

# PGConf.DE 2023



Deep dive into the  
pgBackRest world

Stefan FERCOT

June 27th, 2023

# Who Am I?

- Stefan Fercot
- Database Backup Architect @EDB
- pgBackRest contributor
- aka. pgstef
- <https://pgstef.github.io>

# Agenda

- installation & configuration
- various backup storage types
  - multi-repository feature
- less common operations
  - interact with a standby server
  - asynchronous archiving
- latest features

# pgBackRest

- aims to be a simple, reliable backup and restore system
- current release: 2.46 (May 22, 2023)
- local or remote operation (via SSH or TLS server)
- parallel and asynchronous operations
- S3, Azure, and GCS support
- various compression types: gz, bz2, lz4, zst
- client-side encryption (aes-256-cbc)
- ...

# Installation

- *Use the PGDG repository, Luke!*
  - yum / dnf / apt-get install pgbackrest

# Configuration

- if `/etc/pgbackrest/pgbackrest.conf` does not exist
  - `/etc/pgbackrest.conf` is used

```
[global]
repo1-path=/var/lib/pgsql/15/backups
repo1-retention-full=1
log-level-console=info
```

```
[my_stanza]
pg1-path=/var/lib/pgsql/15/data
pg1-user=postgres
pg1-port=5432
```

- main configuration in the `[global]` part
- each cluster to backup has its own configuration called `stanza`

# Setup - archiving

```
# postgresql.conf
archive_mode = on
archive_command = 'pgbackrest --stanza=my_stanza archive-push %p'
```

- Tips for debugging purposes:
  - look at the PostgreSQL logs!
  - add `--log-level-console=debug`
  - or directly in `pgbackrest.conf`:

```
[my_stanza:archive-push]
log-level-console=debug
```

# Initialization

```
$ pgbackrest --stanza=my_stanza stanza-create
P00  INFO: stanza-create command begin 2.46: ...
P00  INFO: stanza-create for stanza 'my_stanza' on repo1
P00  INFO: stanza-create command end: completed successfully

$ pgbackrest --stanza=my_stanza check
P00  INFO: check command begin 2.46: ...
P00  INFO: check repo1 configuration (primary)
P00  INFO: check repo1 archive for WAL (primary)
P00  INFO: WAL segment ... successfully archived to '...' on repo1
P00  INFO: check command end: completed successfully
```



# Full backup

```
$ pgbackrest --stanza=my_stanza --type=full backup
P00 INFO: backup command begin 2.46: ...
P00 INFO: execute non-exclusive backup start:
backup begins after the next regular checkpoint completes
P00 INFO: backup start archive = 00000001000000000000000003, lsn = 0/3000028
P00 INFO: check archive for prior segment 00000001000000000000000002
P00 INFO: execute non-exclusive backup stop and wait for all WAL segments to archive
P00 INFO: backup stop archive = 00000001000000000000000003, lsn = 0/3000138
P00 INFO: check archive for segment(s) 00000001000000000000000003:00000001000000000000000003
P00 INFO: new backup label = 20230525-120443F
P00 INFO: full backup size = 22MB, file total = 966
P00 INFO: backup command end: completed successfully

P00 INFO: expire command begin 2.46: ...
P00 INFO: repo1: 15-1 remove archive,
start = 00000001000000000000000001, stop = 00000001000000000000000002
P00 INFO: expire command end: completed successfully
```

# Backup types

- full
  - all database cluster files will be copied
  - no dependencies on previous backups
- incr
  - incremental from the last successful backup
  - file-level by default, block-level available since 2.46
- diff
  - like an incremental backup but always based on the last full backup

## INFO command

```
$ pgbackrest info --stanza=my_stanza
stanza: my_stanza
  status: ok
  cipher: none

db (current)
  wal archive min/max (15): 000000010000000000000003/000000010000000000000003

full backup: 20230525-120443F
  timestamp start/stop: 2023-05-25 12:04:43 / 2023-05-25 12:04:53
  wal start/stop: 000000010000000000000003 / 000000010000000000000003
  database size: 22MB, database backup size: 22MB
  repo1: backup set size: 2.9MB, backup size: 2.9MB
```

# Where do I store my backups?

Do not keep your backup storage on the database host!

- directly attached storage ( `repo1-type` )
- dedicated remote host ( `repo1-host` )

# Repository storage types

- `repo1-type`
  - 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 (since 2.46)

# Dedicated remote host

- install pgBackRest
- create a specific user on the backup server
- setup *password-less SSH* or *TLS* connection between the hosts

# Dedicated remote host - configuration

- Database server

```
[global]
repo1-host=backup-srv
repo1-host-user=pgbackrest
```

```
[my_stanza]
pg1-path=/var/lib/pgsql/15/data
pg1-user=postgres
pg1-port=5432
```

- Backup server

```
[global]
repo1-path=/backup_space
```

```
[my_stanza]
pg1-host=database-srv
pg1-host-user=postgres
pg1-path=/var/lib/pgsql/15/data
```

# Command execution with remote storage

- Database server
  - `archive_command`
  - restore
- Backup server
  - backup



# Using multiple repositories

- introduced in 2.33 (April 5, 2021)
  - redundancy
  - various retention settings
  - ...

```
# example
repo1-path=.../repo1
repo1-retention-full=2
repo2-path=.../repo2
repo2-retention-full=1
```

## `--repo` option

- backward compatibility
  - not required when only `repo1` is configured
- when a single repository is configured
  - recommended to use `repo1` in the configuration

## `stanza-create` command

- automatically operates on all configured repositories

```
$ pgbackrest --stanza=my_stanza stanza-create
P00 INFO: stanza-create command begin 2.46: ...
P00 INFO: stanza-create for stanza 'my_stanza' on repo1
P00 INFO: stanza-create for stanza 'my_stanza' on repo2
P00 INFO: stanza-create command end: completed successfully
```

## **check** command

- triggers a new WAL segment to be archived
- tries to push it to all defined repositories

```
$ pgbackrest --stanza=my_stanza check
P00 INFO: check command begin 2.46: ...
P00 INFO: check repo1 configuration (primary)
P00 INFO: check repo2 configuration (primary)
P00 INFO: check repo1 archive for WAL (primary)
P00 INFO: WAL segment ... successfully archived to '...' on repo1
P00 INFO: check repo2 archive for WAL (primary)
P00 INFO: WAL segment ... successfully archived to '...' on repo2
P00 INFO: check command end: completed successfully
```

## `archive-push` command

- tries to push the WAL archive to all reachable repositories
  - an error prevent PostgreSQL to remove/recycle the WAL file!
  - `archive-async=y` brings fault-tolerance

```
P00  DEBUG:      storage/storage::storageNewWrite: => {
    type: posix, name: .../repo1/archive/my_stanza/15-1/0000000100000000/
                00000001000000000000000004-9c9f8c7ba88433a5496ed7b594a93c1a25ca57c2.gz, ...}
...
P00  DEBUG:      storage/storage::storageNewWrite: => {
    type: posix, name: .../repo2/archive/my_stanza/15-1/0000000100000000/
                00000001000000000000000004-9c9f8c7ba88433a5496ed7b594a93c1a25ca57c2.gz, ...}
...
P00  INFO: pushed WAL file '00000001000000000000000004' to the archive
```

# Backups

- scheduled individually for each repository
- without `--repo`, used by priority order
  - (`repo1` > `repo2` > ...)

```
$ pgbackrest backup --stanza=my_stanza --type=full
P00 INFO: backup command begin 2.46: ...
P00 INFO: repo option not specified, defaulting to repo1
P00 INFO: execute non-exclusive backup start:
backup begins after the next regular checkpoint completes
P00 INFO: backup start archive = 00000001000000000000000006, lsn = 0/6000028
P00 INFO: check archive for prior segment 00000001000000000000000005
P00 INFO: execute non-exclusive backup stop and wait for all WAL segments to archive
P00 INFO: backup stop archive = 00000001000000000000000006, lsn = 0/6000138
P00 INFO: check archive for segment(s) 00000001000000000000000006:00000001000000000000000006
P00 INFO: new backup label = 20230525-121742F
P00 INFO: full backup size = 22MB, file total = 966
P00 INFO: backup command end: completed successfully
```

# Show information

- default order sorting backups by dates mixing the repositories
  - might be confusing to find the backups depending on each other

```
$ pgbackrest info --stanza=my_stanza
stanza: my_stanza
  status: ok
  cipher: none

db (current)
  wal archive min/max (15): 00000001000000000000000006/000000010000000000000008

  full backup: 20230525-121742F
    timestamp start/stop: 2023-05-25 12:17:42 / 2023-05-25 12:17:52
    wal start/stop: 00000001000000000000000006 / 000000010000000000000006
    database size: 22MB, database backup size: 22MB
    repo1: backup set size: 2.9MB, backup size: 2.9MB

  full backup: 20230525-121839F
    timestamp start/stop: 2023-05-25 12:18:39 / 2023-05-25 12:18:48
    wal start/stop: 00000001000000000000000008 / 000000010000000000000008
    database size: 22MB, database backup size: 22MB
    repo2: backup set size: 2.9MB, backup size: 2.9MB
```

# Show information per repository

```
$ pgbackrest info --stanza=my_stanza --repo=2
stanza: my_stanza
  status: ok
  cipher: none

db (current)
  wal archive min/max (15): 00000001000000000000000008/00000001000000000000000008

  full backup: 20230525-121839F
    timestamp start/stop: 2023-05-25 12:18:39 / 2023-05-25 12:18:48
    wal start/stop: 00000001000000000000000008 / 00000001000000000000000008
    database size: 22MB, database backup size: 22MB
    repo2: backup set size: 2.9MB, backup size: 2.9MB
```



# Recovery

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

- `archive-get` will look into the repositories in priority order
  - (`repo1` > `repo2` > ...)
- tolerate gaps!

# Less common operations

- refresh Streaming Replication standby
- take backups from the standby server
- asynchronously push or get WAL segments
- selective restore

# Refresh Streaming Replication standby

- repository reachable from both nodes
- add extra stanza configuration on the standby

```
recovery-option=primary_conninfo=host=primary user=replication_user
```

- perform a `delta` restore

```
$ pgbackrest --stanza=my_stanza --type=standby --delta restore
```

- check `primary_conninfo` and `restore_command` before restarting the service

# Test system restore

- use the `--archive-mode=off` restore option
  - disables archiving on restored cluster

# Take backups from the standby server

- `backup-standby` option

```
[global]
...
backup-standby=y

[my_stanza]
pg1-path=/var/lib/pgsql/15/data
pg1-user=postgres
pg1-port=5432
pg2-host=primary
pg2-host-user=postgres
pg2-path=/var/lib/pgsql/15/data
recovery-option=primary_conninfo=host=primary user=replication_user
```

- backup started on primary
  - wait replay location on standby
  - files are copied from the standby

# Asynchronous archiving

- triggered by `archive_command`
- using `archive-async=y`
  - temporary data (acknowledgments) stored into the `spool-path`
  - early archiving using `process-max` processes
- when multiple repositories are defined, and one is failing...
  - archives are pushed asynchronously to working repositories!

# Archiving queue

- `archive-push-queue-max`
  - maximum size of the PostgreSQL archive queue
  - prevent the WAL space from filling up until PostgreSQL stops completely...
  - ...but generate missing archives!
- very important to monitor archiving to ensure it continues working

# Asynchronously get WAL segments

- `archive-get` using `archive-async=y`
  - early fetching `archive-get-queue-max` amount of WAL segments to speed up recovery
  - using `process-max` processes
  - stored in the `spool-path`



# Selective restore

- `--db-include`
  - databases not specifically included will be restored as sparse, zeroed files
  - built-in databases (template0, template1, and postgres) are always restored unless specifically excluded
- `--db-exclude`
  - databases excluded will be restored as sparse, zeroed files
  - with the `--db-include` option, only apply to built-in databases
- `DROP DATABASE` to remove the zeroed databases after recovery

# Latest features

- TLS server
- File bundling
- Backup annotations

# TLS server

- introduced in 2.37 (January 3, 2022)
  - to replace SSH connections
- TLS server must be configured and started on each host
  - `tls-server-*` options used for configuring the TLS server
  - `pg1-host-type=tls` on the backup server
  - `repo1-host-type=tls` on the database server
- certificates generated in the same way as PostgreSQL

See complete example in [EDB docs](#)

# File bundling

- introduced in 2.39 (May 16, 2022)
  - bundle/combine to improve small file support
  - zero-length files are not stored (except in the manifest)
- `repo-bundle`
- `repo-bundle-size`
- `repo-bundle-limit` - size limit for files that will be included in bundles

# Backup annotations

- introduced in 2.41 (September 19, 2022)
  - possibility to annotate backups with user-defined key/value pairs

```
$ pgbackrest backup --stanza=X --type=full \  
  --annotation=comment="this is our initial backup" \  
  --annotation=some-other-key="any text you'd like"
```

```
$ pgbackrest info --stanza=X --set=20230525-122223F
```

```
...
```

```
full backup: 20230525-122223F
```

```
timestamp start/stop: 2023-05-25 12:22:23 / 2023-05-25 12:22:32
```

```
wal start/stop: 000000010000000000000000C / 000000010000000000000000C
```

```
lsn start/stop: 0/C000028 / 0/C000138
```

```
database size: 22.2MB, database backup size: 22.2MB
```

```
repo1: backup set size: 2.9MB, backup size: 2.9MB
```

```
database list: postgres (5)
```

```
annotation(s)
```

```
comment: this is our initial backup
```

```
some-other-key: any text you'd like
```

# Where

- official website: <https://pgbackrest.org>
- user guides: <https://pgbackrest.org/user-guide.html>
- code: <https://github.com/pgbackrest/pgbackrest>
- EDB docs: <https://www.enterprisedb.com/docs/supported-open-source/pgbackrest>

# Conclusion

- pgBackRest is a powerful tool
  - with a lot of features and possibilities

# Questions?



Thank you for your attention!