Using multiple backup repositories with pgBackRest

pgDay Paris 2022

Stefan FERCOT

Thu Mar 24th, 2022

EDB

Who Am I?

- Stefan Fercot
- aka. pgstef
- https://pgstef.github.io
- PostgreSQL user since 2010
- pgBackRest fan & contributor
- Database Backup Architect @EDB

Agenda

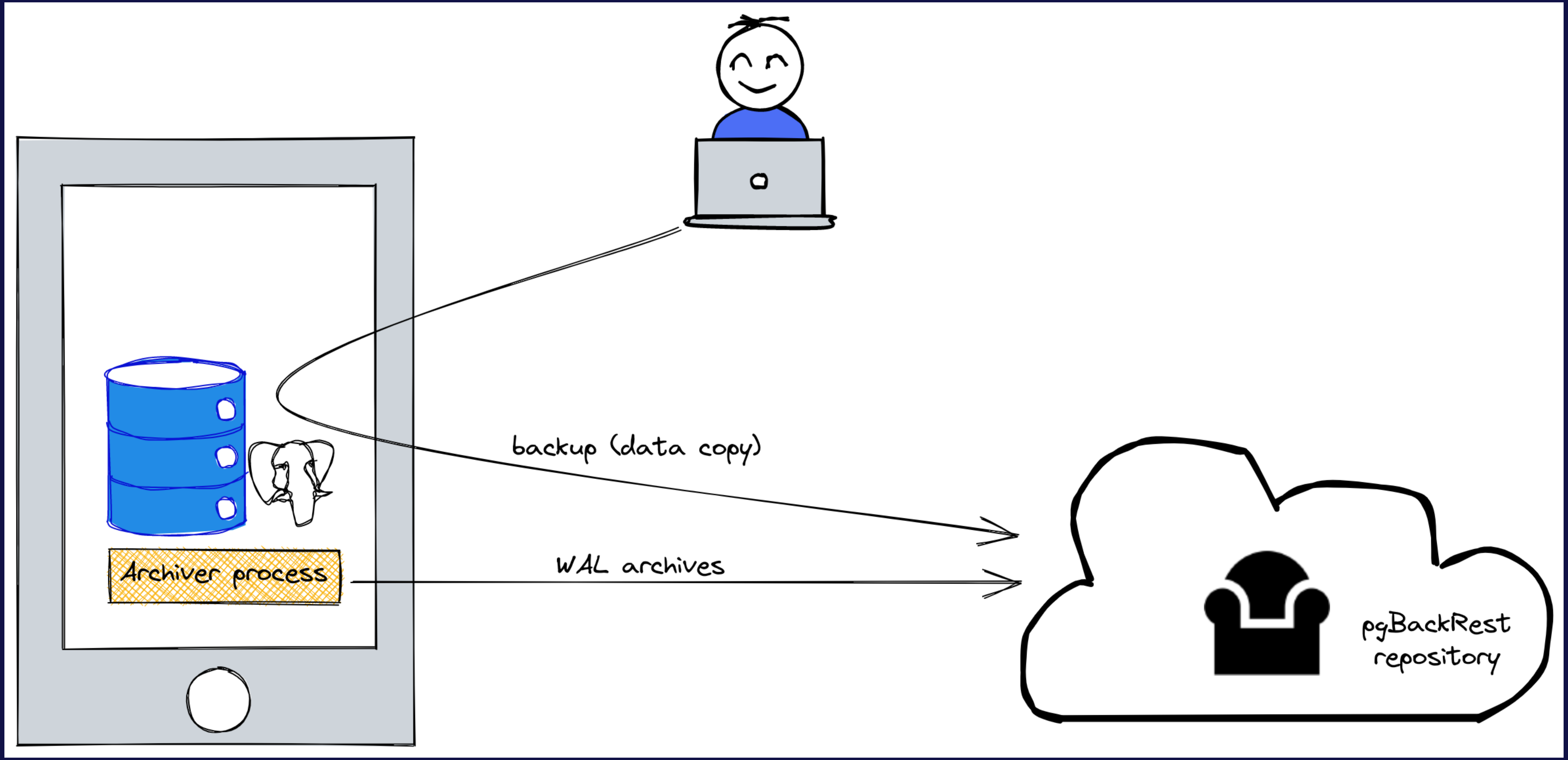
- basic functionalities reminder
- multi-repository feature insights
  - impact on each command
  - combined with asynchronous archiving

EDB

pgBackRest

- aims to be a simple, reliable backup and restore system
- current release: 2.38 (March 6, 2022)
- local or remote operation (via SSH or TLS server)
- parallel and asynchronous operations
- S3, Azure, and GCS support
- …

PG base backup and continuous archiving



Installation

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

Configuration

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

```
[global]
repo1-path=/var/lib/pgsql/14/backups
repo1-retention-full=1
log-level-console=info

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

- main configuration in the `[global]` part
- each PostgreSQL cluster to backup has its own configuration
  - called `stanza`
  - `pg*` options indexing
    - nodes linked together (e.g. using Streaming Replication)

EDB

Options precedence

1. command line argument
2. environment variable
3. `[stanza:command]`
4. `[stanza]`
5. `[global:command]`
6. `[global]`
7. default (internal)

Setup - archiving

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

Tip: add `--log-level-console=debug` for debugging purposes

Debug archive command issues

Tip: look at the PostgreSQL logs!

```
P00   INFO: archive-push command begin 2.38: [pg_wal/000000010000000000000002] ...
ERROR: [103]: unable to find a valid repository:
       repo1: [FileOpenError] unable to load info file ...
       FileOpenError: unable to open file '...' for read: [13] Permission denied
       FileOpenError: unable to open file '...' for read: [13] Permission denied
       HINT: archive.info cannot be opened but is required to push/get WAL segments.
       HINT: is archive_command configured correctly in postgresql.conf?
       HINT: has a stanza-create been performed?
P00   INFO: archive-push command end: aborted with exception [103]
```

Initialization

```
$ pgbackrest --stanza=my_stanza stanza-create
P00   INFO: stanza-create command begin 2.38: ...
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 2.38: ...
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 2.38: ...
P00   INFO: execute non-exclusive pg_start_backup():
backup begins after the next regular checkpoint completes
P00   INFO: backup start archive = 000000010000000000000004, lsn = 0/4000028
P00   INFO: check archive for prior segment 000000010000000000000003
P00   INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments to archive
P00   INFO: backup stop archive = 000000010000000000000004, lsn = 0/4000138
P00   INFO: check archive for segment(s) 000000010000000000000004:000000010000000000000004
P00   INFO: new backup label = 20220309-082913F
P00   INFO: full backup size = 25.2MB, file total = 951
P00   INFO: backup command end: completed successfully
```

```
P00   INFO: expire command begin 2.38: ...
P00   INFO: repo1: 14-1 remove archive,
start = 000000010000000000000001, stop = 000000010000000000000003
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
- `diff`
  - like an incremental backup but always based on the last **full** backup

EDB

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 2.38: ...
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 2.38: ...
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
```

EDB

`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/14-1/0000000100000000/
               000000010000000000000006-0d1ad4fa1e1f926414ad521b75db227f389a464c.gz"},
...
P00  DEBUG:     storage/storage::storageNewWrite: => {
  type: posix, name: {".../repo2/archive/my_stanza/14-1/0000000100000000/
               000000010000000000000006-0d1ad4fa1e1f926414ad521b75db227f389a464c.gz"},
...
P00   INFO: pushed WAL file '000000010000000000000006' to the archive
```

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!

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

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 2.38: ...
P00   INFO: repo option not specified, defaulting to repo1
P00   INFO: execute non-exclusive pg_start_backup():
backup begins after the next regular checkpoint completes
P00   INFO: backup start archive = 000000010000000000000008, lsn = 0/8000028
P00   INFO: check archive for prior segment 000000010000000000000007
P00   INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments to archive
P00   INFO: backup stop archive = 000000010000000000000008, lsn = 0/8000138
P00   INFO: check archive for segment(s) 000000010000000000000008:000000010000000000000008
P00   INFO: new backup label = 20220309-083636F
P00   INFO: full backup size = 25.2MB, file total = 951
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 (14): 000000010000000000000008/00000001000000000000000A

        full backup: 20220309-083636F
            timestamp start/stop: 2022-03-09 08:36:36 / 2022-03-09 08:36:39
            wal start/stop: 000000010000000000000008 / 000000010000000000000008
            database size: 25.2MB, database backup size: 25.2MB
            repo1: backup set size: 3.2MB, backup size: 3.2MB

        full backup: 20220309-083804F
            timestamp start/stop: 2022-03-09 08:38:04 / 2022-03-09 08:38:06
            wal start/stop: 00000001000000000000000A / 00000001000000000000000A
            database size: 25.3MB, database backup size: 25.3MB
            repo2: backup set size: 3.2MB, backup size: 3.2MB
```

Show information per repository

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

    db (current)
        wal archive min/max (14): 00000001000000000000000A/00000001000000000000000A

        full backup: 20220309-083804F
            timestamp start/stop: 2022-03-09 08:38:04 / 2022-03-09 08:38:06
            wal start/stop: 00000001000000000000000A / 00000001000000000000000A
            database size: 25.3MB, database backup size: 25.3MB
            repo2: backup set size: 3.2MB, backup size: 3.2MB
```

pgBackRest restore 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`

EDB

Where

- official website: https://pgbackrest.org
- user guides: https://pgbackrest.org/user-guide.html
- source code and issues: https://github.com/pgbackrest/pgbackrest
- EDB docs: https://www.enterprisedb.com/docs/supported-open-source/pgbackrest
- blog: https://pgstef.github.io

Conclusion

- pgBackRest is a powerful tool
  - with a lot of features and possibilities
- the multi-repositories feature is great for **redundancy**
  - async ops to speed up archiving and recovery + **fault tolerance**!

Questions?

Thank you for your attention!

EDB Goodies: Fill in our survey and receive a gift-card!

https://bit.ly/PGDayParis