FOSDEM PostgreSQL Devroom

pgBackRest Schrödinger's backups

2 February 2020

Stefan Fercot

pgBackRest Schrödinger's backups

FOSDEM PostgreSQL Devroom

TITRE : pgBackRest Schrödinger's backups
SOUS-TITRE : FOSDEM PostgreSQL Devroom

DATE: 2 February 2020

WHO AM I?

- Stefan Fercot
- aka. pgstef
- https://pgstef.github.io
- PostgreSQL user since 2010
- pgBackRest fan
- @dalibo since 2017

DALIBO

- Services

Support	Training	Advice

- Based in France
- Contributing to PostgreSQL community

INTRODUCTION

- pgBackRest
  - what is it ?
  - typical use cases
- check_pgbackrest
  - what do we have to monitor and how to do it ?

pgBackRest is a powerful backup and restore tool, right? Its “info” command is very useful to check your backups state… but…

Are you sure the existing backups match your retention policy? Are you sure you have all the WAL archives you need to restore it?

check_pgbackrest is a tool to help you watch your backups and archives.

It enables you to: * count the number of full backups; * check the age of the newest backup; * make sure that all the archived WAL segments are on disk; * …

During this talk, we’ll see some typical pgBackRest uses and how to monitor them: * backups stored locally, * on a pgBackRest remote host, * in a S3 bucket.

We’ll here show how to use the tool manually or with a Nagios-compatible monitoring system.

WRITE-AHEAD LOG (WAL)

- transactions written sequentially
  - COMMIT when data are 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.

https://www.postgresql.org/docs/current/wal-intro.html

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

---

PGBACKREST

- aims to be a simple, reliable backup and restore system
- written in C (since version 2.21)
- 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
- ...

https://pgbackrest.org

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 2.23: ...
P00   INFO: stanza-create command end: completed successfully

$ pgbackrest --stanza=my_stanza check
P00   INFO: check command begin 2.23: ...
P00   INFO: WAL segment 000000010000000000000001 successfully archived to ...
P00   INFO: check command end: completed successfully
```

FULL BACKUP

```
$ pgbackrest --stanza=my_stanza --type=full backup
P00   INFO: backup command begin 2.23: ...
P00   INFO: execute non-exclusive pg_start_backup() with label "...":
backup begins after the requested immediate checkpoint completes
P00   INFO: backup start archive = 000000010000000000000003, lsn = 0/3000060
P00   INFO: full backup size = 24.2MB
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: new backup label = 20200131-150158F
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.23: ...
P00   INFO: expire command end: completed successfully
```

DIFFERENTIAL BACKUP

```
$ pgbackrest --stanza=my_stanza --type=diff backup
P00   INFO: backup command begin 2.23: ...
P00   INFO: last backup label = 20200131-150158F, version = 2.23
P00   INFO: execute non-exclusive pg_start_backup() with label "...":
backup begins after the requested immediate checkpoint completes
P00   INFO: backup start archive = 000000010000000000000005, lsn = 0/5000028
P00   INFO: diff backup size = 24.2MB
P00   INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments
to archive
P00   INFO: backup stop archive = 000000010000000000000005, lsn = 0/5000138
P00   INFO: new backup label = 20200131-150158F_20200131-150245D
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.23: ...
P00   INFO: expire command end: completed successfully
```

INCREMENTAL BACKUP

```
$ pgbackrest --stanza=my_stanza --type=incr backup
P00   INFO: backup command begin 2.23: ...
P00   INFO: last backup label = 20200131-150158F_20200131-150245D, version = 2.23
P00   INFO: execute non-exclusive pg_start_backup() with label "...":
backup begins after the requested immediate checkpoint completes
P00   INFO: backup start archive = 000000010000000000000007, lsn = 0/7000028
P00   INFO: incr backup size = 24.2MB
P00   INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments
to archive
P00   INFO: backup stop archive = 000000010000000000000007, lsn = 0/7000138
P00   INFO: new backup label = 20200131-150158F_20200131-150410I
P00   INFO: backup command end: completed successfully
P00   INFO: expire command begin 2.23: ...
P00   INFO: expire command end: completed successfully
```

INFO COMMAND

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

db (current)
  wal archive min/max (12-1): 000000010000000000000003/000000010000000000000007

  full backup: 20200131-150158F
    timestamp start/stop: 2020-01-31 15:01:58 / 2020-01-31 15:02:14
    wal start/stop: 000000010000000000000003 / 000000010000000000000003
    database size: 24.2MB, backup size: 24.2MB
    repository size: 2.9MB, repository backup size: 2.9MB
...
```

TYPICAL USE CASES

- Local storage
- Remote storage
- S3 storage

LOCAL STORAGE (1)

```
# /etc/pgbackrest.conf
[global]
repo1-type=cifs
repo1-path=/var/lib/pgbackrest
repo1-retention-full=1
process-max=2
log-level-console=warn
log-level-file=info
start-fast=y
delta=y

[my_stanza]
pg1-path=/var/lib/pgsql/12/data
```

```
--REPO-TYPE

Repository Type Option (--repo-type)
Type of storage used for the repository.
The following repository types are supported:

    cifs - Like posix, but disables links and directory fsyncs
    posix - Posix-compliant file systems
    s3 - AWS Simple Storage Service
```

Be careful with CIFS !

https://www.postgresql.org/docs/current/creating-cluster.html#CREATING-CLUSTER-NFS

REMOTE STORAGE (2)

- **pgsql-srv**

```
[global]
repo1-host=backup-srv
repo1-host-user=postgres
...
[my_stanza]
pg1-path=/var/lib/pgsql/12/data
```

- **backup-srv**

```
[global]
repo1-path=/var/lib/pgbackrest
repo1-retention-full=1
...
[my_stanza]
pg1-host=pgsql-srv
pg1-path=/var/lib/pgsql/12/data
```

COMMAND EXECUTION WITH REMOTE STORAGE

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

S3 STORAGE (3)

```
[global]
repo1-path=/repo1
repo1-type=s3
repo1-s3-endpoint=minio.local
repo1-s3-bucket=pgbackrest
repo1-s3-verify-tls=n
repo1-s3-key=accessKey
repo1-s3-key-secret=***
repo1-s3-region=eu-west-3
```

```
...
[my_stanza]
pg1-path=/var/lib/pgsql/12/data
```

Configuration for MinIO test case (Amazon S3 api).

- `repo1` is a directory inside `pgbackrest` bucket;
- `repo1-s3-verify-tls` disabled for the test case.

CHECK_PGBACKREST

https://github.com/dalibo/check_pgbackrest

INSTALLATION

```
$ sudo yum install nagios-plugins-pgbackrest-1.7-1.noarch.rpm
#==============================================================================
Package                                 Repository
#==============================================================================
Installing:
 nagios-plugins-pgbackrest              nagios-plugins-pgbackrest-1.7-1.noarch

Installing for dependencies:
 nagios-common                          epel
 nagios-plugins                         epel
 perl-IO-Tty                            base
 perl-JSON                              base
 perl-Net-SFTP-Foreign                  epel
 perl-Sort-Key                          epel
#==============================================================================
```

```
$ /usr/lib64/nagios/plugins/check_pgbackrest --version
check_pgbackrest version 1.7, Perl 5.16.3
```

`epel-release` needs to be installed.

https://github.com/dalibo/check_pgbackrest/releases

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`

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

`--RETENTION-FULL`

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

BACKUPS_RETENTION OK - backups policy checks ok |
  full=1 diff=1 incr=1
  latest=incr,20200131-150158F_20200131-150410I latest_age=120s
```

`--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=1
Long message   : incr=1
Long message   : latest=incr,20200131-150158F_20200131-150410I
Long message   : latest_age=2m24s
```

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=1
Long message    : incr=1
Long message    : latest=incr,20200131-150158F_20200131-150410I
Long message    : latest_age=2m47s
Long message    : latest_full=20200131-150158F
Long message    : latest_full_age=5m
```

WHATEVER THE BACKUPS LOCATION ?

Only based on `pgbackrest info` output!

WALS ARCHIVES CHECK

- The `pgbackrest 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
  - doesn't give the age of the most recent archive (until 2.21)
  - ...

HOW ?

- 000000010000000200000003
  - 00000001 : time-line
  - 00000002 : wal
  - 00000003 : segment
- initdb `--wal-segsize=size`
  - since v11
  - by default 16MB
  - 256 segments per wal (>= v9.3)

Check for each segment, wal after wal!

Since version 9.3, segment names are from 00000000 to 000000FF. Previously, to 000000FE.

TIME-LINE SWITCH

```
# 00000002.history
1 0/9000000 no recovery target specified
```

- Segment : 000000010000000000000009

EXAMPLE : OOPS (1)

```
$ createdb bench
$ pgbench -i -s 100 bench
$ pgbackrest info --stanza=my_stanza
stanza: my_stanza
status: ok
cipher: none

db (current)
  wal archive min/max (12-1): 000000010000000000000003/000000010000000000000004D
...
```

LOCAL STORAGE (1)

```
$ check_pgbackrest --stanza=my_stanza
  --service=archives --repo-path=/var/lib/pgbackrest/archive

WAL_ARCHIVES OK - 81 WAL archived, latest archived since 1m27s |
  latest_archive_age=87s num_archives=81
```

```
--OUTPUT=HUMAN

$ check_pgbackrest --stanza=my_stanza
  --service=archives --repo-path=/var/lib/pgbackrest/archive --output=human
Service        : WAL_ARCHIVES
Returns        : 0 (OK)
Message        : 81 WAL archived
Message        : latest archived since 1m59s
Long message   : latest_archive_age=1m59s
Long message   : num_archives=81
Long message   : archives_dir=/var/lib/pgbackrest/archive/my_stanza/12-1
Long message   : min_wal=000000010000000000000003
Long message   : max_wal=000000010000000000000053
Long message   : oldest_archive=000000010000000000000003
Long message   : latest_archive=000000010000000000000053
Long message   : latest_bck_archive_start=000000010000000000000007
Long message   : latest_bck_type=incr
```

Oops (2)

```
$ rm -rf [...]/archive/my_stanza/12-1/0000000100000000/000000010000000000000001*
$ pgbackrest info --stanza=my_stanza
stanza: my_stanza
status: ok
cipher: none

db (current)
  wal archive min/max (12-1): 000000010000000000000003/000000010000000000000053
...
```

pgBackRest doesn't report any error!

OOPS (3)

```
$ check_pgbackrest --stanza=my_stanza
  --service=archives --repo-path=/var/lib/pgbackrest/archive --output=human
Service        : WAL_ARCHIVES
Returns        : 2 (CRITICAL)
Message        : wrong sequence or missing file @ '000000010000000000000010'
...
Message        : wrong sequence or missing file @ '00000001000000000000001F'
...
Long message   : min_wal=000000010000000000000003
Long message   : max_wal=000000010000000000000053
Long message   : oldest_archive=000000010000000000000003
Long message   : latest_archive=000000010000000000000053
Long message   : latest_bck_archive_start=000000010000000000000007
Long message   : latest_bck_type=incr
```

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

REMOTE STORAGE (2)

- from **pgsql-srv**
  - consider remote storage with `--repo-host` and `--repo-host-user`
  - `Net::SFTP::Foreign`
- from **backup-srv**
  - consider local storage

https://metacpan.org/pod/Net::SFTP::Foreign

`--REPO-HOST`, `--REPO-HOST-USER`

```
$ check_pgbackrest --stanza=my_stanza
  --service=archives --repo-path=/var/lib/pgbackrest/archive
  --repo-host="backup-srv" --repo-host-user=postgres

WAL_ARCHIVES OK - 4 WAL archived, latest archived since 25m30s |
  latest_archive_age=25m30s num_archives=4
```

S3 STORAGE (3)

- Get `repo1-s3-key` and `repo1-s3-key-secret`
  - from pgBackRest configuration
  - `Config::IniFiles`
- Connection API `Net::Amazon::S3`
- https://metacpan.org/pod/Config::IniFiles
- https://metacpan.org/pod/Net::Amazon::S3

`--REPO-S3`, `--REPO-S3-OVER-HTTP`

```
$ check_pgbackrest --stanza=my_stanza
  --service=archives --repo-path=/repo1/archive
  --repo-s3 --repo-s3-over-http

WAL_ARCHIVES OK - 4 WAL archived, latest archived since 1m7s |
  latest_archive_age=1m7s num_archives=4
```

GO FURTHER

- `--ignore-archived-before`
  - ignores the archived WALs generated **before** the provided interval
  - used to only check the latest archives
- `--ignore-archived-after`
  - ignores the archived WALs generated **after** the provided interval
  - used under heavy archiving load
- `--latest-archive-age-alert`
  - defines the max age of the latest archived WAL before raising the alert

```
Use the --ignore-archived-before argument to ignore the archived WALs generated before
interval. Used to only check the latest archives.

Use the --ignore-archived-after argument to ignore the archived WALs generated after t

The --latest-archive-age-alert argument defines the max age of the latest archived WAL
before raising a critical alert.
```

MONITORING SYSTEMS

- Output format compatible with many monitoring systems
  - Nagios
  - Naemon
  - Icinga
  - ...

RETENTION (LOCAL STORAGE)

```
object CheckCommand "by_ssh_pgbackrest_retention" {
  import "by_ssh"
  vars.by_ssh_command = "/usr/lib64/nagios/plugins/check_pgbackrest
  --stanza=\$stanza$ --service=retention
  --retention-full=\$retention_full$ --prefix='\$prefix$'"
}

object Service "pgbackrest_retention" {
  import "generic-service"
  host_name = "pgsql-srv"
  check_command = "by_ssh_pgbackrest_retention"
  vars.by_ssh_logname = "accessed_by_ssh"

  vars.stanza = "my_stanza"
  vars.retention_full = 1
  vars.prefix = "sudo -u postgres"
}
```

ARCHIVES (LOCAL STORAGE)

```
object CheckCommand "by_ssh_pgbackrest_archives" {
  import "by_ssh"
  vars.by_ssh_command = "/usr/lib64/nagios/plugins/check_pgbackrest
  --stanza=\$stanza$ --service=archives
  --repo-path=\$repo_path$ --prefix='\$prefix$'"
}

object Service "pgbackrest_archives" {
  import "generic-service"
```

```
  host_name = "pgsql-srv"
  check_command = "by_ssh_pgbackrest_archives"
  vars.by_ssh_logname = "accessed_by_ssh"

  vars.stanza = "my_stanza"
  vars.repo_path = "/var/lib/pgbackrest/archive"
  vars.prefix = "sudo -u postgres"
}
```

TESTS USING VAGRANT

```
[check_pgbackrest]$ ls test
Makefile  perf  provision  README.md  regress  ssh  Vagrantfile
```

- vm.box : CentOS 7
- vm.provider : libvirt

TEST CASE 1

- pgBackRest configured to backup and archive locally on a CIFS mount point
- `icinga-srv` executes `check_pgbackrest` commands through SSH with Icinga 2
- `pgsql-srv` stores the PostgreSQL cluster to backup with pgBackRest
- `backup-srv` stores the CIFS share

TEST CASE 2

- pgBackRest configured to backup and archive remotely
- `icinga-srv` executes `check_pgbackrest` commands through SSH with Icinga 2
- `pgsql-srv` stores the PostgreSQL cluster to backup with pgBackRest
- `backup-srv` stores the pgBackRest backups and archives
- pgBackRest backups are used to build a standby PostgreSQL server (using *Streaming Replication*) on `backup-srv`

TEST CASE 3

- pgBackRest configured to backup and archive on a MinIO S3 bucket
- `icinga-srv` executes `check_pgbackrest` commands through SSH with Icinga 2
- `pgsql-srv` stores the PostgreSQL cluster to backup with pgBackRest
- `backup-srv` stores the MinIO server

EVOLUTION (1)

- Use the `pgbackrest ls` command
  - to get the archives list
  - `mtime` available since 2.21
  - need a `cat` command for the `.history` files

```
$ pgbackrest help ls
pgBackRest 2.23 - 'ls' command help

List paths/files in the repository.

This is intended to be a general purpose list function, but for now it only
works on the repository.

Command Options:

  --filter                        filter output with a regular expression
  --output                        output format [default=text]
  --recurse                       include all subpaths in output [default=n]
  --sort                          sort output ascending, descending, or none
                                  [default=asc]
```

EVOLUTION (2)

- Debian support
  - specific test cases
  - `.deb` package ?

CONTRIBUTION

- Any contribution is welcome!
  - GitHub issues
  - Bugs reports, use cases,...
- CHANGELOG examples
  - *Fix bad behavior on CIFS mount (reported by `renesepp`)*
  - *Add Amazon s3 support for archives service (Andrew E. Bruno)*
  - ...

CONCLUSION

- Use backup tools
- Don't trust their reports
- Try to restore your backups and…
  - monitor them!

QUESTIONS ?