PostgreSQL backups in the age of The Cloud

Current state of PostgreSQL backup tools

Hannu Valtonen

PostgresConf US 2019 - New York

Speaker

@HannuValtonen

- VP Product, co-founder @ Aiven, a cloud company
- Previously: database/distributed systems consultant, SW architect
- PostgreSQL pghoard, pgmemcache, pglookout maintainer

Google Cloud Platform

amazon web services

Microsoft Azure

UpCloud

DigitalOcean

packet

Aiven

- *Your data cloud*
- Based in Helsinki and Boston
- 8 data engines now available in 6 clouds and 80 regions, virtual and bare metal instances
- Launched a fully-managed PostgreSQL service in 2016
- First to offer the latest PostgreSQL features in a managed service

Disclaimers

- Original author and one of the maintainers of PGHoard
- I've tried to make sure that the following content is factually correct but I may have misunderstood or gotten some bits wrong

Backups

Q: So why do people *care* about backups?

A: They don't.

They care very much about their ability to actually restore the data.

Recovery Point Objective (RPO)

- How much data can you lose if something breaks
  - Five minutes, an hour?
  - Ultimately a business decision
  - There may be tradeoffs that you may have to consider to stay within the objective
- It *may* be OK to lose some data in some environments

Recovery Time Objective (RTO)

- How long can it take for you to to recover
  - Ultimately a business decision
  - There may be tradeoffs that you have to do to stay within the objective
- Think about these before you get bitten by the need
- Measure and rehearse to see how long it takes
- Note that bottlenecks whether in IO or in anything else good to identify beforehand

Different ways of taking PostgreSQL backups

- Taking logical backups of your database
  - pg_dump/pg_dumpall with pg_restore
  - Snapshots in time
- File level backups
  - Basebackup + Write Ahead Log (WAL)
  - Plenty of different solutions
  - Point In Time Recovery (PITR)

pg_dump/pg_restore

- Simplest way of taking backups
  - People usually start out this way
  - Performs fairly well when using the -j option
    - Less well when you have just a couple of huge tables
  - Backups are logical snapshots in time
  - Logical dumps useful when you need to move the data to another system

Basebackup + WAL

- Complex to get right
- Physical backup
  - Take a basebackup from PostgreSQL data directory
  - Store all Write Ahead Log (WAL) files from the basebackup onwards
  - Supports restoration to any given point in time (PITR)
  - No conversion between physical format and logical representation
- Backups always consist of the whole cluster (all databases + tables)

Google Cloud Platform

amazon web services

Microsoft Azure

UpCloud

DigitalOcean

packet

Object vs Block Storage

- AWS S3 ~$0.03/GiB
- Google Cloud Storage ~$0.02/GiB
- Azure storage ~$0.03/GiB

- AWS EBS ~$0.10/GiB
- Google PD-SSD $0.17/GiB
- Azure Premium SSD ~same ballpark

Comparison of backup tools

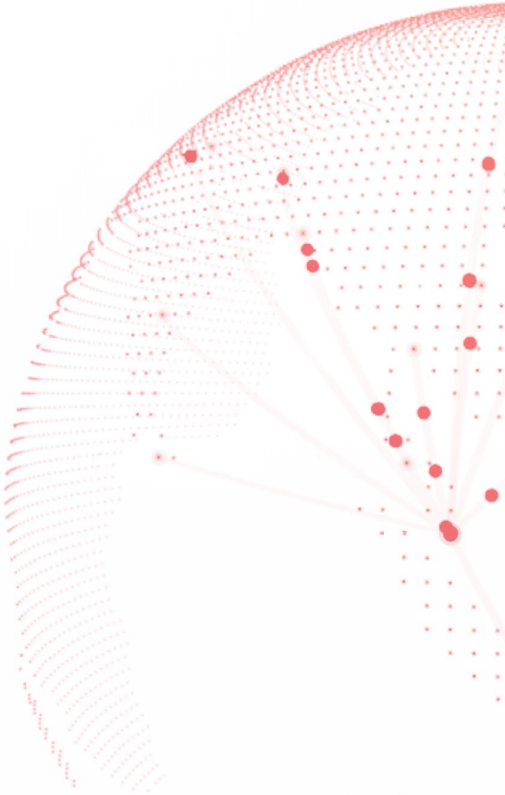
The ones built for the cloud, that is

The contenders:

- WAL-E (BSD-3-clause)
- WAL-G (Apache 2.0)
- PGHoard (Apache 2.0)
- PGBackrest (MIT)

Considerations:

- Monitoring, compression, encryption, ...



Compression

- Compression allows you to save on bandwidth and storage costs
  - Cost savings can be immense

- WAL-E uses LZOP (external command-line tool)
- WAL-G supports LZ4, LZMA and Brotli (zstd?) (called directly)
- PGHoard supports Snappy and LZMA (called directly)
- PGBackrest supports gzip (though able to run it in parallel)

Encryption

- You want to keep your data secure and confidential
  - Often a part of compliance requirements

- WAL-E uses GPG (external command-line tool)
- WAL-G uses GPG (external command-line tool)
- PGHoard has built-in encryption support (called directly)
- PGBackrest has built in encryption support (called directly)

Monitoring

- Crucial to make sure your backups are still working
  - Not much use if your backups haven't been taken in a while

- WAL-E has commands you can run to get backup state
- WAL-G doesn't at least document how you should do this
- PGHoard has extensive metrics available through statsd/Prometheus scraping/JSON dump
- PGBackrest has JSON dump capabilities and PostgreSQL monitoring functions

WAL backup via archive command

- Set a command-line to call in postgresql.conf
- Fairly slow since throughput constrained by a blocking command
  - If you assume say 100ms to backup a single WAL file, your throughput maximum is 10 files/s. (command overhead important to consider)
  - Similar to problems in the HTTP world
  - The reason why increasing the size of WAL segments been under discussion/work

WAL backup via replication connection

- Replication connection based
  - Allows streaming data to be uploaded in parallel greatly increasing throughput
- You need to monitor your backup status externally from PostgreSQL if you do this
- Note that you really want to use a replication slot if you're doing this
- Scales to close to wire-speed

Supported WAL backup modes

- Crucial to make sure your backups are still working
  - Not much use if your backups haven't been taken in a while

- WAL-E `archive_command` (written in Python)
- WAL-G `archive_command` (written in Go)
- PGHoard `pg_receivewal` or `archive_command` or *experimental* native replication
- PGBackrest `archive_command` (written in C)

Basebackup restoration

- Having good compression ratio/decompression speed useful
  - Can bottleneck even on CPU usage depending on compression algorithm used
- Parallelism in restoring basebackups speeds up things considerably
- Restoration of basebackup usually not a bottleneck
  - On i3 instances in AWS restore speeds north of 1 GiB/s are usual

Support for basebackup restores

- Speed helps with RTO
- Parallelism helps with speed

- WAL-E parallel restore supported
- WAL-G parallel restore supported
- PGHoard parallel restore supported
- PGBackrest parallel restore supported

WAL Restore

- Prefetching of WALs mandatory because of the fetch latency
- `restore_command` overhead noticeable if implemented in a slow language
  - Python3 invocation can cost 50ms+ (max 20 WAL files per second)
- PostgreSQL's single threaded replay of WALs a growing issue

- WAL-E `restore_command` (written in Python)
- WAL-G `restore_command` (written in Go)
- PGHoard `restore_command` (written in Go)
- PGBackrest `restore_command` (written in C)

Object storage support

- Typically you want to use the object storage that comes along with your cloud provider to reduce latency and save on costs
    - Price and performance vary among them, but not much

- WAL-E supports Azure storage, GCS, S3 and Swift
- WAL-G supports S3 and GCS
- PGHoard supports Azure storage, GCS, S3 and Swift
- PGBackrest supports S3

PostgreSQL checksums checks

- PostgreSQL has had support for checksums for a while now
  - A good time to calculate the checksums (since you're blowing the disk cache anyway) is during the backup operation since you're reading through all the data anyway

- WAL-E No support for this
- WAL-G No support for this
- PGHoard no support for this
- PGBackrest supports this

Github popularity

on 20th of March 2019

- It would be great if others used the same solution though
  - Github stars can give you at least some sort of idea on if others are using the same backup daemon

- WAL-E 2800
- WAL-G 1021
- PGHoard 881
- PGBackrest 406

Summary

- Many considerations in picking a backup daemon
- Make sure you know what's acceptable for RTO/RPO
  - And make sure the business side of your company understands the limitations as well
- Whichever solution you pick, you're probably better off than rolling your own
- Practice restoration periodically

Future

- Larger WAL segment sizes being considered
  - Driven by the size of `archive_command` overhead
- Single threaded WAL replay is a huge issue for write-heavy databases
  - We're hiring PostgreSQL engineers to work on this and other interesting things, check out https://aiven.io/careers
- Incremental basebackup support for doing delta basebackups
- WAL replay issues over multiple timelines
  - Should follow `.history` file definitions to speed things up

Questions?

Continue the discussion and grab some swag from our booth at the conference and try out Aiven with a free trial at https://aiven.io

aiven

Thanks!

https://aiven.io

@aiven_io

@HannuValtonen