PostgreSQL Backups the Modern Way

Nordic PGDay 2019
Copenhagen, Denmark

Magnus Hagander
magnus@hagander.net

Magnus Hagander

- Redpill Linpro
  - Infrastructure services
  - Principal database consultant
- PostgreSQL
  - Core Team member
  - Committer
  - PostgreSQL Europe

So, backups...

- Do you make them?

Backups

- Are *not* superseded by replication
- Or cloud
- Or containers
- ..

Backups

- Are **boring**
- But I'm glad you have them

Backups

- When did you last restore?

PostgreSQL backups

- Ok, enough generic
- What about backups in PostgreSQL?

Seen this before?

pg_dump options:

```
-Fc    = custom format
-Z     = compression
-j     = parallel
-a = data only, -s = schema only
-n = schema, -t = table
...
```

pg_dump

- Don't use for backups
  - Has other good usecases
- Too slow to restore
- Too much overhead
- No PITR
- Exceptions, of course

Physical backups

- Base backups
- With or without log archive
- Fast restore
- Full cluster only
- Platform specific

Base backups

```
#!/bin/bash
set -e

psql -U postgres -q "SELECT pg_start_backup('foo')"

tar cfz /backup/$(date +%Y%m%d).tar.gz /var/lib/pgsql/data

psql -U postgres -q "SELECT pg_stop_backup()"
```

Base backups

- So many ways to get that wrong
  - Spot one?

Base backups

- This used to be the only way
- Many scripts around that does it
- Most of those are broken...

pg_basebackup

- Base backup over replication protocol
- *Safe*
- Error handling and recovery
- For *most* cases
  - (we'll cover other options later)

pg_basebackup

```
#!/bin/bash
set -e

pg_basebackup -D /backup/$(date +%Y%m%d) -Ft
```

Needs replication

- Enabled by default in 10!
- Older versions:

```
wal_level = hot_standby
max_wal_senders = 10
```

```
local   replication     postgres        peer
```

Backup formats

- plain
  - Safe copy of data directory
  - Not good with multiple tablespaces
- tar
  - Destination still a directory
  - Each tablespace gets one file
    - base.tar

Transaction log

- WAL required to restore backup
- From beginning of backup to end
- In the log archive, right?

Including WAL

- Always use **-x** or **-X** to include WAL
  - Default in 10
- Makes backup *independently consistent*
  - With or without log archive
  - May back up WAL twice
- Use even with log archive!

Including WAL

-X fetch

- Fetches WAL at end of backup
- Can fail if WAL rotated

-X stream

- Replicates WAL over secondary connection
- Fewer failure scenarios
- Does not work with *tar* prior to version 10

-X none

- Turn off (10+ only)

Replication slots

- *pg_basebackup* can fall behind on < 10
- Use replication slot
  - Don't forget to drop!
- PostgreSQL 10 uses ephemeral slot

Backup compression

`pg_basebackup -Z`

- Compression happens in pg_basebackup
- Tar format only
- CPU usage
- Remote server?

Transfer compression

- SSL compression
  - Much harder these days
- ssh tunneling
  - Does not work with WAL

```
ssh mydbserver -c "pg_basebackup -Ft -D- -Z9 -Xnone" > backup.tgz
```

That's it!

- With that, you have backups
- That work
- And are (reasonably) safe

PITR

- Point in time recovery
- You all want it
- A bit more setting up

archive_command

- To use PITR, we use log archiving
- like this?

```
archive_command =
  'test ! -f /mnt/archivedir/%f && cp %p /mnt/archivedir/%f'
```

Don't do that!

pg_receivewal

- Runs on archive server
- Uses streaming replication
- Generates log archive

pg_receivewal

- More granular recovery
- Safe against server restarts
- Can follow timeline switches on master

pg_receivewal

- *Always* use with replication slot
  - As of 9.4
  - But we said modern..
- Backups *should* block

pg_receivewal

```
pg_receivewal -D /log/archive -h master -S backup
```

- Ensure it's restarted!

Backup retention

- How long to keep around?
- What granularity?
- ...

Backup retention

- Recovery needs:
  - Base backup
  - All WAL from start to end
  - All WAL from end to pitr
- (that's why we use -x!)

Backup retention

- `find` is often enough
- Delete logs older than X, base older than Y
  - Safe if -x was used!

```
#!/bin/bash

find /var/backups/basebackup -type f -mtime +30 -print0 |
   xargs -0 -r /bin/rm

find /var/backups/wal -type f -mtime +7 -print0 |
   xargs -0 -r /bin/rm
```

Not enough?

- Handles the simple cases
- But has limitations
- Particularly in management

Other tools

- pgBackRest
- Barman

pgBackRest

- Backup scheduling
- Log archiving
- Retention management
- Multi-server
- Restore shortcuts
- Obsessive validation

pgBackRest

- Developed by CrunchyData
- Perl
  - Moving to C
- MIT license
- ssh but not rsync

pgBackRest

- Custom protocol
- Parallel backups (compression)
- Full/Differential/Incremental
  - Segment based
- Delta restore

pgBackRest

- Validates checksums
- Checksums backups
  - Every time

pgBackRest

- No pg_receivewal support
  - Yet
- No Windows support
  - Yet

Barman

- Backup scheduling
- Log archiving
- Retention management
- Multi-server
- Restore shortcuts

Barman

- Developed by 2ndQuadrant
- Python
- GPLv3
- Primarily ssh+rsync
  - 1.6 learned about pg_receivewal
  - 2.0 learned about pg_basebackup
    - Before that, no (safe) concurrent backup support

What about

Enterprise product X

?

Enterprise product X

- *pg_basebackup*
  - Run as pre-backup command
  - Optionally clean up in post-backup
- *pgbackrest/barman*
  - To *disk*
  - Let backup software take it form there

Summary

Don't roll your own!

Don't roll your own

- Too many pitfalls
- Both base backups and archiving
- Backups are too important!

Don't roll your own

- Primary choice
  - Built-in (*pg_basebackup*)
  - If it's enough
- Secondary choice
  - pgBackRest
  - Barman
- Tertiary choice
  - Restart from top of slide

Thank you!

Magnus Hagander
magnus@hagander.net
@magnushagander
https://www.hagander.net/talks/