Save your data with pgBackRest

DALIBO
L'expertise PostgreSQL

12 July 2018

Stefan Fercot

true

Save your data with pgBackRest

TITRE : Save your data with pgBackRest
SOUS-TITRE :

DATE: 12 July 2018

WHO AM I?

- Stefan Fercot
- aka. pgstef
- PostgreSQL user since 2010
- involved in the community since 2016
- @dalibo since 2017

DALIBO

- Services

Support	Training	Advice

- Based in France
- Contributing to PostgreSQL community
- We're hiring!

INTRODUCTION

Ever heard of Point-in-time recovery? pgBackRest is an awesome tool to handle backups, restores and even helps you build streaming replication !

This talk will introduce the tool, its basic features and how to use it.

WRITE AHEAD LOG (WAL)

- transactions written sequentially
  - considered committed when flushed to disk
- WAL replay after a crash
  - make the database consistent

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.

POINT-IN-TIME RECOVERY (PITR)

- combine
  - file-system-level backup
  - continuous archiving of the WAL files
- restore the file-system-level backup and replay the archived WAL files
- not mandatory to replay the WAL entries all the way to the end

https://www.postgresql.org/docs/current/static/continuous-archiving.html

---

HOW TO DO IT? (1)

- postgresql.conf
  - archive_mode
  - archive_command
  - archive_timeout

HOW TO DO IT? (2)

- pg_start_backup()
- file-system-level backup : tar, rsync,...
- pg_stop_backup()

HOW TO DO IT? (3)

- so many ways to get that wrong
  - what about backup retention?
  - ...

WHAT IS PGBACKREST?

- aims to be a simple, reliable backup and restore system
- developed by David Steele & Stephen Frost (CrunchyData)
- Perl & C
- MIT license

MAIN FEATURES

- custom protocol
  - local or remote operation (via SSH)
- multi-process
- full/differential/incremental backup
- backup rotation and archive expiration
- parallel, asynchronous WAL push and get
- Amazon S3 support
- encryption
- ...

INSTALLATION

- *Use the PGDG repository, Luke!*
  - yum / apt-get install pgbackrest
  - 1 package with some (~ 40) dependencies

CONFIGURATION

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

```
[global]
repo1-path=/var/lib/pgsql/10/backups
log-level-console=info

[my_stanza]
pg1-path=/var/lib/pgsql/10/data
```

true

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

GLOBAL SECTION - SOME EXAMPLES

- **process-max**: max processes to use for compress/transfer
- **repo1-path**: path where backups and archive are stored
- **repo1-cipher-pass**: passphrase used to encrypt/decrypt files of the repository
- **repo1-retention-full**: number of full backups to retain
- **repo1-retention-diff**: number of differential backups to retain
- **repo1-host**: repository host when operating remotely via SSH
- **repo1-host-user**: repository host user when repo1-host is set
- …

To set encryption :

```
# openssl rand -base64 48
```

```
repo1-cipher-pass=xxx
repo1-cipher-type=aes-256-cbc
```

STANZA SECTION

- **pg1-path**: PostgreSQL data directory
- **pg1-port**
- any global configuration can be overridden
- …

POSTGRESQL CONFIGURATION

- `archive_mode = on`
- `wal_level = replica`
- `archive_command = 'pgbackrest --stanza=my_stanza archive-push %p'`

INITIALIZE THE STANZA

- create the stanza

```
# sudo -u postgres pgbackrest --stanza=my_stanza stanza-create
```

- check the configuration and if archiving is working

```
# sudo -u postgres pgbackrest --stanza=my_stanza check
```

The stanza-create command must be run on the host where the repository is located to initialize the stanza.

The check command validates that pgBackRest and the archive_command setting are configured correctly for archiving and backups. It detects misconfigurations, particularly in archiving, that result in incomplete backups because required WAL segments did not reach the archive.

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.

PERFORM A BACKUP

```
# sudo -u postgres pgbackrest --stanza=my_stanza --type=full backup
```

- supported types: incr, diff, full

Example:

```
# sudo -u postgres pgbackrest --stanza=my_stanza --type=full --no-log-timestamp backup
P00   INFO: backup command begin 2.03: --log-level-console=info
--no-log-timestamp --pg1-path=/var/lib/pgsql/10/data
--repo1-path=/var/lib/pgsql/10/backups --stanza=my_stanza --type=full
P00   INFO: execute non-exclusive pg_start_backup() with label
"pgBackRest backup started at 2018-06-20 18:05:24": backup begins after the next regul
```

```
P00   INFO: backup start archive = 000000010000000000000003, lsn = 0/3000060
P00   INFO: full backup size = 23.2MB
P00   INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments to ar
P00   INFO: backup stop archive = 000000010000000000000003, lsn = 0/3000130
P00   INFO: new backup label = 20180620-180524F
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin
P00   INFO: option 'repo1-retention-archive' is not set - archive logs will not be exp
P00   INFO: expire command end: completed successfully
```

BACKUP INFORMATION

```
# sudo -u postgres pgbackrest --stanza=my_stanza info
stanza: my_stanza
    status: ok

    db (current)
        wal archive min/max (10-1):
        000000010000000000000001 / 000000010000000000000003

        full backup: 20180620-180524F
            timestamp start/stop: 2018-06-20 18:05:24 / 2018-06-20 18:05:39
            wal start/stop: 000000010000000000000003 / 000000010000000000000003
            database size: 23.2MB, backup size: 23.2MB
            repository size: 2.7MB, repository backup size: 2.7MB
```

RESTORE A BACKUP

```
# sudo -u postgres pgbackrest --stanza=ma_stanza restore
```

- options
  - `--delta`
  - `--target`
  - …
- `--delta` : by default the PostgreSQL data and tablespace directories are expected to be present but empty. This option performs a delta restore using checksums

- `--target` : defines the recovery target when –type is name, xid, or time.

DEMO - POINT-IN-TIME RECOVERY

STEP 1: BACKUP

```
# sudo -u postgres pgbackrest --stanza=my_stanza --type=full backup
P00   INFO: backup command begin 2.03:
--pg1-path=/var/lib/pgsql/10/data --repo1-path=/var/lib/pgsql/10/backups
--stanza=my_stanza --type=full
P00   INFO: execute non-exclusive pg_start_backup()
with label "pgBackRest backup started at 2018-06-28 16:53:20":
backup begins after the next regular checkpoint completes
P00   INFO: backup start archive = 000000020000000000000019, lsn = 0/19000060
P00   INFO: full backup size = 23.2MB
P00   INFO: execute non-exclusive pg_stop_backup() and
wait for all WAL segments to archive
P00   INFO: backup stop archive = 000000020000000000000019, lsn = 0/19000130
P00   INFO: new backup label = 20180628-165320F
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin
P00   INFO: expire command end: completed successfully
```

STEP 2: CREATE IMPORTANT DATA

```
# sudo -iu 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)
```

STEP 3: GET CURRENT TIME

```
# sudo -iu postgres psql -Atc "select current_timestamp"
2018-06-28 16:54:22.221035+02
```

STEP 4: OOOPS...

```
# sudo -iu postgres psql -c " \
begin; \
drop table important_table; \
commit; \
select * from important_table;"
ERROR:  relation "important_table" does not exist
LINE 1: ...drop table important_table; commit; select * from important_...
                                                             ^
```

STEP 5: STOP POSTGRESQL

```
# sudo -iu postgres psql -c "select pg_switch_wal()"
# systemctl stop postgresql-10.service
```

STEP 6: RESTORE

```
# sudo -u postgres pgbackrest --stanza=my_stanza --delta
--type=time --target="2018-06-28 16:54:22.221035+02" --target-action=promote
restore
P00   INFO: restore command begin 2.03: --delta
--pg1-path=/var/lib/pgsql/10/data --repo1-path=/var/lib/pgsql/10/backups
--stanza=my_stanza --target="2018-06-28 16:54:22.221035+02"
--target-action=promote --type=time
P00   INFO: restore backup set 20180628-165320F
P00   INFO: remove invalid files/paths/links from /var/lib/pgsql/10/data
P00   INFO: cleanup removed 22 files
P00   INFO: write /var/lib/pgsql/10/data/recovery.conf
```

```
P00   INFO: restore global/pg_control
(performed last to ensure aborted restores cannot be started)
P00   INFO: restore command end: completed successfully
```

STEP 7: CHECK RECOVERY.CONF

```
# cat /var/lib/pgsql/10/data/recovery.conf
restore_command = 'pgbackrest --stanza=my_stanza archive-get %f "%p"'
recovery_target_time = '2018-06-28 16:54:22.221035+02'
recovery_target_action = 'promote'
```

STEP 8: START POSTGRESQL

```
# systemctl start postgresql-10.service
# cat /var/lib/pgsql/10/data/log/*
LOG:  recovery stopping before commit of transaction 561,
time 2018-06-28 16:54:59.481247+02
LOG:  redo done at 0/1A01E590
LOG:  last completed transaction was at log time 2018-06-28 16:54:13.370025+02
```

STEP 9: CHECK THE DATA

```
# sudo -iu postgres psql -c "select * from important_table"
    message
----------------
 Important Data
(1 row)
```

STREAMING REPLICATION

- /etc/pgbackrest.conf
  - recovery-option=primary_conninfo=db.mydomain.com
  - recovery-option=standby_mode=on
  - ...

https://pgbackrest.org/user-guide.html#replication/streaming

See http://www.postgresql.org/docs/X.X/static/recovery-config.html for details on recovery.conf options (replace X.X with your PostgreSQL version). This option can be used multiple times.

CONCLUSION

- test it,
- use it!

WHERE

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

THANK YOU FOR YOUR ATTENTION!