Using multiple backup repositories with pgBackRest

data egret

Your remote PostgreSQL DBA team

Stefan FERCOT

contact@dataegret.com

Securing your PostgreSQL database availability and high performance.

- Performance audit
- Backup & restore
- Migration
- Cloud Cost Management
- Architecture review
- DataOps/ CDC projects
- 24/7 Incident support

on premises & cloud

EXPERTISE

Senior DBA team with **10+ years of PostgreSQL experience** each.

DEVELOPMENT

Involved in **new feature and extension development**.

TAILORED APPROACH

Dedicated DBA team that focused on success of your project.

COMMUNITY

Recognised significant **contributing sponsor to PostgreSQL**.

Stefan Fercot

- Senior PostgreSQL Expert
- pgBackRest fan & contributor
- aka. pgstef
- https://pgstef.github.io

Need a Disaster and Recovery Plan? ;-)
*Contact **Data Egret** to talk to me about backups and high-availability!*

Agenda

- basic config setup
- multi-repository feature insights
  - impact on each command
  - combined with asynchronous archiving

pgBackRest

- aims to be a simple, reliable backup and restore system
- current release: 2.55 (*2.55.0* - April 21, 2025)
- local or remote operation (via SSH or TLS server)
- parallel and asynchronous operations
- S3, Azure, and GCS support
- client-side encryption (aes-256-cbc)
- …

Installation

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

Configuration

- `/etc/pgbackrest.conf`, example:

```
[global]
repo1-path=/backup_space/backups
repo1-retention-full=1
log-level-console=info

[my_stanza]
pg1-path=/var/lib/pgsql/17/data
```

- main configuration in the `[global]` part
- each PostgreSQL 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'
```

Initialization

```
$ pgbackrest --stanza=my_stanza stanza-create
P00   INFO: stanza-create command begin 2.54.2: ...
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 2.54.2: ...
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 2.54.2: ...
P00   INFO: execute non-exclusive backup start:
backup begins after the next regular checkpoint completes
P00   INFO: backup start archive = 000000010000000000000003, lsn = 0/3000028
P00   INFO: check archive for prior segment 000000010000000000000002
P00   INFO: execute non-exclusive backup stop and wait for all WAL segments to archive
P00   INFO: backup stop archive = 000000010000000000000003, lsn = 0/3000158
P00   INFO: check archive for segment(s) 000000010000000000000003:000000010000000000000003
P00   INFO: new backup label = 20250218-150349F
P00   INFO: full backup size = 22.2MB, file total = 968
P00   INFO: backup command end: completed successfully
```

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 one repo 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 2.54.2: ...
P00   INFO: stanza-create for stanza 'my_stanza' on repo1
P00   INFO: stanza 'my_stanza' already exists on repo1 and is valid
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

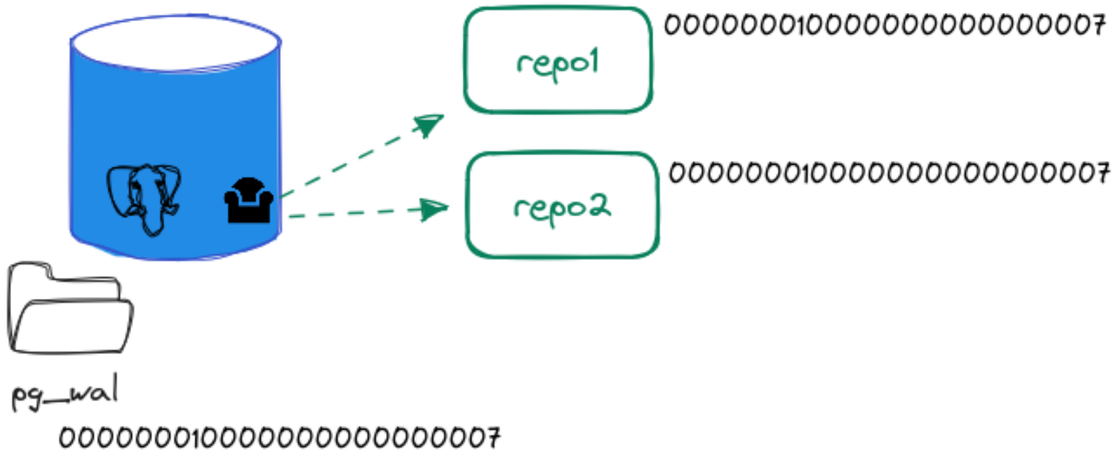
```
$ pgbackrest --stanza=my_stanza check
P00   INFO: check command begin 2.54.2: ...
P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo2 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)
P00   INFO: WAL segment ... successfully archived to '...' on repo1
P00   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

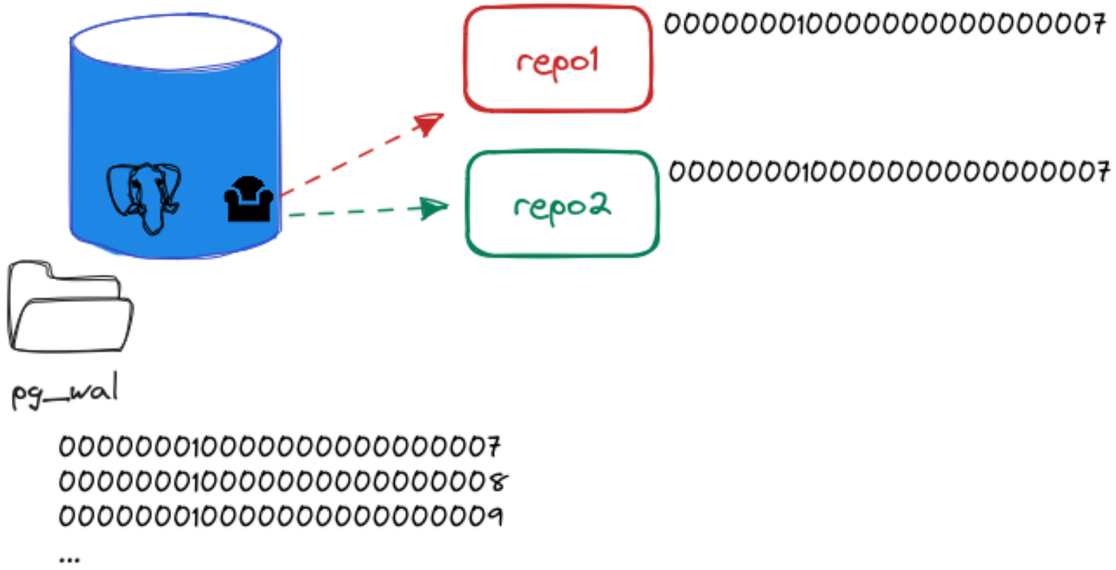
- tries to push the WAL archive to all reachable repositories
  - an error prevent PostgreSQL to remove/recycle the WAL file!
  - `archive-async=y` brings fault-tolerance

```
P00   DEBUG:     storage/storage::storageNewWrite: => {
  type: posix, name: .../repo1/archive/my_stanza/17-1/0000000100000000/
      000000010000000000000007-d0674b05bb800488524c928363238d4c3ad03b01.gz, ...}
...
P00   DEBUG:     storage/storage::storageNewWrite: => {
  type: posix, name: .../repo2/archive/my_stanza/17-1/0000000100000000/
      000000010000000000000007-d0674b05bb800488524c928363238d4c3ad03b01.gz, ...}
...
P00    INFO: pushed WAL file '000000010000000000000007' to the archive
```

Working archiving command



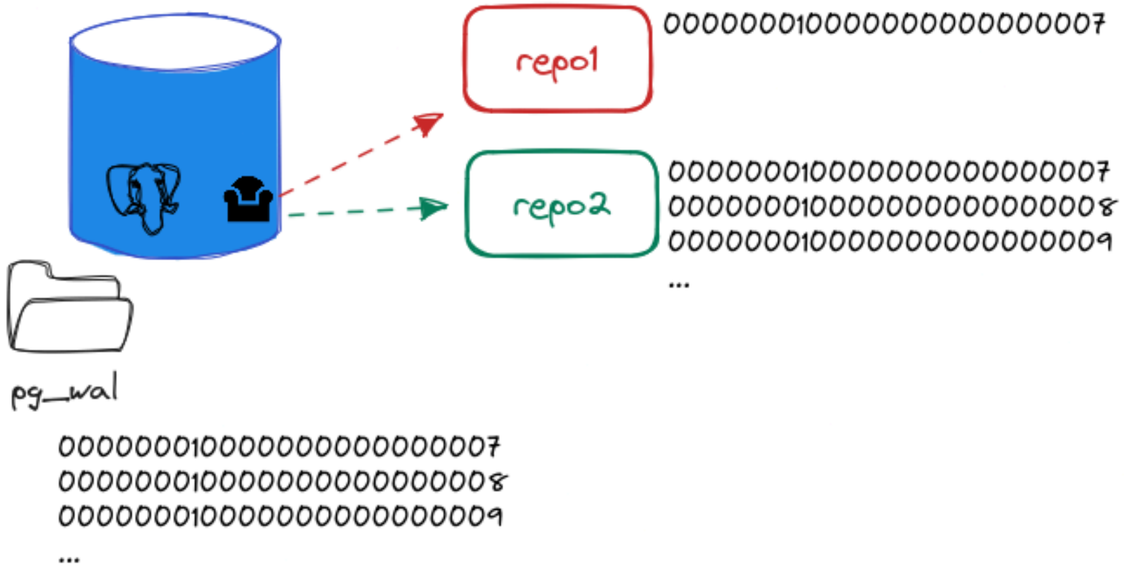
Blocked archiving command



Asynchronous archiving

- 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!

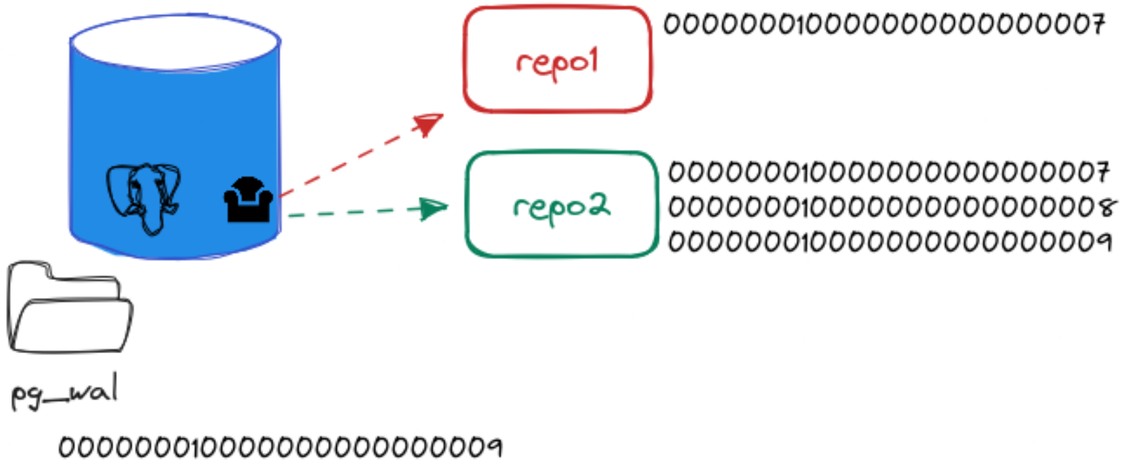
Working asynchronous archiving



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

Archiving queue exceeded



Backups

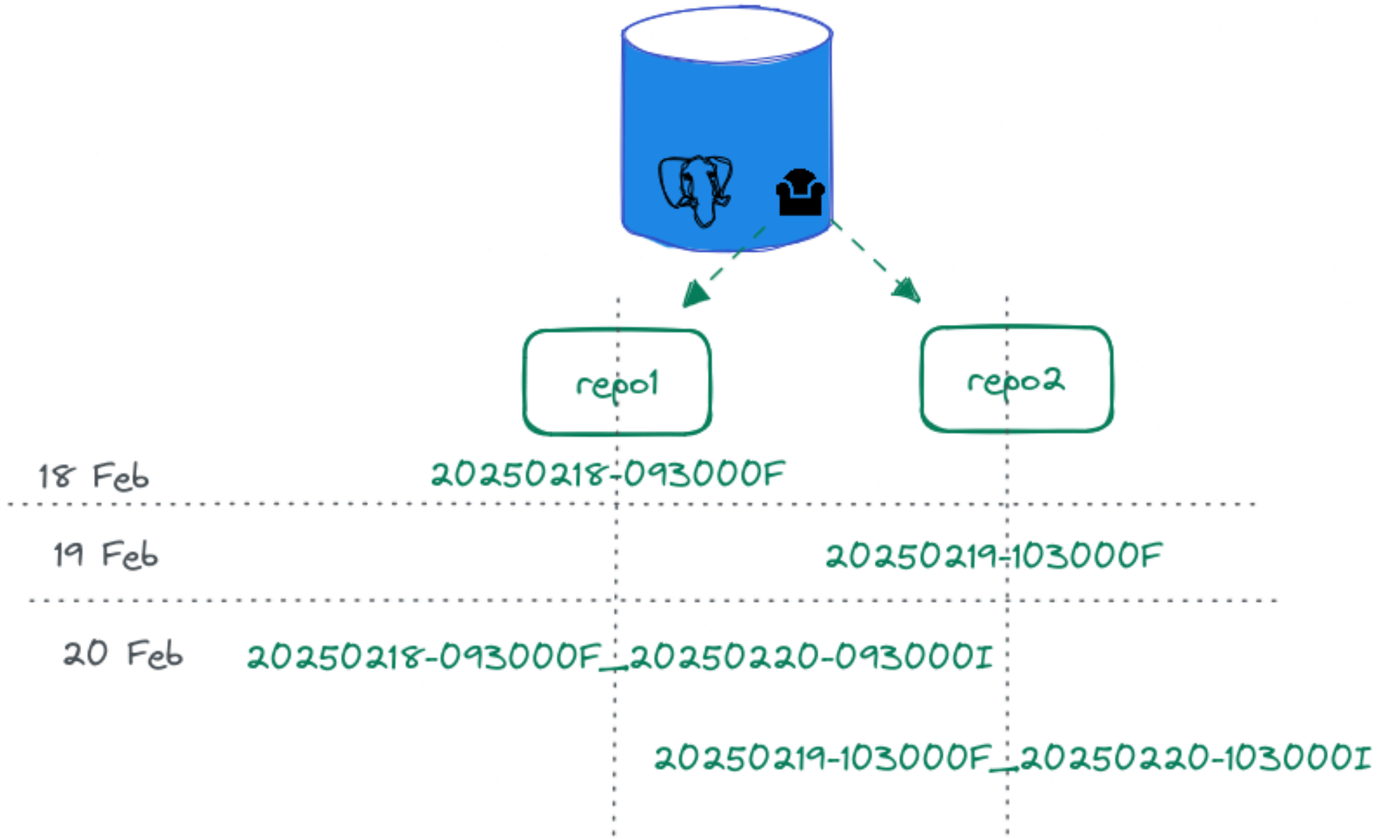
- scheduled individually for each repository
- without `--repo`, used by priority order
  - (`repo1` > `repo2` > ...)

```
$ pgbackrest backup --stanza=my_stanza --type=full
P00   INFO: backup command begin 2.54.2: ...
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 = 000000010000000000000009, lsn = 0/9000028
P00   INFO: check archive for prior segment 000000010000000000000008
P00   INFO: execute non-exclusive backup stop and wait for all WAL segments to archive
P00   INFO: backup stop archive = 000000010000000000000009, lsn = 0/9000120
P00   INFO: check archive for segment(s) 000000010000000000000009:000000010000000000000009
P00   INFO: new backup label = 20250218-152527F
P00   INFO: full backup size = 22.2MB, file total = 968
P00   INFO: backup 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
- `diff`: like `incr` but always based on the last **full** backup
- block-level backups (`diff` / `incr`)
  - saves space and increases the speed of the backup
  - by only copying the parts of files that have changed
  - at least v `2.52.1` is recommended

Mixing backup types



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
  wal archive min/max (17): 000000010000000000000009/000000010000000000000000B

  full backup: 20250218-152527F
      timestamp start/stop: 2025-02-18 15:25:27+00 / 2025-02-18 15:25:31+00
      wal start/stop: 000000010000000000000009 / 000000010000000000000009
      database size: 22.2MB, database backup size: 22.2MB
      repo1: backup set size: 2.9MB, backup size: 2.9MB

  full backup: 20250218-152624F
      timestamp start/stop: 2025-02-18 15:26:24+00 / 2025-02-18 15:26:28+00
      wal start/stop: 00000001000000000000000B / 00000001000000000000000B
      database size: 22.3MB, database backup size: 22.3MB
      repo2: backup set size: 2.9MB, backup size: 2.9MB
```

Show information per repository

```
$ pgbackrest info --stanza=my_stanza --repo=2
  wal archive min/max (17): 000000010000000000000000B/000000010000000000000000B

  full backup: 20250218-152624F
      timestamp start/stop: 2025-02-18 15:26:24+00 / 2025-02-18 15:26:28+00
      wal start/stop: 000000010000000000000000B / 000000010000000000000000B
      database size: 22.3MB, database backup size: 22.3MB
      repo2: backup set size: 2.9MB, backup size: 2.9MB
```

pgBackRest vs PostgreSQL recovery

pgBackRest restore command <> PostgreSQL recovery!

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!

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`

Where to store the repositories?

- supported repository types (`repo-type`)
  - `azure`
  - `gcs`
  - `s3`
  - `sftp`
  - `posix` / `cifs`
- for more tips about performance tuning

Conclusion

- the multi-repositories feature is great for **redundancy**
  - async ops to speed up archiving and recovery + **fault tolerance**!

Questions?

Final Thoughts

- pgBackRest is a powerful and flexible tool, with extensive features
  - to enhance your PostgreSQL backup and recovery strategy
- finding the best setup for your specific use case can be challenging
  - we’re here to help!

Closing words

💡 If you need guidance on optimizing your pgBackRest setup: Let’s Talk!

🎉 **Exclusive Offer:**

Connect with us within the next two weeks and enjoy a 10% discount!

📩 Contact **Data Egret GmbH** to discuss your PostgreSQL needs!

contact@dataegret.com

Ultimate

PostgreSQL Support Plan

08:00 - 17:00 CET
(excl. DE public holidays)

up to 60 Hours / year
up to 20 Hours /month

Unlimited number of servers and questions
Response time: 24 Hours for First Reaction
One-Time interaction is rounded to 30 minutes

Annual Cost **€6,000** excl. VAT
Additional Hours €350/hour excl. VAT

GRAB IT NOW !

Contact@DataEgret.com

On-Demand PostgreSQL Support That Feels In-House

- Query Optimisation
- Configuration Best Practices
- Backup & Recovery guidance
- Bloat Management
- Security advice
- Replication Troubleshooting
- Upgrade Issues and version compatibility
- Installation problems (incl. support for tools and extensions such as pgBackRest, Patroni, PostGIS, pgvector)

The Ultimate PostgreSQL Support Plan is designed for Q&A-style support, offering expert guidance without in-depth installation analysis or requiring the support team to access your database.

Consulting & Support

Packages

- incident support
- hands-on work
- architectural consulting
- remote DBA
- migrations
- data analytics
- machine learning infrastructure
- cloud cost optimisation
- regular proactive health-checks

GRAB IT NOW !

Contact@DataEgret.com