Save your data with pgBackRest

Stefan Fercot

12 July 2018

DALIBO
L'expertise PostgreSQL

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

Write Ahead Log (WAL)

- transactions written sequentially
  - considered committed when flushed to disk
- WAL replay after a crash
  - 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

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
```

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

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
- …

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
```

Perform a Backup

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

- supported types: incr, diff, full

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`
  - ...

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
  - …

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!