:

- Fix memory leak in file bundle backup/restore. (*Reviewed by John Morris, Oscar. Reported by Oscar.*)
- Fix protocol error on short read of remote file. (*Reviewed by Stephen Frost.*)

Improvements:

- Do not store references for zero-length files when bundling. (*Reviewed by Stefan Fercot.*)
- Use more generic descriptions for pg_start_backup()/pg_stop_backup(). (*Reviewed by Greg Sabino Mullane, David Christensen. Suggested by Greg Sabino Mullane.*)

Test Suite Improvements:

- Update test.pl --psql-bin option to match command-line help. (*Contributed by Koshi Shibagaki. Reviewed by David Steele.*)

v2.41 Release Notes

Backup Annotations

Released September 19, 2022

Bug Fixes:

- Fix incorrect time expiration being used for non-default repositories. (*Reviewed by Stefan Fercot. Reported by Adam Brusselback.*)
- Fix issue when listing directories recursively with a filter. (*Reviewed by Stephen Frost. Reported by Efremov Egor.*)

Features:

- Backup key/value annotations. (*Contributed by Stefan Fercot. Reviewed by David Steele. Suggested by Adam Berlin.*)

Improvements:

- Support --set in JSON output for info command. (*Contributed by Stefan Fercot. Reviewed by David Steele. Suggested by Anton Kurochkin.*)
- Allow upload chunk size to be configured for object stores. (*Reviewed by Stefan Fercot. Suggested by Anton Glushakov.*)
- Update archive.info timestamps after a successful backup. (*Reviewed by Stefan Fercot. Suggested by Alex Richman.*)

- Move standby timeline check after checkpoint. (*Reviewed by Stefan Fercot, Keith Fiske. Suggested by Keith Fiske.*)
- Improve warning message on backup resume. (*Suggested by Cynthia Shang.*)

Documentation Improvements:

- Add absolute path for kill in pgbackrest.service. (*Suggested by Don Seiler.*)

v2.40 Release Notes

OpenSSL 3 Support

Released July 18, 2022

NOTE TO PACKAGERS: An experimental meson build has been added but packagers should continue to use the autoconf/make build for the foreseeable future.

Improvements:

- OpenSSL 3 support. (*Reviewed by Stephen Frost.*)
- Create snapshot when listing contents of a path. (*Reviewed by John Morris, Stephen Frost.*)
- Force target-timeline=current when restore type=immediate. (*Reviewed by Stephen Frost.*)
- Truncate files during delta restore when they are larger than expected. (*Reviewed by Stephen Frost.*)
- Disable incremental manifest save when resume=n. (*Contributed by Reid Thompson. Reviewed by David Steele.*)
- Set backup percent complete to zero before copy start. (*Contributed by Reid Thompson. Reviewed by David Steele.*)
- Use S3 IsTruncated flag to determine list continuation. (*Reviewed by John Morris, Soulou. Suggested by Christian Montagne.*)

Documentation Bug Fixes:

- Skip internal options in the configuration reference. (*Reported by Francisco Miguel Biete Banon.*)

Documentation Improvements:

- Add link to PostgreSQL configuration in repository host section. (*Reviewed by Stefan Fercot. Suggested by Julien Cigar.*)

Test Suite Improvements:

- Add experimental Meson build. (*Reviewed by Eli Schwartz, Sam Bassaly.*)
- Allow any path to be passed to the --test-path option. (*Contributed by Andrey Sokolov. Reviewed by David Steele.*)
- Fix compile error when DEBUG_EXEC_TIME is defined without DEBUG. (*Contributed by Andrey Sokolov. Reviewed by David Steele.*)

v2.39 Release Notes

Verify and File Bundling

Released May 16, 2022

Bug Fixes:

- Fix error thrown from FINALLY() causing an infinite loop. (*Reviewed by Stephen Frost.*)
- Error on all lock failures except another process holding the lock. (*Reviewed by Reid Thompson, Geir Råness. Reported by Geir Råness.*)

Features:

- Backup file bundling for improved small file support. (*Reviewed by Reid Thompson, Stefan Fercot, Chris Bandy.*)
- Verify command to validate the contents of a repository. (*Contributed by Cynthia Shang, Reid Thompson. Reviewed by David Steele, Stefan Fercot.*)
- PostgreSQL 15 support. (*Reviewed by Stefan Fercot.*)
- Show backup percent complete in info output. (*Contributed by Reid Thompson. Reviewed by David Steele.*)
- Auto-select backup for restore command --type=lsn. (*Contributed by Reid Thompson. Reviewed by Stefan Fercot, David Steele.*)
- Suppress existing WAL warning when archive-mode-check is disabled. (*Contributed by Reid Thompson. Reviewed by David Steele.*)
- Add AWS IMDSv2 support. (*Contributed by Nuno Pires. Reviewed by David Steele.*)

Improvements:

- Allow repo-hardlink option to be changed after full backup. (*Reviewed by Reid Thompson.*)
- Increase precision of percent complete logging for backup and restore. (*Contributed by Reid Thompson. Reviewed by David Steele.*)
- Improve path validation for repo-* commands. (*Contributed by Reid Thompson. Reviewed by David Steele.*)
- Improve stop command to honor stanza option. (*Contributed by Reid Thompson. Reviewed by David Steele. Suggested by ragaoua.*)
- Improve error message for invalid repo-azure-key. (*Contributed by Reid Thompson. Reviewed by David Steele. Suggested by Seth Daniel.*)
- Add hint to check the log on archive-get/archive-push async error. (*Reviewed by Reid Thompson.*)
- Add ClockError for unexpected clock skew and timezone changes. (*Reviewed by Greg Sabino Mullane, Stefan Fercot. Suggested by Greg Sabino Mullane.*)
- Strip extensions from history manifest before showing in error message. (*Reviewed by Stefan Fercot.*)
- Add user:group to lock permission error. (*Reviewed by Reid Thompson.*)

Documentation Bug Fixes:

- Fix incorrect reference to stanza-update in the user guide. (*Fixed by Abubakar Mohammed. Reviewed by David Steele.*)
- Fix example for repo-gcs-key-type option in configuration reference. (*Reviewed by Reid Thompson.*)
- Fix tls-server-auth example and add clarifications. (*Reviewed by Reid Thompson.*)

Documentation Improvements:

- Simplify messaging around supported versions in the documentation. (*Reviewed by Stefan Fercot, Reid Thompson, Greg Sabino Mullane.*)
- Add option type descriptions. (*Contributed by Reid Thompson. Reviewed by David Steele.*)
- Add FAQ about backup types and restore speed. (*Contributed by David Christensen. Reviewed by Reid Thompson.*)
- Document required base branch for pull requests. (*Contributed by David Christensen. Reviewed by Reid Thompson.*)

v2.38 Release Notes

Minor Bug Fixes and Improvements

Released March 6, 2022

IMPORTANT NOTE: Repository size reported by the info command is now entirely based on what pgBackRest has written to storage. Previously, in certain cases, pgBackRest could detect if additional compression was being applied by the storage but this is no longer supported.

Bug Fixes:

- Retry errors in S3 batch file delete. (*Reviewed by Reid Thompson. Reported by Alex Richman.*)
- Allow case-insensitive matching of HTTP connection header values. (*Reviewed by Reid Thompson. Reported by Rémi Vidier.*)

Features:

- Add support for AWS S3 server-side encryption using KMS. (*Contributed by Christoph Berg. Reviewed by David Steele, Tharindu Amila.*)
- Add archive-missing-retry option. (*Reviewed by Stefan Fercot.*)
- Add backup type filter to info command. (*Contributed by Stefan Fercot. Reviewed by David Steele.*)

Improvements:

- Retry on page validation failure during backup. (*Reviewed by Stephen Frost, David Christensen.*)
- Handle TLS servers that do not close connections gracefully. (*Reviewed by Rémi Vidier, David Christensen, Stephen Frost.*)

- Add backup LSNs to info command output. (*Contributed by Stefan Fercot. Reviewed by David Steele.*)
- Automatically strip trailing slashes for repo-ls paths. (*Contributed by David Christensen. Reviewed by David Steele.*)
- Do not retry fatal errors. (*Reviewed by Reid Thompson.*)
- Remove support for PostgreSQL 8.3/8.4. (*Reviewed by Reid Thompson, Stefan Fercot.*)
- Remove logic that tried to determine additional file system compression. (*Reviewed by Reid Thompson, Stefan Fercot.*)

Documentation Bug Fixes:

- Move repo options in TLS documentation to the global section. (*Reported by Anton Kurochkin.*)
- Remove unused backup-standby option from stanza commands. (*Reported by Stefan Fercot.*)
- Fix typos in help and release notes. (*Fixed by Daniel Gustafsson. Reviewed by David Steele.*)

Documentation Improvements:

- Add aliveness check to systemd service configuration. (*Suggested by Yogesh Sharma.*)
- Add FAQ explaining WAL archive suffix. (*Contributed by Stefan Fercot. Reviewed by David Steele.*)
- Note that replications slots are not restored. (*Contributed by Reid Thompson. Reviewed by David Steele, Stefan Fercot. Suggested by Christophe Courtois.*)

v2.37 Release Notes

TLS Server

Released January 3, 2022

IMPORTANT NOTE: If the restore command is unable to find a backup that matches a specified time target then an error will be thrown, whereas before a warning was logged.

Bug Fixes:

- Fix restore delta link mapping when path/file already exists. (*Reviewed by Reid Thompson. Reported by Younes Alhroub.*)
- Fix socket leak on connection retries. (*Reviewed by Reid Thompson. Reported by James Coleman.*)

Features:

- Add TLS server. (*Reviewed by Stephen Frost, Reid Thompson, Andrew L'Ecuyer.*)
- Add --cmd option. (*Contributed by Reid Thompson. Reviewed by Stefan Fercot, David Steele. Suggested by Virgile CREVON.*)

Improvements:

- Check archive immediately after backup start. (*Reviewed by Reid Thompson, David Christensen.*)
- Add timeline and checkpoint checks to backup. (*Reviewed by Stefan Fercot, Reid Thompson.*)
- Check that clusters are alive and correctly configured during a backup. (*Reviewed by Stefan Fercot.*)
- Error when restore is unable to find a backup to match the time target. (*Reviewed by Reid Thompson, Douglas J Hunley. Suggested by Douglas J Hunley.*)
- Parse protocol/port in S3/Azure endpoints. (*Contributed by Reid Thompson. Reviewed by David Steele.*)
- Add warning when checkpoint_timeout exceeds db-timeout. (*Contributed by Stefan Fercot. Reviewed by David Steele.*)
- Add verb to HTTP error output. (*Contributed by Christoph Berg. Reviewed by David Steele.*)
- Allow y/n arguments for boolean command-line options. (*Contributed by Reid Thompson. Reviewed by David Steele.*)
- Make backup size logging exactly match info command output. (*Contributed by Reid Thompson. Reviewed by David Steele. Suggested by Mahomed Hussein.*)

Documentation Improvements:

- Display size option default and allowed values with appropriate units. (*Reviewed by Reid Thompson.*)
- Fix typos and improve documentation for the tablespace-map-all option. (*Reviewed by Reid Thompson. Suggested by Reid Thompson.*)
- Remove obsolete statement about future multi-repository support. (*Suggested by David Christensen.*)

v2.36 Release Notes

Minor Bug Fixes and Improvements

Released November 1, 2021

Bug Fixes:

- Allow "global" as a stanza prefix. (*Reviewed by Stefan Fercot. Reported by Younes Alhroub.*)
- Fix segfault on invalid GCS key file. (*Reviewed by Stephen Frost. Reported by Henrik Feldt.*)

Improvements:

- Allow link-map option to create new links. (*Reviewed by Don Seiler, Stefan Fercot, Chris Bandy. Suggested by Don Seiler.*)
- Increase max index allowed for pg/repo options to 256. (*Reviewed by Cynthia Shang.*)

- Add WebIdentity authentication for AWS S3. (*Reviewed by James Callahan, Reid Thompson, Benjamin Blattberg, Andrew L'Ecuyer.*)
- Report backup file validation errors in backup.info. (*Contributed by Stefan Fercot. Reviewed by David Steele.*)
- Add recovery start time to online backup restore log. (*Reviewed by Tom Swartz, Stefan Fercot. Suggested by Tom Swartz.*)
- Report original error and retries on local job failure. (*Reviewed by Stefan Fercot.*)
- Rename page checksum error to error list in info text output. (*Reviewed by Stefan Fercot.*)
- Add hints to standby replay timeout message. (*Reviewed by Cynthia Shang, Stefan Fercot. Suggested by Leigh Downs.*)

v2.35 Release Notes

Binary Protocol

Released August 23, 2021

IMPORTANT NOTE: The log level for copied files in the backup/restore commands has been changed to detail. This makes the info log level less noisy but if these messages are required then set the log level for the backup/restore commands to detail

Bug Fixes:

- Detect errors in S3 multi-part upload finalize. (*Reviewed by Cynthia Shang, Marco Montagna. Reported by Marco Montagna, Lev Kokotov, Anderson A. Mallmann.*)
- Fix detection of circular symlinks. (*Reviewed by Stefan Fercot. Reported by Rohit Raveendran.*)
- Only pass selected repo options to the remote. (*Reviewed by David Christensen, Cynthia Shang. Reported by Greg Sabino Mullane, David Christensen.*)

Improvements:

- Binary protocol. (*Reviewed by Cynthia Shang.*)
- Automatically create data directory on restore. (*Contributed by Stefan Fercot. Reviewed by David Steele. Suggested by Chris Bandy.*)
- Allow restore --type=lsn. (*Contributed by Stefan Fercot. Reviewed by Cynthia Shang. Suggested by James Coleman.*)
- Change level of backup/restore copied file logging to detail. (*Reviewed by Stefan Fercot. Suggested by Jens Wilke.*)
- Loop while waiting for checkpoint LSN to reach replay LSN. (*Contributed by Stefan Fercot. Reviewed by David Steele. Suggested by Fatih Mencutekin.*)
- Log backup file total and restore size/file total. (*Reviewed by Cynthia Shang.*)

Documentation Bug Fixes:

- Fix incorrect host names in user guide. (*Reviewed by Stefan Fercot. Reported by Greg Sabino Mullane.*)

Documentation Improvements:

- Update contributing documentation and add pull request template. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Rearrange backup documentation in user guide. (*Reviewed by Cynthia Shang.*)
- Clarify restore --type behavior in command reference. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Fix documentation and comment typos. (*Contributed by Eric Radman. Reviewed by David Steele.*)

Test Suite Improvements:

- Add check for test path inside repo path. (*Reviewed by Greg Sabino Mullane. Suggested by Greg Sabino Mullane.*)
- Add CodeQL static code analysis. (*Reviewed by Cynthia Shang.*)
- Update tests to use standard patterns. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

v2.34 Release Notes

PostgreSQL 14 Support

Released June 7, 2021

Bug Fixes:

- Fix issues with leftover spool files from a prior restore. (*Reviewed by Cynthia Shang, Stefan Fercot, Floris van Nee. Reported by Floris van Nee.*)
- Fix issue when checking links for large numbers of tablespaces. (*Reviewed by Cynthia Shang, Avinash Vallarapu. Reported by Avinash Vallarapu.*)
- Free no longer needed remotes so they do not timeout during restore. (*Reviewed by Cynthia Shang. Reported by Francisco Miguel Biete Banon.*)
- Fix help when a valid option is invalid for the specified command. (*Reviewed by Stefan Fercot. Reported by Cynthia Shang.*)

Features:

- Add PostgreSQL 14 support. (*Reviewed by Cynthia Shang.*)
- Add automatic GCS authentication for GCE instances. (*Reviewed by Jan Wieck, Daniel Farina.*)
- Add repo-retention-history option to expire backup history. (*Contributed by Stefan Fercot. Reviewed by Cynthia Shang, David Steele.*)
- Add db-exclude option. (*Contributed by Stefan Fercot. Reviewed by Cynthia Shang.*)

Improvements:

- Change archive expiration logging from detail to info level. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Remove stanza archive spool path on restore. (*Reviewed by Cynthia Shang, Stefan Fercot.*)
- Do not write files atomically or sync paths during backup copy. (*Reviewed by Stephen Frost, Stefan Fercot, Cynthia Shang.*)

Documentation Improvements:

- Update contributing documentation. (*Contributed by Cynthia Shang. Reviewed by David Steele, Stefan Fercot.*)
- Consolidate RHEL/CentOS user guide into a single document. (*Reviewed by Cynthia Shang.*)
- Clarify that repo-s3-role is not an ARN. (*Contributed by Isaac Yuen. Reviewed by David Steele.*)

v2.33 Release Notes

Multi-Repository and GCS Support

Released April 5, 2021

Bug Fixes:

- Fix option warnings breaking async archive-get/archive-push. (*Reviewed by Cynthia Shang. Reported by Lev Kokotov.*)
- Fix memory leak in backup during archive copy. (*Reviewed by Cynthia Shang. Reported by Christian ROUX, Efremov Egor.*)
- Fix stack overflow in cipher passphrase generation. (*Reviewed by Cynthia Shang. Reported by bsiara.*)
- Fix repo-ls / on S3 repositories. (*Reviewed by Cynthia Shang. Reported by Lesovsky Alexey.*)

Features:

- Multiple repository support. (*Contributed by Cynthia Shang, David Steele. Reviewed by Stefan Fercot, Stephen Frost.*)
- GCS support for repository storage. (*Reviewed by Cynthia Shang, Daniel Farina.*)
- Add archive-header-check option. (*Reviewed by Stephen Frost, Cynthia Shang. Suggested by Hans-Jürgen Schönig.*)

Improvements:

- Include recreated system databases during selective restore. (*Contributed by Stefan Fercot. Reviewed by Cynthia Shang.*)
- Exclude content-length from S3 signed headers. (*Reviewed by Cynthia Shang. Suggested by Brian P Bockelman.*)
- Consolidate less commonly used repository storage options. (*Reviewed by Cynthia Shang.*)

- Allow custom config-path default with ./configure --with-configdir. (*Contributed by Michael Schout. Reviewed by David Steele.*)
- Log archive copy during backup. (*Reviewed by Cynthia Shang, Stefan Fercot.*)

Documentation Improvements:

- Update reference to include links to user guide examples. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Update selective restore documentation with caveats. (*Reviewed by Cynthia Shang, Stefan Fercot.*)
- Add compress-type clarification to archive-copy documentation. (*Reviewed by Cynthia Shang, Stefan Fercot.*)
- Add compress-level defaults per compress-type value. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Add note about required NFS settings being the same as PostgreSQL. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

v2.32 Release Notes

Repository Commands

Released February 8, 2021

Bug Fixes:

- Fix resume after partial delete of backup by prior resume. (*Reviewed by Cynthia Shang. Reported by Tom Swartz.*)

Features:

- Add repo-ls command. (*Reviewed by Cynthia Shang, Stefan Fercot.*)
- Add repo-get command. (*Contributed by Stefan Fercot, David Steele. Reviewed by Cynthia Shang.*)
- Add archive-mode-check option. (*Contributed by Stefan Fercot. Reviewed by David Steele, Michael Banck.*)

Improvements:

- Improve archive-get performance. (*Reviewed by Cynthia Shang.*)

Documentation Improvements:

- Improve expire command documentation. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

v2.31 Release Notes

Minor Bug Fixes and Improvements

Released December 7, 2020

Bug Fixes:

- Allow [, #, and space as the first character in database names. (*Reviewed by Stefan Fercot, Cynthia Shang. Reported by Jefferson Alexandre.*)
- Create standby.signal only on PostgreSQL 12 when restore type is standby. (*Fixed by Stefan Fercot. Reviewed by David Steele. Reported by Keith Fiske.*)

Features:

- Expire history files. (*Contributed by Stefan Fercot. Reviewed by David Steele.*)
- Report page checksum errors in info command text output. (*Contributed by Stefan Fercot. Reviewed by Cynthia Shang.*)
- Add repo-azure-endpoint option. (*Reviewed by Cynthia Shang, Brian Peterson. Suggested by Brian Peterson.*)
- Add pg-database option. (*Reviewed by Cynthia Shang.*)

Improvements:

- Improve info command output when a stanza is specified but missing. (*Contributed by Stefan Fercot. Reviewed by Cynthia Shang, David Steele. Suggested by uspen.*)
- Improve performance of large file lists in backup/restore commands. (*Reviewed by Cynthia Shang, Oscar.*)
- Add retries to PostgreSQL sleep when starting a backup. (*Reviewed by Cynthia Shang. Suggested by Vitaliy Kukharik.*)

Documentation Improvements:

- Replace RHEL/CentOS 6 documentation with RHEL/CentOS 8.

v2.30 Release Notes

PostgreSQL 13 Support

Released October 5, 2020

Bug Fixes:

- Error with hints when backup user cannot read pg_settings. (*Reviewed by Stefan Fercot, Cynthia Shang. Reported by Mohamed Insaf K.*)

Features:

- PostgreSQL 13 support. (*Reviewed by Cynthia Shang.*)

Improvements:

- Improve PostgreSQL version identification. (*Reviewed by Cynthia Shang, Stephen Frost.*)
- Improve working directory error message. (*Reviewed by Stefan Fercot.*)
- Add hint about starting the stanza when WAL segment not found. (*Contributed by David Christensen. Reviewed by David Steele.*)

- Add hint for protocol version mismatch. (*Reviewed by Cynthia Shang. Suggested by loop-evgeny.*)

Documentation Improvements:

- Add note that pgBackRest versions must match when running remotely. (*Reviewed by Cynthia Shang. Suggested by loop-evgeny.*)
- Move info command text to the reference and link to user guide. (*Reviewed by Cynthia Shang. Suggested by Christophe Courtois.*)
- Update yum repository path for CentOS/RHEL user guide. (*Contributed by Heath Lord. Reviewed by David Steele.*)

v2.29 Release Notes

Auto S3 Credentials on AWS

Released August 31, 2020

Bug Fixes:

- Suppress errors when closing local/remote processes. Since the command has completed it is counterproductive to throw an error but still **warn** to indicate that something unusual happened. (*Reviewed by Cynthia Shang. Reported by argdenis.*)
- Fix issue with = character in file or database names. (*Reviewed by Bastian Wegge, Cynthia Shang. Reported by Brad Nicholson, Bastian Wegge.*)

Features:

- Automatically retrieve temporary S3 credentials on AWS instances. (*Contributed by David Steele, Stephen Frost. Reviewed by Cynthia Shang, David Youatt, Aleš Zelený, Jeanette Bromage.*)
- Add archive-mode option to disable archiving on restore. (*Reviewed by Stephen Frost. Suggested by Stephen Frost.*)

Improvements:

- PostgreSQL 13 beta3 support. Changes to the control/catalog/WAL versions in subsequent betas may break compatibility but pgBackRest will be updated with each release to keep pace.
- Asynchronous list/remove for S3/Azure storage. (*Reviewed by Cynthia Shang, Stephen Frost.*)
- Improve memory usage of unlogged relation detection in manifest build. (*Reviewed by Cynthia Shang, Stephen Frost, Brad Nicholson, Oscar. Suggested by Oscar, Brad Nicholson.*)
- Proactively close file descriptors after forking async process. (*Reviewed by Stephen Frost, Cynthia Shang.*)
- Delay backup remote connection close until after archive check. (*Contributed by Floris van Nee. Reviewed by David Steele.*)
- Improve detailed error output. (*Reviewed by Cynthia Shang.*)

- Improve TLS error reporting. (*Reviewed by Cynthia Shang, Stephen Frost.*)

Documentation Bug Fixes:

- Add none to compress-type option reference and fix example. (*Reported by Ugo Bellavance, Don Seiler.*)
- Add missing azure type in repo-type option reference. (*Fixed by Don Seiler. Reviewed by David Steele.*)
- Fix typo in repo-cipher-type option reference. (*Fixed by Don Seiler. Reviewed by David Steele.*)

Documentation Improvements:

- Clarify that expire must be run regularly when expire-auto is disabled. (*Reviewed by Douglas J Hunley. Suggested by Douglas J Hunley.*)

v2.28 Release Notes

Azure Repository Storage

Released July 20, 2020

Bug Fixes:

- Fix restore --force acting like --force --delta. This caused restore to replace files based on timestamp and size rather than overwriting, which meant some files that should have been updated were left unchanged. Normal restore and restore --delta were not affected by this issue. (*Reviewed by Cynthia Shang.*)

Features:

- Azure support for repository storage. (*Reviewed by Cynthia Shang, Don Seiler.*)
- Add expire-auto option. This allows automatic expiration after a successful backup to be disabled. (*Contributed by Stefan Fercot. Reviewed by Cynthia Shang, David Steele.*)

Improvements:

- Asynchronous S3 multipart upload. (*Reviewed by Stephen Frost.*)
- Automatic retry for backup, restore, archive-get, and archive-push. (*Reviewed by Cynthia Shang.*)
- Disable query parallelism in PostgreSQL sessions used for backup control. (*Reviewed by Stefan Fercot.*)
- PostgreSQL 13 beta2 support. Changes to the control/catalog/WAL versions in subsequent betas may break compatibility but pgBackRest will be updated with each release to keep pace.
- Improve handling of invalid HTTP response status. (*Reviewed by Cynthia Shang.*)

- Improve error when pg1-path option missing for archive-get command. (*Reviewed by Cynthia Shang.*)
- Add hint when checksum delta is enabled after a timeline switch. (*Reviewed by Matt Bunter, Cynthia Shang.*)
- Use PostgreSQL instead of postmaster where appropriate. (*Reviewed by Cynthia Shang.*)

Documentation Bug Fixes:

- Fix incorrect example for repo-retention-full-type option. (*Reported by Höseyin Sönmez.*)
- Remove internal commands from HTML and man command references. (*Reported by Cynthia Shang.*)

Documentation Improvements:

- Update PostgreSQL versions used to build user guides. Also add version ranges to indicate that a user guide is accurate for a range of PostgreSQL versions even if it was built for a specific version. (*Reviewed by Stephen Frost.*)
- Update FAQ for expiring a specific backup set. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Update FAQ to clarify default PITR behavior. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

v2.27 Release Notes

Expiration Improvements and Compression Drivers

Released May 26, 2020

Bug Fixes:

- Fix issue checking if file links are contained in path links. (*Reviewed by Cynthia Shang. Reported by Christophe Cavallié.*)
- Allow pg-path1 to be optional for synchronous archive-push. (*Reviewed by Cynthia Shang. Reported by Jerome Peng.*)
- The expire command now checks if a stop file is present. (*Fixed by Cynthia Shang. Reviewed by David Steele.*)
- Handle missing reason phrase in HTTP response. (*Reviewed by Cynthia Shang. Reported by Tenuun.*)
- Increase buffer size for lz4 compression flush. (*Reviewed by Cynthia Shang. Reported by Eric Radman.*)
- Ignore pg-host* and repo-host* options for the remote command. (*Reviewed by Cynthia Shang. Reported by Pavel Suderevsky.*)
- Fix possibly missing pg1-* options for the remote command. (*Reviewed by Cynthia Shang. Reported by Andrew L'Ecuyer.*)

Features:

- Time-based retention for full backups. The --repo-retention-full-type option allows retention of full backups based on a time period, specified in days. (*Contributed by Cynthia Shang, Pierre Ducroquet. Reviewed by David Steele.*)
- Ad hoc backup expiration. Allow the user to remove a specified backup regardless of retention settings. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Zstandard compression support. Note that setting compress-type=zst will make new backups and archive incompatible (unrestorable) with prior versions of pgBackRest. (*Reviewed by Cynthia Shang.*)
- bzip2 compression support. Note that setting compress-type=bz2 will make new backups and archive incompatible (unrestorable) with prior versions of pgBackRest. (*Contributed by Stephen Frost. Reviewed by David Steele, Cynthia Shang.*)
- Add backup/expire running status to the info command. (*Contributed by Stefan Fercot. Reviewed by David Steele.*)

Improvements:

- Expire WAL archive only when repo-retention-archive threshold is met. WAL prior to the first full backup was previously expired after the first full backup. Now it is preserved according to retention settings. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Add local MD5 implementation so S3 works when FIPS is enabled. (*Reviewed by Cynthia Shang, Stephen Frost. Suggested by Brian Almeida, John Kelly.*)
- PostgreSQL 13 beta1 support. 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.*)
- Reduce buffer-size default to 1MiB. (*Reviewed by Stephen Frost.*)
- Throw user-friendly error if expire is not run on repository host. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

v2.26 Release Notes

Non-blocking TLS

Released April 20, 2020

Bug Fixes:

- Remove empty subexpression from manifest regular expression. MacOS was not happy about this though other platforms seemed to work fine. (*Fixed by David Raftis. Reviewed by David Steele.*)

Improvements:

- Non-blocking TLS implementation. (*Reviewed by Slava Moudry, Cynthia Shang, Stephen Frost.*)

- Only limit backup copy size for WAL-logged files. The prior behavior could possibly lead to postgresql.conf or postgresql.auto.conf being truncated in the backup. (*Reviewed by Cynthia Shang.*)
- TCP keep-alive options are configurable. (*Suggested by Marc Cousin.*)
- Add io-timeout option. (*Reviewed by Cynthia Shang.*)

v2.25 Release Notes

LZ4 Compression Support

Released March 26, 2020

Features:

- Add lz4 compression support. Note that setting compress-type=lz4 will make new backups and archive incompatible (unrestorable) with prior versions of pgBackRest. (*Reviewed by Cynthia Shang.*)
- Add --dry-run option to the expire command. Use dry-run to see which backups/archive would be removed by the expire command without actually removing anything. (*Contributed by Cynthia Shang, Luca Ferrari. Reviewed by David Steele. Suggested by Marc Cousin.*)

Improvements:

- Improve performance of remote manifest build. (*Suggested by Jens Wilke.*)
- Fix detection of keepalive options on Linux. (*Contributed by Marc Cousin. Reviewed by David Steele.*)
- Add configure host detection to set standards flags correctly. (*Contributed by Marc Cousin. Reviewed by David Steele.*)
- Remove compress/compress-level options from commands where unused. These commands (e.g. restore, archive-get) never used the compress options but allowed them to be passed on the command line. Now they will error when these options are passed on the command line. If these errors occur then remove the unused options. (*Reviewed by Cynthia Shang.*)
- Limit backup file copy size to size reported at backup start. If a file grows during the backup it will be reconstructed by WAL replay during recovery so there is no need to copy the additional data. (*Reviewed by Cynthia Shang.*)

v2.24 Release Notes

Auto-Select Backup Set for Time Target

Released February 25, 2020

Bug Fixes:

- Prevent defunct processes in asynchronous archive commands. (*Reviewed by Stephen Frost. Reported by Adam Brusselback, ejberdecia.*)
- Error when archive-get/archive-push/restore are not run on a PostgreSQL host. (*Reviewed by Stephen Frost. Reported by Jesper St John.*)

- Read HTTP content to eof when size/encoding not specified. (*Reviewed by Cynthia Shang. Reported by Christian ROUX.*)
- Fix resume when the resumable backup was created by Perl. In this case the resumable backup should be ignored, but the C code was not able to load the partial manifest written by Perl since the format differs slightly. Add validations to catch this case and continue gracefully. (*Reported by Kacey Holston.*)

Features:

- Auto-select backup set on restore when time target is specified. Auto-selection is performed only when --set is not specified. If a backup set for the given target time cannot not be found, the latest (default) backup set will be used. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

Improvements:

- Skip pg_internal.init temp file during backup. (*Reviewed by Cynthia Shang. Suggested by Michael Paquier.*)
- Add more validations to the manifest on backup. (*Reviewed by Cynthia Shang.*)

Documentation Improvements:

- Prevent lock-bot from adding comments to locked issues. (*Suggested by Christoph Berg.*)

v2.23 Release Notes

Bug Fix

Released January 27, 2020

Bug Fixes:

- Fix missing files corrupting the manifest. If a file was removed by PostgreSQL during the backup (or was missing from the standby) then the next file might not be copied and updated in the manifest. If this happened then the backup would error when restored. (*Reviewed by Cynthia Shang. Reported by Vitaliy Kukharik.*)

Improvements:

- Use pkg-config instead of xml2-config for libxml2 build options. (*Contributed by David Steele, Adrian Vondendriesch.*)
- Validate checksums are set in the manifest on backup/restore. (*Reviewed by Cynthia Shang.*)

v2.22 Release Notes

Bug Fix

Released January 21, 2020

Bug Fixes:

- Fix error in timeline conversion. The timeline is required to verify WAL segments in the archive after a backup. The conversion was performed base 10 instead of 16, which led to errors when the timeline was 0xA. (*Reported by Lukas Ertl, Eric Veldhuyzen.*)

v2.21 Release Notes

C Migration Complete

Released January 15, 2020

Bug Fixes:

- Fix options being ignored by asynchronous commands. The asynchronous archive-get/archive-push processes were not loading options configured in command configuration sections, e.g. [global:archive-get]. (*Reviewed by Cynthia Shang. Reported by Urs Kramer.*)
- Fix handling of \ in filenames. \ was not being properly escaped when calculating the manifest checksum which prevented the manifest from loading. Since instances of \ in cluster filenames should be rare to nonexistent this does not seem likely to be a serious problem in the field.

Features:

- pgBackRest is now pure C.
- Add pg-user option. Specifies the database user name when connecting to PostgreSQL. If not specified pgBackRest will connect with the local OS user or PGUSER, which was the previous behavior. (*Contributed by Mike Palmiotto. Reviewed by David Steele.*)
- Allow path-style URIs in S3 driver.

Improvements:

- The backup command is implemented entirely in C. (*Reviewed by Cynthia Shang.*)

v2.20 Release Notes

Bug Fixes

Released December 12, 2019

Bug Fixes:

- Fix archive-push/archive-get when PGDATA is symlinked. These commands tried to use cwd() as PGDATA but this would disagree with the path configured in pgBackRest if PGDATA was symlinked. If cwd() does not match the pgBackRest path then chdir() to the path and make sure the next cwd() matches the result from the first call. (*Reported by Stephen Frost, Milosz Suchy.*)

- Fix reference list when backup.info is reconstructed in expire command. Since the backup command is still using the Perl version of reconstruct this issue will not express unless **1)** there is a backup missing from backup.info and **2)** the expire command is run directly instead of running after backup as usual. This unlikely combination of events means this is probably not a problem in the field.
- Fix segfault on unexpected EOF in gzip decompression. (*Reported by Stephen Frost.*)

v2.19 Release Notes

C Migrations and Bug Fixes

Released November 12, 2019

Bug Fixes:

- Fix remote timeout in delta restore. When performing a delta restore on a largely unchanged cluster the remote could timeout if no files were fetched from the repository within protocol-timeout. Add keep-alives to prevent remote timeout. (*Reported by James Sewell, Jens Wilke.*)
- Fix handling of repeated HTTP headers. When HTTP headers are repeated they should be considered equivalent to a single comma-separated header rather than generating an error, which was the prior behavior. (*Reported by donicrosby.*)

Improvements:

- JSON output from the info command is no longer pretty-printed. Monitoring systems can more easily ingest the JSON without linefeeds. External tools such as jq can be used to pretty-print if desired. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- The check command is implemented entirely in C. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

Documentation Improvements:

- Document how to contribute to pgBackRest. (*Contributed by Cynthia Shang, David Steele.*)
- Document maximum version for auto-stop option. (*Contributed by Brad Nicholson. Reviewed by David Steele.*)

Test Suite Improvements:

- Fix container test path being used when --vm=none. (*Suggested by Stephen Frost.*)
- Fix mismatched timezone in expect test. (*Suggested by Stephen Frost.*)
- Don't autogenerate embedded libc code by default. (*Suggested by Stephen Frost.*)

v2.18 Release Notes

PostgreSQL 12 Support

Released October 1, 2019

Features:

- PostgreSQL 12 support.
- Add info command set option for detailed text output. The additional details include databases that can be used for selective restore and a list of tablespaces and symlinks with their default destinations. (*Contributed by Cynthia Shang. Reviewed by David Steele. Suggested by Stephen Frost, ejberdecia.*)
- Add standby restore type. This restore type automatically adds standby_mode=on to recovery.conf for PostgreSQL < 12 and creates standby.signal for PostgreSQL 12, creating a common interface between PostgreSQL versions. (*Reviewed by Cynthia Shang.*)

Improvements:

- The restore command is implemented entirely in C. (*Reviewed by Cynthia Shang.*)

Documentation Improvements:

- Document the relationship between db-timeout and protocol-timeout. (*Contributed by Cynthia Shang. Reviewed by David Steele. Suggested by James Chanco Jr.*)
- Add documentation clarifications regarding standby repositories. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Add FAQ for time-based Point-in-Time Recovery. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

v2.17 Release Notes

C Migrations and Bug Fixes

Released September 3, 2019

Bug Fixes:

- Improve slow manifest build for very large quantities of tables/segments. (*Reported by Jens Wilke.*)
- Fix exclusions for special files. (*Reported by CluelessTechnologist, Janis Puris, Rachid Broum.*)

Improvements:

- The stanza-create/update/delete commands are implemented entirely in C. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- The start/stop commands are implemented entirely in C. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Create log directories/files with 0750/0640 mode. (*Suggested by Damiano Albani.*)

Documentation Bug Fixes:

- Fix yum.p.o package being installed when custom package specified. (*Reported by Joe Ayers, John Harvey.*)

Documentation Improvements:

- Build pgBackRest as an unprivileged user. (*Suggested by Laurenz Albe.*)

v2.16 Release Notes

C Migrations and Bug Fixes

Released August 5, 2019

Bug Fixes:

- Retry S3 RequestTimeTooSkewed errors instead of immediately terminating. (*Reported by sean0101n, Tim Garton, Jesper St John, Aleš Zelený.*)
- Fix incorrect handling of transfer-encoding response to HEAD request. (*Reported by Pavel Suderevsky.*)
- Fix scoping violations exposed by optimizations in gcc 9. (*Reported by Christian Lange, Ned T. Crigler.*)

Features:

- Add repo-s3-port option for setting a non-standard S3 service port.

Improvements:

- The local command for backup is implemented entirely in C. (*Contributed by David Steele, Cynthia Shang.*)
- The check command is implemented partly in C. (*Reviewed by Cynthia Shang.*)

v2.15 Release Notes

C Implementation of Expire

Released June 25, 2019

Bug Fixes:

- Fix archive retention expiring too aggressively. (*Fixed by Cynthia Shang. Reviewed by David Steele. Reported by Mohamad El-Rifai.*)

Improvements:

- The expire command is implemented entirely in C. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- The local command for restore is implemented entirely in C.
- Remove hard-coded PostgreSQL user so $PGUSER works. (*Suggested by Julian Zhang, Janis Puris.*)
- Honor configure --prefix option. (*Suggested by Daniel Westermann.*)

- Rename repo-s3-verify-ssl option to repo-s3-verify-tls. The new name is preferred because pgBackRest does not support any SSL protocol versions (they are all considered to be insecure). The old name will continue to be accepted.

Documentation Improvements:

- Add FAQ to the documentation. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Use wal_level=replica in the documentation for PostgreSQL 9.6. (*Suggested by Patrick McLaughlin.*)

v2.14 Release Notes

Bug Fix and Improvements

Released May 20, 2019

Bug Fixes:

- Fix segfault when process-max > 8 for archive-push/archive-get. (*Reported by Jens Wilke.*)

Improvements:

- Bypass database checks when stanza-delete issued with force. (*Contributed by Cynthia Shang. Reviewed by David Steele. Suggested by hatifnatt.*)
- Add configure script for improved multi-platform support.

Documentation Features:

- Add user guides for CentOS/RHEL 6/7.

v2.13 Release Notes

Bug Fixes

Released April 18, 2019

Bug Fixes:

- Fix zero-length reads causing problems for IO filters that did not expect them. (*Reported by brunre01, Jens Wilke, Tomasz Kontusz, gurugu-ruguru.*)
- Fix reliability of error reporting from local/remote processes.
- Fix Posix/CIFS error messages reporting the wrong filename on write/sync/close.

v2.12 Release Notes

C Implementation of Archive Push

Released April 11, 2019

IMPORTANT NOTE: The new TLS/SSL implementation forbids dots in S3 bucket names per RFC-2818. This security fix is required for compliant hostname verification.

Bug Fixes:

- Fix issues when a path option is / terminated. (*Reported by Marc Cousin.*)
- Fix issues when log-level-file=off is set for the archive-get command. (*Reported by Brad Nicholson.*)
- Fix C code to recognize host:port option format like Perl does. (*Reported by Kyle Nevins.*)
- Fix issues with remote/local command logging options.

Improvements:

- The archive-push command is implemented entirely in C.
- Increase process-max limit to 999. (*Suggested by Rakshitha-BR.*)
- Improve error message when an S3 bucket name contains dots.

Documentation Improvements:

- Clarify that S3-compatible object stores are supported. (*Suggested by Magnus Hagander.*)

v2.11 Release Notes

C Implementation of Archive Get

Released March 11, 2019

Bug Fixes:

- Fix possible truncated WAL segments when an error occurs mid-write. (*Reported by blogh.*)
- Fix info command missing WAL min/max when stanza specified. (*Fixed by Stefan Fercot. Reviewed by David Steele.*)
- Fix non-compliant JSON for options passed from C to Perl. (*Reported by Leo Khomenko.*)

Improvements:

- The archive-get command is implemented entirely in C.
- Enable socket keep-alive on older Perl versions. (*Contributed by Marc Cousin. Reviewed by David Steele.*)
- Error when parameters are passed to a command that does not accept parameters. (*Suggested by Jason O'Donnell.*)
- Add hints when unable to find a WAL segment in the archive. (*Suggested by Hans-Jürgen Schönig.*)
- Improve error when hostname cannot be found in a certificate. (*Suggested by James Badger.*)
- Add additional options to backup.manifest for debugging purposes. (*Contributed by blogh. Reviewed by David Steele.*)

Documentation Improvements:

- Update default documentation version to PostgreSQL 10.

v2.10 Release Notes

Bug Fixes

Released February 9, 2019

Bug Fixes:

- Add unimplemented S3 driver method required for archive-get. (*Reported by mibiio.*)
- Fix check for improperly configured pg-path. (*Reported by James Chanco Jr.*)

v2.09 Release Notes

Minor Improvements and Bug Fixes

Released January 30, 2019

Bug Fixes:

- Fix issue with multiple async status files causing a hard error. (*Reported by Vidhya Gurumoorthi, Joe Ayers, Douglas J Hunley.*)

Improvements:

- The info command is implemented entirely in C. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Simplify info command text message when no stanzas are present. Replace the repository path with "the repository".
- Add _DARWIN_C_SOURCE flag to Makefile for MacOS builds. (*Contributed by Douglas J Hunley. Reviewed by David Steele.*)
- Update address lookup in C TLS client to use modern methods. (*Suggested by Bruno Friedmann.*)
- Include Posix-compliant header for strcasecmp() and fd_set. (*Suggested by ucando.*)

Documentation Bug Fixes:

- Fix hard-coded repository path. (*Reported by Heath Lord.*)

Documentation Improvements:

- Clarify that encryption is always performed client-side. (*Suggested by Bruce Burdick.*)
- Add examples for building a documentation host.
- Allow if in manifest variables, lists, and list items.

v2.08 Release Notes

Minor Improvements and Bug Fixes

Released January 2, 2019

Bug Fixes:

- Remove request for S3 object info directly after putting it. (*Reported by Matt Kunkel.*)
- Correct archive-get-queue-max to be size type. (*Reported by Ronan Dunklau.*)
- Add error message when current user uid/gid does not map to a name. (*Reported by Camilo Aguilar.*)
- Error when --target-action=shutdown specified for PostgreSQL < 9.5.

Improvements:

- Set TCP keepalives on S3 connections. (*Suggested by Ronan Dunklau.*)
- Reorder info command text output so most recent backup is output last. (*Contributed by Cynthia Shang. Reviewed by David Steele. Suggested by Ryan Lambert.*)
- Change file ownership only when required.
- Redact authentication header when throwing S3 errors. (*Suggested by Brad Nicholson.*)

Documentation Improvements:

- Clarify when target-action is effective and PostgreSQL version support. (*Suggested by Keith Fiske.*)
- Clarify that region/endpoint must be configured correctly for the bucket. (*Suggested by Pritam Barhate.*)
- Add documentation for building the documentation.

v2.07 Release Notes

Automatic Backup Checksum Delta

Released November 16, 2018

Bug Fixes:

- Fix issue with archive-push-queue-max not being honored on connection error. (*Reported by Lardière Sébastien.*)
- Fix static WAL segment size used to determine if archive-push-queue-max has been exceeded.
- Fix error after log file open failure when processing should continue. (*Reported by vthriller.*)

Features:

- Automatically enable backup checksum delta when anomalies (e.g. timeline switch) are detected. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

Improvements:

- Retry all S3 5xx errors rather than just 500 internal errors. (*Suggested by Craig A. James.*)

v2.06 Release Notes

Checksum Delta Backup and PostgreSQL 11 Support

Released October 15, 2018

Bug Fixes:

- Fix missing URI encoding in S3 driver. (*Reported by Dan Farrell.*)
- Fix incorrect error message for duplicate options in configuration files. (*Reported by Jesper St John.*)
- Fix incorrectly reported error return in info logging. A return code of 1 from the archive-get was being logged as an error message at info level but otherwise worked correctly.

Features:

- Add checksum delta for incremental backups. Checksum delta backups uses checksums rather than timestamps to determine if files have changed. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- PostgreSQL 11 support, including configurable WAL segment size.

Improvements:

- Ignore all files in a linked tablespace directory except the subdirectory for the current version of PostgreSQL. Previously an error would be generated if other files were present and not owned by the PostgreSQL user.
- Improve info command to display the stanza cipher type. (*Contributed by Cynthia Shang. Reviewed by David Steele. Suggested by Douglas J Hunley.*)
- Improve support for special characters in filenames.
- Allow delta option to be specified in the pgBackRest configuration file. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

Documentation Improvements:

- Use command in authorized_hosts to improve SSH security. (*Suggested by Stephen Frost, Magnus Hagander.*)
- List allowable values for the buffer-size option in the configuration reference. (*Contributed by Cynthia Shang. Reviewed by David Steele. Suggested by Stéphane Schildknecht.*)

v2.05 Release Notes

Environment Variable Options and Exclude Temporary/Unlogged Relations

Released August 31, 2018

Bug Fixes:

- Fix issue where *relative* links in $PGDATA could be stored in the backup with the wrong path. This issue did not affect absolute links and relative tablespace links were caught by other checks. (*Reported by Cynthia Shang.*)
- Remove incompletely implemented online option from the check command. Offline operation runs counter to the purpose of this command, which is to check if archiving and backups are working correctly. (*Reported by Jason O'Donnell.*)
- Fix issue where errors raised in C were not logged when called from Perl. pgBackRest properly terminated with the correct error code but lacked an error message to aid in debugging. (*Reported by Douglas J Hunley.*)
- Fix issue when a boolean option (e.g. delta) was specified more than once. (*Reported by Yogesh Sharma.*)

Features:

- Allow any option to be set in an environment variable. This includes options that previously could only be specified on the command line, e.g. stanza, and secret options that could not be specified on the command-line, e.g. repo1-s3-key-secret.
- Exclude temporary and unlogged relation (table/index) files from backup. Implemented using the same logic as the patches adding this feature to PostgreSQL, 8694cc96 and 920a5e50. Temporary relation exclusion is enabled in PostgreSQL 9.0. Unlogged relation exclusion is enabled in PostgreSQL 9.1, where the feature was introduced. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Allow arbitrary directories and/or files to be excluded from a backup. Misuse of this feature can lead to inconsistent backups so read the --exclude documentation carefully before using. (*Reviewed by Cynthia Shang.*)
- Add log-subprocess option to allow file logging for local and remote subprocesses.
- PostgreSQL 11 Beta 3 support.

Improvements:

- Allow zero-size files in backup manifest to reference a prior manifest regardless of timestamp delta. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Improve asynchronous archive-get/archive-push performance by directly checking status files. (*Contributed by Stephen Frost. Reviewed by David Steele.*)
- Improve error message when a command is missing the stanza option. (*Suggested by Sarah Conway.*)

Documentation Bug Fixes:

- Fix invalid log level in log-path option reference. (*Reported by Camilo Aguilar.*)

Documentation Improvements:

- Stop trying to arrange contributors in release.xml by last/first name. Contributor names have always been presented in the release notes exactly as given, but we tried to assign internal IDs based on last/first name which can be hard to determine and ultimately doesn't make sense. Inspired by Christophe's PostgresOpen 2017 talk, "Human Beings Do Not Have a Primary Key". (*Suggested by Christophe Pettus.*)

Test Suite Improvements:

- Error if LibC build is performed outside the test environment. LibC is no longer required for production builds.

v2.04 Release Notes

Critical Bug Fix for Backup Resume

Released July 5, 2018

IMPORTANT NOTE: This release fixes a critical bug in the backup resume feature. All resumed backups prior to this release should be considered inconsistent. A backup will be resumed after a prior backup fails, unless resume=n has been specified. A resumed backup can be identified by checking the backup log for the message "aborted backup of same type exists, will be cleaned to remove invalid files and resumed". If the message exists, do not use this backup or any backup in the same set for a restore and check the restore logs to see if a resumed backup was restored. If so, there may be inconsistent data in the cluster.

Bug Fixes:

- Fix critical bug in resume that resulted in inconsistent backups. A regression in v0.82 removed the timestamp comparison when deciding which files from the aborted backup to keep on resume. See note above for more details. (*Reported by David Youatt, Yogesh Sharma, Stephen Frost.*)
- Fix error in selective restore when only one user database exists in the cluster. (*Fixed by Cynthia Shang. Reviewed by David Steele. Reported by Nj Baliyan.*)
- Fix non-compliant ISO-8601 timestamp format in S3 authorization headers. AWS and some gateways were tolerant of space rather than zero-padded hours while others were not. (*Fixed by Andrew Schwartz. Reviewed by David Steele.*)

Features:

- PostgreSQL 11 Beta 2 support.

Improvements:

- Improve the HTTP client to set content-length to 0 when not specified by the server. S3 (and gateways) always set content-length or transfer-

encoding but HTTP 1.1 does not require it and proxies (e.g. HAProxy) may not include either. (*Suggested by Adam K. Sumner.*)
- Set search_path = 'pg_catalog' on PostgreSQL connections. (*Suggested by Stephen Frost.*)

Documentation Improvements:

- Create a new section to describe building pgBackRest and build on a separate host.
- Add sample S3 policy to restrict bucket privileges. (*Suggested by Douglas J Hunley, Jason O'Donnell.*)

v2.03 Release Notes

Single Executable to Deploy

Released May 22, 2018

Bug Fixes:

- Fix potential buffer overrun in error message handling. (*Reported by Lætitia.*)
- Fix archive write lock being taken for the synchronous archive-get command. (*Reported by uspen.*)

Improvements:

- Embed exported C functions and Perl modules directly into the pgBackRest executable.
- Use time_t instead of ___time_t for better portability. (*Suggested by Nick Floersch.*)
- Print total runtime in milliseconds at command end.

v2.02 Release Notes

Parallel Asynchronous Archive Get and Configuration Includes

Released May 6, 2018

Bug Fixes:

- Fix directory syncs running recursively when only the specified directory should be synced. (*Reported by Craig A. James.*)
- Fix archive-copy throwing "path not found" error for incr/diff backups. (*Reported by yummyliu, Vitaliy Kukharik.*)
- Fix failure in manifest build when two or more files in PGDATA are linked to the same directory. (*Reported by Vitaliy Kukharik.*)
- Fix delta restore failing when a linked file is missing.
- Fix rendering of key/value and list options in help. (*Reported by Clinton Adams.*)

Features:

- Add asynchronous, parallel archive-get. This feature maintains a queue of WAL segments to help reduce latency when PostgreSQL requests a WAL segment with restore_command.
- Add support for additional pgBackRest configuration files. The directory is specified by the --config-include-path option. Add --config-path option for overriding the default base path of the --config and --config-include-path option. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Add repo-s3-token option to allow temporary credentials tokens to be configured. pgBackRest currently has no way to request new credentials so the entire command (e.g. backup, restore) must complete before the credentials expire. (*Contributed by Yogesh Sharma. Reviewed by David Steele.*)

Improvements:

- Update the archive-push-queue-max, manifest-save-threshold, and buffer-size options to accept values in KB, MB, GB, TB, or PB where the multiplier is a power of 1024. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Make backup/restore path sync more efficient. Scanning the entire directory can be very expensive if there are a lot of small tables. The backup manifest contains the path list so use it to perform syncs instead of scanning the backup/restore path.
- Show command parameters as well as command options in initial info log message.
- Rename archive-queue-max option to archive-push-queue-max. This is consistent with the new archive-get-queue-max option. The old option name will continue to be accepted.

Documentation Bug Fixes:

- Update docs with 32-bit support and caveats. 32-bit support was added in v1.26. (*Reported by Viorel Tabara.*)

Documentation Improvements:

- Add monitoring examples using PostgreSQL and jq. (*Suggested by Stephen Frost, Brian Faherty.*)
- Add example of command section usage to archiving configuration. (*Suggested by Christophe Courtois.*)
- Remove documentation describing info --output=json as experimental.
- Update out-of-date description for the spool-path option.

Test Suite Features:

- Use lcov for C unit test coverage reporting. Switch from Devel::Cover because it would not report on branch coverage for reports converted from gcov. Incomplete branch coverage for a module now generates an error. Coverage of unit tests is not displayed in the report unless they are incomplete for either statement or branch coverage.

v2.01 Release Notes

Minor Bug Fixes and Improvements

Released March 19, 2018

Bug Fixes:

- Fix --target-action and --recovery-option options being reported as invalid when restoring with --type=immediate. (*Reported by Brad Nicholson.*)
- Immediately error when a secure option (e.g. repo1-s3-key) is passed on the command line. Since pgBackRest would not pass secure options on to sub-processes an obscure error was thrown. The new error is much clearer and provides hints about how to fix the problem. Update command documentation to omit secure options that cannot be specified on the command-line. (*Reported by Brad Nicholson.*)
- Fix issue passing --no-config to embedded Perl. (*Reported by Ibrahim Edib Kokdemir.*)
- Fix issue where specifying log-level-stderr > warn would cause a local/remote process to error on exit due to output found on stderr when none was expected. The max value for a local/remote process is now error since there is no reason for these processes to emit warnings. (*Reported by Clinton Adams.*)
- Fix manifest test in the check command when tablespaces are present. (*Fixed by Cynthia Shang. Reviewed by David Steele. Reported by Thomas Flatley.*)

Improvements:

- Error when multiple arguments are set in the config file for an option that does not accept multiple arguments. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Remove extraneous sudo commands from src/Makefile. (*Contributed by Adrian Vondendriesch. Reviewed by David Steele.*)

Documentation Improvements:

- Show index in examples for indexed options, i.e. repo-*, pg-*. (*Suggested by Stephen Frost.*)
- Simplify table of contents on command page by only listing commands. (*Suggested by Stephen Frost.*)
- Remove references to the C library being optional.

Test Suite Features:

- Add CentOS/RHEL package builds.
- Use clang for static code analysis. Nothing found initially except for some functions that should have been marked ___noreturn___.

v2.00 Release Notes

Performance Improvements for Archive Push

Released February 23, 2018

Features:

- The archive-push command is now partially coded in C which allows the PostgreSQL archive_command to run significantly faster when processing status messages from the asynchronous archive process. (*Reviewed by Cynthia Shang.*)

Improvements:

- Improve check command to verify that the backup manifest can be built. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Improve performance of HTTPS client. Buffering now takes the pending bytes on the socket into account (when present) rather than relying entirely on select(). In some instances the final bytes would not be flushed until the connection was closed.
- Improve S3 delete performance. The constant S3_BATCH_MAX had been replaced with a hard-coded value of 2, probably during testing.
- Allow any non-command-line option to be reset to default on the command-line. This allows options in pgbackrest.conf to be reset to default which reduces the need to write new configuration files for specific needs.
- The C library is now required. This eliminates conditional loading and eases development of new library features.
- The pgbackrest executable is now a C binary instead of Perl. This allows certain time-critical commands (like async archive-push) to run more quickly.
- Rename db-* options to pg-* and backup-* options to repo-* to improve consistency. repo-* options are now indexed although currently only one is allowed.

Documentation Features:

- All clusters in the documentation are initialized with checksums.

Documentation Improvements:

- List deprecated option names in documentation and command-line help.
- Clarify that S3 buckets must be created by the user. (*Suggested by David Youatt.*)

v1.29 Release Notes

Critical Bug Fix for Backup Resume

Released July 5, 2018

IMPORTANT NOTE: This release fixes a critical bug in the backup resume feature. All resumed backups prior to this release should be considered inconsistent. A backup will be resumed after a prior backup fails, unless resume=n has been specified. A resumed backup can be identified by checking the backup

log for the message "aborted backup of same type exists, will be cleaned to remove invalid files and resumed". If the message exists, do not use this backup or any backup in the same set for a restore and check the restore logs to see if a resumed backup was restored. If so, there may be inconsistent data in the cluster.

Bug Fixes:

- Fix critical bug in resume that resulted in inconsistent backups. A regression in v0.82 removed the timestamp comparison when deciding which files from the aborted backup to keep on resume. See note above for more details. (*Reported by David Youatt, Yogesh Sharma, Stephen Frost.*)
- Fix non-compliant ISO-8601 timestamp format in S3 authorization headers. AWS and some gateways were tolerant of space rather than zero-padded hours while others were not. (*Fixed by Andrew Schwartz. Reviewed by David Steele.*)
- Fix directory syncs running recursively when only the specified directory should be synced. (*Reported by Craig A. James.*)
- Fix --target-action and --recovery-option options being reported as invalid when restoring with --type=immediate. (*Reported by Brad Nicholson.*)
- Fix archive-copy throwing "path not found" error for incr/diff backups. (*Reported by yummyliu, Vitaliy Kukharik.*)
- Fix failure in manifest build when two or more files in PGDATA are linked to the same directory. (*Reported by Vitaliy Kukharik.*)
- Fix delta restore failing when a linked file was missing.
- Fix error in selective restore when only one user database exists in the cluster. (*Fixed by Cynthia Shang. Reviewed by David Steele. Reported by Nj Baliyan.*)

Improvements:

- Improve the HTTP client to set content-length to 0 when not specified by the server. S3 (and gateways) always set content-length or transfer-encoding but HTTP 1.1 does not require it and proxies (e.g. HAProxy) may not include either. (*Suggested by Adam K. Sumner.*)
- Improve performance of HTTPS client. Buffering now takes the pending bytes on the socket into account (when present) rather than relying entirely on select(). In some instances the final bytes would not be flushed until the connection was closed.
- Improve S3 delete performance. The constant S3_BATCH_MAX had been replaced with a hard-coded value of 2, probably during testing.
- Make backup/restore path sync more efficient. Scanning the entire directory can be very expensive if there are a lot of small tables. The backup manifest contains the path list so use it to perform syncs instead of scanning the backup/restore path. Remove recursive path sync functionality since it is no longer used.

Documentation Bug Fixes:

- Update docs with 32-bit support and caveats. 32-bit support was added in v1.26. (*Reported by Viorel Tabara.*)

Documentation Improvements:

- Clarify that S3 buckets must be created by the user. (*Suggested by David Youatt.*)
- Update out-of-date description for the spool-path option.

v1.28 Release Notes

Stanza Delete

Released February 1, 2018

Bug Fixes:

- Fixed inability to restore a single database contained in a tablespace using --db-include. (*Fixed by Cynthia Shang. Reviewed by David Steele. Reported by Chiranjeevi Ravilla.*)
- Ensure latest db-id is selected on when matching archive.info to backup.info. This provides correct matching in the event there are system-id and db-version duplicates (e.g. after reverting a pg_upgrade). (*Fixed by Cynthia Shang. Reviewed by David Steele. Reported by Adam K. Sumner.*)
- Fixed overly chatty error message when reporting an invalid command. (*Reported by Jason O'Donnell.*)

Features:

- Add stanza-delete command to cleanup unused stanzas. (*Contributed by Cynthia Shang. Reviewed by David Steele. Suggested by Magnus Hagander.*)

Improvements:

- Improve stanza-create command so that it does not error when the stanza already exists. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

Documentation Improvements:

- Update stanza-create --force documentation to urge caution when using. (*Suggested by Jason O'Donnell.*)

v1.27 Release Notes

Bug Fixes and Documentation

Released December 19, 2017

Bug Fixes:

- Fixed an issue that suppressed locality errors for backup and restore. When a backup host is present, backups should only be allowed on the backup host and restores should only be allowed on the database host

unless an alternate configuration is created that ignores the remote host. (*Reported by Lardière Sébastien.*)
- Fixed an issue where WAL was not expired on PostgreSQL 10. This was caused by a faulty regex that expected all PostgreSQL major versions to be X.X. (*Reported by Adam Brusselback.*)
- Fixed an issue where the --no-config option was not passed to child processes. This meant the child processes would still read the local config file and possibly cause unexpected behaviors.
- Fixed info command to eliminate "db (prior)" output if no backups or archives exist for a prior version of the cluster. (*Fixed by Cynthia Shang. Reviewed by David Steele. Reported by Stephen Frost.*)

Documentation Features:

- Document the relationship between the archive-copy and archive-check options. (*Suggested by Markus Nullmeier.*)
- Improve archive-copy reference documentation.

v1.26 Release Notes

Repository Encryption

Released November 21, 2017

Bug Fixes:

- Fixed an issue that could cause copying large manifests to fail during restore. (*Reported by Craig A. James.*)
- Fixed incorrect WAL offset for 32-bit architectures. (*Fixed by Javier Wilson. Reviewed by David Steele.*)
- Fixed an issue retrieving WAL for old database versions. After a stanza-upgrade it should still be possible to restore backups from the previous version and perform recovery with archive-get. However, archive-get only checked the most recent db version/id and failed. Also clean up some issues when the same db version/id appears multiple times in the history. (*Fixed by Cynthia Shang. Reviewed by David Steele. Reported by Clinton Adams.*)
- Fixed an issue with invalid backup groups being set correctly on restore. If the backup cannot map a group to a name it stores the group in the manifest as false then uses either the owner of $PGDATA to set the group during restore or failing that the group of the current user. This logic was not working correctly because the selected group was overwriting the user on restore leaving the group undefined and the user incorrectly set to the group. (*Reported by Jeff McCormick.*)
- Fixed an issue passing parameters to remotes. When more than one db was specified the path, port, and socket path would for db1 were passed no matter which db was actually being addressed. (*Reported by uspen.*)

Features:

- Repository encryption support. (*Contributed by Cynthia Shang, David Steele.*)

Improvements:

- Disable gzip filter when --compress-level-network=0. The filter was used with compress level set to 0 which added overhead without any benefit.
- Inflate performance improvement for gzip filter.

Documentation Features:

- Add template to improve initial information gathered for issue submissions. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

Documentation Improvements:

- Clarify usage of the archive-timeout option and describe how it is distinct from the PostgreSQL archive_timeout setting. (*Contributed by Cynthia Shang. Reviewed by David Steele. Suggested by Keith Fiske.*)

Test Suite Features:

- Automated tests for 32-bit i386/i686 architecture.

v1.25 Release Notes

S3 Performance Improvements

Released October 24, 2017

Bug Fixes:

- Fix custom settings for compress-level option being ignored. (*Reported by Jens Wilke.*)
- Remove error when overlapping timelines are detected. Overlapping timelines are valid in many Point-in-Time-Recovery (PITR) scenarios. (*Reported by blogh.*)
- Fix instances where database-id was not rendered as an integer in JSON info output. (*Fixed by Cynthia Shang. Reviewed by David Steele. Reported by Jason O'Donnell.*)

Features:

- Improve performance of list requests on S3. Any beginning literal portion of a filter expression is used to generate a search prefix which often helps keep the request small enough to avoid rate limiting. (*Suggested by Mihail Shvein.*)

Test Suite Features:

- Add I/O performance tests.

v1.24 Release Notes

New Backup Exclusions

Released September 28, 2017

Bug Fixes:

- Fixed an issue where warnings were being emitted in place of lower priority log messages during backup from standby initialization. (*Reported by uspen.*)
- Fixed an issue where some db-* options (e.g. db-port) were not being passed to remotes. (*Reported by uspen.*)

Features:

- Exclude contents of pg_snapshots, pg_serial, pg_notify, and pg_dynshmem from backup since they are rebuilt on startup.
- Exclude pg_internal.init files from backup since they are rebuilt on startup.

Improvements:

- Open log file after async process is completely separated from the main process to prevent the main process from also logging to the file. (*Suggested by Jens Wilke.*)

Documentation Features:

- Add passwordless SSH configuration.

Documentation Improvements:

- Rename master to primary in documentation to align with PostgreSQL convention.

v1.23 Release Notes

Multiple Standbys and PostgreSQL 10 Support

Released September 3, 2017

Bug Fixes:

- Fixed an issue that could cause compression to abort on growing files. (*Reported by Jesper St John, Aleksandr Rogozin.*)
- Fixed an issue with keep-alives not being sent to the remote from the local process. (*Reported by William Cox.*)

Features:

- Up to seven standbys can be configured for backup from standby. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- PostgreSQL 10 support.
- Allow content-length (in addition to chunked encoding) when reading XML data to improve compatibility with third-party S3 gateways. (*Suggested by Victor Gdalevich.*)

Improvements:

- Increase HTTP timeout for S3.
- Add HTTP retries to harden against transient S3 network errors.

Documentation Bug Fixes:

- Fixed document generation to include section summaries on the Configuration page. (*Fixed by Cynthia Shang. Reviewed by David Steele.*)

v1.22 Release Notes

Fixed S3 Retry

Released August 9, 2017

Bug Fixes:

- Fixed authentication issue in S3 retry.

v1.21 Release Notes

Improved Info Output and SSH Port Option

Released August 8, 2017

Bug Fixes:

- The archive_status directory is now recreated on restore to support PostgreSQL 8.3 which does not recreate it automatically like more recent versions do. (*Reported by Stephen Frost.*)
- Fixed an issue that could cause the empty archive directory for an old PostgreSQL version to be left behind after a stanza-upgrade. (*Fixed by Cynthia Shang. Reviewed by David Steele.*)

Features:

- Modified the info command (both text and JSON output) to display the archive ID and minimum/maximum WAL currently present in the archive for the current and prior, if any, database cluster version. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Added --backup-ssh-port and --db-ssh-port options to support non-default SSH ports. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

Improvements:

- Retry when S3 returns an internal error (500).

Documentation Bug Fixes:

- Fix description of --online based on the command context.

Documentation Features:

- Add creation of /etc/pgbackrest.conf to manual installation instructions.

Documentation Improvements:

- Move repository options into a separate section in command/command-line help. (*Suggested by Stephen Frost.*)

v1.20 Release Notes

Critical 8.3/8.4 Bug Fix

Released June 27, 2017

IMPORTANT NOTE: PostgreSQL 8.3 and 8.4 installations utilizing tablespaces should upgrade immediately from any v1 release and run a full backup. A bug prevented tablespaces from being backed up on these versions only. PostgreSQL 9.0

Bug Fixes:

- Fixed an issue that prevented tablespaces from being backed up on PostgreSQL 8.4.
- Fixed missing flag in C library build that resulted in a mismatched binary on 32-bit systems. (*Reported by Adrian Vondendriesch.*)

Features:

- Add s3-repo-ca-path and s3-repo-ca-file options to accommodate systems where CAs are not automatically found by IO::Socket::SSL, i.e. RHEL7, or to load custom CAs. (*Suggested by Scott Frazer.*)

Test Suite Features:

- Add documentation builds to CI.

v1.19 Release Notes

S3 Support

Released June 12, 2017

Bug Fixes:

- Fixed the info command so the WAL archive min/max displayed is for the current database version. (*Fixed by Cynthia Shang. Reviewed by David Steele.*)
- Fixed the backup command so the backup-standby option is reset (and the backup proceeds on the primary) if the standby is not configured and/or reachable. (*Fixed by Cynthia Shang. Reviewed by David Steele.*)
- Fixed config warnings raised from a remote process causing errors in the master process. (*Fixed by Cynthia Shang. Reviewed by David Steele.*)

Features:

- Amazon S3 repository support. (*Reviewed by Cynthia Shang.*)

Documentation Bug Fixes:

- Changed invalid max-archive-mb option in configuration reference to archive-queue-max.
- Fixed missing sudo in installation section. (*Fixed by Lætitia. Reviewed by David Steele.*)

v1.18 Release Notes

Stanza Upgrade, Refactoring, and Locking Improvements

Released April 12, 2017

Bug Fixes:

- Fixed an issue where read-only operations that used local worker processes (i.e. restore) were creating write locks that could interfere with parallel archive-push. (*Reported by Jens Wilke.*)

Features:

- Added the stanza-upgrade command to provide a mechanism for upgrading a stanza after upgrading to a new major version of PostgreSQL. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Added validation of pgbackrest.conf to display warnings if options are not valid or are not in the correct section. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

Improvements:

- Simplify locking scheme. Now, only the master process will hold write locks (for archive-push and backup commands) and not all local and remote worker processes as before.
- Do not set timestamps of files in the backup directories to match timestamps in the cluster directory. This was originally done to enable backup resume, but that process is now implemented with checksums.
- Improved error message when the restore command detects the presence of postmaster.pid. (*Suggested by Yogesh Sharma.*)
- Renumber return codes between 25 and 125 to avoid PostgreSQL interpreting some as fatal signal exceptions. (*Suggested by Yogesh Sharma.*)

v1.17 Release Notes

Page Checksum Bug Fix

Released March 13, 2017

Bug Fixes:

- Fixed an issue where newly initialized (but unused) pages would cause page checksum warnings. (*Reported by Stephen Frost.*)

v1.16 Release Notes

Page Checksum Improvements, CI, and Package Testing

Released March 2, 2017

Bug Fixes:

- Fixed an issue where tables over 1GB would report page checksum warnings after the first segment. (*Reported by Stephen Frost.*)
- Fixed an issue where databases created with a non-default tablespace would raise bogus warnings about pg_filenode.map and pg_internal.init not being page aligned. (*Reported by blogh.*)

Test Suite Features:

- Continuous integration using travis-ci.
- Automated builds of Debian packages for all supported distributions.

v1.15 Release Notes

Refactoring and Bug Fixes

Released February 13, 2017

Bug Fixes:

- Fixed a regression introduced in v1.13 that could cause backups to fail if files were removed (e.g. tables dropped) while the manifest was being built. (*Reported by Navid Golpayegani.*)

v1.14 Release Notes

Refactoring and Bug Fixes

Released February 13, 2017

Bug Fixes:

- Fixed an issue where an archive-push error would not be retried and would instead return errors to PostgreSQL indefinitely (unless the .error file was manually deleted). (*Reported by Jens Wilke.*)
- Fixed a race condition in parallel archiving where creation of new paths generated an error when multiple processes attempted to do so at the same time. (*Reported by Jens Wilke.*)

Improvements:

- Improved performance of wal archive min/max provided by the info command. (*Suggested by Jens Wilke.*)

Documentation Features:

- Updated async archiving documentation to more accurately describe how the new method works and how it differs from the old method. (*Suggested by Jens Wilke.*)

v1.13 Release Notes

Parallel Archiving, Stanza Create, Improved Info and Check

Released February 5, 2017

IMPORTANT NOTE: The new implementation of asynchronous archiving no longer copies WAL to a separate queue. If there is any WAL left over in the old queue after upgrading to 1.13, it will be abandoned and **not** pushed to the repository.

To prevent this outcome, stop archiving by setting archive_command = false. Next, drain the async queue by running pgbackrest --stanza=[stanza-name] archive-push and wait for the process to complete. Check that the queue in [spool-path]/archive/[stanza-name]/out is empty. Finally, install 1.13 and restore the original archive_command.

IMPORTANT NOTE: The stanza-create command is not longer optional and must be executed before backup or archiving can be performed on a **new** stanza. Pre-existing stanzas do not require stanza-create to be executed.

Bug Fixes:

- Fixed const assignment giving compiler warning in C library. (*Fixed by Adrian Vondendriesch. Reviewed by David Steele.*)
- Fixed a few directory syncs that were missed for the --repo-sync option.
- Fixed an issue where a missing user/group on restore could cause an "uninitialized value" error in File->owner(). (*Reported by Leonardo GG Avellar.*)
- Fixed an issue where protocol mismatch errors did not output the expected value.
- Fixed a spurious archive-get log message that indicated an exit code of 1 was an abnormal termination.

Features:

- Improved, multi-process implementation of asynchronous archiving.
- Improved stanza-create command so that it can repair broken repositories in most cases and is robust enough to be made mandatory. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Improved check command to run on a standby, though only basic checks are done because pg_switch_xlog() cannot be executed on a replica. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Added archive and backup WAL ranges to the info command.
- Added warning to update pg_tablespace.spclocation when remapping tablespaces in PostgreSQL < 9.2. (*Contributed by blogh. Reviewed by David Steele.*)
- Remove remote lock requirements for the archive-get, restore, info, and check commands since they are read-only operations. (*Suggested by Michael Vitale.*)

Improvements:

- Log file banner is not output until the first log entry is written. (*Suggested*

by Jens Wilke.)
- Reduced the likelihood of torn pages causing a false positive in page checksums by filtering on start backup LSN.
- Remove Intel-specific optimization from C library build flags. (*Contributed by Adrian Vondendriesch. Reviewed by David Steele.*)
- Remove --lock option. This option was introduced before the lock directory could be located outside the repository and is now obsolete.
- Added --log-timestamp option to allow timestamps to be suppressed in logging. This is primarily used to avoid filters in the automated documentation.
- Return proper error code when unable to convert a relative path to an absolute path. (*Suggested by Yogesh Sharma.*)

Documentation Features:

- Added documentation to the User Guide for the process-max option. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

v1.12 Release Notes

Page Checksums, Configuration, and Bug Fixes

Released December 12, 2016

IMPORTANT NOTE: In prior releases it was possible to specify options on the command-line that were invalid for the current command without getting an error. An error will now be generated for invalid options so it is important to carefully check command-line options in your environment to prevent disruption.

Bug Fixes:

- Fixed an issue where options that were invalid for the specified command could be provided on the command-line without generating an error. The options were ignored and did not cause any change in behavior, but it did lead to some confusion. Invalid options will now generate an error. (*Reported by Nikhilchandra Kulkarni.*)
- Fixed an issue where internal symlinks were not being created for tablespaces in the repository. This issue was only apparent when trying to bring up clusters in-place manually using filesystem snapshots and did not affect normal backup and restore.
- Fixed an issue that prevented errors from being output to the console before the logging system was initialized, i.e. while parsing options. Error codes were still being returned accurately so this would not have made a process look like it succeeded when it did not. (*Reported by Adrian Vondendriesch.*)
- Fixed an issue where the db-port option specified on the backup server would not be properly passed to the remote unless it was from the first configured database. (*Reported by Michael Vitale.*)

Features:

- Added the --checksum-page option to allow pgBackRest to validate page checksums in data files when checksums are enabled on PostgreSQL >= 9.3. Note that this functionality requires a C library which may not initially be available in OS packages. The option will automatically be enabled when the library is present and checksums are enabled on the cluster. (*Suggested by Stephen Frost.*)
- Added the --repo-link option to allow internal symlinks to be suppressed when the repository is located on a filesystem that does not support symlinks. This does not affect any pgBackRest functionality, but the convenience link latest will not be created and neither will internal tablespace symlinks, which will affect the ability to bring up clusters in-place manually using filesystem snapshots.
- Added the --repo-sync option to allow directory syncs in the repository to be disabled for file systems that do not support them, e.g. NTFS.
- Added a predictable log entry to signal that a command has completed successfully. For example a backup ends successfully with: INFO: backup command end: completed successfully. (*Suggested by Jens Wilke.*)

Improvements:

- For simplicity, the pg_control file is now copied with the rest of the files instead of by itself of at the end of the process. The backup command does not require this behavior and the restore copies to a temporary file which is renamed at the end of the restore.

Documentation Bug Fixes:

- Fixed an issue that suppressed exceptions in PDF builds.
- Fixed regression in section links introduced in v1.10.

Documentation Features:

- Added Retention to QuickStart section.

v1.11 Release Notes

Bug Fix for Asynchronous Archiving Efficiency

Released November 17, 2016

Bug Fixes:

- Fixed an issue where asynchronous archiving was transferring one file per execution instead of transferring files in batches. This regression was introduced in v1.09 and affected efficiency only, all WAL segments were correctly archived in asynchronous mode. (*Reported by Stephen Frost.*)

v1.10 Release Notes

Stanza Creation and Minor Bug Fixes

Released November 8, 2016

Bug Fixes:

- Fixed an issue where a backup could error if no changes were made to a database between backups and only pg_control changed.
- Fixed an issue where tablespace paths with the same prefix would cause an invalid link error. (*Reported by Nikhilchandra Kulkarni.*)

Features:

- Added the stanza-create command to formalize creation of stanzas in the repository. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

Improvements:

- Removed extraneous use lib directives from Perl modules. (*Suggested by Devrim Gündüz.*)

v1.09 Release Notes

9.6 Support, Configurability, and Bug Fixes

Released October 10, 2016

Bug Fixes:

- Fixed the check command to prevent an error message from being logged if the backup directory does not exist. (*Fixed by Cynthia Shang. Reviewed by David Steele.*)
- Fixed error message to properly display the archive command when an invalid archive command is detected. (*Reported by Jason O'Donnell.*)
- Fixed an issue where the async archiver would not be started if archive-push did not have enough space to queue a new WAL segment. This meant that the queue would never be cleared without manual intervention (such as calling archive-push directly). PostgreSQL now receives errors when there is not enough space to store new WAL segments but the async process will still be started so that space is eventually freed. (*Reported by Jens Wilke.*)
- Fixed a remote timeout that occurred when a local process generated checksums (during resume or restore) but did not copy files, allowing the remote to go idle. (*Reported by Jens Wilke.*)

Features:

- Non-exclusive backups will automatically be used on PostgreSQL 9.6.
- Added the cmd-ssh option to allow the ssh client to be specified. (*Suggested by Jens Wilke.*)
- Added the log-level-stderr option to control whether console log messages are sent to stderr or stdout. By default this is set to warn which represents a change in behavior from previous versions, even though it may be more intuitive. Setting log-level-stderr=off will preserve the old behavior. (*Suggested by Sascha Biberhofer.*)

- Set application_name to "pgBackRest [command]" for database connections. (*Suggested by Jens Wilke.*)
- Check that archive_mode is enabled when archive-check option enabled.

Improvements:

- Clarified error message when unable to acquire pgBackRest advisory lock to make it clear that it is not a PostgreSQL backup lock. (*Suggested by Jens Wilke.*)
- pgBackRest version number included in command start INFO log output.
- Process ID logged for local process start/stop INFO log output.

Documentation Features:

- Added archive-timeout option documentation to the user guide. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

v1.08 Release Notes

Bug Fixes and Log Improvements

Released September 14, 2016

Bug Fixes:

- Fixed an issue where local processes were not disconnecting when complete and could later timeout. (*Reported by Todd Vernick.*)
- Fixed an issue where the protocol layer could timeout while waiting for WAL segments to arrive in the archive. (*Reported by Todd Vernick.*)

Improvements:

- Cache file log output until the file is created to create a more complete log.

v1.07 Release Notes

Thread to Process Conversion and Bug Fixes

Released September 7, 2016

Bug Fixes:

- Fixed an issue where tablespaces were copied from the primary during standby backup.
- Fixed the check command so backup info is checked remotely and not just locally. (*Fixed by Cynthia Shang. Reviewed by David Steele.*)
- 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. Reviewed by David Steele.*)

Features:

- Converted Perl threads to processes to improve compatibility and performance.
- Exclude contents of $PGDATA/pg_replslot directory so that replication slots on the primary do not become part of the backup.
- The archive-start and archive-stop settings are now filled in backup.manifest even when archive-check=n. (*Suggested by Jens Wilke.*)
- Additional warnings when archive retention settings may not have the intended effect or would allow indefinite retention. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- 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.

Documentation Bug Fixes:

- Fixed minor documentation reproducibility issues related to binary paths.

Documentation Features:

- Documentation for archive retention. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

v1.06 Release Notes

Backup from Standby and Bug Fixes

Released August 25, 2016

Bug Fixes:

- Fixed an issue where a tablespace link that referenced another link would not produce an error, but instead skip the tablespace entirely. (*Reported by Michael Vitale.*)
- Fixed an issue where options that should not allow multiple values could be specified multiple times in pgbackrest.conf without an error being raised. (*Reported by Michael Vitale.*)
- Fixed an issue where the protocol-timeout option was not automatically increased when the db-timeout option was increased. (*Reported by Todd Vernick.*)

Features:

- Backup from a standby cluster. A connection to the primary cluster is still required to start/stop the backup and copy files that are not replicated, but the vast majority of files are copied from the standby in order to reduce load on the primary.
- More flexible configuration for databases. Master and standby can both be configured on the backup server and pgBackRest will automatically determine which is the primary. This means no configuration changes for

backup are required after failing over from a primary to standby when a separate backup server is used.
- Exclude directories during backup that are cleaned, recreated, or zeroed by PostgreSQL at startup. These include pgsql_tmp and pg_stat_tmp. The postgresql.auto.conf.tmp file is now excluded in addition to files that were already excluded: backup_label.old, postmaster.opts, postmaster.pid, recovery.conf, recovery.done.
- Experimental support for non-exclusive backups in PostgreSQL 9.6 beta4. Changes to the control/catalog/WAL versions in subsequent betas may break compatibility but pgBackRest will be updated with each release to keep pace.

Improvements:

- Improve error message for links that reference links in manifest build.
- Added hints to error message when relative paths are detected in archive-push or archive-get.
- Improve backup log messages to indicate which host the files are being copied from.

v1.05 Release Notes

Bug Fix for Tablespace Link Checking

Released August 9, 2016

Bug Fixes:

- Fixed an issue where tablespace paths that had $PGDATA as a substring would be identified as a subdirectories of $PGDATA even when they were not. Also hardened relative path checking a bit. (*Reported by Chris Fort.*)

Documentation Features:

- Added documentation for scheduling backups with cron. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

Documentation Improvements:

- Moved the backlog from the pgBackRest website to the GitHub repository wiki. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

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. Reviewed by David Steele.*)
- Fixed an issue where a remote could try to start its own remote when the backup-host option was not present in pgbackrest.conf on the database server. (*Reported by Lardière Sébastien.*)
- Fixed an issue where the contents of pg_xlog were being backed up if the directory was symlinked. This didn't cause any issues during restore but was a waste of space.
- Fixed an invalid log() call in lock routines.

Features:

- Experimental support for non-exclusive backups in PostgreSQL 9.6 beta3. Changes to the control/catalog/WAL versions in subsequent betas may break compatibility but pgBackRest will be updated with each release to keep pace.

Improvements:

- Suppress banners on SSH protocol connections.
- Improved remote error messages to identify the host where the error was raised.
- All remote types now take locks. The exceptions date to when the test harness and pgBackRest were running in the same VM and no longer apply.

Documentation Features:

- Added clarification on why the default for the backrest-user option is backrest. (*Suggested by Michael Vitale.*)
- Updated information about package availability on supported platforms. (*Suggested by Michael Vitale.*)

v1.03 Release Notes

Check Command and Bug Fixes

Released July 2, 2016

Bug Fixes:

- Fixed an issue where keep-alives could be starved out by lots of small files during multi-threaded backup. They were also completely absent from single/multi-threaded backup resume and restore checksumming. (*Reported by Janice Parkinson, Chris Barber.*)
- Fixed an issue where the expire command would refuse to run when explicitly called from the command line if the db-host option was set. This was

not an issue when expire was run automatically after a backup (*Reported by Chris Barber.*)
- Fixed an issue where validation was being running on archive_command even when the archive-check option was disabled.

Features:

- Added check command to validate that pgBackRest is configured correctly for archiving and backups. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Added the protocol-timeout option. Previously protocol-timeout was set as db-timeout + 30 seconds.
- Failure to shutdown remotes at the end of the backup no longer throws an exception. Instead a warning is generated that recommends a higher protocol-timeout.
- Experimental support for non-exclusive backups in PostgreSQL 9.6 beta2. Changes to the control/catalog/WAL versions in subsequent betas may break compatibility but pgBackRest will be updated with each release to keep pace.

Improvements:

- Improved handling of users/groups captured during backup that do not exist on the restore host. Also explicitly handle the case where user/group is not mapped to a name.
- Option handling is now far more strict. Previously it was possible for a command to use an option that was not explicitly assigned to it. This was especially true for the backup-host and db-host options which are used to determine locality.

Documentation Improvements:

- Allow a static date to be used for documentation to generate reproducible builds. (*Suggested by Adrian Vondendriesch.*)
- Added documentation for asynchronous archiving to the user guide. (*Contributed by Cynthia Shang. Reviewed by David Steele.*)
- Recommended install location for pgBackRest modules is now /usr/share/perl5 since /usr/lib/perl5 has been removed from the search path in newer versions of Perl.
- Added instructions for removing prior versions of pgBackRest.

v1.02 Release Notes

Bug Fix for Perl 5.22

Released June 2, 2016

Bug Fixes:

- Fix usage of sprintf() due to new constraints in Perl 5.22. Parameters not referenced in the format string are no longer allowed. (*Fixed by Adrian*

Vondendriesch. Reviewed by David Steele.)

Documentation Bug Fixes:

- Fixed syntax that was not compatible with Perl 5.2X. (*Fixed by Christoph Berg, Adrian Vondendriesch. Reviewed by David Steele.*)
- Fixed absolute paths that were used for the PDF logo. (*Reported by Adrian Vondendriesch.*)

Documentation Features:

- Release notes are now broken into sections so that bugs, features, and refactors are clearly delineated. An "Additional Notes" section has been added for changes to documentation and the test suite that do not affect the core code.
- Added man page generation. (*Contributed by Adrian Vondendriesch, David Steele.*)
- The change log was the last piece of documentation to be rendered in Markdown only. Wrote a converter so the document can be output by the standard renderers. The change log will now be located on the website and has been renamed to "Releases". (*Contributed by Cynthia Shang. Reviewed by David Steele.*)

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. Reviewed by David Steele.*)
- 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 log directory. (*Suggested by Stephen Frost.*)
- Executable filename changed from pg_backrest to pgbackrest. (*Suggested by Michael Renner, Stephen Frost.*)
- All files and directories linked from PGDATA are now included in the backup. By default links will be restored directly into PGDATA as files or directories. The --link-all option can be used to restore all links to their original locations. The --link-map option can be used to remap a link to a new location.
- Removed --tablespace option and replaced with --tablespace-map-all option which should more clearly indicate its function.

- Added detail log level which will output more information than info without being as verbose as debug.

Pre-Stable Releases

v0.92 Release Notes

Command-line Repository Path Fix

Released April 6, 2016

Bug Fixes:

- Fixed an issue where the master process was passing --repo-remote-path instead of --repo-path to the remote and causing the lock files to be created in the default repository directory (/var/lib/backup), generally ending in failure. This was only an issue when --repo-remote-path was defined on the command line rather than in pg_backrest.conf. (*Reported by Jan Wieck.*)

v0.91 Release Notes

Tablespace Bug Fix and Minor Enhancements

Released March 22, 2016

IMPORTANT BUG FIX FOR TABLESPACES: A change to the repository format was accidentally introduced in 0.90 which means the on-disk backup was no longer a valid PostgreSQL cluster when the backup contained tablespaces. This only affected users who directly copied the backups to restore PostgreSQL clusters rather than using the restore command. However, the fix breaks compatibility with older backups that contain tablespaces no matter how they are being restored (pgBackRest will throw errors and refuse to restore). New full backups should be taken immediately after installing version 0.91 for any clusters that contain tablespaces. If older backups need to be restored then use a version of pgBackRest that matches the backup version.

Bug Fixes:

- Fixed repository incompatibility introduced in pgBackRest 0.90. (*Reported by Evan Benoit.*)

Features:

- Copy global/pg_control last during backups.
- Write .info and .manifest files to temp before moving them to their final locations and fsync'ing.
- Rename --no-start-stop option to --no-online.

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. Reviewed by David Steele.*)
- Improved error message when repo-path or repo-remote-path does not exist.
- Added checks for --delta and --force restore options to ensure that the destination is a valid $PGDATA directory. pgBackRest will check for the presence of PG_VERSION or backup.manifest (left over from an aborted restore). If neither file is found then --delta and --force will be disabled but the restore will proceed unless there are files in the $PGDATA directory (or any tablespace directories) in which case the operation will be aborted.
- When restore --set=latest (the default) the actual backup restored will be output to the log.
- Support for PostgreSQL 9.5 partial WAL segments and recovery_target_action setting. The archive_mode = 'always' setting is not yet supported.
- Support for recovery_target = 'immediate' recovery setting introduced in PostgreSQL 9.4.
- The following tablespace checks have been added: paths or files in pg_tblspc, relative links in pg_tblspc, tablespaces in $PGDATA. All three will generate errors.

v0.89 Release Notes

Timeout Bug Fix and Restore Read-Only Repositories

Released December 24, 2015

Bug Fixes:

- Fixed an issue where longer-running backups/restores would timeout when remote and threaded. Keepalives are now used to make sure the remote for the main process does not timeout while the thread remotes do all the work. The error message for timeouts was also improved to make debugging easier. (*Reported by Stephen Frost.*)

Features:

- Allow restores to be performed on a read-only repository by using --no-lock and --log-level-file=off. The --no-lock option can only be used with restores.

v0.88 Release Notes

Documentation and Minor Bug Fixes

Released November 22, 2015

Bug Fixes:

- Fixed an issue where the start/stop commands required the --config option. (*Reported by Dmitry Didovicher.*)
- Fixed an issue where log files were being overwritten instead of appended. (*Reported by Stephen Frost, Dmitry Didovicher.*)
- Fixed an issue where backup-user was not optional.

Features:

- Symlinks are no longer created in backup directories in the repository. These symlinks could point virtually anywhere and potentially be dangerous. Symlinks are still recreated during a restore. (*Suggested by Stephen Frost.*)
- Added better messaging for backup expiration. Full and differential backup expirations are logged on a single line along with a list of all dependent backups expired.
- Archive retention is automatically set to full backup retention if not explicitly configured.

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.

Documentation Features:

- Added a new user guide that covers pgBackRest basics and some advanced topics including PITR. Much more to come, but it's a start. (*Contributed by David Steele, Stephen Frost. Reviewed by Michael Renner, Cynthia Shang, Eric Radman, Dmitry Didovicher.*)

v0.85 Release Notes

Start/Stop Commands and Minor Bug Fixes

Released October 8, 2015

Bug Fixes:

- Fixed an issue where an error could be returned after a backup or restore completely successfully.
- Fixed an issue where a resume would fail if temp files were left in the root backup directory when the backup failed. This scenario was likely if the backup process got terminated during the copy phase.

Features:

- Added stop and start commands to prevent pgBackRest processes from running on a system where PostgreSQL is shutdown or the system needs to be quiesced for some other reason.
- Experimental support for PostgreSQL 9.5 beta1. This may break when the control version or WAL magic changes in future versions but will be updated in each pgBackRest release to keep pace. All regression tests pass except for --target-resume tests (this functionality has changed in 9.5) and there is no testing yet for .partial WAL segments.

v0.82 Release Notes

Refactoring, Command-line Help, and Minor Bug Fixes

Released September 14, 2015

Bug Fixes:

- Fixed an issue where resumed compressed backups were not preserving existing files.
- Fixed an issue where resume and incr/diff would not ensure that the prior backup had the same compression and hardlink settings.
- Fixed an issue where a cold backup using --no-start-stop could be started on a running PostgreSQL cluster without --force specified.
- Fixed an issue where a thread could be started even when none were requested.
- Fixed an issue where the pgBackRest version number was not being updated in backup.info and archive.info after an upgrade/downgrade.

- Fixed an issue where the info command was throwing an exception when the repository contained no stanzas. (*Reported by Stephen Frost.*)
- Fixed an issue where the PostgreSQL pg_stop_backup() NOTICEs were being output to stderr. (*Reported by Stephen Frost.*)

Features:

- Experimental support for PostgreSQL 9.5 alpha2. This may break when the control version or WAL magic changes in future versions but will be updated in each pgBackRest release to keep pace. All regression tests pass except for --target-resume tests (this functionality has changed in 9.5) and there is no testing yet for .partial WAL segments.

Improvements:

- Renamed recovery-setting option and section to recovery-option to be more consistent with pgBackRest naming conventions.
- Added dynamic module loading to speed up commands, especially asynchronous archiving.

Documentation Features:

- Command-line help is now extracted from the same XML source that is used for the other documentation and includes much more detail.

v0.80 Release Notes

DBI Support, Stability, and Convenience Features

Released August 9, 2015

Bug Fixes:

- Fixed an issue that caused the formatted timestamp for both the oldest and newest backups to be reported as the current time by the info command. Only text output was affected -- json output reported the correct epoch values. (*Reported by Michael Renner.*)
- Fixed protocol issue that was preventing ssh errors (especially on connection) from being logged.

Features:

- The repository is now created and updated with consistent directory and file modes. By default umask is set to 0000 but this can be disabled with the neutral-umask setting. (*Suggested by Cynthia Shang.*)
- Added the stop-auto option to allow failed backups to automatically be stopped when a new backup starts.
- Added the db-timeout option to limit the amount of time pgBackRest will wait for pg_start_backup() and pg_stop_backup() to return.
- Remove pg_control file at the beginning of the restore and copy it back at the very end. This prevents the possibility that a partial restore can be started by PostgreSQL.

- Added checks to be sure the db-path setting is consistent with db-port by comparing the data_directory as reported by the cluster against the db-path setting and the version as reported by the cluster against the value read from pg_control. The db-socket-path setting is checked to be sure it is an absolute path.
- Experimental support for PostgreSQL 9.5 alpha1. This may break when the control version or WAL magic changes in future versions but will be updated in each pgBackRest release to keep pace. All regression tests pass except for --target-resume tests (this functionality has changed in 9.5) and there is no testing yet for .partial WAL segments.

Improvements:

- Now using Perl DBI and DBD::Pg for connections to PostgreSQL rather than psql. The cmd-psql and cmd-psql-option settings have been removed and replaced with db-port and db-socket-path. Follow the instructions in the Installation Guide to install DBD::Pg on your operating system.

Test Suite Features:

- Added vagrant test configurations for Ubuntu 14.04 and CentOS 7.

v0.78 Release Notes

Remove CPAN Dependencies, Stability Improvements

Released July 13, 2015

Improvements:

- Removed dependency on CPAN packages for multi-threaded operation. While it might not be a bad idea to update the threads and Thread::Queue packages, it is no longer necessary.
- Modified wait backoff to use a Fibonacci rather than geometric sequence. This will make wait time grow less aggressively while still giving reasonable values.

Test Suite Features:

- Added vagrant test configurations for Ubuntu 12.04 and CentOS 6.

v0.77 Release Notes

CentOS/RHEL 6 Support and Protocol Improvements

Released June 30, 2015

Features:

- Added file and directory syncs to the File object for additional safety during backup/restore and archiving. (*Suggested by Andres Freund.*)
- Added support for Perl 5.10.1 and OpenSSH 5.3 which are default for CentOS/RHEL 6. (*Suggested by Eric Radman.*)

- Improved error message when backup is run without archive_command set and without --no-archive-check specified. (*Suggested by Eric Radman.*)

v0.75 Release Notes

New Repository Format, Info Command and Experimental 9.5 Support

Released June 14, 2015

IMPORTANT NOTE: This flag day release breaks compatibility with older versions of pgBackRest. The manifest format, on-disk structure, and the binary names have all changed. You must create a new repository to hold backups for this version of pgBackRest and keep your older repository for a time in case you need to do a restore. The pg_backrest.conf file has not changed but you'll need to change any references to pg_backrest.pl in cron (or elsewhere) to pg_backrest (without the .pl extension).

Features:

- Added the info command.
- Logging now uses unbuffered output. This should make log files that are being written by multiple threads less chaotic. (*Suggested by Michael Renner.*)
- Experimental support for PostgreSQL 9.5. This may break when the control version or WAL magic changes but will be updated in each release.

Improvements:

- More efficient file ordering for backup. Files are copied in descending size order so a single thread does not end up copying a large file at the end. This had already been implemented for restore.

v0.70 Release Notes

Stability Improvements for Archiving, Improved Logging and Help

Released June 1, 2015

Bug Fixes:

- Fixed an issue where archive-copy would fail on an incr/diff backup when hardlink=n. In this case the pg_xlog path does not already exist and must be created. (*Reported by Michael Renner.*)
- Fixed an issue in async archiving where archive-push was not properly returning 0 when archive-max-mb was reached and moved the async check after transfer to avoid having to remove the stop file twice. Also added unit tests for this case and improved error messages to make it clearer to the user what went wrong. (*Reported by Michael Renner.*)
- Fixed a locking issue that could allow multiple operations of the same type against a single stanza. This appeared to be benign in terms of data integrity but caused spurious errors while archiving and could lead to errors in backup/restore. (*Reported by Michael Renner.*)

Features:

- Allow duplicate WAL segments to be archived when the checksum matches. This is necessary for some recovery scenarios.
- Allow comments/disabling in pg_backrest.conf using the # character. Only # characters in the first character of the line are honored. (*Suggested by Michael Renner.*)
- Better logging before pg_start_backup() to make it clear when the backup is waiting on a checkpoint. (*Suggested by Michael Renner.*)
- Various command behavior and logging fixes. (*Reviewed by Michael Renner. Suggested by Michael Renner.*)

Improvements:

- Replaced JSON module with JSON::PP which ships with core Perl.

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.

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.

Test Suite Features:

- More comprehensive unit tests in all areas.

v0.30 Release Notes

Core Restructuring and Unit Tests

Released October 5, 2014

Documentation Features:

- Added much needed documentation

Test Suite Features:

- Fairly comprehensive unit tests for all the basic operations. More work to be done here for sure, but then there is always more work to be done on unit tests.

v0.19 Release Notes

Improved Error Reporting/Handling

Released May 13, 2014

Bug Fixes:

- Found and squashed a nasty bug where file_copy() was defaulted to ignore errors. There was also an issue in file_exists() that was causing the test to fail when the file actually did exist. Together they could have resulted in a corrupt backup with no errors, though it is very unlikely.

v0.18 Release Notes

Return Soft Error When Archive Missing

Released April 13, 2014

Bug Fixes:

- The archive-get command now returns a 1 when the archive file is missing to differentiate from hard errors (ssh connection failure, file copy error, etc.) This lets PostgreSQL know that 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.

Documentation Features:

- Absolutely no documentation (outside the code). Well, excepting these release notes.