pgBackRest

Command Reference

Table of Contents

1 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

2 Annotate Command (annotate)

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

2.1 Command Options

2.1.1 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"
```

2.1.2 Set Option (--set)

Backup set to annotate.

The backup set to annotate.

```
example: --set=20150131-153358F_20150131-153401I
```

2.2 General Options

2.2.1 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
```

2.2.2 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 but can be disabled by setting `compress-level-network=0`. 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: 3
allowed: 0-9
example: --compress-level-network=1
```

2.2.3 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
```

2.2.4 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
```

2.2.5 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
```

2.2.6 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

2.2.7 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
```

2.2.8 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

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

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

2.2.9 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: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

2.2.10 Keep Alive Option (`--sck-keep-alive`)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

2.2.11 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
```

2.2.12 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
```

2.2.13 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
```

2.2.14 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
```

2.3 Log Options

2.3.1 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
```

2.3.2 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
```

2.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

2.3.4 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
```

2.3.5 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
```

2.3.6 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
```

2.4 Repository Options

2.4.1 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
```

2.4.2 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
```

2.4.3 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

```
default: shared
example: --repo1-azure-key-type=sas
```

2.4.4 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
```

2.4.5 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
```

2.4.6 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
```

2.4.7 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
```

2.4.8 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
```

2.4.9 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

2.4.10 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
```

2.4.11 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
```

2.4.12 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
```

2.4.13 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.

```
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

2.4.14 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

2.4.15 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
```

2.4.16 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
```

2.4.17 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
```

2.4.18 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.

```
allowed: 0-65535
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

2.4.19 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
```

2.4.20 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

2.4.21 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
```

2.4.22 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
```

2.4.23 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
```

2.4.24 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

```
default: shared
example: --repo1-s3-key-type=auto
```

2.4.25 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

2.4.26 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
```

2.4.27 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
```

2.4.28 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
```

2.4.29 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

2.4.30 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
```

2.4.31 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
```

2.4.32 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
```

2.4.33 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
```

2.4.34 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
```

2.4.35 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
```

2.4.36 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
```

2.4.37 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
```

2.4.38 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

2.4.39 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

2.4.40 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

2.4.41 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

2.4.42 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
```

2.4.43 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.

Default chunk sizes by repo type:

- `azure` - 4MiB
- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined behavior.

```
allowed: 64KiB-1TiB
example: --repo1-storage-upload-chunk-size=16MiB
```

2.4.44 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

2.4.45 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
```

3 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.

3.1 Command Options

3.1.1 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
```

3.1.2 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: 0-4PiB
example: --archive-get-queue-max=1GiB
```

3.1.3 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
```

3.1.4 Archive Timeout Option (`--archive-timeout`)

Archive timeout.

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

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

3.2 General Options

3.2.1 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
```

3.2.2 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.*

```
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

3.2.3 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
```

3.2.4 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 but can be disabled by setting `compress-level-network=0`. 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: 3
allowed: 0-9
example: --compress-level-network=1
```

3.2.5 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
```

3.2.6 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
```

3.2.7 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
```

3.2.8 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: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

3.2.9 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

3.2.10 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
```

3.2.11 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

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

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

3.2.12 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
```

3.2.13 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: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

3.2.14 Keep Alive Option (`--sck-keep-alive`)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

3.2.15 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`.

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
```

3.2.16 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
```

3.2.17 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
```

3.2.18 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
```

3.2.19 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
```

3.3 Log Options

3.3.1 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
```

3.3.2 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
```

3.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

3.3.4 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
```

3.3.5 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
```

3.3.6 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
```

3.4 Maintainer Options

3.4.1 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
```

3.5 Repository Options

3.5.1 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
```

3.5.2 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
```

3.5.3 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

```
default: shared
example: --repo1-azure-key-type=sas
```

3.5.4 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
```

3.5.5 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
```

3.5.6 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
```

3.5.7 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
```

3.5.8 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
```

3.5.9 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

3.5.10 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
```

3.5.11 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
```

3.5.12 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
```

3.5.13 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.

```
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

3.5.14 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

3.5.15 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
```

3.5.16 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
```

3.5.17 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
```

3.5.18 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.

```
allowed: 0-65535
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

3.5.19 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
```

3.5.20 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

3.5.21 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
```

3.5.22 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
```

3.5.23 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
```

3.5.24 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

```
default: shared
example: --repo1-s3-key-type=auto
```

3.5.25 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

3.5.26 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
```

3.5.27 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
```

3.5.28 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
```

3.5.29 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

3.5.30 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
```

3.5.31 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
```

3.5.32 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
```

3.5.33 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
```

3.5.34 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
```

3.5.35 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
```

3.5.36 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
```

3.5.37 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
```

3.5.38 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

3.5.39 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

3.5.40 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

3.5.41 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

3.5.42 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
```

3.5.43 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.

Default chunk sizes by repo type:

- `azure` - 4MiB
- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined behavior.

```
allowed: 64KiB-1TiB
example: --repo1-storage-upload-chunk-size=16MiB
```

3.5.44 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

3.5.45 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
```

3.6 Stanza Options

3.6.1 PostgreSQL Path Option (`--pg-path`)

PostgreSQL data directory.

This should be the same as the `data_directory` setting in `postgresql.conf`. Even though this value can be read from `postgresql.conf` or PostgreSQL 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

4 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.

4.1 Command Options

4.1.1 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
```

4.1.2 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
```

4.1.3 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
```

4.1.4 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.

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: 0-4PiB
example: --archive-push-queue-max=1TiB
```

Deprecated Name: archive-queue-max

4.1.5 Archive Timeout Option (`--archive-timeout`)

Archive timeout.

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

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

4.2 General Options

4.2.1 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
```

4.2.2 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.*

```
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

4.2.3 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
```

4.2.4 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
```

4.2.5 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).

The following are the defaults levels based on `compress-type` when `compress-level` is not specified:

- `bz2` - 9
- `gz` - 6
- `lz4` - 1
- `zst` - 3

```
example: --compress-level=9
```

4.2.6 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 but can be disabled by setting `compress-level-network=0`. 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: 3
allowed: 0-9
example: --compress-level-network=1
```

4.2.7 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
```

4.2.8 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
```

4.2.9 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
```

4.2.10 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
```

4.2.11 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: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

4.2.12 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

4.2.13 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
```

4.2.14 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

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

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

4.2.15 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
```

4.2.16 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: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

4.2.17 Keep Alive Option (`--sck-keep-alive`)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

4.2.18 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`.

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
```

4.2.19 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
```

4.2.20 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
```

4.2.21 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
```

4.2.22 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
```

4.3 Log Options

4.3.1 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
```

4.3.2 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
```

4.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

4.3.4 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
```

4.3.5 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
```

4.3.6 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
```

4.4 Maintainer Options

4.4.1 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
```

4.4.2 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
```

4.5 Repository Options

4.5.1 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
```

4.5.2 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

```
default: shared
example: --repo1-azure-key-type=sas
```

4.5.3 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
```

4.5.4 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
```

4.5.5 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
```

4.5.6 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
```

4.5.7 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
```

4.5.8 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

4.5.9 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
```

4.5.10 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
```

4.5.11 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
```

4.5.12 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.

```
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

4.5.13 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

4.5.14 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
```

4.5.15 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
```

4.5.16 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
```

4.5.17 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.

```
allowed: 0-65535
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

4.5.18 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
```

4.5.19 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

4.5.20 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
```

4.5.21 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
```

4.5.22 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
```

4.5.23 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

```
default: shared
example: --repo1-s3-key-type=auto
```

4.5.24 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

4.5.25 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
```

4.5.26 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
```

4.5.27 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
```

4.5.28 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

4.5.29 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
```

4.5.30 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
```

4.5.31 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
```

4.5.32 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
```

4.5.33 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
```

4.5.34 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
```

4.5.35 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
```

4.5.36 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
```

4.5.37 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

4.5.38 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

4.5.39 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

4.5.40 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

4.5.41 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
```

4.5.42 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.

Default chunk sizes by repo type:

- `azure` - 4MiB
- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined

behavior.

```
allowed: 64KiB-1TiB
example: --repo1-storage-upload-chunk-size=16MiB
```

4.5.43 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

4.5.44 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
```

4.6 Stanza Options

4.6.1 PostgreSQL Path Option (`--pg-path`)

PostgreSQL data directory.

This should be the same as the `data_directory` setting in `postgresql.conf`. Even though this value can be read from `postgresql.conf` or PostgreSQL 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

5 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.

5.1 Command Options

5.1.1 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"
```

5.1.2 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
```

5.1.3 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
```

5.1.4 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
```

5.1.5 Archive Timeout Option (`--archive-timeout`)

Archive timeout.

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

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

5.1.6 Backup from Standby Option (`--backup-standby`)

Backup from the standby cluster.

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

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

5.1.7 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
```

5.1.8 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/
```

5.1.9 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.

```
default: y
example: --expire-auto
```

5.1.10 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
```

5.1.11 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: 1-1TiB
example: --manifest-save-threshold=8GiB
```

5.1.12 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
```

5.1.13 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
```

5.1.14 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
```

5.1.15 Stop Auto Option (`--stop-auto`)

Stop prior failed backup on new backup.

This will only be done if an exclusive advisory lock can be acquired to demonstrate that the prior failed backup process has really stopped.

This feature is not supported for PostgreSQL >= `9.6` since backups are run in non-exclusive mode.

The setting is disabled by default because it assumes that pgBackRest is the only process doing exclusive online backups. It depends on an advisory lock that only pgBackRest sets so it may abort other processes that do exclusive online backups. Note that base_backup and pg_dump are safe to use with this setting because they do not call `pg_start_backup()` so are not exclusive.

```
default: n
example: --stop-auto
```

5.1.16 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
```

5.2 General Options

5.2.1 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
```

5.2.2 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.*

```
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

5.2.3 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
```

5.2.4 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
```

5.2.5 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).

The following are the defaults levels based on `compress-type` when `compress-level` is not specified:

- `bz2` - 9
- `gz` - 6
- `lz4` - 1
- `zst` - 3

```
example: --compress-level=9
```

5.2.6 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 but can be disabled by setting `compress-level-network=0`. 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: 3
allowed: 0-9
example: --compress-level-network=1
```

5.2.7 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
```

5.2.8 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
```

5.2.9 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
```

5.2.10 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
```

5.2.11 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: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

5.2.12 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
```

5.2.13 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

5.2.14 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
```

5.2.15 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

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

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

5.2.16 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
```

5.2.17 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: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

5.2.18 Keep Alive Option (`--sck-keep-alive`)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

5.2.19 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
```

5.2.20 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
```

5.2.21 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
```

5.2.22 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
```

5.3 Log Options

5.3.1 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
```

5.3.2 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
```

5.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

5.3.4 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
```

5.3.5 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
```

5.3.6 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
```

5.4 Maintainer Options

5.4.1 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
```

5.4.2 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
```

5.5 Repository Options

5.5.1 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
```

5.5.2 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
```

5.5.3 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

```
default: shared
example: --repo1-azure-key-type=sas
```

5.5.4 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
```

5.5.5 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
```

5.5.6 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
```

5.5.7 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
```

5.5.8 Repository Bundle Size Option (`--repo-bundle-size`)

Target size for file bundles.

Defines the total size of files that will be added to a single bundle. Most bundles will be smaller than this size but it is possible that some will be slightly larger, 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
```

5.5.9 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
```

5.5.10 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
```

5.5.11 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
```

5.5.12 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
```

5.5.13 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

5.5.14 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
```

5.5.15 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

5.5.16 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

5.5.17 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.

```
allowed: 1-9999999
example: --repo1-retention-diff=3
```

Deprecated Name: retention-diff

5.5.18 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

5.5.19 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 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.

```
default: count
example: --repo1-retention-full-type=time
```

5.5.20 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
```

5.5.21 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
```

5.5.22 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
```

5.5.23 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

```
default: shared
example: --repo1-s3-key-type=auto
```

5.5.24 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

5.5.25 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
```

5.5.26 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
```

5.5.27 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
```

5.5.28 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

5.5.29 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
```

5.5.30 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
```

5.5.31 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
```

5.5.32 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
```

5.5.33 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
```

5.5.34 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
```

5.5.35 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
```

5.5.36 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
```

5.5.37 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

5.5.38 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

5.5.39 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

5.5.40 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

5.5.41 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
```

5.5.42 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.

Default chunk sizes by repo type:

- `azure` - 4MiB

- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined behavior.

```
allowed: 64KiB-1TiB
example: --repo1-storage-upload-chunk-size=16MiB
```

5.5.43 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

5.5.44 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
```

5.6 Stanza Options

5.6.1 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
```

5.6.2 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

5.6.3 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
```

5.6.4 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
```

5.6.5 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
```

5.6.6 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.

```
example: --pg1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: db-cmd

5.6.7 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

5.6.8 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
```

5.6.9 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
```

5.6.10 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
```

5.6.11 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.

```
allowed: 0-65535
example: --pg1-host-port=25
```

Deprecated Name: db-ssh-port

5.6.12 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
```

5.6.13 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 `postgres`

correctly the user should be the PostgreSQL database cluster owner which is generally , the default.

```
default: postgres
example: --pg1-host-user=db_owner
```

Deprecated Name: db-user

5.6.14 PostgreSQL Path Option (--pg-path)

PostgreSQL data directory.

This should be the same as the `data_directory` setting in `postgresql.conf`. Even though this value can be read from `postgresql.conf` or PostgreSQL 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

5.6.15 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

5.6.16 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_directory` setting in `postgresql.conf`.

```
example: --pg1-socket-path=/var/run/postgresql
```

Deprecated Name: db-socket-path

5.6.17 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
```

6 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.

6.1 Command Options

6.1.1 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
```

6.1.2 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
```

6.1.3 Archive Timeout Option (`--archive-timeout`)

Archive timeout.

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

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

6.1.4 Backup from Standby Option (`--backup-standby`)

Backup from the standby cluster.

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

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

6.2 General Options

6.2.1 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
```

6.2.2 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.*

```
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

6.2.3 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
```

6.2.4 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 but can be disabled by setting `compress-level-network=0`. 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: 3
allowed: 0-9
example: --compress-level-network=1
```

6.2.5 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
```

6.2.6 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
```

6.2.7 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
```

6.2.8 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: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

6.2.9 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

6.2.10 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

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

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

6.2.11 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: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

6.2.12 Keep Alive Option (`--sck-keep-alive`)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

6.2.13 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
```

6.2.14 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
```

6.2.15 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
```

6.2.16 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
```

6.3 Log Options

6.3.1 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
```

6.3.2 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
```

6.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

6.3.4 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
```

6.3.5 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
```

6.3.6 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
```

6.4 Maintainer Options

6.4.1 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
```

6.5 Repository Options

6.5.1 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
```

6.5.2 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

```
default: shared
example: --repo1-azure-key-type=sas
```

6.5.3 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
```

6.5.4 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
```

6.5.5 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
```

6.5.6 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
```

6.5.7 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
```

6.5.8 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

6.5.9 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
```

6.5.10 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
```

6.5.11 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
```

6.5.12 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.

```
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

6.5.13 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

6.5.14 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
```

6.5.15 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
```

6.5.16 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
```

6.5.17 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.

```
allowed: 0-65535
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

6.5.18 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
```

6.5.19 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

6.5.20 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
```

6.5.21 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
```

6.5.22 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
```

6.5.23 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

```
default: shared
example: --repo1-s3-key-type=auto
```

6.5.24 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

6.5.25 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
```

6.5.26 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
```

6.5.27 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
```

6.5.28 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

6.5.29 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
```

6.5.30 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
```

6.5.31 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
```

6.5.32 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
```

6.5.33 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
```

6.5.34 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
```

6.5.35 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
```

6.5.36 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
```

6.5.37 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

6.5.38 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

6.5.39 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

6.5.40 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

6.5.41 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
```

6.5.42 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.

Default chunk sizes by repo type:

- `azure` - 4MiB
- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined behavior.

```
allowed: 64KiB-1TiB
example: --repo1-storage-upload-chunk-size=16MiB
```

6.5.43 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

6.5.44 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
```

6.6 Stanza Options

6.6.1 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
```

6.6.2 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

6.6.3 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
```

6.6.4 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
```

6.6.5 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
```

6.6.6 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.

```
example: --pg1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: db-cmd

6.6.7 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

6.6.8 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
```

6.6.9 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
```

6.6.10 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
```

6.6.11 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.

```
allowed: 0-65535
example: --pg1-host-port=25
```

Deprecated Name: db-ssh-port

6.6.12 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
```

6.6.13 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

6.6.14 PostgreSQL Path Option (--pg-path)

PostgreSQL data directory.

This should be the same as the data_directory setting in postgresql.conf. Even though this value can be read from postgresql.conf or PostgreSQL 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

6.6.15 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

6.6.16 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_directory` setting in `postgresql.conf`.

```
example: --pg1-socket-path=/var/run/postgresql
```

Deprecated Name: db-socket-path

6.6.17 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
```

7 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 and Incremental backups are count-based but will always be expired when the backup they depend on is expired. 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.

7.1 Command Options

7.1.1 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
```

7.2 General Options

7.2.1 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
```

7.2.2 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
```

7.2.3 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
```

7.2.4 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
```

7.2.5 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
```

7.2.6 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

7.2.7 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
```

7.2.8 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

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

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

7.2.9 Keep Alive Option (`--sck-keep-alive`)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

7.2.10 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
```

7.2.11 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
```

7.2.12 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
```

7.2.13 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
```

7.3 Log Options

7.3.1 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
```

7.3.2 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
```

7.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

7.3.4 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
```

7.3.5 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
```

7.4 Repository Options

7.4.1 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
```

7.4.2 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
```

7.4.3 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

```
default: shared
example: --repo1-azure-key-type=sas
```

7.4.4 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
```

7.4.5 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
```

7.4.6 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
```

7.4.7 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
```

7.4.8 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
```

7.4.9 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
```

7.4.10 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

7.4.11 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

7.4.12 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.

```
allowed: 1-9999999
example: --repo1-retention-diff=3
```

Deprecated Name: retention-diff

7.4.13 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

7.4.14 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 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.

```
default: count
example: --repo1-retention-full-type=time
```

7.4.15 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
```

7.4.16 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
```

7.4.17 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
```

7.4.18 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

```
default: shared
example: --repo1-s3-key-type=auto
```

7.4.19 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

7.4.20 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
```

7.4.21 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
```

7.4.22 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
```

7.4.23 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

7.4.24 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
```

7.4.25 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
```

7.4.26 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
```

7.4.27 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
```

7.4.28 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
```

7.4.29 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
```

7.4.30 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
```

7.4.31 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
```

7.4.32 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

7.4.33 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

7.4.34 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

7.4.35 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

7.4.36 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
```

7.4.37 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.

Default chunk sizes by repo type:

- `azure` - 4MiB
- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined behavior.

```
allowed: 64KiB-1TiB
example: --repo1-storage-upload-chunk-size=16MiB
```

7.4.38 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

7.4.39 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
```

8 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.

9 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.

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`' message will appear beside the '`status`' information if one of those commands is currently running on the host.

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

9.1 Command Options

9.1.1 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
```

9.1.2 Set Option (`--set`)

Backup set to detail.

Details include 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
```

9.1.3 Type Option (`--type`)

Filter on backup type.

Filter the output using one of the following backup types: `full`, `incr`, `diff`.

```
example: --type=full
```

9.2 General Options

9.2.1 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
```

9.2.2 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.*

```
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

9.2.3 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
```

9.2.4 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 but can be disabled by setting `compress-level-network=0`. 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: 3
allowed: 0-9
example: --compress-level-network=1
```

9.2.5 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
```

9.2.6 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
```

9.2.7 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
```

9.2.8 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

9.2.9 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
```

9.2.10 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: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

9.2.11 Keep Alive Option (`--sck-keep-alive`)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

9.2.12 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
```

9.2.13 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
```

9.2.14 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
```

9.2.15 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
```

9.3 Log Options

9.3.1 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
```

9.3.2 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
```

9.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

9.3.4 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
```

9.3.5 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
```

9.3.6 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
```

9.4 Repository Options

9.4.1 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
```

9.4.2 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
```

9.4.3 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

```
default: shared
example: --repo1-azure-key-type=sas
```

9.4.4 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
```

9.4.5 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
```

9.4.6 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
```

9.4.7 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
```

9.4.8 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
```

9.4.9 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

9.4.10 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
```

9.4.11 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
```

9.4.12 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
```

9.4.13 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.

```
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

9.4.14 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

9.4.15 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
```

9.4.16 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
```

9.4.17 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
```

9.4.18 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.

```
allowed: 0-65535
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

9.4.19 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
```

9.4.20 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

9.4.21 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
```

9.4.22 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
```

9.4.23 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
```

9.4.24 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

```
default: shared
example: --repo1-s3-key-type=auto
```

9.4.25 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

9.4.26 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
```

9.4.27 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
```

9.4.28 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
```

9.4.29 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

9.4.30 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
```

9.4.31 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
```

9.4.32 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
```

9.4.33 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
```

9.4.34 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
```

9.4.35 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
```

9.4.36 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
```

9.4.37 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
```

9.4.38 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

9.4.39 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

9.4.40 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

9.4.41 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

9.4.42 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
```

9.4.43 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.

Default chunk sizes by repo type:

- `azure` - 4MiB
- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined behavior.

```
allowed: 64KiB-1TiB
example: --repo1-storage-upload-chunk-size=16MiB
```

9.4.44 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

9.4.45 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
```

10 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.

10.1 Command Options

10.1.1 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
```

10.2 General Options

10.2.1 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
```

10.2.2 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.*

```
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

10.2.3 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
```

10.2.4 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 but can be disabled by setting `compress-level-network=0`. 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: 3
allowed: 0-9
example: --compress-level-network=1
```

10.2.5 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
```

10.2.6 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
```

10.2.7 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
```

10.2.8 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: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

10.2.9 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

10.2.10 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

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

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

10.2.11 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: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

10.2.12 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
```

10.2.13 Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

10.2.14 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
```

10.2.15 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
```

10.2.16 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
```

10.2.17 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
```

10.3 Log Options

10.3.1 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
```

10.3.2 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
```

10.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

10.3.4 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
```

10.3.5 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
```

10.3.6 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
```

10.4 Repository Options

10.4.1 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
```

10.4.2 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
```

10.4.3 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

```
default: shared
example: --repo1-azure-key-type=sas
```

10.4.4 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
```

10.4.5 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
```

10.4.6 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
```

10.4.7 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
```

10.4.8 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
```

10.4.9 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

10.4.10 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
```

10.4.11 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
```

10.4.12 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
```

10.4.13 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.

```
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

10.4.14 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

10.4.15 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
```

10.4.16 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
```

10.4.17 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
```

10.4.18 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.

```
allowed: 0-65535
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

10.4.19 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
```

10.4.20 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

10.4.21 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
```

10.4.22 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
```

10.4.23 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
```

10.4.24 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

```
default: shared
example: --repo1-s3-key-type=auto
```

10.4.25 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

10.4.26 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
```

10.4.27 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
```

10.4.28 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
```

10.4.29 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

10.4.30 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
```

10.4.31 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
```

10.4.32 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
```

10.4.33 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
```

10.4.34 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
```

10.4.35 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
```

10.4.36 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
```

10.4.37 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
```

10.4.38 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

10.4.39 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

10.4.40 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

10.4.41 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

10.4.42 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
```

10.4.43 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.

Default chunk sizes by repo type:

- `azure` - 4MiB

- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined behavior.

```
allowed: 64KiB-1TiB
example: --repo1-storage-upload-chunk-size=16MiB
```

10.4.44 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

10.4.45 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
```

11 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.

11.1 Command Options

11.1.1 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)$"
```

11.1.2 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
```

11.1.3 Recurse Subpaths Option (`--recurse`)

Include all subpaths in output.

All subpaths and their files will be included in the output.

```
default: n
example: --recurse
```

11.1.4 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
```

11.2 General Options

11.2.1 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
```

11.2.2 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.*

```
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

11.2.3 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
```

11.2.4 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 but can be disabled by setting compress-level-network=0. 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: 3
allowed: 0-9
example: --compress-level-network=1
```

11.2.5 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
```

11.2.6 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
```

11.2.7 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
```

11.2.8 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: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

11.2.9 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

11.2.10 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

respectively.

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

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

11.2.11 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: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

11.2.12 Keep Alive Option (`--sck-keep-alive`)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

11.2.13 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
```

11.2.14 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
```

11.2.15 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
```

11.2.16 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
```

11.3 Log Options

11.3.1 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
```

11.3.2 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
```

11.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

11.3.4 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
```

11.3.5 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
```

11.3.6 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
```

11.4 Repository Options

11.4.1 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
```

11.4.2 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
```

11.4.3 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

```
default: shared
example: --repo1-azure-key-type=sas
```

11.4.4 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
```

11.4.5 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
```

11.4.6 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
```

11.4.7 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
```

11.4.8 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
```

11.4.9 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

11.4.10 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
```

11.4.11 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
```

11.4.12 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
```

11.4.13 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.

```
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

11.4.14 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

11.4.15 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
```

11.4.16 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
```

11.4.17 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
```

11.4.18 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.

```
allowed: 0-65535
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

11.4.19 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
```

11.4.20 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

11.4.21 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
```

11.4.22 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
```

11.4.23 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
```

11.4.24 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

```
default: shared
example: --repo1-s3-key-type=auto
```

11.4.25 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

11.4.26 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
```

11.4.27 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
```

11.4.28 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
```

11.4.29 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

11.4.30 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
```

11.4.31 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
```

11.4.32 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
```

11.4.33 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
```

11.4.34 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
```

11.4.35 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
```

11.4.36 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
```

11.4.37 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
```

11.4.38 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

11.4.39 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

11.4.40 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

11.4.41 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

11.4.42 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
```

11.4.43 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.

Default chunk sizes by repo type:

- `azure` - 4MiB
- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined behavior.

```
allowed: 64KiB-1TiB
example: --repo1-storage-upload-chunk-size=16MiB
```

11.4.44 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

11.4.45 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
```

12 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.

12.1 Command Options

12.1.1 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
```

12.1.2 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
```

12.1.3 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
```

12.1.4 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
```

12.1.5 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
```

12.1.6 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
```

12.1.7 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
```

12.1.8 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
```

12.1.9 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
```

12.1.10 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.

```
example: --tablespace-map-all=/data/tablespace
```

12.1.11 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
```

12.1.12 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
```

12.1.13 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
```

12.1.14 Target Timeline Option (`--target-timeline`)

Recover along a timeline.

See `recovery_target_timeline` in the PostgreSQL docs for more information.

```
example: --target-timeline=3
```

12.1.15 Type Option (`--type`)

Recovery type.

The following recovery types are supported:

- `default` - recover to the end of the archive stream.
- `immediate` - recover only until the database becomes consistent. This option is only supported on PostgreSQL >= 9.4.
- `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
```

12.2 General Options

12.2.1 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
```

12.2.2 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.*

```
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

12.2.3 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
```

12.2.4 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 but can be disabled by setting `compress-level-network=0`. 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: 3
allowed: 0-9
example: --compress-level-network=1
```

12.2.5 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
```

12.2.6 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
```

12.2.7 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
```

12.2.8 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
```

12.2.9 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

12.2.10 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
```

12.2.11 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

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

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

12.2.12 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
```

12.2.13 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: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

12.2.14 Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

12.2.15 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
```

12.2.16 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
```

12.2.17 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
```

12.2.18 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
```

12.3 Log Options

12.3.1 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
```

12.3.2 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
```

12.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

- `off` - No logging at all (not recommended)
- `error` - Log only errors
- `warn` - Log warnings and errors

`info` - Log info, warnings, and errors

- `detail` - Log detail, info, warnings, and errors
- `debug` - Log debug, detail, info, warnings, and errors
- `trace` - Log trace (very verbose debugging), debug, info, warnings, and errors

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

12.3.4 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
```

12.3.5 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
```

12.3.6 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
```

12.4 Maintainer Options

12.4.1 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
```

12.5 Repository Options

12.5.1 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
```

12.5.2 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
```

12.5.3 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

```
default: shared
example: --repo1-azure-key-type=sas
```

12.5.4 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
```

12.5.5 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
```

12.5.6 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
```

12.5.7 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
```

12.5.8 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
```

12.5.9 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

12.5.10 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
```

12.5.11 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
```

12.5.12 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
```

12.5.13 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.

```
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

12.5.14 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

12.5.15 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
```

12.5.16 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
```

12.5.17 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
```

12.5.18 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.

```
allowed: 0-65535
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

12.5.19 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
```

12.5.20 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

12.5.21 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
```

12.5.22 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
```

12.5.23 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
```

12.5.24 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

```
default: shared
example: --repo1-s3-key-type=auto
```

12.5.25 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

12.5.26 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
```

12.5.27 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
```

12.5.28 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
```

12.5.29 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

12.5.30 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
```

12.5.31 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
```

12.5.32 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
```

12.5.33 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
```

12.5.34 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
```

12.5.35 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
```

12.5.36 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
```

12.5.37 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
```

12.5.38 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

12.5.39 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

12.5.40 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

12.5.41 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

12.5.42 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
```

12.5.43 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.

Default chunk sizes by repo type:

- `azure` - 4MiB
- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined behavior.

```
allowed: 64KiB-1TiB
example: --repo1-storage-upload-chunk-size=16MiB
```

12.5.44 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

12.5.45 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
```

12.6 Stanza Options

12.6.1 PostgreSQL Path Option (`--pg-path`)

PostgreSQL data directory.

This should be the same as the `data_directory` setting in `postgresql.conf`. Even though this value can be read from `postgresql.conf` or PostgreSQL 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

13 Server Command (`server`)

The pgBackRest server allows access to remote hosts without using the SSH protocol.

13.1 Command Options

13.1.1 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=*
```

13.1.2 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 repeating 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
```

13.1.3 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
```

13.1.4 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
```

13.1.5 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
```

13.1.6 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
```

13.2 General Options

13.2.1 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
```

13.2.2 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
```

13.2.3 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
```

13.2.4 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
```

13.2.5 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

13.2.6 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: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

13.2.7 Keep Alive Option (`--sck-keep-alive`)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

13.2.8 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
```

13.2.9 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
```

13.2.10 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
```

13.3 Log Options

13.3.1 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
```

13.3.2 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
```

13.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

13.3.4 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
```

13.3.5 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
```

14 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.

14.1 Command Options

14.1.1 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=*
```

14.1.2 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
```

14.2 General Options

14.2.1 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
```

14.2.2 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
```

14.2.3 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
```

14.2.4 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
```

14.2.5 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

14.2.6 Keep Alive Option (`--sck-keep-alive`)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

14.2.7 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
```

14.2.8 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
```

14.2.9 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
```

14.3 Log Options

14.3.1 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
```

14.3.2 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
```

14.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

Specifies which log levels will output to `stderr` rather than `stdout` (specified by `log-level-console`). The

timestamp and process will not be output to `stderr`.

The following log levels are supported:

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

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

14.3.4 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
```

14.3.5 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
```

15 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.

15.1 Command Options

15.1.1 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
```

15.2 General Options

15.2.1 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
```

15.2.2 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.*

```
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

15.2.3 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
```

15.2.4 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 but can be disabled by setting compress-level-network=0. 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: 3
allowed: 0-9
example: --compress-level-network=1
```

15.2.5 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
```

15.2.6 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
```

15.2.7 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
```

15.2.8 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: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

15.2.9 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

15.2.10 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
```

15.2.11 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

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

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

15.2.12 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: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

15.2.13 Keep Alive Option (`--sck-keep-alive`)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

15.2.14 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
```

15.2.15 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
```

15.2.16 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
```

15.2.17 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
```

15.3 Log Options

15.3.1 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
```

15.3.2 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
```

15.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

15.3.4 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
```

15.3.5 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
```

15.3.6 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
```

15.4 Maintainer Options

15.4.1 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
```

15.5 Repository Options

15.5.1 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
```

15.5.2 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

```
default: shared
example: --repo1-azure-key-type=sas
```

15.5.3 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
```

15.5.4 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
```

15.5.5 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
```

15.5.6 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
```

15.5.7 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
```

15.5.8 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

15.5.9 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
```

15.5.10 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
```

15.5.11 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
```

15.5.12 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.

```
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

15.5.13 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

15.5.14 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
```

15.5.15 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
```

15.5.16 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
```

15.5.17 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.

```
allowed: 0-65535
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

15.5.18 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
```

15.5.19 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

15.5.20 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
```

15.5.21 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
```

15.5.22 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
```

15.5.23 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

```
default: shared
```

```
example: --repo1-s3-key-type=auto
```

15.5.24 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

15.5.25 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
```

15.5.26 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
```

15.5.27 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
```

15.5.28 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

15.5.29 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
```

15.5.30 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
```

15.5.31 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
```

15.5.32 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
```

15.5.33 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
```

15.5.34 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
```

15.5.35 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
```

15.5.36 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
```

15.5.37 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

15.5.38 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

15.5.39 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

15.5.40 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

15.5.41 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
```

15.5.42 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.

Default chunk sizes by repo type:

- `azure` - 4MiB
- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined behavior.

```
allowed: 64KiB-1TiB
example: --repo1-storage-upload-chunk-size=16MiB
```

15.5.43 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

15.5.44 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
```

15.6 Stanza Options

15.6.1 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
```

15.6.2 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

15.6.3 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
```

15.6.4 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
```

15.6.5 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
```

15.6.6 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.

```
example: --pg1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: db-cmd

15.6.7 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

15.6.8 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
```

15.6.9 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
```

15.6.10 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
```

15.6.11 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.

```
allowed: 0-65535
example: --pg1-host-port=25
```

Deprecated Name: db-ssh-port

15.6.12 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
```

15.6.13 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

15.6.14 PostgreSQL Path Option (`--pg-path`)

PostgreSQL data directory.

This should be the same as the `data_directory` setting in `postgresql.conf`. Even though this value can be read from `postgresql.conf` or PostgreSQL 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

15.6.15 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

15.6.16 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_directory` setting in `postgresql.conf`.

```
example: --pg1-socket-path=/var/run/postgresql
```

Deprecated Name: db-socket-path

15.6.17 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
```

16 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.

16.1 Command Options

16.1.1 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
```

16.2 General Options

16.2.1 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
```

16.2.2 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.*

```
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

16.2.3 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
```

16.2.4 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 but can be disabled by setting `compress-level-network=0`. 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: 3
allowed: 0-9
example: --compress-level-network=1
```

16.2.5 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
```

16.2.6 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
```

16.2.7 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
```

16.2.8 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: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

16.2.9 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

16.2.10 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
```

16.2.11 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

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

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

16.2.12 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: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

16.2.13 Keep Alive Option (`--sck-keep-alive`)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

16.2.14 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
```

16.2.15 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
```

16.2.16 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
```

16.2.17 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
```

16.3 Log Options

16.3.1 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
```

16.3.2 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
```

16.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

16.3.4 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
```

16.3.5 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
```

16.3.6 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
```

16.4 Maintainer Options

16.4.1 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
```

16.5 Repository Options

16.5.1 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
```

16.5.2 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
```

16.5.3 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

```
default: shared
example: --repo1-azure-key-type=sas
```

16.5.4 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
```

16.5.5 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
```

16.5.6 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
```

16.5.7 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
```

16.5.8 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
```

16.5.9 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

16.5.10 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
```

16.5.11 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
```

16.5.12 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
```

16.5.13 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.

```
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

16.5.14 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

16.5.15 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
```

16.5.16 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
```

16.5.17 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
```

16.5.18 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.

```
allowed: 0-65535
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

16.5.19 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
```

16.5.20 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

16.5.21 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
```

16.5.22 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
```

16.5.23 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
```

16.5.24 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

```
default: shared
example: --repo1-s3-key-type=auto
```

16.5.25 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

16.5.26 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
```

16.5.27 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
```

16.5.28 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
```

16.5.29 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

16.5.30 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
```

16.5.31 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
```

16.5.32 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
```

16.5.33 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
```

16.5.34 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
```

16.5.35 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
```

16.5.36 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
```

16.5.37 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
```

16.5.38 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

16.5.39 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

16.5.40 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

16.5.41 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

16.5.42 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
```

16.5.43 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.

Default chunk sizes by repo type:

- `azure` - 4MiB
- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined

behavior.

```
allowed: 64KiB-1TiB
example: --repo1-storage-upload-chunk-size=16MiB
```

16.5.44 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

16.5.45 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
```

16.6 Stanza Options

16.6.1 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
```

16.6.2 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

16.6.3 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
```

16.6.4 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
```

16.6.5 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
```

16.6.6 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.

```
example: --pg1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: db-cmd

16.6.7 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

16.6.8 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
```

16.6.9 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
```

16.6.10 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
```

16.6.11 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.

```
allowed: 0-65535
example: --pg1-host-port=25
```

Deprecated Name: db-ssh-port

16.6.12 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
```

16.6.13 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

16.6.14 PostgreSQL Path Option (`--pg-path`)

PostgreSQL data directory.

This should be the same as the `data_directory` setting in `postgresql.conf`. Even though this value can be read from `postgresql.conf` or PostgreSQL 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

16.6.15 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

16.6.16 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_directory` setting in `postgresql.conf`.

```
example: --pg1-socket-path=/var/run/postgresql
```

Deprecated Name: db-socket-path

16.6.17 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
```

17 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.

17.1 Command Options

17.1.1 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
```

17.2 General Options

17.2.1 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
```

17.2.2 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.*

```
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

17.2.3 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
```

17.2.4 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 but can be disabled by setting compress-level-network=0. 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: 3
allowed: 0-9
example: --compress-level-network=1
```

17.2.5 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
```

17.2.6 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
```

17.2.7 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
```

17.2.8 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: 1800
allowed: 0.1-604800
example: --db-timeout=600
```

17.2.9 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: 60
allowed: 0.1-3600
```

```
example: --io-timeout=120
```

17.2.10 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
```

17.2.11 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

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

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

17.2.12 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: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

17.2.13 Keep Alive Option (`--sck-keep-alive`)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

17.2.14 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
```

17.2.15 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
```

17.2.16 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
```

17.2.17 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
```

17.3 Log Options

17.3.1 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
```

17.3.2 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
```

17.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

17.3.4 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
```

17.3.5 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
```

17.3.6 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
```

17.4 Maintainer Options

17.4.1 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
```

17.5 Repository Options

17.5.1 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
```

17.5.2 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

```
default: shared
example: --repo1-azure-key-type=sas
```

17.5.3 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
```

17.5.4 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
```

17.5.5 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
```

17.5.6 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
```

17.5.7 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
```

17.5.8 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

17.5.9 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
```

17.5.10 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
```

17.5.11 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
```

17.5.12 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.

```
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

17.5.13 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

17.5.14 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
```

17.5.15 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
```

17.5.16 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
```

17.5.17 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.

```
allowed: 0-65535
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

17.5.18 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
```

17.5.19 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

17.5.20 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
```

17.5.21 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
```

17.5.22 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
```

17.5.23 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

```
default: shared
example: --repo1-s3-key-type=auto
```

17.5.24 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

17.5.25 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
```

17.5.26 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
```

17.5.27 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
```

17.5.28 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

17.5.29 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
```

17.5.30 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
```

17.5.31 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
```

17.5.32 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
```

17.5.33 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
```

17.5.34 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
```

17.5.35 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
```

17.5.36 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
```

17.5.37 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

17.5.38 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

17.5.39 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

17.5.40 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

17.5.41 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
```

17.5.42 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.

Default chunk sizes by repo type:

- `azure` - 4MiB
- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined behavior.

```
allowed: 64KiB-1TiB
example: --repo1-storage-upload-chunk-size=16MiB
```

17.5.43 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

17.5.44 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
```

17.6 Stanza Options

17.6.1 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
```

17.6.2 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

17.6.3 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
```

17.6.4 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
```

17.6.5 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
```

17.6.6 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.

```
example: --pg1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: db-cmd

17.6.7 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

17.6.8 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
```

17.6.9 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
```

17.6.10 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
```

17.6.11 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.

```
allowed: 0-65535
example: --pg1-host-port=25
```

Deprecated Name: db-ssh-port

17.6.12 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
```

17.6.13 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

17.6.14 PostgreSQL Path Option (`--pg-path`)

PostgreSQL data directory.

This should be the same as the `data_directory` setting in `postgresql.conf`. Even though this value can be read from `postgresql.conf` or PostgreSQL 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

17.6.15 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

17.6.16 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_directory` setting in `postgresql.conf`.

```
example: --pg1-socket-path=/var/run/postgresql
```

Deprecated Name: db-socket-path

17.6.17 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
```

18 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.

18.1 General Options

18.1.1 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
```

18.1.2 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
```

18.1.3 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
```

18.1.4 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
```

18.1.5 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

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

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

18.1.6 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
```

18.2 Log Options

18.2.1 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
```

18.2.2 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
```

18.2.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

- `off` - No logging at all (not recommended)
- `error` - Log only errors
- `warn` - Log warnings and errors

- `info` - Log info, warnings, and errors
- `detail` - Log detail, info, warnings, and errors
- `debug` - Log debug, detail, info, warnings, and errors
- `trace` - Log trace (very verbose debugging), debug, info, warnings, and errors

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

18.2.4 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
```

18.2.5 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
```

19 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.

19.1 Command Options

19.1.1 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
```

19.2 General Options

19.2.1 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
```

19.2.2 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
```

19.2.3 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
```

19.2.4 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
```

19.2.5 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

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

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

19.2.6 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
```

19.3 Log Options

19.3.1 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
```

19.3.2 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
```

19.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

19.3.4 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
```

19.3.5 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
```

20 Verify Command (`verify`)

Verify determines if the backups and archives in the repository are valid.

20.1 Command Options

20.1.1 Output Option (`--output`)

Output type.

Output may be none (default) or text. Requesting text generates output to stdout.

```
default: none
example: --output=text
```

20.1.2 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
```

20.2 General Options

20.2.1 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
```

20.2.2 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.*

```
example: --cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

20.2.3 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
```

20.2.4 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 but can be disabled by setting compress-level-network=0. 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: 3
allowed: 0-9
```

```
example: --compress-level-network=1
```

20.2.5 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
```

20.2.6 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
```

20.2.7 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
```

20.2.8 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: 60
allowed: 0.1-3600
example: --io-timeout=120
```

20.2.9 Neutral Umask Option (--neutral-umask)

Use a neutral umask.

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

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

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

20.2.10 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
```

20.2.11 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: 1830
allowed: 0.1-604800
example: --protocol-timeout=630
```

20.2.12 Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: --no-sck-keep-alive
```

20.2.13 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
```

20.2.14 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
```

20.2.15 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
```

20.2.16 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
```

20.3 Log Options

20.3.1 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
```

20.3.2 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
```

20.3.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

20.3.4 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
```

20.3.5 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
```

20.3.6 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
```

20.4 Maintainer Options

20.4.1 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
```

20.5 Repository Options

20.5.1 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
```

20.5.2 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
```

20.5.3 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

```
default: shared
example: --repo1-azure-key-type=sas
```

20.5.4 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
```

20.5.5 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
```

20.5.6 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
```

20.5.7 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
```

20.5.8 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
```

20.5.9 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

20.5.10 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
```

20.5.11 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
```

20.5.12 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
```

20.5.13 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.

```
example: --repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

20.5.14 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

20.5.15 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
```

20.5.16 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
```

20.5.17 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
```

20.5.18 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.

```
allowed: 0-65535
example: --repo1-host-port=25
```

Deprecated Name: backup-ssh-port

20.5.19 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
```

20.5.20 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

20.5.21 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
```

20.5.22 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
```

20.5.23 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
```

20.5.24 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

```
default: shared
example: --repo1-s3-key-type=auto
```

20.5.25 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: --repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

20.5.26 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
```

20.5.27 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
```

20.5.28 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
```

20.5.29 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: --repo1-sftp-host=sftprepo.domain
```

20.5.30 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
```

20.5.31 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
```

20.5.32 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
```

20.5.33 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
```

20.5.34 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
```

20.5.35 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
```

20.5.36 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
```

20.5.37 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
```

20.5.38 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

20.5.39 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

20.5.40 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

20.5.41 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

20.5.42 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
```

20.5.43 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.

Default chunk sizes by repo type:

- `azure` - 4MiB
- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined behavior.

```
allowed: 64KiB-1TiB
example: --repo1-storage-upload-chunk-size=16MiB
```

20.5.44 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

20.5.45 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
```

21 Version Command (`version`)

Displays installed pgBackRest version.

Copyright © 2015-2024, The PostgreSQL Global Development Group, MIT License. Updated March 24, 2024

pgBackRest

Configuration Reference

Table of Contents

1 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`

2 Archive Options

The `archive` section defines options for the archive-push and archive-get commands.

2.1 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
```

2.2 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: 0-4PiB
example: archive-get-queue-max=1GiB
```

2.3 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
```

2.4 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.

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: 0-4PiB
example: archive-push-queue-max=1TiB
```

Deprecated Name: archive-queue-max

2.5 Archive Timeout Option (`--archive-timeout`)

Archive timeout.

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

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

3 Backup Options

The `backup` section defines settings related to backup.

3.1 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"
```

3.2 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
```

3.3 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
```

3.4 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
```

3.5 Backup from Standby Option (`--backup-standby`)

Backup from the standby cluster.

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

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

3.6 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
```

3.7 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/
```

3.8 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.

```
default: y
example: expire-auto=y
```

3.9 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: 1-1TiB
example: manifest-save-threshold=8GiB
```

3.10 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
```

3.11 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
```

3.12 Stop Auto Option (--stop-auto)

Stop prior failed backup on new backup.

This will only be done if an exclusive advisory lock can be acquired to demonstrate that the prior failed backup process has really stopped.

This feature is not supported for PostgreSQL >= `9.6` since backups are run in non-exclusive mode.

The setting is disabled by default because it assumes that pgBackRest is the only process doing exclusive online backups. It depends on an advisory lock that only pgBackRest sets so it may abort other processes that do exclusive online backups. Note that base_backup and pg_dump are safe to use with this setting because they do not call `pg_start_backup()` so are not exclusive.

```
default: n
example: stop-auto=y
```

4 General Options

The `general` section defines options that are common for many commands.

4.1 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
```

4.2 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.*

```
example: cmd=/var/lib/pgsql/bin/pgbackrest_wrapper.sh
```

4.3 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
```

4.4 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
```

4.5 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).

The following are the defaults levels based on `compress-type` when `compress-level` is not specified:

- `bz2` - 9
- `gz` - 6
- `lz4` - 1
- `zst` - 3

```
example: compress-level=9
```

4.6 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 but can be disabled by setting `compress-level-network=0`. 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: 3
allowed: 0-9
example: compress-level-network=1
```

4.7 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
```

4.8 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
```

4.9 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
```

4.10 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
```

4.11 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: 1800
allowed: 0.1-604800
example: db-timeout=600
```

4.12 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
```

4.13 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=y
```

4.14 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: 60
allowed: 0.1-3600
example: io-timeout=120
```

4.15 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
```

4.16 Neutral Umask Option (`--neutral-umask`)

Use a neutral umask.

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

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

```
default: y
example: neutral-umask=n
```

4.17 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
```

4.18 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: 1830
allowed: 0.1-604800
example: protocol-timeout=630
```

4.19 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=y
```

4.20 Keep Alive Option (--sck-keep-alive)

Keep-alive enable.

Enables keep-alive messages on socket connections.

```
default: y
example: sck-keep-alive=n
```

4.21 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`.

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
```

4.22 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
```

4.23 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
```

4.24 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
```

4.25 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
```

5 Log Options

The `log` section defines logging-related settings.

> ***CAUTION:*** *Trace-level logging may expose secrets such as keys and passwords. Use with caution!*

5.1 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
```

5.2 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
```

5.3 Std Error Log Level Option (`--log-level-stderr`)

Level for stderr logging.

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

The following log levels are supported:

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

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

5.4 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
```

5.5 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
```

5.6 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
```

6 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.

6.1 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
```

6.2 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
```

6.3 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
```

7 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.

7.1 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
```

7.2 Azure Repository Account Option (--repo-azure-account)

Azure repository account.

Azure account used to store the repository.

```
example: repo1-azure-account=pg-backup
```

7.3 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
```

7.4 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
```

7.5 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==
```

7.6 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

```
default: shared
example: repo1-azure-key-type=sas
```

7.7 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
```

7.8 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
```

7.9 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
```

7.10 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
```

7.11 Repository Bundle Size Option (`--repo-bundle-size`)

Target size for file bundles.

Defines the total size of files that will be added to a single bundle. Most bundles will be smaller than this size but it is possible that some will be slightly larger, 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
```

7.12 Repository Cipher Passphrase Option (`--repo-cipher-pass`)

Repository cipher passphrase.

Passphrase used to encrypt/decrypt files of the repository.

```
example: repo1-cipher-pass=zWaf6XtpjIVZC5444yXB+cgFDFl7MxGlgkZSaoPvTGirhPygu4jOKOXf9LO4vjfO
```

7.13 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
```

7.14 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
```

7.15 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
```

7.16 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
```

7.17 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
```

7.18 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

7.19 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

7.20 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
```

7.21 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
```

7.22 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
```

7.23 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.

```
example: repo1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: backup-cmd

7.24 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

7.25 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
```

7.26 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
```

7.27 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
```

7.28 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.

```
allowed: 0-65535
example: repo1-host-port=25
```

Deprecated Name: backup-ssh-port

7.29 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
```

7.30 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

7.31 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
```

7.32 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

7.33 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

7.34 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.

```
allowed: 1-9999999
example: repo1-retention-diff=3
```

Deprecated Name: retention-diff

7.35 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

7.36 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 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.

```
default: count
example: repo1-retention-full-type=time
```

7.37 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
```

7.38 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
```

7.39 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
```

7.40 S3 Repository Access Key Option (`--repo-s3-key`)

S3 repository access key.

AWS key used to access this bucket.

```
example: repo1-s3-key=AKIAIOSFODNN7EXAMPLE
```

7.41 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
```

7.42 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

```
default: shared
example: repo1-s3-key-type=auto
```

7.43 S3 Repository KMS Key ID Option (`--repo-s3-kms-key-id`)

S3 repository KMS key.

Setting this option enables S3 server-side encryption using the specified AWS key management service key.

```
example: repo1-s3-kms-key-id=bceb4f13-6939-4be3-910d-df54dee817b7
```

7.44 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
```

7.45 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
```

7.46 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 ...
```

7.47 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
```

7.48 SFTP Repository Host Option (`--repo-sftp-host`)

SFTP repository host.

The SFTP host containing the repository.

```
example: repo1-sftp-host=sftprepo.domain
```

7.49 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
```

7.50 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
```

7.51 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
```

7.52 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
```

7.53 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
```

7.54 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
```

7.55 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
```

7.56 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
```

7.57 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
```

7.58 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

7.59 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

7.60 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

7.61 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

7.62 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
```

7.63 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.

Default chunk sizes by repo type:

- `azure` - 4MiB
- `gcs` - 4MiB
- `s3` - 5MiB

Note that valid chunk sizes vary by storage type and by platform. For example, AWS S3 has a minimum chunk size of 5MiB but S3 clones may accept lower values. 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. No attempt is made to validate configured chunk sizes so selecting an invalid value will lead to errors from the storage service or undefined behavior.

```
allowed: 64KiB-1TiB
example: repo1-storage-upload-chunk-size=16MiB
```

7.64 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

7.65 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
```

8 Restore Options

The `restore` section defines settings used for restoring backups.

8.1 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
```

8.2 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
```

8.3 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
```

8.4 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
```

8.5 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
```

8.6 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
```

8.7 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
```

8.8 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.

```
example: tablespace-map-all=/data/tablespace
```

9 Server Options

The `server` section defines options used for configuring the TLS server.

9.1 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=*
```

9.2 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 repeating 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
```

9.3 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
```

9.4 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
```

9.5 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
```

9.6 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
```

10 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).

10.1 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
```

10.2 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

10.3 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
```

10.4 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
```

10.5 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
```

10.6 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.

```
example: pg1-host-cmd=/usr/lib/backrest/bin/pgbackrest
```

Deprecated Name: db-cmd

10.7 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

10.8 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
```

10.9 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
```

10.10 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
```

10.11 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.

```
allowed: 0-65535
example: pg1-host-port=25
```

Deprecated Name: db-ssh-port

10.12 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
```

10.13 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

10.14 PostgreSQL Path Option (--pg-path)

PostgreSQL data directory.

This should be the same as the data_directory setting in postgresql.conf. Even though this value can be read from postgresql.conf or PostgreSQL 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

10.15 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

10.16 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_directory setting in postgresql.conf.

```
example: pg1-socket-path=/var/run/postgresql
```

Deprecated Name: db-socket-path

10.17 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
```

Copyright © 2015-2024, The PostgreSQL Global Development Group, MIT License. Updated March 24, 2024

pgBackRest

Frequently Asked Questions

Table of Contents

1 Introduction

Frequently Asked Questions are intended to provide details for specific questions that may or may not be covered in the User Guide, Configuration, or Command reference. If you are unable to find details for your specific issue here, remember that the pgBackRest Issues List in GitHub is also a valuable resource.

2 What if I get the "could not find WAL segment" error?

The cause of this error can be a result of many different issues, some of which may be:

- misconfigured archive_command
- misconfigured pgBackRest configuration files
- network or permissions issue
- third party product (e.g. S3, Swift or Minio) configuration issue
- large amount of WAL queueing to be archived

It is advisable to:

- check the archive_command in PostgreSQL

- check the pgBackRest configuration settings on each host (e.g. pg* settings are set on the repository host and repo* settings on the pg host)
- run the check command with `--archive-timeout` set to a higher value than in the pgBackRest configuration file (or default) to see if the WAL queue needs more time to clear. If the system is generating a lot of WAL, then consider configuring asynchronous archiving

3 How do I manually purge a backup set?

A full backup set can be expired using the `--set` option as explained in Command Reference: Expire.

4 How can I configure options independently for each command?

pgBackRest has the ability to set options independently in the configuration file for each command. Configure Cluster Stanza details this feature as well as option precedence.

For example, the `process-max` option can be optimized for each command:

```
[global]
# used where not overridden
process-max=2

[global:backup]
# more cores for backup
process-max=4

[global:restore]
# all the cores for restore
process-max=8

[global:archive-push]
# more cores for archive-push
process-max=3

[global:archive-get]
# fewer cores for archive-get
process-max=1
```

5 Can I use dots (periods) in my S3 bucket name?

RFC-2818 does not allow wildcards to match on a dot (.) so s3 bucket names must not contain dots. If there are dots in the S3 bucket name then an error such as "unable to find hostname 'my.backup.bucket.s3.amazonaws.com' in certificate common name or subject alternative names" will occur.

6 Where can I find packages for older versions of pgBackRest?

The apt.postgresql.org repository maintains an archive of older versions. Debian also maintains snapshots of all test builds.

7 Why does a backup attempt fail when `backup-standby=y` and the standby database is down?

Configuring backup from standby is generally intended to reduce load on the primary, so switching backups to the primary when the standby is down often defeats the point. Putting more load on the primary in a situation where there are already failures in the system is not recommended. Backups are not critical as long as you have one that is fairly recent -- the important thing is to keep up with WAL archiving. There is plenty of time to get a backup when the system is stable again.

If you really need a backup, the solution is to have more standbys or remove `backup-standby`. This can be overridden on the command line with `--no-backup-standby`, so there is no need to reconfigure for a one-off backup.

8 Should I setup my repository on a standby host?

No. When primary and standby databases are configured, the pgBackRest configuration files should be symmetric in order to seamlessly handle failovers. If they are not, the configurations will need to be changed on failover or further problems may result.

See the Dedicated Repository Host section of the User Guide for more information.

9 Time-based Point-in-Time Recovery does not appear to work, why?

The most common mistake when using time-based Point-in-Time Recovery is forgetting to choose a backup set that is before the target time. pgBackRest will attempt to discover a backup to play forward from the time specified by the `--target=` if the `--set` option is not specified. If a backup set cannot be found, then restore will default to the latest backup. However, if the latest backup is after the target time, then `--target=` is not considered valid by PostgreSQL and is therefore ignored, resulting in WAL recovery to the latest time available.

To use the `--set` option, choose a backup set by running the info command and finding the backup with a timestamp stop that is before the target time. Then when running the restore, specify the option `--set=BACKUP_LABEL` where `BACKUP_LABEL` is the chosen backup set.

See the Point-in-Time Recovery section of the User Guide for more information.

10 What does the WAL archive suffix mean?

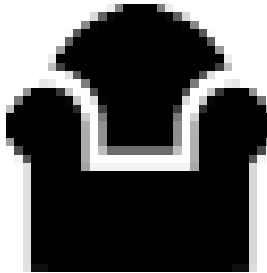
The suffix is the SHA1 checksum used to verify file integrity. There is no way to omit it.

11 Does it take longer to restore specific backup types (full,

differential, incremental)?

The various backup types require the same amount of time to restore. Restore retrieves files based on the backup manifest, which may reference files from a previous backup in the case of incremental or differential backups. While there could be differences in time spent *making* a given backup (depending on backup type), database size determines restore time (disk I/O, network I/O, etc. being equal).

Copyright © 2015-2024, The PostgreSQL Global Development Group, MIT License. Updated March 24, 2024



pgBackRest

Reliable PostgreSQL Backup & Restore

[User Guides] [Configuration] [Commands] [FAQ] [Metrics]

Introduction

pgBackRest is a reliable backup and restore solution for PostgreSQL that seamlessly scales up to the largest databases and workloads.

pgBackRest v2.51 is the current stable release. Release notes are on the Releases page.

Please find us on GitHub and give us a star if you like pgBackRest!

Features

Parallel Backup & Restore

Compression is usually the bottleneck during backup operations so pgBackRest solves this problem with parallel processing and more efficient compression algorithms such as lz4 and zstd.

Local or Remote Operation

A custom protocol allows pgBackRest to backup, restore, and archive locally or remotely via TLS/SSH with minimal configuration. An interface to query PostgreSQL is also provided via the protocol layer so that remote access to PostgreSQL is never required, which enhances security.

Multiple Repositories

Multiple repositories allow, for example, a local repository with minimal retention for fast restores and a remote repository with a longer retention for redundancy and access across the enterprise.

Full, Differential, & Incremental Backups (at File or Block Level)

Full, differential, and incremental backups are supported. pgBackRest is not susceptible to the time resolution issues of rsync, making differential and incremental backups safe without the requirement to checksum each file. Block-level backups save space by only copying the parts of files that have changed.

Backup Rotation & Archive Expiration

Retention polices can be set for full and differential backups to create coverage for any time frame. The WAL archive can be maintained for all backups or strictly for the most recent backups. In the latter case WAL required to make older backups consistent will be maintained in the archive.

Backup Integrity

Checksums are calculated for every file in the backup and rechecked during a restore or verify. After a backup finishes copying files, it waits until every WAL segment required to make the backup consistent reaches the repository.

Backups in the repository may be stored in the same format as a standard PostgreSQL cluster (including tablespaces). If compression is disabled and hard links are enabled it is possible to snapshot a backup in the repository and bring up a PostgreSQL cluster directly on the snapshot. This is advantageous for terabyte-scale databases that are time consuming to restore in the traditional way.

All operations utilize file and directory level fsync to ensure durability.

Page Checksums

If page checksums are enabled pgBackRest will validate the checksums for every file that is copied during a backup. All page checksums are validated during a full backup and checksums in files that have changed are validated during differential and incremental backups.

Validation failures do not stop the backup process, but warnings with details of exactly which pages have failed validation are output to the console and file log.

This feature allows page-level corruption to be detected early, before backups that contain valid copies of the data have expired.

Backup Resume

An interrupted backup can be resumed from the point where it was stopped. Files that were already copied are compared with the checksums in the manifest to ensure integrity. Since this operation can take place entirely on the repository host, it reduces load on the PostgreSQL host and saves time since checksum calculation is faster than compressing and retransmitting data.

Streaming Compression & Checksums

Compression and checksum calculations are performed in stream while files are being copied to the repository, whether the repository is located locally or remotely.

If the repository is on a repository host, compression is performed on the PostgreSQL host and files are transmitted in a compressed format and simply stored on the repository host. When compression is disabled a lower level of compression is utilized to make efficient use of available bandwidth while keeping CPU cost to a minimum.

Delta Restore

The manifest contains checksums for every file in the backup so that during a restore it is possible to use these checksums to speed processing enormously. On a delta restore any files not present in the backup are first removed and then checksums are generated for the remaining files. Files that match the backup are left in place and the rest of the files are restored as usual. Parallel processing can lead to a dramatic reduction in restore times.

Parallel, Asynchronous WAL Push & Get

Dedicated commands are included for pushing WAL to the archive and getting WAL from the archive. Both commands support parallelism to accelerate processing and run asynchronously to provide the fastest possible response time to PostgreSQL.

WAL push automatically detects WAL segments that are pushed multiple times and de-duplicates when the segment is identical, otherwise an error is raised. Asynchronous WAL push allows transfer to be offloaded to another process which compresses WAL segments in parallel for maximum throughput. This can be a critical feature for databases with

extremely high write volume.

Asynchronous WAL get maintains a local queue of WAL segments that are decompressed and ready for replay. This reduces the time needed to provide WAL to PostgreSQL which maximizes replay speed. Higher-latency connections and storage (such as S3) benefit the most.

The push and get commands both ensure that the database and repository match by comparing PostgreSQL versions and system identifiers. This virtually eliminates the possibility of misconfiguring the WAL archive location.

Tablespace & Link Support

Tablespaces are fully supported and on restore tablespaces can be remapped to any location. It is also possible to remap all tablespaces to one location with a single command which is useful for development restores.

File and directory links are supported for any file or directory in the PostgreSQL cluster. When restoring it is possible to restore all links to their original locations, remap some or all links, or restore some or all links as normal files or directories within the cluster directory.

S3, Azure, and GCS Compatible Object Store Support

pgBackRest repositories can be located in S3, Azure, and GCS compatible object stores to allow for virtually unlimited capacity and retention.

Encryption

pgBackRest can encrypt the repository to secure backups wherever they are stored.

Compatibility with ten versions of PostgreSQL

pgBackRest includes support for ten versions of PostgreSQL, the five supported versions and the last five EOL versions. This allows ample time to upgrade to a supported version.

Getting Started

pgBackRest strives to be easy to configure and operate:

- User guides for various operating systems and PostgreSQL versions.
- Command reference for command-line operations.
- Configuration reference for creating pgBackRest configurations.

Documentation for v1 can be found here. No further releases are planned for v1 because v2 is backward-compatible with v1 options and repositories.

Contributions

Contributions to pgBackRest are always welcome! Please see our Contributing Guidelines for details on how to contribute features, improvements or issues.

Support

pgBackRest is completely free and open source under the MIT license. You may use it for personal or commercial purposes without any restrictions whatsoever. Bug reports are taken very seriously and will be addressed as quickly as possible.

Creating a robust disaster recovery policy with proper replication and backup strategies can be a very complex and daunting task. You may find that you need help during the architecture phase and ongoing support to ensure that your enterprise continues running smoothly.

Crunchy Data provides packaged versions of pgBackRest for major operating systems and expert full life-cycle commercial support for pgBackRest and all things PostgreSQL. Crunchy Data is committed to providing open source solutions with no vendor lock-in, ensuring that cross-compatibility with the community version of pgBackRest is always strictly maintained.

Please visit Crunchy Data for more information.

Recognition

Primary recognition goes to Stephen Frost for all his valuable advice and criticism during the development of pgBackRest.

Crunchy Data has contributed significant time and resources to pgBackRest and continues to actively support development. Resonate also contributed to the development of pgBackRest and allowed early (but well tested) versions to be installed as their primary PostgreSQL backup solution.

Armchair graphic by Sandor Szabo.

Copyright © 2015-2024, The PostgreSQL Global Development Group, MIT License. Updated March 24, 2024

pgBackRest

Reliable PostgreSQL Backup & Restore

[User Guides] [Configuration] [Commands] [FAQ] [Metrics]

Introduction

pgBackRest is a reliable backup and restore solution for PostgreSQL that seamlessly scales up to the largest databases and workloads.

pgBackRest v2.51 is the current stable release. Release notes are on the Releases page.

Please find us on GitHub and give us a star if you like pgBackRest!

Features

Parallel Backup & Restore

Compression is usually the bottleneck during backup operations so pgBackRest solves this problem with parallel processing and more efficient compression algorithms such as lz4 and zstd.

Local or Remote Operation

A custom protocol allows pgBackRest to backup, restore, and archive locally or remotely via TLS/SSH with minimal configuration. An interface to query PostgreSQL is also provided via the protocol layer so that remote access to PostgreSQL is never required, which enhances security.

Multiple Repositories

Multiple repositories allow, for example, a local repository with minimal retention for fast restores and a remote repository with a longer retention for redundancy and access across the enterprise.

Full, Differential, & Incremental Backups (at File or Block Level)

Full, differential, and incremental backups are supported. pgBackRest is not susceptible to the time resolution issues of rsync, making differential and incremental backups safe without the requirement to checksum each file. Block-level backups save space by only copying the parts of files that have changed.

Backup Rotation & Archive Expiration

Retention polices can be set for full and differential backups to create coverage for any time frame. The WAL archive can be maintained for all backups or strictly for the most recent backups. In the latter case WAL required to make older backups consistent will be maintained in the archive.

Backup Integrity

Checksums are calculated for every file in the backup and rechecked during a restore or verify. After a backup finishes copying files, it waits until every WAL segment required to make the backup consistent reaches the repository.

Backups in the repository may be stored in the same format as a standard PostgreSQL cluster (including tablespaces). If compression is disabled and hard links are enabled it is possible to snapshot a backup in the repository and bring up a PostgreSQL cluster directly on the snapshot. This is advantageous for terabyte-scale databases that are time consuming to restore in the traditional way.

All operations utilize file and directory level fsync to ensure durability.

Page Checksums

If page checksums are enabled pgBackRest will validate the checksums for every file that is copied during a backup. All page checksums are validated during a full backup and checksums in files that have changed are validated during differential and incremental backups.

Validation failures do not stop the backup process, but warnings with details of exactly which pages have failed validation are output to the console and file log.

This feature allows page-level corruption to be detected early, before backups that contain valid copies of the data have expired.

Backup Resume

An interrupted backup can be resumed from the point where it was stopped. Files that were already copied are compared with the checksums in the manifest to ensure integrity. Since this operation can take place entirely on the repository host, it reduces load on the PostgreSQL host and saves time since checksum calculation is faster than compressing and retransmitting data.

Streaming Compression & Checksums

Compression and checksum calculations are performed in stream while files are being copied to the repository, whether the repository is located locally or remotely.

If the repository is on a repository host, compression is performed on the PostgreSQL host and files are transmitted in a compressed format and simply stored on the repository host. When compression is disabled a lower level of compression is utilized to make efficient use of available bandwidth while keeping CPU cost to a minimum.

Delta Restore

The manifest contains checksums for every file in the backup so that during a restore it is possible to use these checksums to speed processing enormously. On a delta restore any files not present in the backup are first removed and then checksums are generated for the remaining files. Files that match the backup are left in place and the rest of the files are restored as usual. Parallel processing can lead to a dramatic reduction in restore times.

Parallel, Asynchronous WAL Push & Get

Dedicated commands are included for pushing WAL to the archive and getting WAL from the archive. Both commands support parallelism to accelerate processing and run asynchronously to provide the fastest possible response time to PostgreSQL.

WAL push automatically detects WAL segments that are pushed multiple times and de-duplicates when the segment is identical, otherwise an error is raised. Asynchronous WAL push allows transfer to be offloaded to another process which compresses WAL segments in parallel for maximum throughput. This can be a critical feature for databases with

extremely high write volume.

Asynchronous WAL get maintains a local queue of WAL segments that are decompressed and ready for replay. This reduces the time needed to provide WAL to PostgreSQL which maximizes replay speed. Higher-latency connections and storage (such as S3) benefit the most.

The push and get commands both ensure that the database and repository match by comparing PostgreSQL versions and system identifiers. This virtually eliminates the possibility of misconfiguring the WAL archive location.

Tablespace & Link Support

Tablespaces are fully supported and on restore tablespaces can be remapped to any location. It is also possible to remap all tablespaces to one location with a single command which is useful for development restores.

File and directory links are supported for any file or directory in the PostgreSQL cluster. When restoring it is possible to restore all links to their original locations, remap some or all links, or restore some or all links as normal files or directories within the cluster directory.

S3, Azure, and GCS Compatible Object Store Support

pgBackRest repositories can be located in S3, Azure, and GCS compatible object stores to allow for virtually unlimited capacity and retention.

Encryption

pgBackRest can encrypt the repository to secure backups wherever they are stored.

Compatibility with ten versions of PostgreSQL

pgBackRest includes support for ten versions of PostgreSQL, the five supported versions and the last five EOL versions. This allows ample time to upgrade to a supported version.

Getting Started

pgBackRest strives to be easy to configure and operate:

- User guides for various operating systems and PostgreSQL versions.
- Command reference for command-line operations.
- Configuration reference for creating pgBackRest configurations.

Documentation for v1 can be found here. No further releases are planned for v1 because v2 is backward-compatible with v1 options and repositories.

Contributions

Contributions to pgBackRest are always welcome! Please see our Contributing Guidelines for details on how to contribute features, improvements or issues.

Support

pgBackRest is completely free and open source under the MIT license. You may use it for personal or commercial purposes without any restrictions whatsoever. Bug reports are taken very seriously and will be addressed as quickly as possible.

Creating a robust disaster recovery policy with proper replication and backup strategies can be a very complex and daunting task. You may find that you need help during the architecture phase and ongoing support to ensure that your enterprise continues running smoothly.

Crunchy Data provides packaged versions of pgBackRest for major operating systems and expert full life-cycle commercial support for pgBackRest and all things PostgreSQL. Crunchy Data is committed to providing open source solutions with no vendor lock-in, ensuring that cross-compatibility with the community version of pgBackRest is always strictly maintained.

Please visit Crunchy Data for more information.

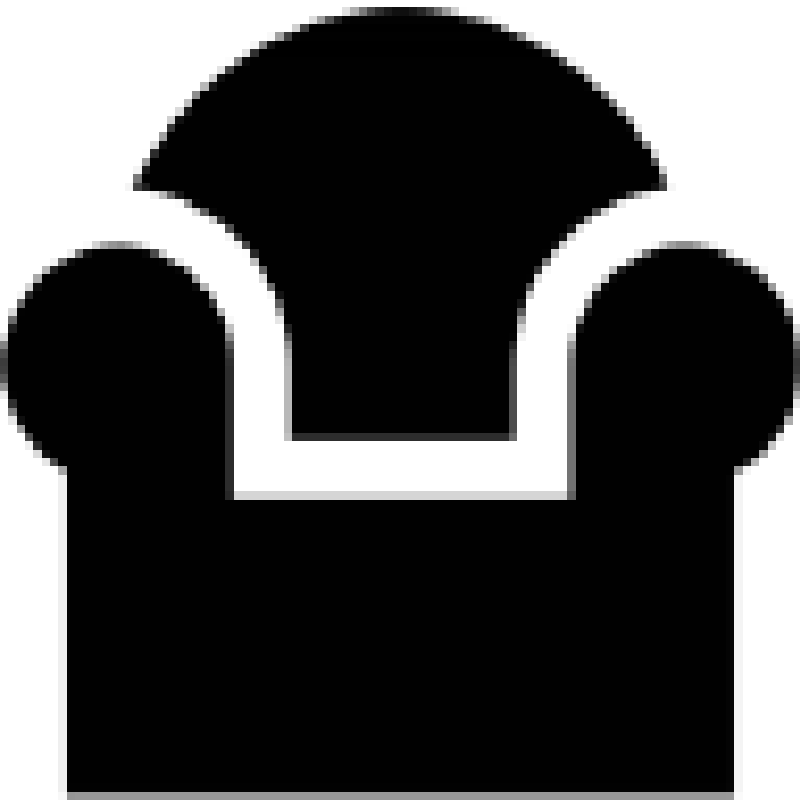
Recognition

Primary recognition goes to Stephen Frost for all his valuable advice and criticism during the development of pgBackRest.

Crunchy Data has contributed significant time and resources to pgBackRest and continues to actively support development. Resonate also contributed to the development of pgBackRest and allowed early (but well tested) versions to be installed as their primary PostgreSQL backup solution.

Armchair graphic by Sandor Szabo.

Copyright © 2015-2024, The PostgreSQL Global Development Group, MIT License. Updated March 24, 2024



pgBackRest

Metrics

Table of Contents

1 Code Coverage

pgBackRest aims to have complete function/branch/line coverage for the core C code in `/src`.

Function/line coverage is complete with no exceptions.

Branch coverage excludes branches inside macros and `assert()` calls. Macros have their own unit tests so they do not need to be tested everywhere they appear. Asserts are not expected to have complete branch coverage since they test cases that should always be true.

Directory	Functions	Branches	Lines
build/common	22/22 (100.0%)	56/56 (100.0%)	290/290 (100.0%)
build/config	31/31 (100.0%)	462/462 (100.0%)	980/980 (100.0%)
build/error	6/6 (100.0%)	26/26 (100.0%)	78/78 (100.0%)
build/help	13/13 (100.0%)	134/134 (100.0%)	262/262 (100.0%)
build/postgres	8/8 (100.0%)	60/60 (100.0%)	151/151 (100.0%)
command	9/9 (100.0%)	72/72 (100.0%)	158/158 (100.0%)
command/annotate	1/1 (100.0%)	12/12 (100.0%)	33/33 (100.0%)
command/archive	15/15 (100.0%)	104/104 (100.0%)	281/281 (100.0%)
command/archive/get	9/9 (100.0%)	200/200 (100.0%)	469/469 (100.0%)
command/archive/push	12/12 (100.0%)	130/130 (100.0%)	407/407 (100.0%)
command/backup	50/50 (100.0%)	790/790 (100.0%)	1895/1895 (100.0%)
command/check	13/13 (100.0%)	106/106 (100.0%)	272/272 (100.0%)
command/control	4/4 (100.0%)	32/32 (100.0%)	60/60 (100.0%)
command/expire	10/10 (100.0%)	240/240 (100.0%)	398/398 (100.0%)
command/help	5/5 (100.0%)	134/134 (100.0%)	243/243 (100.0%)
command/info	13/13 (100.0%)	346/346 (100.0%)	712/712 (100.0%)
command/local	1/1 (100.0%)	---	7/7 (100.0%)
command/remote	1/1 (100.0%)	6/6 (100.0%)	21/21 (100.0%)
command/repo	9/9 (100.0%)	108/108 (100.0%)	235/235 (100.0%)
command/restore	34/34 (100.0%)	696/696 (100.0%)	1448/1448 (100.0%)
command/server	6/6 (100.0%)	24/24 (100.0%)	88/88 (100.0%)
command/stanza	5/5 (100.0%)	106/106 (100.0%)	135/135 (100.0%)
command/verify	21/21 (100.0%)	314/314 (100.0%)	757/757 (100.0%)
common	149/149 (100.0%)	634/634 (100.0%)	1856/1856 (100.0%)
common/compress	13/13 (100.0%)	24/24 (100.0%)	122/122 (100.0%)

common/compress/bz2	13/13 (100.0%)	20/20 (100.0%)	173/173 (100.0%)
common/compress/gz	13/13 (100.0%)	26/26 (100.0%)	174/174 (100.0%)
common/compress/lz4	15/15 (100.0%)	24/24 (100.0%)	179/179 (100.0%)
common/compress/zst	13/13 (100.0%)	12/12 (100.0%)	146/146 (100.0%)
common/crypto	32/32 (100.0%)	88/88 (100.0%)	540/540 (100.0%)
common/error	33/33 (100.0%)	62/62 (100.0%)	188/188 (100.0%)
common/io	59/59 (100.0%)	182/182 (100.0%)	740/740 (100.0%)
common/io/filter	31/31 (100.0%)	92/92 (100.0%)	385/385 (100.0%)
common/io/http	47/47 (100.0%)	250/250 (100.0%)	708/708 (100.0%)
common/io/socket	28/28 (100.0%)	104/104 (100.0%)	440/440 (100.0%)
common/io/tls	34/34 (100.0%)	98/98 (100.0%)	509/509 (100.0%)
common/type	320/320 (100.0%)	842/842 (100.0%)	4141/4141 (100.0%)
config	90/90 (100.0%)	984/984 (100.0%)	1876/1876 (100.0%)
db	24/24 (100.0%)	116/116 (100.0%)	442/442 (100.0%)
doc/src/command/build	8/8 (100.0%)	134/134 (100.0%)	301/301 (100.0%)
info	92/92 (100.0%)	944/944 (100.0%)	2504/2504 (100.0%)
postgres	37/37 (100.0%)	136/136 (100.0%)	475/475 (100.0%)
postgres/interface	5/5 (100.0%)	12/12 (100.0%)	52/52 (100.0%)
protocol	55/55 (100.0%)	228/228 (100.0%)	963/963 (100.0%)
storage	54/54 (100.0%)	228/228 (100.0%)	811/811 (100.0%)
storage/azure	25/25 (100.0%)	116/116 (100.0%)	533/533 (100.0%)
storage/cifs	2/2 (100.0%)	---	18/18 (100.0%)
storage/gcs	32/32 (100.0%)	130/130 (100.0%)	678/678 (100.0%)
storage/posix	28/28 (100.0%)	167/168 (99.40%)	485/485 (100.0%)
storage/remote	35/35 (100.0%)	120/120 (100.0%)	801/801 (100.0%)
storage/s3	30/30 (100.0%)	150/150 (100.0%)	726/726 (100.0%)
storage/sftp	31/31 (100.0%)	404/404 (100.0%)	834/834 (100.0%)
TOTAL	1646/1646 (100.0%)	10485/10486 (99.99%)	31180/31180 (100.0%)

The C unit test modules in `/test/src/module` also have complete function/line coverage but are not included in the report.

Copyright © 2015-2024, The PostgreSQL Global Development Group, MIT License. Updated March 24, 2024

pgBackRest User Guide

Debian & Ubuntu

Table of Contents

1 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 15, 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.

2 Concepts

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

2.1 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.

2.2 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.

2.3 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).

2.4 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.

3 Upgrading pgBackRest

3.1 Upgrading pgBackRest from v1 to v2

Upgrading from v1 to v2 is fairly straight-forward. The repository format has not changed and all non-deprecated options from v1 are accepted, so for most installations it is simply a matter of installing the new version.

However, there are a few caveats:

- The deprecated `thread-max` option is no longer valid. Use `process-max` instead.
- The deprecated `archive-max-mb` option is no longer valid. This has been replaced with the `archive-push-queue-max` option which has different semantics.
- The default for the `backup-user` option has changed from `backrest` to `pgbackrest`.
- In v2.02 the default location of the pgBackRest configuration file has changed from `/etc/pgbackrest.conf` to `/etc/pgbackrest/pgbackrest.conf`. If `/etc/pgbackrest/pgbackrest.conf` does not exist, the `/etc/pgbackrest.conf` file will be loaded instead, if it exists.

Many option names have changed to improve consistency although the old names from v1 are still accepted. In general, `db-*` options have been renamed to `pg-*` and `backup-*`/`retention-*` options have been renamed to `repo-*` when appropriate.

PostgreSQL and repository options must be indexed when using the new names introduced in v2, e.g. `pg1-host`, `pg1-path`, `repo1-path`, `repo1-type`, etc.

3.2 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.

4 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.

The preferred build method is meson/ninja as shown below. The autoconf/make method is also provided for legacy purposes, see Build.

build ⇒ Download version `2.51` of pgBackRest to `/build` path

```
$ mkdir -p /build
$ wget -q -O - \
       https://github.com/pgbackrest/pgbackrest/archive/release/2.51.tar.gz | \
       tar zx -C /build
```

build ⇒ Install build dependencies

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

build ⇒ Configure and compile pgBackRest

```
$ meson setup /build/pgbackrest /build/pgbackrest-release-2.51
$ ninja -C /build/pgbackrest
```

5 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 2.51 - 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 the repository.
    version         Get version.

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

6 Quick Start

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

6.1 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/15/bin/initdb \
      -D /var/lib/postgresql/15/demo -k -A peer
$ sudo pg_createcluster 15 demo
Configuring already existing cluster (configuration: /etc/postgresql/15/demo, data:
/var/lib/postgresql/15/demo, owner: 102:103)
Ver Cluster Port Status Owner    Data directory              Log file
15  demo    5432 down   postgres /var/lib/postgresql/15/demo
/var/log/postgresql/postgresql-15-demo.log
```

6.2 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` in `postgresql.conf`.

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

When creating the `/etc/pgbackrest/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/15/demo
```

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

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.

The files are concatenated as if they were one big file; 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 2.51 - '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
```

6.3 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/15/demo

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

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

6.4 Configure Archiving

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

pg-primary:`/etc/postgresql/15/demo/postgresql.conf` ⇒ Configure archive settings

```
  archive_command = 'pgbackrest --stanza=demo archive-push %p'
  archive_mode = on
  max_wal_senders = 3
  wal_level = replica
```

`%p` is how PostgreSQL specifies the location of the WAL segment to be archived. Setting `wal_level` to at least `replica` and increasing `max_wal_senders` is a good idea even if there are currently no replicas as this will allow them to be added later without restarting the primary cluster.

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

pg-primary ⇒ Restart the demo cluster

```
$ sudo pg_ctlcluster 15 demo restart
```

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

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/15/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`.

6.5 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/15/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.

6.6 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/15/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
```

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

6.7 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 2.51: --exec-id=429-7ad4af6d --log-level-
console=info --log-level-stderr=off --no-log-timestamp --pg1-
path=/var/lib/postgresql/15/demo --repo1-cipher-pass= --repo1-cipher-type=aes-256-cbc --
repo1-path=/var/lib/pgbackrest --stanza=demo
P00   INFO: stanza-create for stanza 'demo' on repo1
P00   INFO: stanza-create command end: completed successfully
```

6.8 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 2.51: --exec-id=438-4b9091f1 --log-level-console=info --
log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/postgresql/15/demo --repo1-
cipher-pass= --repo1-cipher-type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --
stanza=demo
P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)
P00   INFO: WAL segment 000000010000000000000001 successfully archived to
'/var/lib/pgbackrest/archive/demo/15-1/0000000100000000/000000010000000000000001-
427f252ef49175611d0f6581b25edbbef146a6b9.gz' on repo1
P00   INFO: check command end: completed successfully
```

6.9 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/15/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 2.51: --exec-id=465-2c592f2e --log-level-console=info -
-log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/postgresql/15/demo --repo1-
cipher-pass= --repo1-cipher-type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo1-
retention-full=2 --stanza=demo --start-fast
P00   WARN: no prior backup exists, incr backup has been changed to full
P00   INFO: execute non-exclusive backup start: backup begins after the requested
immediate checkpoint completes
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 = 20240320-211134F
P00   INFO: full backup size = 21.8MB, file total = 961
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.51: --exec-id=465-2c592f2e --log-level-console=info -
-log-level-stderr=off --no-log-timestamp --repo1-cipher-pass= --repo1-cipher-type=aes-
256-cbc --repo1-path=/var/lib/pgbackrest --repo1-retention-full=2 --stanza=demo
```

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 = 20240320-211134F_20240320-211138D
P00   INFO: diff backup size = 8.3KB, file total = 961
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.51: --exec-id=492-5729be80 --log-level-console=info -
-log-level-stderr=off --no-log-timestamp --repo1-cipher-pass= --repo1-cipher-type=aes-
256-cbc --repo1-path=/var/lib/pgbackrest --repo1-retention-full=2 --stanza=demo
```

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.

6.10 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.

6.11 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 (15): 000000010000000000000001/000000010000000000000005
          full backup: 20240320-211134F
              timestamp start/stop: 2024-03-20 21:11:34+00 / 2024-03-20 21:11:37+00
              wal start/stop: 000000010000000000000002 / 000000010000000000000003
              database size: 21.8MB, database backup size: 21.8MB
              repo1: backup set size: 2.9MB, backup size: 2.9MB
          diff backup: 20240320-211134F_20240320-211138D
              timestamp start/stop: 2024-03-20 21:11:38+00 / 2024-03-20 21:11:40+00
              wal start/stop: 000000010000000000000004 / 000000010000000000000005
              database size: 21.8MB, database backup size: 8.3KB
              repo1: backup set size: 2.9MB, backup size: 464B
              backup reference list: 20240320-211134F
```

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.

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`' message will appear beside the '`status`' information if one of those commands is currently running on the host.

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

6.12 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 15 demo stop
$ sudo -u postgres rm /var/lib/postgresql/15/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 15 demo start
  Error: /usr/lib/postgresql/15/bin/pg_ctl /usr/lib/postgresql/15/bin/pg_ctl start -D
  /var/lib/postgresql/15/demo -l /var/log/postgresql/postgresql-15-demo.log -s -o  -c
  config_file="/etc/postgresql/15/demo/postgresql.conf"  exited with status 1:
  postgres: could not find the database system
  Expected to find it in the directory "/var/lib/postgresql/15/demo",
  but could not open file "/var/lib/postgresql/15/demo/global/pg_control": No such file or
  directory
  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/15/demo -mindepth 1 -delete
```

pg-primary ⇒ Restore the demo cluster and start PostgreSQL

```
$ sudo -u postgres pgbackrest --stanza=demo restore
$ sudo pg_ctlcluster 15 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.

7 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.

7.1 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" | 2024-03-20 21:11:40+00 | 000000010000000000000005
(1 row)
```

7.2 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'
1710969100
```

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.*

8 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.

8.1 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. 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/15/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-size` too high.

8.2 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/15/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
```

8.3 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=20240320-211154F info
  stanza: demo
      status: ok
      cipher: aes-256-cbc
```

```
      db (current)
          wal archive min/max (15): 000000020000000000000007/000000020000000000000009

          full backup: 20240320-211154F
              timestamp start/stop: 2024-03-20 21:11:54+00 / 2024-03-20 21:11:57+00
              wal start/stop: 000000020000000000000008 / 000000020000000000000009
              lsn start/stop: 0/8000028 / 0/9000050
              database size: 21.8MB, database backup size: 21.8MB
              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=20240320-211154F \
      --annotation=key= --annotation=new_key=new_value annotate
$ sudo -u postgres pgbackrest --stanza=demo --set=20240320-211154F info
  stanza: demo
      status: ok
      cipher: aes-256-cbc

      db (current)
          wal archive min/max (15): 000000020000000000000007/000000020000000000000009

          full backup: 20240320-211154F
              timestamp start/stop: 2024-03-20 21:11:54+00 / 2024-03-20 21:11:57+00
              wal start/stop: 000000020000000000000008 / 000000020000000000000009
              lsn start/stop: 0/8000028 / 0/9000050
              database size: 21.8MB, database backup size: 21.8MB
              repo1: backup size: 2.9MB
              database list: postgres (5)
              annotation(s)
                  new_key: new_value
                  source: demo backup
```

9 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 and Incremental backups are count-based but will always be expired when the backup they depend on is expired. 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.

9.1 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/15/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 973 lines of output]
  P00   INFO: repo1: remove expired backup 20240320-211151F
  P00 DETAIL: repo1: 15-1 archive retention on backup 20240320-211154F, start =
  000000020000000000000008
  P00   INFO: repo1: 15-1 remove archive, start = 000000020000000000000007, stop =
  000000020000000000000007
  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 20240320-211154F
  P00   INFO: repo1: remove expired backup 20240320-211154F
  P00   INFO: repo1: 15-1 remove archive, start = 000000020000000000000008, stop =
  00000002000000000000000A
  P00   INFO: expire command end: completed successfully
```

The `20240320-211134F` full backup is expired and archive retention is based on the `20240320-211159F` which is now the oldest full backup.

9.2 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/15/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. Incremental backups cannot be expired independently — they are always expired with their related full or differential backup.

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 2.51: --exec-id=972-f3c64275 --log-level-console=info --log-level-stderr=off --no-log-timestamp --repo1-cipher-pass= --repo1-cipher-type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo1-retention-diff=1 --repo1-retention-full=2 --stanza=demo
P00   INFO: repo1: expire diff backup set 20240320-211202F_20240320-211206D, 20240320-211202F_20240320-211209I
P00   INFO: repo1: remove expired backup 20240320-211202F_20240320-211209I
P00   INFO: repo1: remove expired backup 20240320-211202F_20240320-211206D
P00   INFO: expire command end: completed successfully
```

9.3 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/15/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 = 20240320-211202F_20240320-211215D
P00   INFO: diff backup size = 8.3KB, file total = 961
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 2.51: --exec-id=1054-067bb1b8 --log-level-
console=detail --log-level-stderr=off --no-log-timestamp --repo1-cipher-pass= --repo1-
cipher-type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo1-retention-archive=1 --
repo1-retention-archive-type=diff --repo1-retention-diff=2 --repo1-retention-full=2 --
stanza=demo
P00 DETAIL: repo1: 15-1 archive retention on backup 20240320-211159F, start =
00000002000000000000000B, stop = 00000002000000000000000B
P00 DETAIL: repo1: 15-1 archive retention on backup 20240320-211202F, start =
00000002000000000000000C, stop = 00000002000000000000000D
P00 DETAIL: repo1: 15-1 archive retention on backup 20240320-211202F_20240320-211212D,
start = 000000020000000000000012, stop = 000000020000000000000013
P00 DETAIL: repo1: 15-1 archive retention on backup 20240320-211202F_20240320-211215D,
start = 000000020000000000000016
P00   INFO: repo1: 15-1 remove archive, start = 00000002000000000000000E, stop =
000000020000000000000011
P00   INFO: repo1: 15-1 remove archive, start = 000000020000000000000014, stop =
000000020000000000000015
P00   INFO: expire command end: completed successfully
```

The `20240320-211202F_20240320-211212D` 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 `20240320-211202F_20240320-211212D` but before `20240320-211202F_20240320-211215D` are removed. WAL segments generated after the new backup `20240320-211202F_20240320-211215D` 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.

10 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.

10.1 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.

10.2 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 15 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/15/demo' exists
P00 DETAIL: remove 'global/pg_control' so cluster will not start if restore does not
complete
P00   INFO: remove invalid files/links/paths from '/var/lib/postgresql/15/demo'
P00 DETAIL: remove invalid file '/var/lib/postgresql/15/demo/backup_label.old'
P00 DETAIL: remove invalid file '/var/lib/postgresql/15/demo/base/17pg_internal.init'
          [filtered 14 lines of output]
P01 DETAIL: restore file /var/lib/postgresql/15/demo/backup_label (260B, 0.00%) checksum
3246d2affce4dc025b9349fdaff4cc39b69d6fa0
P01 DETAIL: restore file /var/lib/postgresql/15/demo/pg_multixact/members/0000 - exists
and matches backup (bundle 20240320-211202F/1/0, 8KB, 0.04%) checksum
0631457264ff7f8d5fb1edc2c0211992a67c73e6
P01 DETAIL: restore file /var/lib/postgresql/15/demo/PG_VERSION - exists and matches
backup (bundle 20240320-211202F/1/40, 3B, 0.04%) checksum
587b596f04f7db9c2cad3d6b87dd2b3a05de4f35
P01 DETAIL: restore file /var/lib/postgresql/15/demo/global/pg_filenode.map - exists and
matches backup (bundle 20240320-211202F/1/64, 512B, 0.04%) checksum
8426f71eec225fb3087aa80427d8e6b4e6a8a65b
P01 DETAIL: restore file /var/lib/postgresql/15/demo/global/6247 - exists and matches
backup (bundle 20240320-211202F/1/232, 8KB, 0.07%) checksum
ea40c8171261ed36b40f1597297f0a111790313c
          [filtered 985 lines of output]
```

pg-primary ⇒ Restart PostgreSQL

```
$ sudo pg_ctlcluster 15 demo start
```

10.3 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/15/demo/base/32768
  7.3M    /var/lib/postgresql/15/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=20240320-211202F_20240320-211224I info
        [filtered 12 lines of output]
            repo1: backup size: 2.0MB
            backup reference list: 20240320-211202F, 20240320-211202F_20240320-211215D
            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 15 demo stop
$ sudo -u postgres pgbackrest --stanza=demo --delta \
      --db-include=test2 --type=immediate --target-action=promote restore
$ sudo pg_ctlcluster 15 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:
FATAL:  relation mapping file "base/32768/pg_filenode.map" contains invalid data
```

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

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

pg-primary ⇒ Show space used by test1 database after recovery

```
$ sudo -u postgres du -sh /var/lib/postgresql/15/demo/base/32768
8.0K    /var/lib/postgresql/15/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)
```

11 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"
  2024-03-20 21:12:35.138287+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 list: 20240320-211202F, 20240320-211202F_20240320-211215D
         incr backup: 20240320-211202F_20240320-211237I
            timestamp start/stop: 2024-03-20 21:12:37+00 / 2024-03-20 21:12:38+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 pg_ctlcluster 15 demo stop
$ sudo -u postgres pgbackrest --stanza=demo --delta \
      --set=20240320-211202F_20240320-211237I --target-timeline=current \
      --type=time "--target=2024-03-20 21:12:35.138287+00" --target-action=promote
restore
$ sudo pg_ctlcluster 15 demo start
        [filtered 13 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 s
  FATAL:  recovery ended before configured recovery target was reached
  LOG:  startup process (PID 1460) 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 `2024-03-20 21:12:35.138287+00`

```
$ sudo -u postgres pgbackrest --stanza=demo --delta \
      --type=time "--target=2024-03-20 21:12:35.138287+00" \
      --target-action=promote restore
$ sudo -u postgres cat /var/lib/postgresql/15/demo/postgresql.auto.conf
        [filtered 9 lines of output]
  # Recovery settings generated by pgBackRest restore on 2024-03-20 21:12:40
  restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
  recovery_target_time = '2024-03-20 21:12:35.138287+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 15 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-15-demo.log
        [filtered 4 lines of output]
  LOG:  database system was interrupted; last known up at 2024-03-20 21:12:25 UTC
  LOG:  restored log file "00000004.history" from archive
  LOG:  starting point-in-time recovery to 2024-03-20 21:12:35.138287+00
  LOG:  starting backup recovery with redo LSN 0/19000028, checkpoint LSN 0/19000060, on
  timeline ID 3
  LOG:  restored log file "00000004.history" from archive
        [filtered 5 lines of output]
  LOG:  database system is ready to accept read-only connections
  LOG:  restored log file "000000040000000000000001A" from archive
  LOG:  recovery stopping before commit of transaction 734, time 2024-03-20
  21:12:36.418519+00
  LOG:  redo done at 0/19025398 system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed:
  0.06 s
  LOG:  last completed transaction was at log time 2024-03-20 21:12:33.849995+00
  LOG:  restored log file "000000040000000000000019" from archive
  LOG:  selected new timeline ID: 5
        [filtered 5 lines of output]
```

12 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 15 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 2.51: --exec-id=1591-b8d242a4 --log-level-console=info --
log-level-stderr=off --no-log-timestamp --stanza=demo
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 2.51: --exec-id=1599-9626d400 --log-level-
console=info --log-level-stderr=off --no-log-timestamp --pg1-
path=/var/lib/postgresql/15/demo --repo=1 --repo1-cipher-pass= --repo1-cipher-type=aes-
256-cbc --repo1-path=/var/lib/pgbackrest --stanza=demo
P00   INFO: stanza-delete command end: completed successfully
```

13 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 but backups will 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.

14 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/15/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
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 2.51: --exec-id=1671-c310cc0e --log-level-
console=info --log-level-stderr=off --no-log-timestamp --pg1-
path=/var/lib/postgresql/15/demo --repo2-azure-account= --repo2-azure-container=demo-
container --repo2-azure-key= --repo1-cipher-pass= --repo1-cipher-type=aes-256-cbc --
repo1-path=/var/lib/pgbackrest --repo2-path=/demo-repo --repo2-type=azure --stanza=demo
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 object stores is relatively slow so commands benefit by increasing `process-max` to parallelize file creation.

pg-primary ⇒ Backup the demo cluster

```
$ sudo -u postgres pgbackrest --stanza=demo --repo=2 \
      --log-level-console=info backup
P00   INFO: backup command begin 2.51: --exec-id=1680-43678308 --log-level-console=info
--log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/postgresql/15/demo --
process-max=4 --repo=2 --repo2-azure-account= --repo2-azure-container=demo-container --
repo2-azure-key= --repo1-block --repo1-bundle --repo1-cipher-pass= --repo1-cipher-
type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo2-path=/demo-repo --repo1-
retention-diff=2 --repo1-retention-full=2 --repo2-retention-full=4 --repo2-type=azure --
stanza=demo --start-fast
P00   WARN: no prior backup exists, incr backup has been changed to full
P00   INFO: execute non-exclusive backup start: backup begins after the requested
immediate checkpoint completes
P00   INFO: backup start archive = 000000050000000000000001B, lsn = 0/1B000028
       [filtered 3 lines of output]
P00   INFO: check archive for segment(s)
```

```
00000005000000000000001B:00000005000000000000001B
P00   INFO: new backup label = 20240320-211250F
P00   INFO: full backup size = 29.0MB, file total = 1263
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.51: --exec-id=1680-43678308 --log-level-console=info
--log-level-stderr=off --no-log-timestamp --repo=2 --repo2-azure-account= --repo2-azure-
container=demo-container --repo2-azure-key= --repo1-cipher-pass= --repo1-cipher-
type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo2-path=/demo-repo --repo1-
retention-diff=2 --repo1-retention-full=2 --repo2-retention-full=4 --repo2-type=azure --
stanza=demo
```

15 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/15/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
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: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 object stores is relatively slow so commands benefit by increasing `process-max` to parallelize file creation.

pg-primary ⇒ Backup the demo cluster

```
$ sudo -u postgres pgbackrest --stanza=demo --repo=3 \
       --log-level-console=info backup
P00   INFO: backup command begin 2.51: --exec-id=1745-57b1ad64 --log-level-console=info
--log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/postgresql/15/demo --
process-max=4 --repo=3 --repo2-azure-account= --repo2-azure-container=demo-container --
repo2-azure-key= --repo1-block --repo1-bundle --repo1-cipher-pass= --repo1-cipher-
type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo2-path=/demo-repo --repo3-
path=/demo-repo --repo1-retention-diff=2 --repo1-retention-full=2 --repo2-retention-
full=4 --repo3-retention-full=4 --repo3-s3-bucket=demo-bucket --repo3-s3-endpoint=s3.us-
east-1.amazonaws.com --repo3-s3-key= --repo3-s3-key-secret= --repo3-s3-region=us-east-1
--repo2-type=azure --repo3-type=s3 --stanza=demo --start-fast
P00   WARN: no prior backup exists, incr backup has been changed to full
P00   INFO: execute non-exclusive backup start: backup begins after the requested
immediate checkpoint completes
P00   INFO: backup start archive = 000000050000000000000001C, lsn = 0/1C000028
         [filtered 3 lines of output]
P00   INFO: check archive for segment(s)
000000050000000000000001C:000000050000000000000001D
P00   INFO: new backup label = 20240320-211258F
P00   INFO: full backup size = 29.0MB, file total = 1263
```

```
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.51: --exec-id=1745-57b1ad64 --log-level-console=info
--log-level-stderr=off --no-log-timestamp --repo=3 --repo2-azure-account= --repo2-azure-
container=demo-container --repo2-azure-key= --repo1-cipher-pass= --repo1-cipher-
type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo2-path=/demo-repo --repo3-
path=/demo-repo --repo1-retention-diff=2 --repo1-retention-full=2 --repo2-retention-
full=4 --repo3-retention-full=4 --repo3-s3-bucket=demo-bucket --repo3-s3-endpoint=s3.us-
east-1.amazonaws.com --repo3-s3-key= --repo3-s3-key-secret= --repo3-s3-region=us-east-1
--repo2-type=azure --repo3-type=s3 --stanza=demo
```

16 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/15/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 2.51: --exec-id=1827-ce4d36e9 --log-level-console=info
--log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/postgresql/15/demo --
process-max=4 --repo=4 --repo2-azure-account= --repo2-azure-container=demo-container --
repo2-azure-key= --repo1-block --repo1-bundle --repo4-bundle --repo1-cipher-pass= --
repo1-cipher-type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo2-path=/demo-repo -
-repo3-path=/demo-repo --repo4-path=/demo-repo --repo1-retention-diff=2 --repo1-
retention-full=2 --repo2-retention-full=4 --repo3-retention-full=4 --repo3-s3-
bucket=demo-bucket --repo3-s3-endpoint=s3.us-east-1.amazonaws.com --repo3-s3-key= --
repo3-s3-key-secret= --repo3-s3-region=us-east-1 --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 --repo2-type=azure --repo3-type=s3 --
repo4-type=sftp --stanza=demo --start-fast
P00   WARN: option 'repo4-retention-full' is not set for 'repo4-retention-full-
type=count', the repository may run out of space
            HINT: to retain full backups indefinitely (without warning), set option
'repo4-retention-full' to the maximum.
P00   WARN: no prior backup exists, incr backup has been changed to full
P00   INFO: execute non-exclusive backup start: backup begins after the requested
immediate checkpoint completes
P00   INFO: backup start archive = 000000050000000000000001E, lsn = 0/1E000028
        [filtered 3 lines of output]
P00   INFO: check archive for segment(s)
000000050000000000000001E:000000050000000000000001F
P00   INFO: new backup label = 20240320-211306F
P00   INFO: full backup size = 29.0MB, file total = 1263
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.51: --exec-id=1827-ce4d36e9 --log-level-console=info
--log-level-stderr=off --no-log-timestamp --repo=4 --repo2-azure-account= --repo2-azure-
container=demo-container --repo2-azure-key= --repo1-cipher-pass= --repo1-cipher-
type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo2-path=/demo-repo --repo3-
path=/demo-repo --repo4-path=/demo-repo --repo1-retention-diff=2 --repo1-retention-
full=2 --repo2-retention-full=4 --repo3-retention-full=4 --repo3-s3-bucket=demo-bucket -
-repo3-s3-endpoint=s3.us-east-1.amazonaws.com --repo3-s3-key= --repo3-s3-key-secret= --
repo3-s3-region=us-east-1 --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 --repo2-type=azure --repo3-type=s3 --
repo4-type=sftp --stanza=demo
P00   INFO: expire command end: completed successfully
```

17 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/15/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 object stores is relatively slow so commands benefit by increasing `process-max` to parallelize file creation.

18 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.

18.1 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.

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
```

18.2 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.*

18.3 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/15/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/15/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.

18.4 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
```

18.5 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.

18.6 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 15 demo stop
$ sudo -u postgres pgbackrest --stanza=demo --delta restore
$ sudo pg_ctlcluster 15 demo start
```

19 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/15/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 (15): 000000060000000000000025/000000060000000000000027

          full backup: 20240320-211336F
              timestamp start/stop: 2024-03-20 21:13:36+00 / 2024-03-20 21:13:40+00
              wal start/stop: 000000060000000000000025 / 000000060000000000000025
              database size: 29.0MB, database backup size: 29.0MB
              repo1: backup set size: 3.9MB, backup size: 3.9MB

          full backup: 20240320-211341F
              timestamp start/stop: 2024-03-20 21:13:41+00 / 2024-03-20 21:13:45+00
              wal start/stop: 000000060000000000000026 / 000000060000000000000027
              database size: 29.0MB, database backup size: 29.0MB
              repo1: backup set size: 3.9MB, backup size: 3.9MB
```

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.

20 Starting and Stopping

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

pg-primary ⇒ Stop the pgBackRest services

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

New pgBackRest processes 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-
  primary': stop file exists for all stanzas
  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 process that are currently running. 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 processes again with the start command.

pg-primary ⇒ Start the pgBackRest services

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

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

pg-primary ⇒ Stop pgBackRest services for the `demo` stanza

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

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

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-
  primary': stop file exists for stanza demo
  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 the pgBackRest processes for a single stanza.

pg-primary ⇒ Start the pgBackRest services for the demo stanza

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

21 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.

21.1 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
```

21.2 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
```

21.3 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/15/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 15 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/15/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 2024-03-20 21:11:42
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2024-03-20 21:12:18
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
```

```
# Recovery settings generated by pgBackRest restore on 2024-03-20 21:12:40
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
# Removed by pgBackRest restore on 2024-03-20 21:13:29 # recovery_target_time = '2024-
03-20 21:12:35.138287+00'
# Removed by pgBackRest restore on 2024-03-20 21:13:29 # recovery_target_action =
'promote'

# Recovery settings generated by pgBackRest restore on 2024-03-20 21:13:29
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2024-03-20 21:13:57
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
```

The `hot_standby` setting must be enabled before starting PostgreSQL to allow read-only connections on pg-standby. Otherwise, connection attempts will be refused. The rest of the configuration is in case the standby is promoted to a primary.

pg-standby:`/etc/postgresql/15/demo/postgresql.conf` ⇒ Configure PostgreSQL

```
archive_command = 'pgbackrest --stanza=demo archive-push %p'
archive_mode = on
hot_standby = on
max_wal_senders = 3
wal_level = replica
```

pg-standby ⇒ Start PostgreSQL

```
$ sudo pg_ctlcluster 15 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-15-demo.log
        [filtered 3 lines of output]
LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
LOG:  database system was interrupted; last known up at 2024-03-20 21:13:42 UTC
LOG:  entering standby mode
LOG:  starting backup recovery with redo LSN 0/26000028, checkpoint LSN 0/26000060, on
timeline ID 6
LOG:  restored log file "00000006.history" from archive
       [filtered 6 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
---------------+-------------------------------
 0/28019A10    | 2024-03-20 21:14:04.941779+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 | 2024-03-20 21:14:05.921459+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 2.51: --exec-id=510-07ffc384 --log-level-console=info --
log-level-file=detail --log-level-stderr=off --no-log-timestamp --pg1-
path=/var/lib/postgresql/15/demo --repo1-host=repository --stanza=demo
P00   INFO: check repo1 (standby)
P00   INFO: switch wal not performed because this is a standby
P00   INFO: check command end: completed successfully
```

21.4 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/15/demo/pg_hba.conf'
$ sudo pg_ctlcluster 15 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/15/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 15 demo stop
$ sudo -u postgres pgbackrest --stanza=demo --delta --type=standby restore
$ sudo -u postgres cat /var/lib/postgresql/15/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 2024-03-20 21:11:42
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2024-03-20 21:12:18
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2024-03-20 21:12:40
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
# Removed by pgBackRest restore on 2024-03-20 21:13:29 # recovery_target_time = '2024-
03-20 21:12:35.138287+00'
# Removed by pgBackRest restore on 2024-03-20 21:13:29 # recovery_target_action =
'promote'

# Recovery settings generated by pgBackRest restore on 2024-03-20 21:13:29
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2024-03-20 21:14:09
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 15 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-15-demo.log
```

```
        [filtered 13 lines of output]
LOG:  consistent recovery state reached at 0/27000088
LOG:  database system is ready to accept read-only connections
LOG:  started streaming WAL from primary at 0/29000000 on timeline 6
```

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 | 2024-03-20 21:14:15.427743+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 | 2024-03-20 21:14:15.612202+00
  (1 row)
```

22 Multiple Stanzas

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

22.1 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
```

22.2 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
```

22.3 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/15/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/15/demo

[demo-alt]
pg1-host=pg-alt
pg1-path=/var/lib/postgresql/15/demo

[global]
process-max=3
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y
```

22.4 Setup Demo Cluster

pg-alt ⇒ Create the demo cluster

```
$ sudo -u postgres /usr/lib/postgresql/15/bin/initdb \
       -D /var/lib/postgresql/15/demo -k -A peer
$ sudo pg_createcluster 15 demo
Configuring already existing cluster (configuration: /etc/postgresql/15/demo, data:
/var/lib/postgresql/15/demo, owner: 102:103)
Ver Cluster Port Status Owner    Data directory              Log file
15  demo    5432 down   postgres /var/lib/postgresql/15/demo
/var/log/postgresql/postgresql-15-demo.log
```

pg-alt:`/etc/postgresql/15/demo/postgresql.conf` ⇒ Configure PostgreSQL settings

```
archive_command = 'pgbackrest --stanza=demo-alt archive-push %p'
archive_mode = on
max_wal_senders = 3
wal_level = replica
```

pg-alt ⇒ Start the demo cluster

```
$ sudo pg_ctlcluster 15 demo restart
```

22.5 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 2.51: --exec-id=392-ec781da6 --log-level-
console=info --log-level-file=detail --log-level-stderr=off --no-log-timestamp --pg1-
path=/var/lib/postgresql/15/demo --repo1-host=repository --stanza=demo-alt
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 2.51: --exec-id=402-bef8f92a --log-level-console=info --
log-level-file=detail --log-level-stderr=off --no-log-timestamp --repo1-host=repository
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/pgbackrest/archive/demo-alt/15-1/0000000100000000/000000010000000000000001-
618c3be51569dcd807fed44ff895551c6e0decf5.gz' on repo1
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 2.51: --exec-id=1249-38c39446 --log-level-console=info -
-log-level-stderr=off --no-log-timestamp --repo1-path=/var/lib/pgbackrest
P00   INFO: check stanza 'demo'
P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)
P00   INFO: WAL segment 000000060000000000000029 successfully archived to
'/var/lib/pgbackrest/archive/demo/15-1/0000000600000000/000000060000000000000029-
eda5c9bea9a8e898894d8e81e635cff5ce7d8fb8.gz' on repo1
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/pgbackrest/archive/demo-alt/15-1/0000000100000000/000000010000000000000002-
6c72b74d5fbdeee8f47ae40f976d163955575f3a.gz' on repo1
P00   INFO: check command end: completed successfully
```

23 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/15/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/15/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 15 demo restart
```

23.1 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 2.51: --exec-id=2436-2d3122fa --log-level-console=info --log-level-file=detail --log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/postgresql/15/demo --repo1-host=repository --stanza=demo
  P00   INFO: check repo1 configuration (primary)
  P00   INFO: check repo1 archive for WAL (primary)
  P00   INFO: WAL segment 0000000600000000000002F successfully archived to '/var/lib/pgbackrest/archive/demo/15-1/0000000600000000/0000000600000000000002F-083e1c564e6e9695ecd12297fcb0b7187e955f1d.gz' on repo1
  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 2.51: [/var/lib/postgresql/15/demo/pg_wal] --archive-async --exec-id=2422-8ad4f0e5 --log-level-console=off --log-level-file=detail --log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/postgresql/15/demo --process-max=2 --repo1-host=repository --spool-path=/var/spool/pgbackrest --stanza=demo
  P00   INFO: push 1 WAL file(s) to archive: 0000000600000000000002A
  P01 DETAIL: pushed WAL file '0000000600000000000002A' to the archive
  P00   INFO: archive-push:async command end: completed successfully

  -------------------PROCESS START-------------------
  P00   INFO: archive-push:async command begin 2.51: [/var/lib/postgresql/15/demo/pg_wal] --archive-async --exec-id=2439-6934dd67 --log-level-console=off --log-level-file=detail --log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/postgresql/15/demo --process-max=2 --repo1-host=repository --spool-path=/var/spool/pgbackrest --stanza=demo
  P00   INFO: push 4 WAL file(s) to archive: 0000000600000000000002B...0000000600000000000002E
  P02 DETAIL: pushed WAL file '0000000600000000000002C' to the archive
  P01 DETAIL: pushed WAL file '0000000600000000000002B' to the archive
  P02 DETAIL: pushed WAL file '0000000600000000000002D' to the archive
  P01 DETAIL: pushed WAL file '0000000600000000000002E' to the archive
  P00   INFO: archive-push:async command end: completed successfully

  -------------------PROCESS START-------------------
  P00   INFO: archive-push:async command begin 2.51: [/var/lib/postgresql/15/demo/pg_wal] --archive-async --exec-id=2455-9043b356 --log-level-console=off --log-level-file=detail --log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/postgresql/15/demo --process-max=2 --repo1-host=repository --spool-path=/var/spool/pgbackrest --stanza=demo
  P00   INFO: push 1 WAL file(s) to archive: 0000000600000000000002F
  P01 DETAIL: pushed WAL file '0000000600000000000002F' to the archive
  P00   INFO: archive-push:async command end: completed successfully
```

23.2 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 2.51: [000000060000000000000026,
000000060000000000000027, 000000060000000000000028, 000000060000000000000029,
00000006000000000000002A, 00000006000000000000002B, 00000006000000000000002C,
00000006000000000000002D] --archive-async --exec-id=730-3eb6c8a2 --log-level-console=off
--log-level-file=detail --log-level-stderr=off --no-log-timestamp --pg1-
path=/var/lib/postgresql/15/demo --process-max=2 --repo1-host=repository --spool-
path=/var/spool/pgbackrest --stanza=demo
P00   INFO: get 8 WAL file(s) from archive:
000000060000000000000026...00000006000000000000002D
P01 DETAIL: found 000000060000000000000026 in the repo1: 15-1 archive
P02 DETAIL: found 000000060000000000000027 in the repo1: 15-1 archive
P01 DETAIL: found 000000060000000000000028 in the repo1: 15-1 archive
P02 DETAIL: found 000000060000000000000029 in the repo1: 15-1 archive
P00 DETAIL: unable to find 00000006000000000000002A in the archive
P00   INFO: archive-get:async command end: completed successfully
      [filtered 14 lines of output]
P00   INFO: archive-get:async command begin 2.51: [00000006000000000000002A,
00000006000000000000002B, 00000006000000000000002C, 00000006000000000000002D,
00000006000000000000002E, 00000006000000000000002F, 000000060000000000000030,
000000060000000000000031] --archive-async --exec-id=773-5c7e1c42 --log-level-console=off
--log-level-file=detail --log-level-stderr=off --no-log-timestamp --pg1-
path=/var/lib/postgresql/15/demo --process-max=2 --repo1-host=repository --spool-
path=/var/spool/pgbackrest --stanza=demo
P00   INFO: get 8 WAL file(s) from archive:
00000006000000000000002A...000000060000000000000031
P01 DETAIL: found 00000006000000000000002A in the repo1: 15-1 archive
P02 DETAIL: found 00000006000000000000002B in the repo1: 15-1 archive
P01 DETAIL: found 00000006000000000000002C in the repo1: 15-1 archive
P02 DETAIL: found 00000006000000000000002D in the repo1: 15-1 archive
P02 DETAIL: found 00000006000000000000002F in the repo1: 15-1 archive
P01 DETAIL: found 00000006000000000000002E in the repo1: 15-1 archive
P00 DETAIL: unable to find 000000060000000000000030 in the archive
P00   INFO: archive-get:async command end: completed successfully
      [filtered 17 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
```

24 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/15/demo
pg2-host=pg-standby
pg2-path=/var/lib/postgresql/15/demo
```

```
[demo-alt]
pg1-host=pg-alt
pg1-path=/var/lib/postgresql/15/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 non-exclusive backup start: backup begins after the requested
immediate checkpoint completes
P00   INFO: backup start archive = 000000060000000000000031, lsn = 0/31000028
P00   INFO: wait for replay on the standby to reach 0/31000028
P00   INFO: replay on the standby reached 0/31000028
P00   INFO: check archive for prior segment 000000060000000000000030
P01 DETAIL: backup file pg-primary:/var/lib/postgresql/15/demo/global/pg_control (8KB,
0.53%) checksum e76b973464a159923348ea5fbfd57623caa58651
P02 DETAIL: backup file pg-standby:/var/lib/postgresql/15/demo/base/5/1249 (456KB,
31.18%) checksum 7e45991b888245980cad32947d863f4a79f9e354
P03 DETAIL: backup file pg-standby:/var/lib/postgresql/15/demo/base/5/2658 (120KB,
39.24%) checksum 28ba128001c5c18538aa9c4839031da7346ffbdf
       [filtered 1276 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.

25 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 15 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 15 demo stop
```

Create the new cluster and perform upgrade.

pg-primary ⇒ Create new cluster and perform the upgrade

```
$ sudo -u postgres /usr/lib/postgresql/16/bin/initdb \
      -D /var/lib/postgresql/16/demo -k -A peer
$ sudo pg_createcluster 16 demo
$ sudo -u postgres sh -c 'cd /var/lib/postgresql && \
      /usr/lib/postgresql/16/bin/pg_upgrade \
      --old-bindir=/usr/lib/postgresql/15/bin \
      --new-bindir=/usr/lib/postgresql/16/bin \
      --old-datadir=/var/lib/postgresql/15/demo \
      --new-datadir=/var/lib/postgresql/16/demo \
      --old-options=" -c config_file=/etc/postgresql/15/demo/postgresql.conf" \
      --new-options=" -c config_file=/etc/postgresql/16/demo/postgresql.conf"'
        [filtered 42 lines of output]
Checking for extension updates                             ok

Upgrade Complete

----------------
Optimizer statistics are not transferred by pg_upgrade.
       [filtered 3 lines of output]
```

Configure the new cluster settings and port.

pg-primary:`/etc/postgresql/16/demo/postgresql.conf` ⇒ Configure PostgreSQL

```
archive_command = 'pgbackrest --stanza=demo archive-push %p'
archive_mode = on
max_wal_senders = 3
wal_level = replica
```

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/16/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/16/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/16/demo
pg2-host=pg-standby
pg2-path=/var/lib/postgresql/16/demo

[demo-alt]
pg1-host=pg-alt
pg1-path=/var/lib/postgresql/15/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/15/demo/pg_hba.conf \
      /etc/postgresql/16/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 2.51: --exec-id=2864-ed4fe8b0 --log-level-
console=info --log-level-file=detail --log-level-stderr=off --no-log-timestamp --no-
online --pg1-path=/var/lib/postgresql/16/demo --repo1-host=repository --stanza=demo
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 16 demo start
```

Test configuration using the check command.

pg-primary ⇒ Check configuration

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

Remove the old cluster.

pg-primary ⇒ Remove old cluster

```
$ sudo pg_dropcluster 15 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 15 demo
$ sudo pg_createcluster 16 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-standby': unable to connect to 'dbname='postgres' port=5432': connection to server
on socket "/var/run/postgresql/.s.PGSQL.5432" failed: No such file or directory
                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: `/etc/postgresql/16/demo/postgresql.conf` ⇒ Configure PostgreSQL

```
hot_standby = on
```

pg-standby ⇒ Start PostgreSQL and check the pgBackRest configuration

```
$ sudo pg_ctlcluster 16 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/16/demo
pg2-host=pg-standby
pg2-path=/var/lib/postgresql/16/demo

[demo-alt]
pg1-host=pg-alt
pg1-path=/var/lib/postgresql/15/demo

[global]
backup-standby=y
process-max=3
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2
start-fast=y
```

Copyright © 2015-2024, The PostgreSQL Global Development Group, MIT License. Updated March 24, 2024

pgBackRest

User Guides

Available User Guides

- Debian & Ubuntu
- RHEL

Copyright © 2015-2024, The PostgreSQL Global Development Group, MIT License. Updated March 24, 2024

pgBackRest User Guide

RHEL

Table of Contents

1 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 7-8 and PostgreSQL 12, 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.

2 Concepts

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

2.1 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.

2.2 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.

2.3 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).

2.4 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.

3 Upgrading pgBackRest

3.1 Upgrading pgBackRest from v1 to v2

Upgrading from v1 to v2 is fairly straight-forward. The repository format has not changed and all non-deprecated options from v1 are accepted, so for most installations it is simply a matter of installing the new version.

However, there are a few caveats:

- The deprecated `thread-max` option is no longer valid. Use `process-max` instead.
- The deprecated `archive-max-mb` option is no longer valid. This has been replaced with the `archive-push-queue-max` option which has different semantics.
- The default for the `backup-user` option has changed from `backrest` to `pgbackrest`.
- In v2.02 the default location of the pgBackRest configuration file has changed from `/etc/pgbackrest.conf` to `/etc/pgbackrest/pgbackrest.conf`. If `/etc/pgbackrest/pgbackrest.conf` does not exist, the `/etc/pgbackrest.conf` file will be loaded instead, if it exists.

Many option names have changed to improve consistency although the old names from v1 are still accepted. In general, `db-*` options have been renamed to `pg-*` and `backup-*`/`retention-*` options have been renamed to `repo-*` when appropriate.

PostgreSQL and repository options must be indexed when using the new names introduced in v2, e.g. `pg1-host`, `pg1-path`, `repo1-path`, `repo1-type`, etc.

3.2 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.

4 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.

The preferred build method is meson/ninja, see Build. The autoconf/make method is shown here for legacy purposes.

build ⇒ Download version 2.51 of pgBackRest to /build path

```
$ mkdir -p /build
$ wget -q -O - \
       https://github.com/pgbackrest/pgbackrest/archive/release/2.51.tar.gz | \
       tar zx -C /build
```

build ⇒ Install build dependencies

```
$ sudo yum install make gcc postgresql12-devel openssl-devel \
       libxml2-devel lz4-devel libzstd-devel bzip2-devel libyaml-devel libssh2-devel
```

build ⇒ Configure and compile pgBackRest

```
$ cd /build/pgbackrest-release-2.51/src && ./configure && make
```

5 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 7-8 packages for pgBackRest are available from Crunchy Data or 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-release-2.51/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 2.51 - 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 the repository.
      version          Get version.

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

6 Quick Start

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

6.1 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-12/bin/initdb \
       -D /var/lib/pgsql/12/data -k -A peer
```

By default RHEL 7-8 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/12/data/postgresql.conf` ⇒ Set `log_filename`

```
  log_filename = 'postgresql.log'
```

6.2 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` in `postgresql.conf`.

By default RHEL 7-8 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/12/data
```

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

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.

The files are concatenated as if they were one big file; 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 2.51 - '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
```

6.3 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/12/data

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

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

6.4 Configure Archiving

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

pg-primary:`/var/lib/pgsql/12/data/postgresql.conf` ⇒ Configure archive settings

```
  archive_command = 'pgbackrest --stanza=demo archive-push %p'
  archive_mode = on
  log_filename = 'postgresql.log'
  max_wal_senders = 3
  wal_level = replica
```

`%p` is how PostgreSQL specifies the location of the WAL segment to be archived. Setting `wal_level` to at least `replica` and increasing `max_wal_senders` is a good idea even if there are currently no replicas as this will allow

them to be added later without restarting the primary cluster.

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

pg-primary ⇒ Restart the demo cluster

```
$ sudo systemctl restart postgresql-12.service
```

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/12/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`.

6.5 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/12/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.

6.6 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/12/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
```

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

6.7 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 2.51: --exec-id=1037-e7e2a455 --log-level-
console=info --log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/pgsql/12/data
--repo1-cipher-pass= --repo1-cipher-type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --
stanza=demo
P00   INFO: stanza-create for stanza 'demo' on repo1
P00   INFO: stanza-create command end: completed successfully
```

6.8 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 2.51: --exec-id=1065-fbf787f1 --log-level-console=info -
-log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/pgsql/12/data --repo1-
cipher-pass= --repo1-cipher-type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --
stanza=demo
P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)
P00   INFO: WAL segment 000000010000000000000001 successfully archived to
'/var/lib/pgbackrest/archive/demo/12-1/0000000100000000/000000010000000000000001-
e08b19d2ec1f234a6b7bab79bb849ba33283d293.gz' on repo1
P00   INFO: check command end: completed successfully
```

6.9 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/12/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 2.51: --exec-id=1138-e9d4ea7e --log-level-console=info
--log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/pgsql/12/data --repo1-
cipher-pass= --repo1-cipher-type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo1-
retention-full=2 --stanza=demo --start-fast
P00   WARN: no prior backup exists, incr backup has been changed to full
P00   INFO: execute non-exclusive backup start: backup begins after the requested
immediate checkpoint completes
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 = 20240320-210638F
P00   INFO: full backup size = 23.5MB, file total = 981
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.51: --exec-id=1138-e9d4ea7e --log-level-console=info
--log-level-stderr=off --no-log-timestamp --repo1-cipher-pass= --repo1-cipher-type=aes-
256-cbc --repo1-path=/var/lib/pgbackrest --repo1-retention-full=2 --stanza=demo
```

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 = 20240320-210638F_20240320-210642D
P00   INFO: diff backup size = 8.8KB, file total = 981
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.51: --exec-id=1199-0465df2b --log-level-console=info
--log-level-stderr=off --no-log-timestamp --repo1-cipher-pass= --repo1-cipher-type=aes-
256-cbc --repo1-path=/var/lib/pgbackrest --repo1-retention-full=2 --stanza=demo
```

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.

6.10 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.

6.11 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 (12): 000000010000000000000001/000000010000000000000005
          full backup: 20240320-210638F
              timestamp start/stop: 2024-03-20 21:06:38+00 / 2024-03-20 21:06:41+00
              wal start/stop: 000000010000000000000002 / 000000010000000000000003
              database size: 23.5MB, database backup size: 23.5MB
              repo1: backup set size: 2.8MB, backup size: 2.8MB
          diff backup: 20240320-210638F_20240320-210642D
              timestamp start/stop: 2024-03-20 21:06:42+00 / 2024-03-20 21:06:44+00
              wal start/stop: 000000010000000000000004 / 000000010000000000000005
              database size: 23.5MB, database backup size: 8.8KB
              repo1: backup set size: 2.8MB, backup size: 688B
              backup reference list: 20240320-210638F
```

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.

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`' message will appear beside the '`status`' information if one of those commands is currently running on the host.

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

6.12 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-12.service
$ sudo -u postgres rm /var/lib/pgsql/12/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-12.service
$ sudo systemctl status postgresql-12.service
        [filtered 12 lines of output]
Mar 20 21:06:45 pg-primary systemd[1]: postgresql-12.service: Main process exited,
code=exited, status=2/INVALIDARGUMENT
Mar 20 21:06:45 pg-primary systemd[1]: postgresql-12.service: Failed with result 'exit-
code'.
Mar 20 21:06:45 pg-primary systemd[1]: Failed to start PostgreSQL 12 database server.
```

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/12/data -mindepth 1 -delete
```

pg-primary ⇒ Restore the demo cluster and start PostgreSQL

```
$ sudo -u postgres pgbackrest --stanza=demo restore
$ sudo systemctl start postgresql-12.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.

7 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.

7.1 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" | 2024-03-20 21:06:44+00 | 000000010000000000000005
  (1 row)
```

8 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.

8.1 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. 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/12/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-size` too high.

8.2 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/12/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
```

8.3 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=20240320-210654F info
stanza: demo
    status: ok
    cipher: aes-256-cbc

    db (current)
        wal archive min/max (12): 000000020000000000000007/000000020000000000000009

        full backup: 20240320-210654F
            timestamp start/stop: 2024-03-20 21:06:54+00 / 2024-03-20 21:06:57+00
            wal start/stop: 000000020000000000000008 / 000000020000000000000009
            lsn start/stop: 0/8000028 / 0/9000050
            database size: 23.5MB, database backup size: 23.5MB
            repo1: backup size: 2.8MB
            database list: postgres (13396)
            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=20240320-210654F \
      --annotation=key= --annotation=new_key=new_value annotate
$ sudo -u postgres pgbackrest --stanza=demo --set=20240320-210654F info
stanza: demo
    status: ok
    cipher: aes-256-cbc

    db (current)
        wal archive min/max (12): 000000020000000000000007/000000020000000000000009

        full backup: 20240320-210654F
            timestamp start/stop: 2024-03-20 21:06:54+00 / 2024-03-20 21:06:57+00
            wal start/stop: 000000020000000000000008 / 000000020000000000000009
            lsn start/stop: 0/8000028 / 0/9000050
            database size: 23.5MB, database backup size: 23.5MB
            repo1: backup size: 2.8MB
            database list: postgres (13396)
            annotation(s)
```

```
                    new_key: new_value
                    source: demo backup
```

9 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 and Incremental backups are count-based but will always be expired when the backup they depend on is expired. 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.

9.1 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/12/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 993 lines of output]
P00   INFO: repo1: remove expired backup 20240320-210651F
```

```
P00 DETAIL: repo1: 12-1 archive retention on backup 20240320-210654F, start =
000000020000000000000008
P00   INFO: repo1: 12-1 remove archive, start = 000000020000000000000007, stop =
000000020000000000000007
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 20240320-210654F
P00   INFO: repo1: remove expired backup 20240320-210654F
P00   INFO: repo1: 12-1 remove archive, start = 000000020000000000000008, stop =
000000020000000000000009
P00   INFO: expire command end: completed successfully
```

The `20240320-210638F` full backup is expired and archive retention is based on the `20240320-210659F` which is now the oldest full backup.

9.2 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/12/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. Incremental backups cannot be expired independently — they are always expired with their related full or differential backup.

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 2.51: --exec-id=2245-e82141fe --log-level-console=info
--log-level-stderr=off --no-log-timestamp --repo1-cipher-pass= --repo1-cipher-type=aes-
256-cbc --repo1-path=/var/lib/pgbackrest --repo1-retention-diff=1 --repo1-retention-
full=2 --stanza=demo
```

```
P00   INFO: repo1: expire diff backup set 20240320-210702F_20240320-210705D, 20240320-
210702F_20240320-210708I
```

```
P00   INFO: repo1: remove expired backup 20240320-210702F_20240320-210708I
P00   INFO: repo1: remove expired backup 20240320-210702F_20240320-210705D
P00   INFO: expire command end: completed successfully
```

9.3 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/12/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 = 20240320-210702F_20240320-210714D
```

```
P00   INFO: diff backup size = 10.8KB, file total = 981
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 2.51: --exec-id=2441-13e385ba --log-level-
console=detail --log-level-stderr=off --no-log-timestamp --repo1-cipher-pass= --repo1-
cipher-type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo1-retention-archive=1 --
repo1-retention-archive-type=diff --repo1-retention-diff=2 --repo1-retention-full=2 --
stanza=demo
P00 DETAIL: repo1: 12-1 archive retention on backup 20240320-210659F, start =
00000002000000000000000A, stop = 00000002000000000000000B
P00 DETAIL: repo1: 12-1 archive retention on backup 20240320-210702F, start =
00000002000000000000000D, stop = 00000002000000000000000D
```

```
P00 DETAIL: repo1: 12-1 archive retention on backup 20240320-210702F_20240320-210711D,
start = 000000020000000000000012, stop = 000000020000000000000013
```

```
P00 DETAIL: repo1: 12-1 archive retention on backup 20240320-210702F_20240320-210714D,
start = 000000020000000000000016
```

```
P00   INFO: repo1: 12-1 remove archive, start = 00000002000000000000000C, stop =
00000002000000000000000C
P00   INFO: repo1: 12-1 remove archive, start = 00000002000000000000000E, stop =
000000020000000000000011
P00   INFO: repo1: 12-1 remove archive, start = 000000020000000000000014, stop =
000000020000000000000015
```

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

The `20240320-210702F_20240320-210711D` 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 `20240320-210702F_20240320-210711D` but before `20240320-210702F_20240320-210714D` are removed. WAL segments generated after the new backup `20240320-210702F_20240320-210714D` 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.

10 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.

10.1 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.

10.2 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-12.service
$ sudo -u postgres pgbackrest --stanza=demo --delta \
      --log-level-console=detail restore
          [filtered 2 lines of output]
  P00 DETAIL: check '/var/lib/pgsql/12/data' exists
  P00 DETAIL: remove 'global/pg_control' so cluster will not start if restore does not
  complete
  P00   INFO: remove invalid files/links/paths from '/var/lib/pgsql/12/data'
  P00 DETAIL: remove invalid file '/var/lib/pgsql/12/data/backup_label.old'
  P00 DETAIL: remove invalid file '/var/lib/pgsql/12/data/base/13396/pg_internal.init'
          [filtered 1027 lines of output]
```

pg-primary ⇒ Restart PostgreSQL

```
$ sudo systemctl start postgresql-12.service
```

10.3 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
  INSERT 0 1
$ sudo -u postgres psql -c "create table test2_table (id int); \
      insert into test2_table (id) values (2);" test2
  INSERT 0 1
```

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/12/data/base/32768
  7.8M    /var/lib/pgsql/12/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=20240320-210702F_20240320-210720I info
        [filtered 12 lines of output]
            repo1: backup size: 1.9MB
            backup reference list: 20240320-210702F, 20240320-210702F_20240320-210714D
            database list: postgres (13396), 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-12.service
$ sudo -u postgres pgbackrest --stanza=demo --delta \
      --db-include=test2 --type=immediate --target-action=promote restore
$ sudo systemctl start postgresql-12.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: FATAL:  relation mapping file "base/32768/pg_filenode.map" contains invalid
data
```

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

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

pg-primary ⇒ Show space used by test1 database after recovery

```
$ sudo -u postgres du -sh /var/lib/pgsql/12/data/base/32768
16K     /var/lib/pgsql/12/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
 13395 | template0
 13396 | postgres
 32769 | test2
(4 rows)
```

11 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;"
     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"
2024-03-20 21:07:29.317767+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;"
ERROR:  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 list: 20240320-210702F, 20240320-210702F_20240320-210714D
         incr backup: 20240320-210702F_20240320-210731I
            timestamp start/stop: 2024-03-20 21:07:31+00 / 2024-03-20 21:07:33+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 systemctl stop postgresql-12.service
$ sudo -u postgres pgbackrest --stanza=demo --delta \
      --set=20240320-210702F_20240320-210731I --target-timeline=current \
      --type=time "--target=2024-03-20 21:07:29.317767+00" --target-action=promote
restore
$ sudo systemctl start postgresql-12.service
$ sudo -u postgres psql -c "select * from important_table"
  ERROR:  relation "important_table" does not exist
  LINE 1: select * from important_table
                        ^
```

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

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

```
$ sudo -u postgres cat /var/lib/pgsql/12/data/log/postgresql.log
  LOG:  database system was interrupted; last known up at 2024-03-20 21:07:31 UTC
  LOG:  starting point-in-time recovery to 2024-03-20 21:07:29.317767+00
  LOG:  restored log file "00000004.history" from archive
  LOG:  restored log file "00000004000000000000001A" from archive
  LOG:  redo starts at 0/1A000028
  LOG:  consistent recovery state reached at 0/1A000100
  LOG:  database system is ready to accept read only connections
  LOG:  redo done at 0/1A000100
        [filtered 6 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 `2024-03-20 21:07:29.317767+00`

```
$ sudo systemctl stop postgresql-12.service
$ sudo -u postgres pgbackrest --stanza=demo --delta \
      --type=time "--target=2024-03-20 21:07:29.317767+00" \
      --target-action=promote restore
$ sudo -u postgres cat /var/lib/pgsql/12/data/postgresql.auto.conf
        [filtered 9 lines of output]
  # Recovery settings generated by pgBackRest restore on 2024-03-20 21:07:38
  restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
  recovery_target_time = '2024-03-20 21:07:29.317767+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-12.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/12/data/log/postgresql.log
  LOG:  database system was interrupted; last known up at 2024-03-20 21:07:20 UTC
  LOG:  restored log file "00000004.history" from archive
  LOG:  restored log file "00000005.history" from archive
  LOG:  starting point-in-time recovery to 2024-03-20 21:07:29.317767+00
  LOG:  restored log file "00000005.history" from archive
  LOG:  restored log file "000000040000000000000019" from archive
        [filtered 3 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 496, time 2024-03-20
  21:07:30.613133+00
  LOG:  redo done at 0/1901D128
  LOG:  last completed transaction was at log time 2024-03-20 21:07:27.981685+00
  LOG:  selected new timeline ID: 6
  LOG:  archive recovery complete
  LOG:  database system is ready to accept connections
```

12 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-12.service
```

pg-primary ⇒ Stop pgBackRest for the stanza

```
$ sudo -u postgres pgbackrest --stanza=demo --log-level-console=info stop
  P00   INFO: stop command begin 2.51: --exec-id=3842-0f76ae18 --log-level-console=info --
  log-level-stderr=off --no-log-timestamp --stanza=demo
  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 2.51: --exec-id=3869-ea3f46be --log-level-
console=info --log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/pgsql/12/data
--repo=1 --repo1-cipher-pass= --repo1-cipher-type=aes-256-cbc --repo1-
path=/var/lib/pgbackrest --stanza=demo
P00   INFO: stanza-delete command end: completed successfully
```

13 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 but backups will 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.

14 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/12/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
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 2.51: --exec-id=4032-eec23e60 --log-level-
console=info --log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/pgsql/12/data
--repo2-azure-account= --repo2-azure-container=demo-container --repo2-azure-key= --
repo1-cipher-pass= --repo1-cipher-type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --
repo2-path=/demo-repo --repo2-type=azure --stanza=demo
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 object stores is relatively slow so commands benefit by increasing `process-max` to parallelize file creation.

pg-primary ⇒ Backup the demo cluster

```
$ sudo -u postgres pgbackrest --stanza=demo --repo=2 \
      --log-level-console=info backup
P00   INFO: backup command begin 2.51: --exec-id=4060-f8c0709e --log-level-console=info
--log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/pgsql/12/data --process-
max=4 --repo=2 --repo2-azure-account= --repo2-azure-container=demo-container --repo2-
azure-key= --repo1-block --repo1-bundle --repo1-cipher-pass= --repo1-cipher-type=aes-
256-cbc --repo1-path=/var/lib/pgbackrest --repo2-path=/demo-repo --repo1-retention-
diff=2 --repo1-retention-full=2 --repo2-retention-full=4 --repo2-type=azure --
stanza=demo --start-fast
P00   WARN: no prior backup exists, incr backup has been changed to full
P00   INFO: execute non-exclusive backup start: backup begins after the requested
immediate checkpoint completes
P00   INFO: backup start archive = 00000006000000000000001B, lsn = 0/1B000028
       [filtered 3 lines of output]
P00   INFO: check archive for segment(s)
00000006000000000000001B:00000006000000000000001B
P00   INFO: new backup label = 20240320-210744F
P00   INFO: full backup size = 31.1MB, file total = 1287
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.51: --exec-id=4060-f8c0709e --log-level-console=info
--log-level-stderr=off --no-log-timestamp --repo=2 --repo2-azure-account= --repo2-azure-
container=demo-container --repo2-azure-key= --repo1-cipher-pass= --repo1-cipher-
type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo2-path=/demo-repo --repo1-
retention-diff=2 --repo1-retention-full=2 --repo2-retention-full=4 --repo2-type=azure --
stanza=demo
```

15 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/12/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
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: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 object stores is relatively slow so commands benefit by increasing `process-max` to parallelize file creation.

pg-primary ⇒ Backup the demo cluster

```
$ sudo -u postgres pgbackrest --stanza=demo --repo=3 \
      --log-level-console=info backup
P00   INFO: backup command begin 2.51: --exec-id=4220-f20e1322 --log-level-console=info
--log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/pgsql/12/data --process-
max=4 --repo=3 --repo2-azure-account= --repo2-azure-container=demo-container --repo2-
azure-key= --repo1-block --repo1-bundle --repo1-cipher-pass= --repo1-cipher-type=aes-
256-cbc --repo1-path=/var/lib/pgbackrest --repo2-path=/demo-repo --repo3-path=/demo-repo
--repo1-retention-diff=2 --repo1-retention-full=2 --repo2-retention-full=4 --repo3-
retention-full=4 --repo3-s3-bucket=demo-bucket --repo3-s3-endpoint=s3.us-east-
1.amazonaws.com --repo3-s3-key= --repo3-s3-key-secret= --repo3-s3-region=us-east-1 --
repo2-type=azure --repo3-type=s3 --stanza=demo --start-fast
P00   WARN: no prior backup exists, incr backup has been changed to full
P00   INFO: execute non-exclusive backup start: backup begins after the requested
immediate checkpoint completes
P00   INFO: backup start archive = 00000006000000000000001C, lsn = 0/1C000028
       [filtered 3 lines of output]
P00   INFO: check archive for segment(s)
00000006000000000000001C:00000006000000000000001D
P00   INFO: new backup label = 20240320-210752F
P00   INFO: full backup size = 31.1MB, file total = 1287
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.51: --exec-id=4220-f20e1322 --log-level-console=info
--log-level-stderr=off --no-log-timestamp --repo=3 --repo2-azure-account= --repo2-azure-
container=demo-container --repo2-azure-key= --repo1-cipher-pass= --repo1-cipher-
type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo2-path=/demo-repo --repo3-
path=/demo-repo --repo1-retention-diff=2 --repo1-retention-full=2 --repo2-retention-
full=4 --repo3-retention-full=4 --repo3-s3-bucket=demo-bucket --repo3-s3-endpoint=s3.us-
east-1.amazonaws.com --repo3-s3-key= --repo3-s3-key-secret= --repo3-s3-region=us-east-1
--repo2-type=azure --repo3-type=s3 --stanza=demo
```

16 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/12/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 2.51: --exec-id=4444-af89d79d --log-level-console=info
--log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/pgsql/12/data --process-
max=4 --repo=4 --repo2-azure-account= --repo2-azure-container=demo-container --repo2-
azure-key= --repo1-block --repo1-bundle --repo4-bundle --repo1-cipher-pass= --repo1-
cipher-type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo2-path=/demo-repo --
repo3-path=/demo-repo --repo4-path=/demo-repo --repo1-retention-diff=2 --repo1-
retention-full=2 --repo2-retention-full=4 --repo3-retention-full=4 --repo3-s3-
bucket=demo-bucket --repo3-s3-endpoint=s3.us-east-1.amazonaws.com --repo3-s3-key= --
repo3-s3-key-secret= --repo3-s3-region=us-east-1 --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 --repo2-type=azure --repo3-type=s3 --repo4-
type=sftp --stanza=demo --start-fast
P00   WARN: option 'repo4-retention-full' is not set for 'repo4-retention-full-
type=count', the repository may run out of space
            HINT: to retain full backups indefinitely (without warning), set option
'repo4-retention-full' to the maximum.
P00   WARN: no prior backup exists, incr backup has been changed to full
P00   INFO: execute non-exclusive backup start: backup begins after the requested
immediate checkpoint completes
P00   INFO: backup start archive = 00000006000000000000001F, lsn = 0/1F000028
       [filtered 3 lines of output]
P00   INFO: check archive for segment(s)
00000006000000000000001F:00000006000000000000001F
P00   INFO: new backup label = 20240320-210759F
```

```
P00   INFO: full backup size = 31.1MB, file total = 1287
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.51: --exec-id=4444-af89d79d --log-level-console=info
--log-level-stderr=off --no-log-timestamp --repo=4 --repo2-azure-account= --repo2-azure-
container=demo-container --repo2-azure-key= --repo1-cipher-pass= --repo1-cipher-
type=aes-256-cbc --repo1-path=/var/lib/pgbackrest --repo2-path=/demo-repo --repo3-
path=/demo-repo --repo4-path=/demo-repo --repo1-retention-diff=2 --repo1-retention-
full=2 --repo2-retention-full=4 --repo3-retention-full=4 --repo3-s3-bucket=demo-bucket -
-repo3-s3-endpoint=s3.us-east-1.amazonaws.com --repo3-s3-key= --repo3-s3-key-secret= --
repo3-s3-region=us-east-1 --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 --repo2-type=azure --repo3-type=s3 --repo4-
type=sftp --stanza=demo
P00   INFO: expire command end: completed successfully
```

17 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/12/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 object stores is relatively slow so commands benefit by increasing `process-max` to parallelize

file creation.

18 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.

18.1 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.

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 7-8 packages for pgBackRest are available from Crunchy Data or 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-release-2.51/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
```

18.2 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/12/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/12/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.

18.3 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=simple
  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=simple
  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
```

18.4 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
```

18.5 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.

18.6 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-12.service
$ sudo -u postgres pgbackrest --stanza=demo --delta restore
$ sudo systemctl start postgresql-12.service
```

19 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/12/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 (12): 000000070000000000000025/000000070000000000000027
```

```
            full backup: 20240320-210830F
                timestamp start/stop: 2024-03-20 21:08:30+00 / 2024-03-20 21:08:34+00
                wal start/stop: 000000070000000000000025 / 000000070000000000000025
                database size: 31.1MB, database backup size: 31.1MB
                repo1: backup set size: 3.7MB, backup size: 3.7MB

            full backup: 20240320-210835F
                timestamp start/stop: 2024-03-20 21:08:35+00 / 2024-03-20 21:08:38+00
                wal start/stop: 000000070000000000000026 / 000000070000000000000027
                database size: 31.1MB, database backup size: 31.1MB
                repo1: backup set size: 3.7MB, backup size: 3.7MB
```

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.

20 Starting and Stopping

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

pg-primary ⇒ Stop the pgBackRest services

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

New pgBackRest processes 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-
  primary': stop file exists for all stanzas
  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 process that are currently running. 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 processes again with the start command.

pg-primary ⇒ Start the pgBackRest services

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

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

pg-primary ⇒ Stop pgBackRest services for the `demo` stanza

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

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

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-
  primary': stop file exists for stanza demo
  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 the pgBackRest processes for a single stanza.

pg-primary ⇒ Start the pgBackRest services for the `demo` stanza

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

21 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.

21.1 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 7-8 packages for pgBackRest are available from Crunchy Data or 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-release-2.51/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
```

21.2 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/12/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=simple
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/12/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/12/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 2024-03-20 21:06:46
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2024-03-20 21:07:16
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2024-03-20 21:07:38
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
# Removed by pgBackRest restore on 2024-03-20 21:08:27 # recovery_target_time = '2024-
03-20 21:07:29.317767+00'
# Removed by pgBackRest restore on 2024-03-20 21:08:27 # recovery_target_action =
'promote'

# Recovery settings generated by pgBackRest restore on 2024-03-20 21:08:27
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

# Recovery settings generated by pgBackRest restore on 2024-03-20 21:08:51
restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
```

The `hot_standby` setting must be enabled before starting PostgreSQL to allow read-only connections on pg-standby. Otherwise, connection attempts will be refused. The rest of the configuration is in case the standby is promoted to a primary.

pg-standby:`/var/lib/pgsql/12/data/postgresql.conf` ⇒ Configure PostgreSQL

```
archive_command = 'pgbackrest --stanza=demo archive-push %p'
archive_mode = on
hot_standby = on
log_filename = 'postgresql.log'
max_wal_senders = 3
wal_level = replica
```

pg-standby ⇒ Start PostgreSQL

```
$ sudo systemctl start postgresql-12.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/12/data/log/postgresql.log
LOG:  database system was interrupted; last known up at 2024-03-20 21:08:35 UTC
LOG:  entering standby mode
LOG:  restored log file "00000007.history" from archive
LOG:  restored log file "000000070000000000000026" from archive
LOG:  redo starts at 0/26000028
LOG:  restored log file "000000070000000000000027" from archive
LOG:  consistent recovery state reached at 0/27000050
LOG:  database system is ready to accept read only connections
```

An easy way to test that replication is properly configured is to create a table on 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";
     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
---------------+-------------------------------
 0/280217D8    | 2024-03-20 21:08:55.448036+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 | 2024-03-20 21:08:57.932964+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 2.51: --exec-id=1289-ae55e99f --log-level-console=info --log-level-file=detail --log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/pgsql/12/data --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 --stanza=demo
P00   INFO: check repo1 (standby)
P00   INFO: switch wal not performed because this is a standby
P00   INFO: check command end: completed successfully
```

21.3 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/12/data/pg_hba.conf'
$ sudo systemctl reload postgresql-12.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/12/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-12.service
$ sudo -u postgres pgbackrest --stanza=demo --delta --type=standby restore
$ sudo -u postgres cat /var/lib/pgsql/12/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 2024-03-20 21:06:46
  restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

  # Recovery settings generated by pgBackRest restore on 2024-03-20 21:07:16
  restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

  # Recovery settings generated by pgBackRest restore on 2024-03-20 21:07:38
  restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'
  # Removed by pgBackRest restore on 2024-03-20 21:08:27 # recovery_target_time = '2024-
  03-20 21:07:29.317767+00'
  # Removed by pgBackRest restore on 2024-03-20 21:08:27 # recovery_target_action =
  'promote'

  # Recovery settings generated by pgBackRest restore on 2024-03-20 21:08:27
  restore_command = 'pgbackrest --stanza=demo archive-get %f "%p"'

  # Recovery settings generated by pgBackRest restore on 2024-03-20 21:09:00
  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 7-8 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/12/data/postgresql.conf` ⇒ Enable `hot_standby`

```
  archive_command = 'pgbackrest --stanza=demo archive-push %p'
  archive_mode = on
  hot_standby = on
  log_filename = 'postgresql.log'
  max_wal_senders = 3
  wal_level = replica
```

pg-standby ⇒ Start PostgreSQL

```
$ sudo systemctl start postgresql-12.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/12/data/log/postgresql.log
        [filtered 7 lines of output]
  LOG:  database system is ready to accept read only connections
  LOG:  restored log file "000000070000000000000028" from archive
  LOG:  started streaming WAL from primary at 0/29000000 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";
     message      |       current_timestamp
----------------+-------------------------------
 Important Data | 2024-03-20 21:09:04.220613+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 | 2024-03-20 21:09:04.430511+00
(1 row)
```

22 Multiple Stanzas

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

22.1 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 7-8 packages for pgBackRest are available from Crunchy Data or 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-release-2.51/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
```

22.2 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/12/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/12/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/12/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=simple
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
```

22.3 Setup Demo Cluster

pg-alt ⇒ Create the demo cluster

```
$ sudo -u postgres /usr/pgsql-12/bin/initdb \
      -D /var/lib/pgsql/12/data -k -A peer
```

pg-alt:`/var/lib/pgsql/12/data/postgresql.conf` ⇒ Configure PostgreSQL settings

```
archive_command = 'pgbackrest --stanza=demo-alt archive-push %p'
archive_mode = on
log_filename = 'postgresql.log'
max_wal_senders = 3
wal_level = replica
```

pg-alt ⇒ Start the demo cluster

```
$ sudo systemctl restart postgresql-12.service
```

22.4 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 2.51: --exec-id=848-89eeb550 --log-level-
console=info --log-level-file=detail --log-level-stderr=off --no-log-timestamp --pg1-
path=/var/lib/pgsql/12/data --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 --
stanza=demo-alt
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 2.51: --exec-id=876-3779c874 --log-level-console=info --
log-level-file=detail --log-level-stderr=off --no-log-timestamp --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
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/pgbackrest/archive/demo-alt/12-1/0000000100000000/000000010000000000000001-
a161ad3854eb1398bc559bcbad31ec5804ff8a0e.gz' on repo1
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 2.51: --exec-id=1236-4e98cadc --log-level-console=info -
-log-level-stderr=off --no-log-timestamp --repo1-path=/var/lib/pgbackrest
P00   INFO: check stanza 'demo'
P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)
P00   INFO: WAL segment 000000070000000000000029 successfully archived to
'/var/lib/pgbackrest/archive/demo/12-1/0000000700000000/000000070000000000000029-
9731dc63334f04b604c18b4169f3bd2ac698462f.gz' on repo1
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/pgbackrest/archive/demo-alt/12-1/0000000100000000/000000010000000000000002-
baabc00c9fe06e86ac5fe0d6c591362c0ee0eafc.gz' on repo1
P00   INFO: check command end: completed successfully
```

23 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/12/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/12/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-12.service
```

23.1 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 2.51: --exec-id=5506-c03e5a0f --log-level-console=info --log-level-file=detail --log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/pgsql/12/data --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 --stanza=demo
  P00   INFO: check repo1 configuration (primary)
  P00   INFO: check repo1 archive for WAL (primary)
  P00   INFO: WAL segment 000000070000000000000002F successfully archived to '/var/lib/pgbackrest/archive/demo/12-1/0000000700000000/000000070000000000000002F-1be413a5dccc946080bf9936efe8396cc838d0ad.gz' on repo1
  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 2.51: [/var/lib/pgsql/12/data/pg_wal] --
archive-async --exec-id=5474-ebdcd94c --log-level-console=off --log-level-file=detail --
log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/pgsql/12/data --process-
max=2 --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 --stanza=demo
P00   INFO: push 1 WAL file(s) to archive: 00000007000000000000002A
P01 DETAIL: pushed WAL file '00000007000000000000002A' to the archive
P00 DETAIL: statistics: {"socket.client":{"total":1},"socket.session":
{"total":1},"tls.client":{"total":1},"tls.session":{"total":1}}
P00   INFO: archive-push:async command end: completed successfully

-------------------PROCESS START-------------------
P00   INFO: archive-push:async command begin 2.51: [/var/lib/pgsql/12/data/pg_wal] --
archive-async --exec-id=5508-78eab1ef --log-level-console=off --log-level-file=detail --
log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/pgsql/12/data --process-
max=2 --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 --stanza=demo
P00   INFO: push 4 WAL file(s) to archive:
00000007000000000000002B...00000007000000000000002E
P02 DETAIL: pushed WAL file '00000007000000000000002C' to the archive
P01 DETAIL: pushed WAL file '00000007000000000000002B' to the archive
P01 DETAIL: pushed WAL file '00000007000000000000002E' to the archive
P02 DETAIL: pushed WAL file '00000007000000000000002D' to the archive
P00 DETAIL: statistics: {"socket.client":{"total":1},"socket.session":
{"total":1},"tls.client":{"total":1},"tls.session":{"total":1}}
P00   INFO: archive-push:async command end: completed successfully

-------------------PROCESS START-------------------
P00   INFO: archive-push:async command begin 2.51: [/var/lib/pgsql/12/data/pg_wal] --
archive-async --exec-id=5516-74267498 --log-level-console=off --log-level-file=detail --
log-level-stderr=off --no-log-timestamp --pg1-path=/var/lib/pgsql/12/data --process-
max=2 --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 --stanza=demo
P00   INFO: push 1 WAL file(s) to archive: 00000007000000000000002F
P01 DETAIL: pushed WAL file '00000007000000000000002F' to the archive
P00 DETAIL: statistics: {"socket.client":{"total":1},"socket.session":
{"total":1},"tls.client":{"total":1},"tls.session":{"total":1}}
P00   INFO: archive-push:async command end: completed successfully
```

23.2 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 2.51: [000000070000000000000026,
000000070000000000000027, 000000070000000000000028, 000000070000000000000029,
00000007000000000000002A, 00000007000000000000002B, 00000007000000000000002C,
00000007000000000000002D] --archive-async --exec-id=1846-0f262ffe --log-level-
console=off --log-level-file=detail --log-level-stderr=off --no-log-timestamp --pg1-
path=/var/lib/pgsql/12/data --process-max=2 --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 --stanza=demo
P00   INFO: get 8 WAL file(s) from archive:
000000070000000000000026...00000007000000000000002D
```

```
P01 DETAIL: found 000000070000000000000026 in the repo1: 12-1 archive
P02 DETAIL: found 000000070000000000000027 in the repo1: 12-1 archive
P02 DETAIL: found 000000070000000000000029 in the repo1: 12-1 archive
P01 DETAIL: found 000000070000000000000028 in the repo1: 12-1 archive
```

```
P00 DETAIL: unable to find 00000007000000000000002A in the archive
P00 DETAIL: statistics: {"socket.client":{"total":1},"socket.session":
{"total":1},"tls.client":{"total":1},"tls.session":{"total":1}}
      [filtered 31 lines of output]
P00   INFO: archive-get:async command begin 2.51: [000000070000000000000002A,
000000070000000000000002B, 000000070000000000000002C, 000000070000000000000002D,
000000070000000000000002E, 000000070000000000000002F, 000000070000000000000030,
000000070000000000000031] --archive-async --exec-id=1905-2de2c0e3 --log-level-
console=off --log-level-file=detail --log-level-stderr=off --no-log-timestamp --pg1-
path=/var/lib/pgsql/12/data --process-max=2 --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 --stanza=demo
P00   INFO: get 8 WAL file(s) from archive:
000000070000000000000002A...000000070000000000000031
```

```
P02 DETAIL: found 000000070000000000000002B in the repo1: 12-1 archive
P01 DETAIL: found 000000070000000000000002A in the repo1: 12-1 archive
P02 DETAIL: found 000000070000000000000002C in the repo1: 12-1 archive
P01 DETAIL: found 000000070000000000000002D in the repo1: 12-1 archive
P02 DETAIL: found 000000070000000000000002E in the repo1: 12-1 archive
P01 DETAIL: found 000000070000000000000002F in the repo1: 12-1 archive
```

```
P00 DETAIL: unable to find 000000070000000000000030 in the archive
P00 DETAIL: statistics: {"socket.client":{"total":1},"socket.session":
{"total":1},"tls.client":{"total":1},"tls.session":{"total":1}}
      [filtered 14 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
```

24 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/12/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/12/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/12/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 non-exclusive backup start: backup begins after the requested
immediate checkpoint completes
P00   INFO: backup start archive = 000000070000000000000031, lsn = 0/31000028
P00   INFO: wait for replay on the standby to reach 0/31000028
P00   INFO: replay on the standby reached 0/31000028
P00   INFO: check archive for prior segment 000000070000000000000030
P01 DETAIL: backup file pg-primary:/var/lib/pgsql/12/data/global/pg_control (8KB, 0.34%)
checksum be3626e17470417c9e2522e428bcd859a454db71
P02 DETAIL: backup file pg-standby:/var/lib/pgsql/12/data/base/13396/2608 (456KB,
19.77%) checksum b8a6e41d5dad306781c877be0fc6ade524b9d171
P03 DETAIL: backup file pg-standby:/var/lib/pgsql/12/data/base/13396/1249 (440KB,
38.52%) checksum 58b34f74f7c66922e3b8de566b8b4aeebb9c8775
P04 DETAIL: backup file pg-standby:/var/lib/pgsql/12/data/base/13396/2674 (344KB,
53.18%) checksum 437be9641d5181d251265527b64bd243fd1e4f5f
P01 DETAIL: backup file pg-primary:/var/lib/pgsql/12/data/log/postgresql.log (6.0KB,
53.44%) checksum 98a718c1e70728610eb85a159e284a30f7619c3e
P01 DETAIL: backup file pg-primary:/var/lib/pgsql/12/data/pg_hba.conf (4.5KB, 53.63%)
checksum 65e54ae24bda87b2542351cb16a7fecc7e5aceeb
P01 DETAIL: match file from prior backup pg-
primary:/var/lib/pgsql/12/data/current_logfiles (26B, 53.63%) checksum
78a9f5c10960f0d91fcd3139374698248617795a2
P04 DETAIL: backup file pg-standby:/var/lib/pgsql/12/data/base/13396/1259 (104KB,
58.06%) checksum 6aa134e928bdf36cb818005b04bacd2f2acc93b9
        [filtered 1298 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.

25 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-12.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-12.service
```

Create the new cluster and perform upgrade.

pg-primary ⇒ Create new cluster and perform the upgrade

```
$ sudo -u postgres /usr/pgsql-13/bin/initdb \
      -D /var/lib/pgsql/13/data -k -A peer
$ sudo -u postgres sh -c 'cd /var/lib/pgsql && \
      /usr/pgsql-13/bin/pg_upgrade \
      --old-bindir=/usr/pgsql-12/bin \
      --new-bindir=/usr/pgsql-13/bin \
      --old-datadir=/var/lib/pgsql/12/data \
      --new-datadir=/var/lib/pgsql/13/data \
      --old-options=" -c config_file=/var/lib/pgsql/12/data/postgresql.conf" \
      --new-options=" -c config_file=/var/lib/pgsql/13/data/postgresql.conf"'
        [filtered 66 lines of output]
Checking for extension updates                              ok

Upgrade Complete

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

Configure the new cluster settings and port.

pg-primary:`/var/lib/pgsql/13/data/postgresql.conf` ⇒ Configure PostgreSQL

```
archive_command = 'pgbackrest --stanza=demo archive-push %p'
archive_mode = on
log_filename = 'postgresql.log'
max_wal_senders = 3
wal_level = replica
```

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/13/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/13/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/13/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/13/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/12/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/12/data/pg_hba.conf \
      /var/lib/pgsql/13/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 2.51: --exec-id=6021-d2bfba9d --log-level-
console=info --log-level-file=detail --log-level-stderr=off --no-log-timestamp --no-
online --pg1-path=/var/lib/pgsql/13/data --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 --
stanza=demo
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-13.service
```

Test configuration using the check command.

pg-primary ⇒ Check configuration

```
$ sudo -u postgres systemctl status postgresql-13.service
$ sudo -u postgres pgbackrest --stanza=demo check
```

Remove the old cluster.

pg-primary ⇒ Remove old cluster

```
$ sudo rm -rf /var/lib/pgsql/12/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/12/data
$ sudo -u postgres mkdir -p -m 700 /usr/pgsql-13/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-standby': unable to connect to 'dbname='postgres' port=5432': could not connect to
  server: No such file or directory
                  Is the server running locally and accepting
                  connections on Unix domain socket "/run/postgresql/.s.PGSQL.5432"?
```

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:`/var/lib/pgsql/13/data/postgresql.conf` ⇒ Configure PostgreSQL

```
  hot_standby = on
```

pg-standby ⇒ Start PostgreSQL and check the pgBackRest configuration

```
$ sudo systemctl start postgresql-13.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/13/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/13/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/12/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
```

Copyright © 2015-2024, The PostgreSQL Global Development Group, MIT License. Updated March 24, 2024