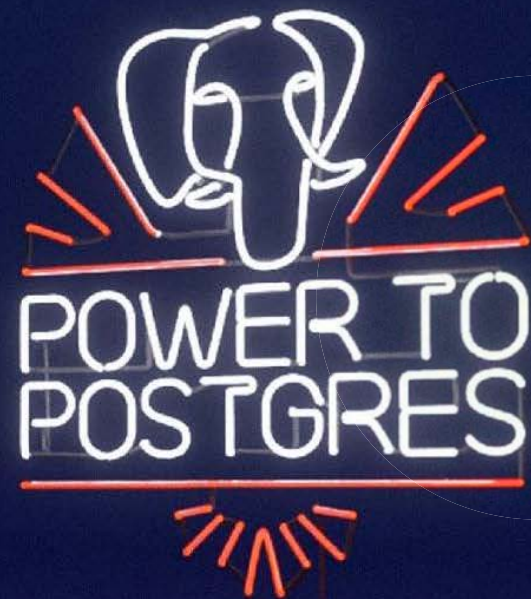


How can I be **ready** for the next
~~backup~~ **recovery** of a **Postgres**
database?

Postgres continuous backup and PITR with Barman

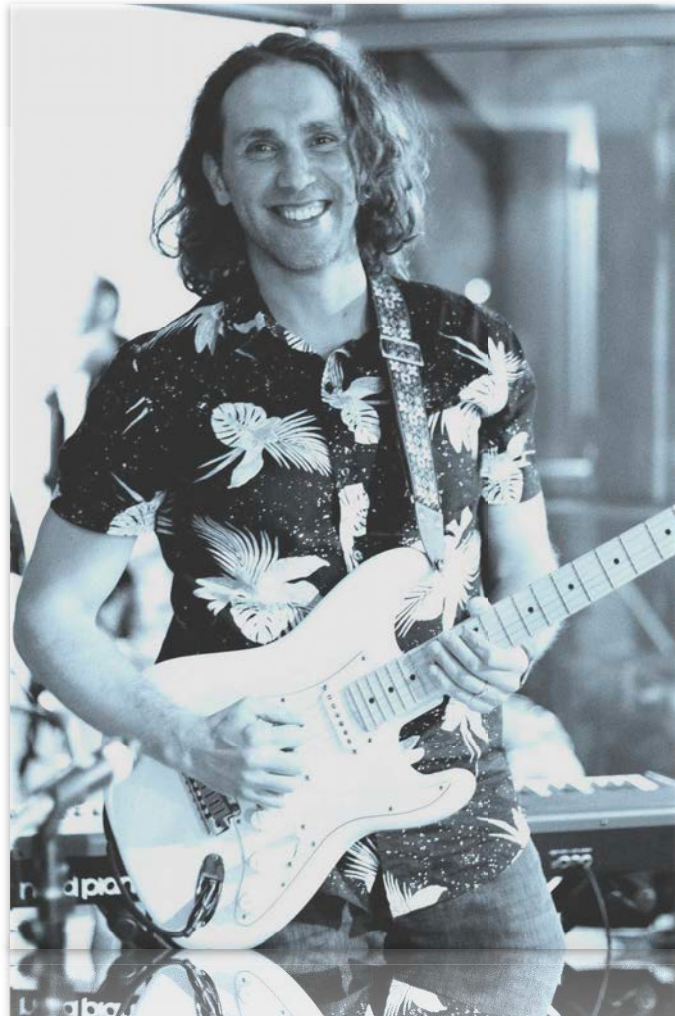
Gabriele Bartolini

9 December 2020 - Postgres Build 2020



About Gabriele Bartolini

- PostgreSQL user since ~2000
 - Community member since 2006
- 2ndQuadrant, from 2008 to 2020
 - Head of Global Support
 - Cloud Native Initiative Lead
 - Founding member of Barman
- **Now with EDB**



Today's menu

- About Barman
- Useful Patterns
- What lies ahead of us
- Final thoughts



Follow me:

@_GBartolini_

Tweet:

#pgbarman

#PostgresBuild2020

pgbarman.org



“Always design a thing by considering it in its next larger context – a chair in a room, a room in a house, a house in an environment, an environment in a city plan.”

Eliel Saarinen

About Barman



About Barman

Backup And Recovery MANager for PostgreSQL

- Disaster Recovery of PostgreSQL databases
 - Based on continuous physical backup
- First backup and recovery tool for PostgreSQL
 - Conceived in 2011, open sourced in 2012
 - **Goal: foster migrations from Oracle**
- Written in Python
- Open Source, distributed under GNU GPL 3



Barman
Backup and recovery
manager for PostgreSQL

Innovative concepts

First Introduced by Barman in the Postgres ecosystem

- Remote backup
- Remote recovery
- Multiple PostgreSQL servers
- Backup catalogue
- Independent WAL archive and hub
- WAL compression
- Multiple backup methods
- Incremental backup and recovery
- WAL streaming
- Retention policies
- Monitoring integration
- Hook scripts

Useful Patterns



#0 - Focus on business goals

Let goals drive your initiatives, not the technical means

- **Make decisions based on:**
 - Recovery Point Objectives (RPO)
 - Recovery Time Objectives (RTO)
- **Your context is unique, and you are the expert there**
- **Technological aspects are important**
 - But they are just the means to our business goals

#1 - Separate Barman from Postgres

Let Barman and Postgres run on different servers and storage

- Resilience
- Commodity storage
 - Physical servers with high capacity local disks
- Integration with standby servers
 - Zero data loss clusters



#2 - Avoid network disks (if you can)

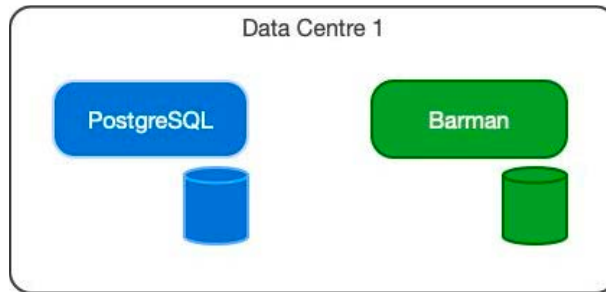
Some customers run Barman on NFS/CIFS/Ceph/...

- **File lock issues**
 - Place “barman_lock_directory” in the local Linux disk
- **Performance issues**
 - Latency
 - Throughput (bottleneck)
 - Shared storage (variability)
 - Accentuated by large databases

#3 - Favour locality over distance

Let Barman and Postgres be next to each other, in the same data centre

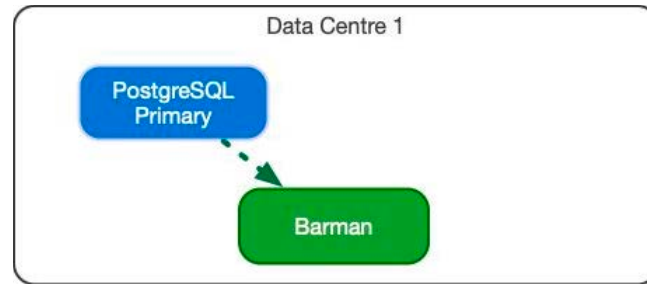
- Better resilience of the local cluster
 - Next tips will provide the whys and the hows
- Preferably dedicated network
- Data centre as a single point of failure?
 - Yes, but ...
 - ... don't let that scare you, it can be solved



#4 - WAL streaming

Let Postgres stream WAL records to Barman and reduce RPO

- Ensure all WALs are shipped
 - Rely on replication slots
 - Fully automated with “create_slot” option
- Asynchronous replication by default
- Synchronous replication if ...
 - You add a standby server



#5 - WAL fetching (get-wal)

Use Barman as an “infinite” hub of WAL files through “get-wal”

- On-demand remote fetching:
 - Standby servers
 - Recovery operations
- “barman-wal-restore”
- Parallel pre-fetch (performance boost)
- On the fly decompression
- Forget about configuring “wal_keep_segments” or “max_wal_size”



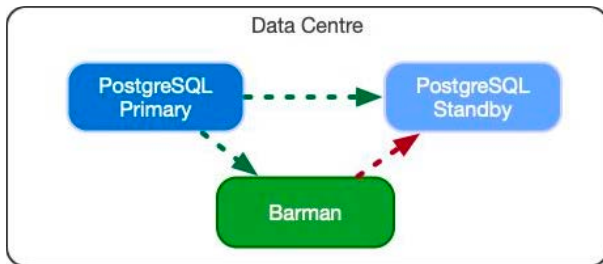
#6 - Compress your WALs

- Set “compression” option, globally
- Performed by the WAL archiver process
 - Invoked automatically by the “cron” command
- gzip is normally fine
- Supported algorithms:
 - bzip2
 - pigz
- There’s more: custom compression methods

#7 - Your PostgreSQL building block

The “Flux Capacitor”

1. Primary instance
2. Standby instance
3. Barman instance

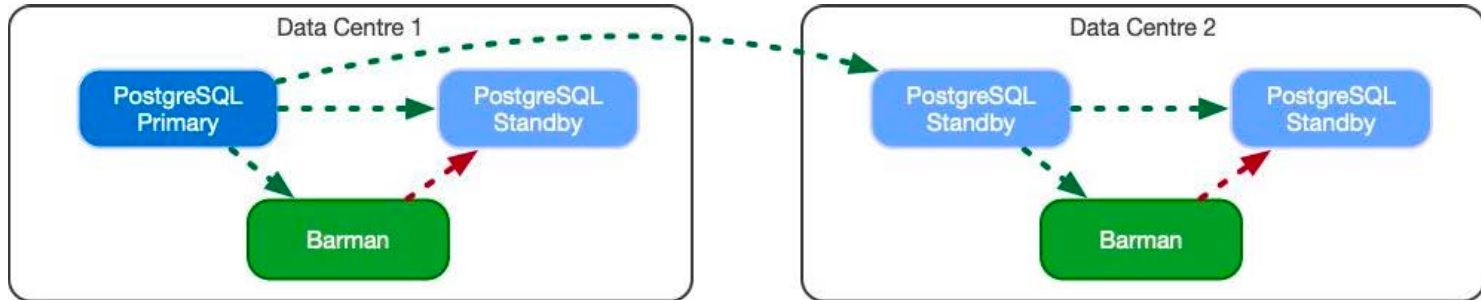


Food for thought:

- High Availability
- Disaster Recovery
- Local standby servers
 - You can always add more
 - Simple architecture
 - Yet very effective
- Symmetric architecture (next rule)

#8 - Multi-data centre architecture

Symmetric architecture is key



#9 - Encryption

Very important for GDPR compliance

Encrypt at rest

- PostgreSQL servers
- Backup servers
- Delegated to the operating system
 - LUKS
 - Volmetric

Food for thought:

- Secure connections
- Secure access to backup servers
- Encryption of exported backup files:
 - Tar files on tape
 - Tar files on Cloud object stores

#10 - Constrain your RPO

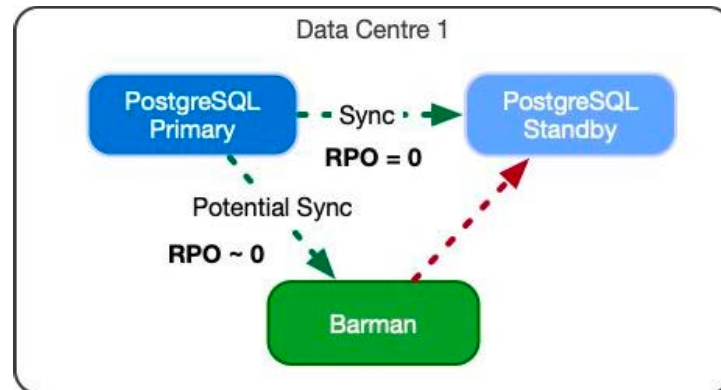
Clearly define your Recovery Point Objective in PostgreSQL

- By default it is approximately a WAL file
- **Recommendation: set it to a maximum of 5 minutes**
 - Use “archive_timeout”
 - For example “archive_timeout = 5min”
- **Important for geographical redundancy**
 - Beyond the local region

#11 - Reduce your RPO to 0

Trust PostgreSQL native synchronous streaming replication

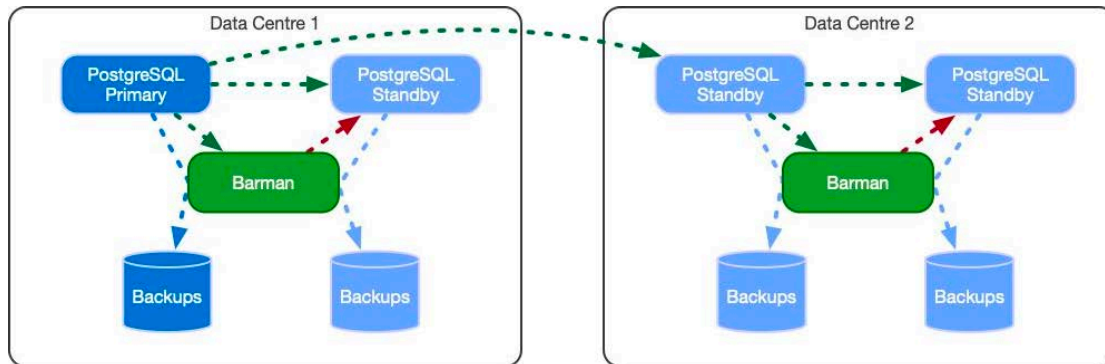
- Requirements:
 - A local standby
 - A local Barman
- Two options:
 - **Zero data loss standby**
 - Zero data loss backup



#12 - Backup from a standby

Available from PostgreSQL 9.6

- **Symmetric architecture**
- Off-load the primary
- Redundancy
 - Feature request: cluster awareness



#13 - Let “check” be your compass

“barman check” is the most critical command

- It guides you
 - Setup process
 - Problem solving
- Integration with alerting tools

#14 - Monitoring

Observability is fundamental in business continuity environments

- Barman resides on a Linux system
- That system must be under monitoring
 - Standard metrics
 - Disk usage
- “barman check --nagios all”

#15 - Weekly backups

Define the frequency of backups through cron

- Start with weekly backups
 - Evaluate daily backups if you require shorter RTO
- “barman backup all”

#16 - Retention policies

Key capability in disaster recovery solutions

- **Automatically purge old backups**
- **Retention policies based on:**
 - Redundancy (quantity)
 - Recovery window (time, Point of Recoverability)
- **Food for thought:**
 - Delete hook scripts

#17 - Use rsync/SSH for backups

Barman has an optimised copy algorithm based on rsync

- Enables incremental copy
 - Set “reuse_backup = link” if your file system supports hard links
- Enables parallel copy
 - Set “parallel_jobs” option
- Used for incremental and parallel recovery too
- Real example from a production customer:
 - Weekly backup of a **25TB** database takes 13 hours to complete
 - Reuses over 70% of the previous backup

#18 - Rely on object stores

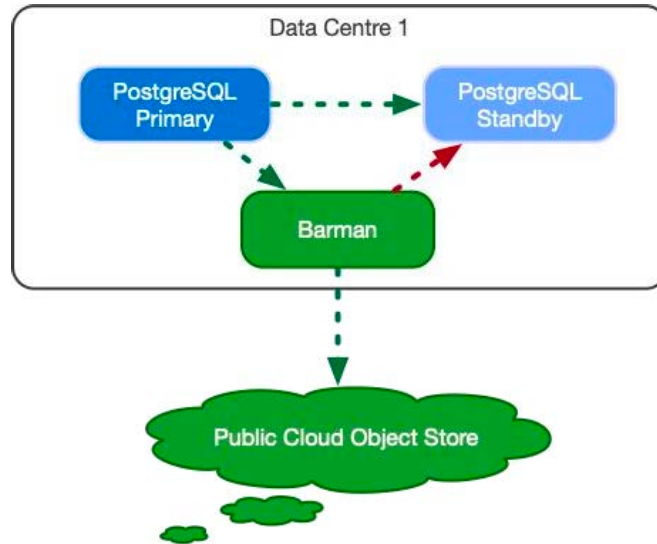
By ... relaying to AWS S3 compatible object stores

- Public/Private/Hybrid cloud
 - You can use a gateway too
- Requires Boto3 library
- “barman-cloud-wal-archive” (WALs)
- “barman-cloud-backup” (Backups)

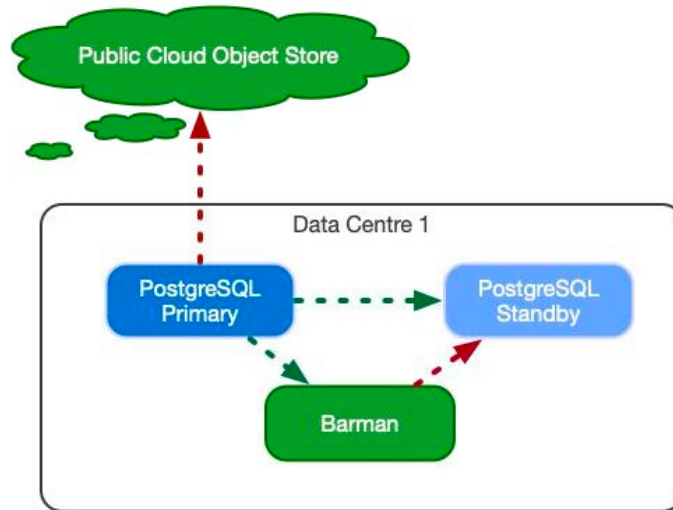
Food for thought:

- Removing SPoF for data availability
 - Data centre, Provider, Continent, Planet, ... that’s it for now
 - Multi-tiered backup and recovery
- Enhanced DR capabilities
- Remember encryption!

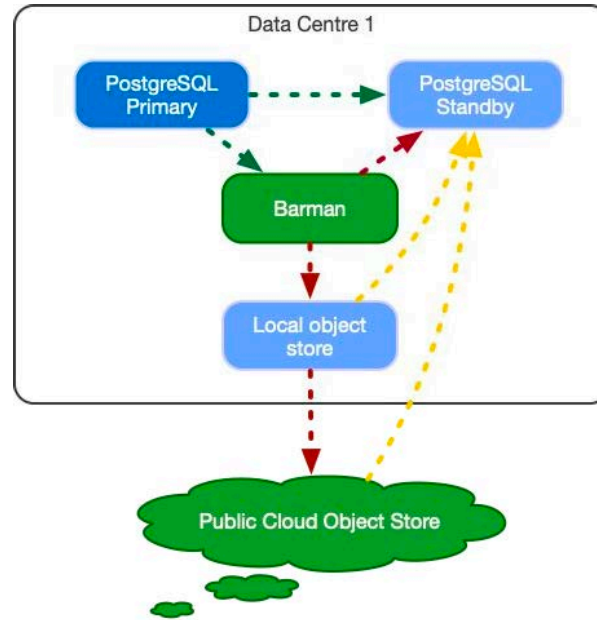
#19 - Public cloud, via Barman




#20 - Public cloud, direct

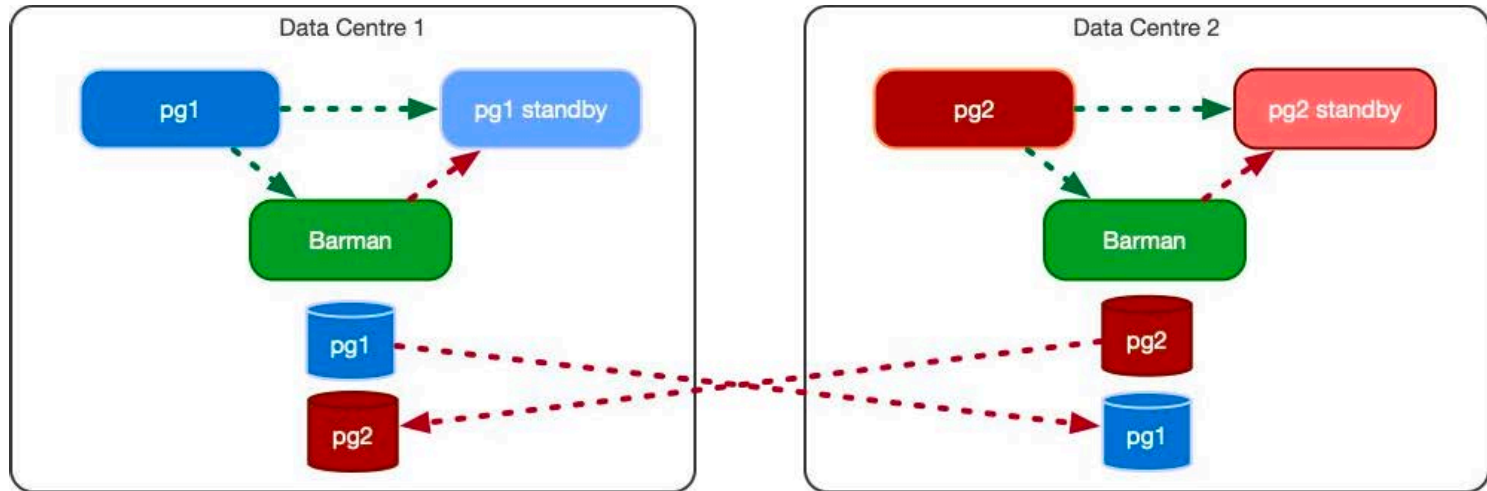


#21 - Local object store with gateway



 To be implemented

#22 - Geo-redundancy



#23 - Aggressive start

Enable “immediate_checkpoint”

- Speed up the start of the backup
 - Request a checkpoint without waiting for the scheduled one
- **Not available when backing up from a standby server**

#24 - Precautions

Best practices that increase the resilience through “barman check”

- “`minimum_redundancy`”
 - Safety measure: set it to 1
- “`last_backup_maximum_age`”
 - Based on the backup frequency
 - E.g. “1 week” in case of weekly backups
- “`max_incoming_wals_queue`”
 - Let “barman check” fail if your incoming queue of WALs gets too high

#25 - Server configuration files

Ideal for configuration management tools and automation

- Use a separate configuration file per server
 - .conf suffix is required
- Place the file in the `/etc/barman.conf.d` folder
- Hint: add the major PostgreSQL version as a suffix to the file (and server ID)
 - `SERVER_ID-PGVERSION.conf`
 - Example: `chat-13.conf`

#26 - Last version always wins

Always install the latest version of Barman

- Barman is backwards compatible
- Trunk based development
- Comprehensive set of automated tests

#27 - Enjoy convention over configuration

- **Most options in Barman:**
 - Can be set globally in the configuration file
 - Can be overridden at server level
 - Have default values
- **If you use our packages, system configuration is already taken care of**
 - User
 - Cron
 - Log rotation

#28 - Get hooked

Integrate Barman with external systems through hook scripts

- Available Before/After certain events
- Two types:
 - Standard: in case of failure, no retry
 - Retry: in case of failure, retry
 - Typical: before WAL archive to relay WAL files in the Cloud
 - See “barman-cloud-wal-archive” man page

#29 - Work in small batches

Configure the WAL archiver batch size

- **WAL archiver is automatically run by the “cron” command**
 - Every minute
 - Processes incoming WAL files and archives them compressed (if required)
 - By default, the batch is unlimited
- **Batch size can be set for both standard archiving and streaming channels**
 - Tune it based on the number of expected WALs between two cron runs
 - Good value to start with is between 10 and 100

#30 - JSON output

- Every command supports “-f json”
- Integration with other applications

#31 - Principle Of Least Authority (POLA)

- Avoid using a superuser for Barman
- The “barman” user can be a standard user
 - With specific grants for backup and read operations
 - barman_streaming can be used for replication connections
- **Requirements:**
 - PostgreSQL 10+
 - Barman 2.11+

#32 - Periodically recover your backups

Use hook scripts to systematically recover and test your backups

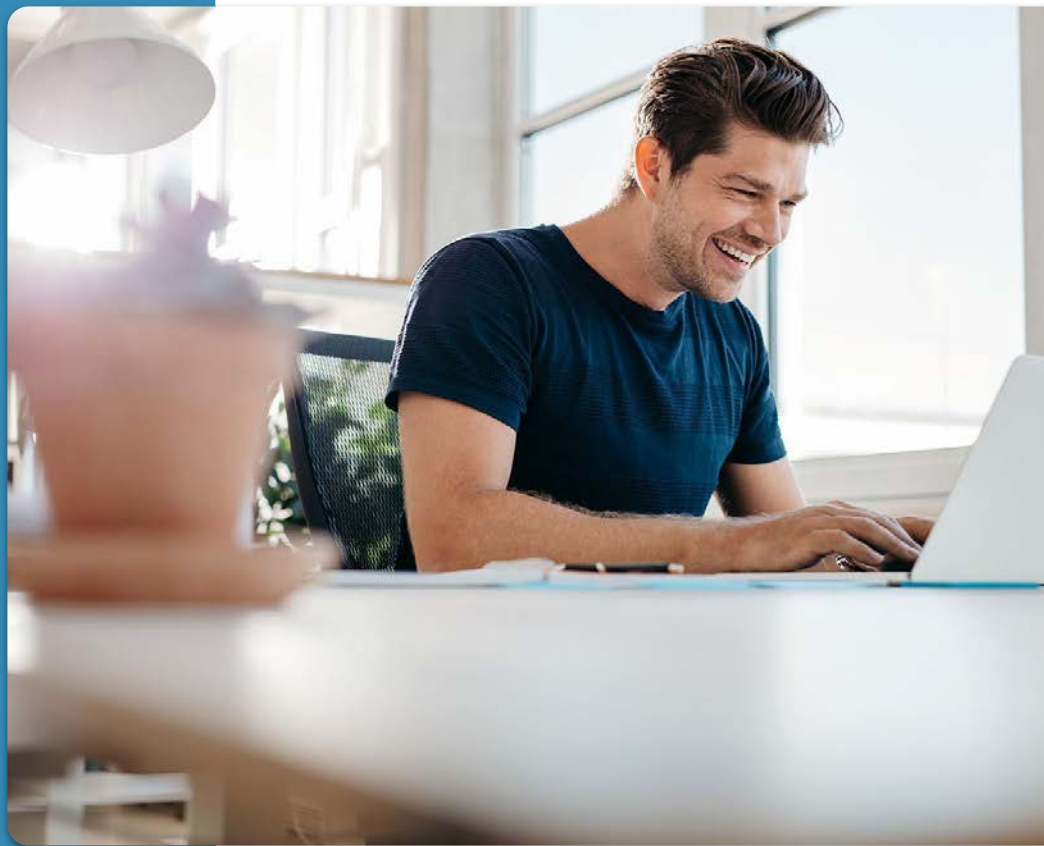
- Use post backup hook scripts to issue a remote recovery
- Use pre recovery scripts to control and prepare remote environments
- Use post recovery scripts to perform data reconciliation and transformation
- You can regenerate a standby server, a reporting/BI database or a QA environment
- The known unknowns:
 - Will my backup work?
 - How long will it take to recover from a backup (worst case scenario RTO)?

#33 - Enhance gradually

Consider a cluster with just a PostgreSQL instance and a Barman instance

- Can I achieve Disaster Recovery?
- Can I achieve High Availability?
- Can I achieve Business Continuity?
- What are my RPO and RTO?
- Start with your goals, add components gradually to improve them
 - KISS
 - Remove your next Single Point of Failure

What lies ahead of us



EDB powers Barman

EDB acquired 2ndQuadrant on 29 Sept 2020

- **EDB is committed to Barman**
 - New leadership, management and dev team
- **Original team stepping down after ~ 10 years**
 - We will serve as advisors
 - Focus on Kubernetes integration
- **Want a scoop?**
 - Abhijit Menon-Sen will lead Barman Development
 - Sebastiaan Mannem will be the Product Manager



Some ideas

An anticipation of possible future developments

- **Barman Operator for Kubernetes**
 - Stay tuned, lots of cool stuff around the corner!
- **Tier-2: compressed/encrypted backups**
 - `tar_retention_policy`
- **Tier-3: object store backups**
 - `S3_retention_policy`
- **Seamless integration with `pg_verifybackup`**
- **Snapshot backups**



Final thoughts



Conclusion

- Use these patterns as enablers for dialog in your team.
- Enhance gradually! Focus on the goals, not the means!
 - **Constrain your RPO** and don't underestimate RPO=0 scenarios
 - Test your backups. Always. **Measure your RTO**. Be ready for the next incident.
- Remember the “Flux capacitor”
- Barman is stable, robust & backwards compatible
- Barman is Open Source

Follow me:

@_GBartolini_

Tweet:

#pgbarman

#PostgresBuild2020

pgbarman.org

