Unleash the Power within pgBackRest

PostgreSQL User Group NL

Stefan FERCOT

11 March 2021

Who Am I?

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

Agenda

- pgBackRest
  - basic functionalities reminder
  - typical setups
  - less common operations
    - interact with a standby server
    - asynchronous archiving
  - diagnostic
  - monitoring
  - release radar

pgBackRest

- aims to be a simple, reliable backup and restore system
- written in C (migration completed in 2019)
- custom protocol
  - local or remote operation (via SSH)
- parallelism and asynchronous operation
- full/differential/incremental backup
- backup rotation and archive expiration
- S3/Azure support
- encryption
- multiple compression types (gzip, bzip, lz4, zstd)
- …

Installation

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

Configuration

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

```
[global]
repo1-path=/var/lib/pgsql/13/backups
repo1-retention-full=1
log-level-console=info

[my_stanza]
pg1-path=/var/lib/pgsql/13/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 2.32: ...
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 2.32: ...
P00   INFO: check repo1 configuration (primary)
P00   INFO: check repo1 archive for WAL (primary)
P00   INFO: WAL segment 000000010000000000000001 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 2.32: ...
P00   INFO: execute non-exclusive pg_start_backup():
backup begins after the next regular checkpoint completes
P00   INFO: backup start archive = 000000010000000000000003, lsn = 0/3000028
P00   INFO: full backup size = 23.1MB
P00   INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments to archive
P00   INFO: backup stop archive = 000000010000000000000003, lsn = 0/3000138
P00   INFO: check archive for segment(s) 000000010000000000000003:000000010000000000000003
P00   INFO: new backup label = 20210304-090917F
P00   INFO: backup command end: completed successfully

P00   INFO: expire command begin 2.32: ...
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

INFO command

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

    db (current)
        wal archive min/max (13): 000000010000000000000003/000000010000000000000007

        full backup: 20210304-090917F
            timestamp start/stop: 2021-03-04 09:09:17 / 2021-03-04 09:09:29
            wal start/stop: 000000010000000000000003 / 000000010000000000000003
            database size: 23.1MB, database backup size: 23.1MB
            repo1: backup set size: 2.8MB, backup size: 2.8MB
...
```

Typical setups

Do not keep your backup storage on the PostgreSQL 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
  - posix - Posix-compliant file systems
  - s3 - AWS Simple Storage Service
  - **gcs - Google Cloud Storage** coming soon!

Dedicated remote host

- create a specific user on the backup server
- install pgBackRest
- setup passwordless SSH connection between the hosts

Dedicated remote host - configuration

- pgsql-srv

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

[my_stanza]
pg1-path=/var/lib/pgsql/13/data
```

- backup-srv

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

[my_stanza]
pg1-host=pgsql-srv
pg1-host-user=postgres
pg1-path=/var/lib/pgsql/13/data
```

Command execution with remote storage

- **pgsql-srv**
  - `archive_command`
  - restore
- **backup-srv**
  - backup

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

Take backups from the standby server

- `backup-standby` option

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

[my_stanza]
pg1-path=/var/lib/pgsql/13/data
pg2-host=primary
pg2-path=/var/lib/pgsql/13/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

Asynchronously push WAL segments

- `archive-async=y` and `spool-path`: store temporary data into a local directory
- `archive-push` for `archive_command`
  - write WAL segments into the spool path and acknowledgments when successfully stored in the archive
  - `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-async=y` and `spool-path`
- `archive-get` for `restore_command`
  - prefetch `archive-get-queue-max` amount of WAL segments to speed up recovery

Selective restore

- `--db-include` restore option
  - databases not specifically included will be restored as sparse, zeroed files
  - `DROP DATABASE` to remove the zeroed databases after recovery

Diagnostic

- checksums
- `check` command
- `verify` command

Checksums

- PostgreSQL `initdb --data-checksums`
  - PGSETUP_INITDB_OPTIONS
- pg_checksums
  - enable, disable or check data checksums **offline**
- pgBackRest `--checksum-page`
  - validate all data page checksums while backing up a cluster
  - automatically enabled when data page checksums are enabled on the cluster

EDB

`check` command

- validates configuration and `archive_command` setting
- calls `pg_create_restore_point('pgBackRest Archive Check')` and `pg_switch_wal()`

`verify` command

- internal command only, **work in progress**

```
pgBackRest 2.32 - 'verify' command help

Verify the contents of the repository.

Verify will attempt to determine if the backups and archives in the repository
are valid.
```

- WAL validation and backup files verification

```
INFO: Results:
  archiveId: 13-1, total WAL checked: 4, total valid WAL: 4
    missing: 0, checksum invalid: 0, size invalid: 0, other: 0
  backup: 20210304-100800F, status: valid, total files checked: 939, total valid files: 939
    missing: 0, checksum invalid: 0, size invalid: 0, other: 0
```

Monitoring

Schrödinger's Law of Backups

> *The condition/state of any backup is unknown until a restore is attempted.*

- play with `pgbackrest info --output=json` within PostgreSQL…
- … or use check_pgbackrest

check_pgbackrest

- whatever the backups location?
  - only based on `pgbackrest info` output and `repo` commands!

Available services

```
$ check_pgbackrest --list

List of available services:
  archives           Check WAL archives.
  check_pgb_version  Check the version of this check_pgbackrest script.
  retention          Check the retention policy.
```

Retention

- Fails when
  - the number of full backups is less than `--retention-full`
  - the newest backup is older than `--retention-age`
  - the newest full backup is older than `--retention-age-to-full`

EDB

--retention-full

```
$ check_pgbackrest --stanza=my_stanza \
  --service=retention --retention-full=1

BACKUPS_RETENTION OK - backups policy checks ok |
  full=1 diff=0 incr=0
  latest=full,20210304-100800F latest_age=163s
```

EDB

--output=human

```
$ check_pgbackrest --stanza=my_stanza \
  --service=retention --retention-full=1 --output=human

Service         : BACKUPS_RETENTION
Returns         : 0 (OK)
Message         : backups policy checks ok
Long message    : full=1
Long message    : diff=0
Long message    : incr=0
Long message    : latest=full,20210304-100800F
Long message    : latest_age=3m16s
```

Multiple arguments together

```
$ check_pgbackrest --stanza=my_stanza \
  --service=retention --retention-full=1 --output=human \
  --retention-age=24h --retention-age-to-full=7d

Service         : BACKUPS_RETENTION
Returns         : 0 (OK)
Message         : backups policy checks ok
Long message    : full=1
Long message    : diff=0
Long message    : incr=0
Long message    : latest=full,20210304-100800F
Long message    : latest_age=3m30s
Long message    : latest_full=20210304-100800F
Long message    : latest_full_age=3m30s
```

Archives

- `info` command
  - shows the oldest (min) archive and the most recent one (max)
  - doesn’t check if all the archives in between are really on the disk
  - …
- `verify` command is still experimental

Archives (2)

```
$ check_pgbackrest --stanza=my_stanza  --service=archives

WAL_ARCHIVES OK - 4 WAL archived, latest archived since 3m21s |
  latest_archive_age=201s num_archives=4
```

EDB

--output=human

```
$ check_pgbackrest --stanza=my_stanza --service=archives --output=human

Service         : WAL_ARCHIVES
Returns         : 0 (OK)
Message         : 4 WAL archived
Message         : latest archived since 3m40s
Long message    : latest_archive_age=3m40s
Long message    : num_archives=4
...
Long message    : latest_archive=000000030000000000000000E
Long message    : latest_bck_archive_start=000000030000000000000000B
Long message    : latest_bck_type=full
...
```

Oops (1)

```
$ rm -rf [...]/archive/my_stanza/13-1/0000000300000000/000000030000000000000000C-*
$ pgbackrest info --stanza=my_stanza
stanza: my_stanza
    status: ok
    cipher: none

    db (current)
        wal archive min/max (13): 000000030000000000000000B/000000030000000000000000E
...

$ pgbackrest verify --stanza=my_stanza
P00   INFO: Results:
  archiveId: 13-1, total WAL checked: 3, total valid WAL: 3
    missing: 0, checksum invalid: 0, size invalid: 0, other: 0
  backup: 20210304-100800F, status: valid, total files checked: 939, total valid files: 939
    missing: 0, checksum invalid: 0, size invalid: 0, other: 0
```

pgBackRest doesn’t report any error!

Oops (2)

```
$ check_pgbackrest --stanza=my_stanza --service=archives --output=human
Service        : WAL_ARCHIVES
Returns        : 2 (CRITICAL)
Message        : wrong sequence, 1 missing file(s) (...)
Long message   : latest_archive_age=5m49s
Long message   : num_archives=3
Long message   : num_missing_archives=1
Long message   : oldest_missing_archive=00000003000000000000000C
Long message   : latest_missing_archive=00000003000000000000000C
...
```

- WARNING if missing archive < `latest_bck_archive_start`
  - CRITICAL otherwise

Release Radar

https://pgbackrest.org/release.html

2.21 (January 15, 2020)

- C migration complete
- Minor bug fixes and improvements
  - 2.22 (January 21, 2020)
  - 2.23 (January 27, 2020)

2.24 (February 25, 2020)

- Auto-select backup set for time target

2.25 (March 26, 2020)

- LZ4 Compression Support
- `--dry-run` option for the expire command

2.26 (April 20, 2020)

- Non-blocking TLS
- TCP keep-alive options

2.27 (May 26, 2020)

- Time-based retention for full backup
- Expire a specific backup set
- **Zstandard** and **bzip2** compression support

2.28 (July 20, 2020)

- Azure support for repository storage
- `expire-auto` option

2.29 (August 31, 2020)

- Automatically retrieve temporary S3 credentials on AWS instances
- `archive-mode` option to disable archiving on restore

2.30 (October 5, 2020)

- PostgreSQL 13 support

2.31 (December 7, 2020)

- `pg-database` option
- Report page checksums errors in info command text output
  - only when `--set` is provided

2.32 (February 8, 2021)

- `repo-ls` and `repo-get` commands

What’s next?

- Multi-repository support
- GCS driver

Multi-repository

- `archive-push` to multiple repositories
  - very useful for fault-tolerance using `archive-async`
- `backup` to specified location
  - allow multiple retention policies
- `archive-get` able to find archives and history files across all defined repositories
  - very handy in case of gaps - missing archives

Where

- official website: https://pgbackrest.org
- code: https://github.com/pgbackrest/pgbackrest
- rpm and deb: in the PGDG repositories!

User-guides

- official: https://pgbackrest.org/user-guide.html
- blog: https://pgstef.github.io/

Conclusion

- pgBackRest is a powerful tool
  - with a lot of features and possibilities
- don’t forget *Schrödinger’s Law of Backups*
  - monitor backups and archiving system

Questions?

Thank you for your attention!