

# Démystifier les sauvegardes incrémentales dans l'écosystème PostgreSQL



**data egret**

Your remote PostgreSQL DBA team

Stefan FERCOT

[stefan.fercot@dataegret.com](mailto:stefan.fercot@dataegret.com)

# SECURING YOUR DATABASE AVAILABILITY, SO THAT YOUR TEAM CAN FOCUS ON NEW FEATURE DEVELOPMENT.

- Migrations
- DB audit
- Performance optimisation
- Backup & restore
- Architectural review
- Advising Data Science teams
- Developer training

**on premise & cloud**



## EXPERTISE

Senior DBA with **10+ years**  
of PostgreSQL  
administration **experience**



## DEVELOPMENT

Involved in  
**new feature and**  
**extension development**



## TAILORED APPROACH

Felxible approach and  
**dedicated team** focused  
on success of your project



## COMMUNITY

**Contributing Sponsor.**  
Deeply involved in the  
PostgreSQL community

# Stefan Fercot

- Senior PostgreSQL Expert @Data Egret
- pgBackRest fan & contributor
- aka. pgstef
- <https://pgstef.github.io>

# Agenda

- Sauvegardes incrémentales dans PostgreSQL 17
  - Extraits des journaux de transaction (“WAL summaries”)
  - Copie incrémentale avec `pg_basebackup`
  - `pg_combinebackup`
- Limitations et autres considérations utiles
  - `pg_verifybackup`, *evergreen* backup
- L'écosystème
  - Exemple comparatif : `pg_basebackup` vs pgBackRest

# PostgreSQL 17

*PostgreSQL 17 Beta 1 Released on May 23, 2024.*

<https://www.postgresql.org/about/news/postgresql-17-beta-1-released-2865/>

# Sauvegardes incrémentales

```
$ git show dc212340058b4e7ecfc5a7a81ec50e7a207bf288
```

```
Author: Robert Haas <rhaas@postgresql.org>
```

```
Date:   Wed Dec 20 09:49:12 2023 -0500
```

```
    Add support for incremental backup.
```

An incremental backup is like a regular full backup except that some relation files are replaced with files with names like `INCREMENTAL.${ORIGINAL_NAME}`, and the `backup_label` file contains additional lines identifying it as an incremental backup.

The new `pg_combinebackup` tool can be used to reconstruct a data directory from a full backup and a series of incremental backups.

Patch by me. Reviewed by Matthias van de Meent, Dilip Kumar, Jakub Wartak, Peter Eisentraut, and Álvaro Herrera. Thanks especially to Jakub for incredibly helpful and extensive testing.

encore plus de travail réalisé après ça,  
avec de nombreux hackers impliqués !



# Toujours se référer à la doc !

<https://www.postgresql.org/docs/devel/continuous-archiving.html#BACKUP-INCREMENTAL-BACKUP>

## 25.3.3. Making an Incremental Backup

You can use `pg_basebackup` to take an incremental backup by specifying the `--incremental` option. You must supply, as an argument to `--incremental`, the backup manifest to an earlier backup from the same server. In the resulting backup, non-relation files will be included in their entirety, but some relation files may be replaced by smaller incremental files which contain only the blocks which have been changed since the earlier backup and enough metadata to reconstruct the current version of the file.




To figure out which blocks need to be backed up, the server uses WAL summaries, which are stored in the data directory, inside the directory `pg_wal/summaries`. If the required summary files are not present, an attempt to take an incremental backup will fail. The summaries present in this directory must cover all LSNs from the start LSN of the prior backup to the start LSN of the current backup. Since the server looks for WAL summaries just after establishing the start LSN of the current backup, the necessary summary files probably won't be instantly present on disk, but the server will wait for any missing files to show up. This also helps if the WAL summarization process has fallen behind. However, if the necessary files have already been removed, or if the WAL summarizer doesn't catch up quickly enough, the incremental backup will fail.

When restoring an incremental backup, it will be necessary to have not only the incremental backup itself but also all earlier backups that are required to supply the blocks omitted from the incremental backup. See `pg_combinebackup` for further information about this requirement. Note that there are restrictions on the use of `pg_combinebackup` when the checksum status of the cluster has been changed; see `pg_combinebackup limitations`.

Note that all of the requirements for making use of a full backup also apply to an incremental backup. For instance, you still need all of the WAL segment files generated during and after the file system backup, and any relevant WAL history files. And you still need to create a `recovery.signal` (or `standby.signal`) and perform recovery, as described in [Section 25.3.5](#). The requirement to have earlier backups available at restore time and to use `pg_combinebackup` is an additional requirement on top of everything else. Keep in mind that PostgreSQL has no built-in mechanism to figure out which backups are still needed as a basis for restoring later incremental backups. You must keep track of the relationships between your full and incremental backups on your own, and be certain not to remove earlier backups if they might be needed when restoring later incremental backups.

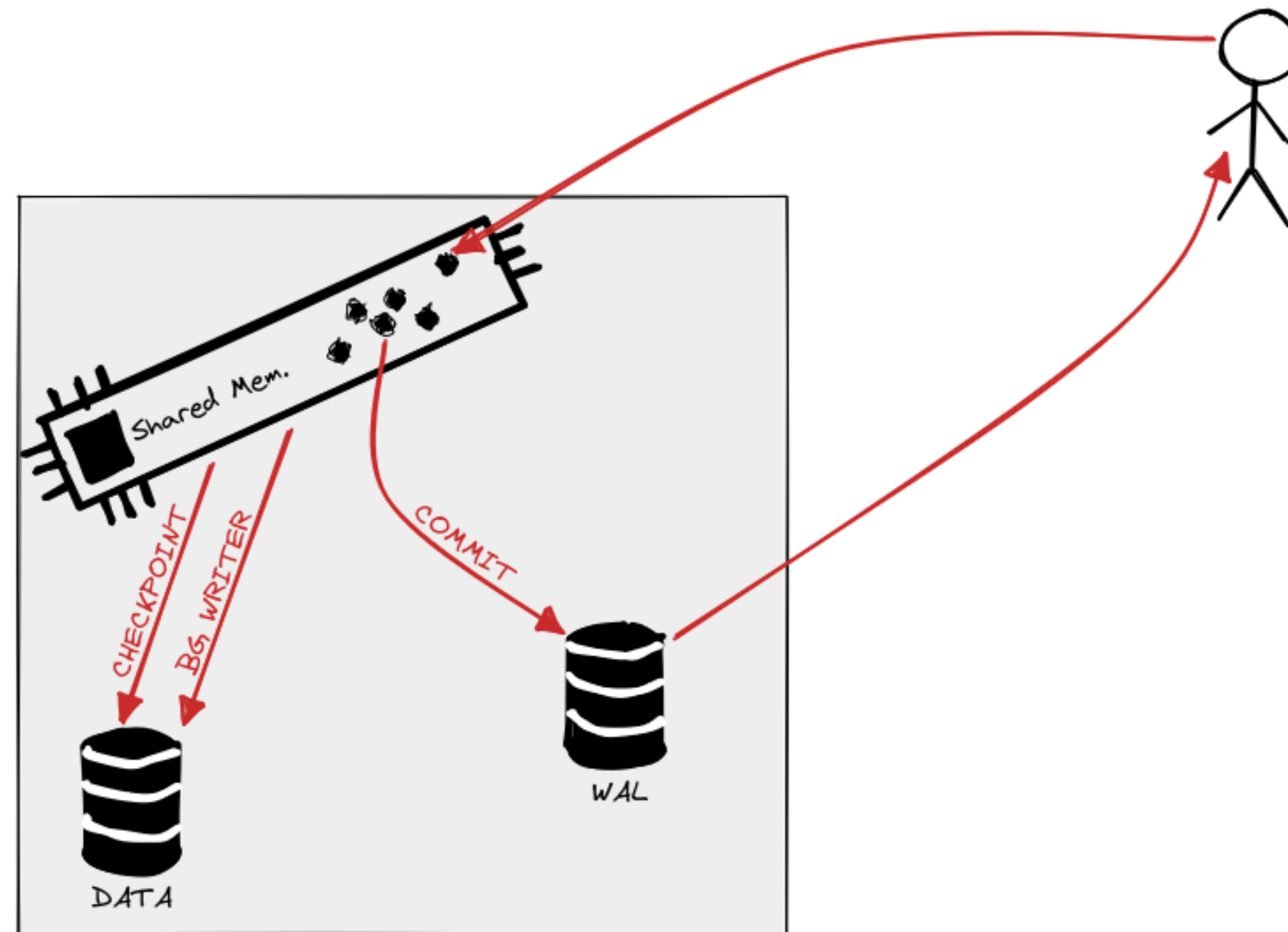
Incremental backups typically only make sense for relatively large databases where a significant portion of the data does not change, or only changes slowly. For a small database, it's simpler to ignore the existence of incremental backups and simply take full backups, which are simpler to manage. For a large database all of which is heavily modified, incremental backups won't be much smaller than full backups.

# Qu'est-ce qu'une sauvegarde incrémentale ?

- plutôt que de copier l'instance complète...
  - ne copier que les données réellement modifiées
-  plus rapide, moins d'espace disque nécessaire
-  plus de travail nécessaire pour la restauration !
-  comment identifier rapidement et de manière fiable quelles données ont changé ?



# Journaux de transaction (Write Ahead Log)



# Extraits des journaux de transaction - WAL summarizer

```
$ git show 174c480508ac25568561443e6d4a82d5c1103487
Author: Robert Haas <rhaas@postgresql.org>
Date:   Wed Dec 20 08:41:09 2023 -0500
    Add a new WAL summarizer process.
```

- `summarize_wal` active ou désactive ce nouveau *background process*
- crée des extraits des journaux de transaction
  - dans `pg_wal/summaries`
  - un fichier par cycle de *checkpoint*
- supprime automatiquement les fichiers
  - plus vieux que `wal_summarize_keep_time` (10 jours par défaut)

# Exemple

```
postgres=# ALTER SYSTEM SET summarize_wal TO on;
```

```
ALTER SYSTEM
```

```
postgres=# SELECT pg_reload_conf();
```

```
pg_reload_conf
```

```
-----
```

```
t
```

```
(1 row)
```

```
postgres=# SELECT * FROM pg_settings WHERE name='summarize_wal';
```

```
name          | summarize_wal
```

```
setting       | on
```

```
category      | Write-Ahead Log / Summarization
```

```
short_desc    | Starts the WAL summarizer process to enable incremental backup.
```

```
context       | sighup
```

## Exemple (2)

```
$ pgbench -i -s 60 pgbench
$ du -hs 17/data/pg_wal/summaries/*
20K 17/data/pg_wal/summaries/000000010000000022402E68000000002F9DEC80.summary
20K 17/data/pg_wal/summaries/00000001000000002F9DEC80000000005041A5D0.summary

$ psql -c "select * from pg_available_wal_summaries();"
 tli | start_lsn | end_lsn
-----+-----+-----
   1 | 0/22402E68 | 0/2F9DEC80
   1 | 0/2F9DEC80 | 0/5041A5D0
```

- pour une analyse plus approfondie :
  - `pg_available_wal_summaries()`,
  - `pg_get_wal_summarizer_state()`
  - `pg_wal_summary_contents()`, `pg_walsummary` client

## pg\_basebackup

```
$ pg_basebackup --help|grep incremental
-i, --incremental=OLDMANIFEST
    take incremental backup
```

- ne fonctionne qu'avec le format `plain`
- il suffit de fournir le manifeste de la sauvegarde précédente (référence)

# Qu'est-ce qu'un manifeste de sauvegarde ?

```
$ pg_basebackup -D full_1 --checkpoint=fast
$ cat full_1/backup_manifest
{"System-Identifier": 7362166132444199273,
  "Files": [
    ...
    { "Path": "base/1/1255", "Size": 811008,
      "Last-Modified": "2024-04-26 13:24:39 GMT",
      "Checksum-Algorithm": "CRC32C", "Checksum": "02b18ea4" },
  ],
  "WAL-Ranges": [
    { "Timeline": 1, "Start-LSN": "0/60000028", "End-LSN": "0/60000120" }
  ],
```



# Exemple

```
$ pgbench -i -s 10 pgbench
$ pg_basebackup -D incr_1 --incremental=full_1/backup_manifest -c fast

$ pgbench -i -s 10 pgbench
$ pg_basebackup -D incr_from_full1 --incremental=full_1/backup_manifest -c fast
$ pg_basebackup -D incr_from_incr1 --incremental=incr_1/backup_manifest -c fast
$ pg_basebackup -D full_2 --checkpoint=fast
```



## pg\_combinebackup

- reconstruit un répertoire de données (PGDATA)
  - à partir d'une sauvegarde incrémentale
  - et de toutes les sauvegardes dont elle dépend

```
$ pg_combinebackup --output=restore_1 full_1 incr_1 incr_from_incr1
```

```
$ pg_combinebackup --output=restore_2 full_1 incr_from_full1
```

```
$ pg_combinebackup --output=restore_3 full_1 incr_from_incr1/  
error: backup at "full_1" starts at LSN 0/60000028, but expected 0/69000028
```

- pas d'historique des dépendances entre les sauvegardes !

# Limitations

- ne recalcule pas les sommes de contrôle des pages (*page checksums*)
  - prendre une nouvelle sauvegarde complète après avoir activé/désactivé les sommes de contrôle
- ne vérifie que si les sauvegardes sont correctement ordonnées...
  - pas que chaque sauvegarde est correcte/intacte
  - pour ça, utilisez `pg_verifybackup` !

## pg\_verifybackup

*orthogonalité : un outil doit faire une seule tâche et la faire aussi bien que possible*

```
$ cp -rfP incr_from_incr1 incr_from_incr1_corrupted
$ rm incr_from_incr1_corrupted/base/1/INCREMENTAL.2675
$ pg_verifybackup incr_from_incr1_corrupted
error: "base/1/INCREMENTAL.2675" is present in the manifest but not on disk

$ pg_combinebackup --output=restore_3 full_1 incr_1 incr_from_incr1_corrupted
$ pg_verifybackup restore_3
backup successfully verified
```

# The “evergreen” backup

- `pg_combinebackup` va produire...
  - une sauvegarde complète “*artificielle*” !
  - qui peut être utilisée comme source pour de nouvelles sauvegardes
  - ou pour d’autres invocations de `pg_combinebackup`

```
# idée initiale par Robert Haas
$ pg_basebackup -D sunday_full -c fast
$ pg_basebackup -D monday_incr --incremental=sunday_full/backup_manifest -c fast
$ pg_combinebackup -o monday_full sunday_full monday_incr
$ rm -rf sunday_full monday_incr
```

# L'écosystème

- pgBarman
- WAL-G
- pgBackRest



# pgBarman

- version 3.10.0, sortie le 24 Janvier 2024
- <https://github.com/EnterpriseDB/barman>
- `rsync` (via SSH) ou `pg_basebackup`

*IMPORTANT: Because Barman transparently makes use of `pg_basebackup`, features such as incremental backup, parallel backup, and deduplication are currently not available.*

# WAL-G

- version 3.0.0, sortie le 17 Mars 2024
- <https://github.com/wal-g/wal-g>
- supporte PostgreSQL, MS SQL Server, et Greenplum
  - *unstable* pour MongoDB, MySQL/MariaDB et Redis
- delta backup

*Delta computation is based on ModTime of file system and LSN number of pages in datafiles*

# WAL-G (2)

WALG\_DELTA\_MAX\_STEPS

Delta-backup is the difference between previously taken backup and present state.

WALG\_DELTA\_MAX\_STEPS determines how many delta backups can be between full backups.

Delta computation is based on ModTime of file system and LSN number of pages in datafiles.

WALG\_DELTA\_FROM\_NAME

To configure base for next delta backup.

WALG\_DELTA\_ORIGIN can be LATEST (chaining increments),

LATEST\_FULL (for bases where volatile part is compact and chaining has no meaning).

# pgBackRest

- version 2.52, sortie le 27 Mai 2024
- <https://github.com/pgbackrest/pgbackrest>

*Block incremental backups save space by only storing the parts of a file that have changed since the prior backup rather than storing the entire file.*

# Exemple comparatif : pg\_basebackup vs pgBackRest

```
$ createdb pgbench
$ pgbench -i -s 600 pgbench

$ cat /var/lib/pgsql/17/data/postgresql.auto.conf
summarize_wal = 'on'
archive_mode = on
archive_command = 'pgbackrest --stanza=demo archive-push %p'
```

# pgBackRest config

```
$ cat /etc/pgbackrest.conf
[global]
repo1-path=/var/lib/pgbackrest/1
repo1-retention-full=2

repo2-path=/var/lib/pgbackrest/2
repo2-retention-full=2
repo2-bundle=y
repo2-block=y

compress-type=zst
start-fast=y
[demo]
pg1-path=/var/lib/pgsql/17/data
```



# Sauvegarde complète (full)

```
$ pg_basebackup -D /var/lib/postgresql/17/backups/full_1 \
  --checkpoint=fast --wal-method=none --progress
$ du -hs /var/lib/postgresql/17/backups/full_1
8.8G  /var/lib/postgresql/17/backups/full_1
$ find /var/lib/postgresql/17/backups/full_1 -type f | wc -l
1288
```

```
$ pgbackrest --stanza=demo --type=full --repo=1 backup
$ du -hs /var/lib/pgbackrest/1/backup/demo/20240604-122538F
415M  /var/lib/pgbackrest/1/backup/demo/20240604-122538F
$ find /var/lib/pgbackrest/1/backup/demo/latest -type f | wc -l
1293
```

```
$ pgbackrest --stanza=demo --type=full --repo=2 backup
$ du -hs /var/lib/pgbackrest/2/backup/demo/20240604-122646F
395M  /var/lib/pgbackrest/2/backup/demo/20240604-122646F
$ find /var/lib/pgbackrest/2/backup/demo/latest -type f | wc -l
15
```

# Sauvegarde incrémentale

```
$ pgbench -c 4 -n -b simple-update -t 100 pgbench
```

```
$ pg_basebackup -D /var/lib/pgsql/17/backups/incr_1 \  
--incremental=/var/lib/pgsql/17/backups/full_1/backup_manifest \  
--checkpoint=fast --wal-method=none --progress
```

```
$ du -hs /var/lib/pgsql/17/backups/incr_1  
16M /var/lib/pgsql/17/backups/incr_1
```

```
$ pgbackrest --stanza=demo --type=incr --repo=1 backup
```

```
$ du -hs /var/lib/pgbackrest/1/backup/demo/20240604-122538F_20240604-123415I  
407M /var/lib/pgbackrest/1/backup/demo/20240604-122538F_20240604-123415I
```

```
$ pgbackrest --stanza=demo --type=incr --repo=2 backup
```

```
$ du -hs /var/lib/pgbackrest/2/backup/demo/20240604-122646F_20240604-123447I  
7.3M /var/lib/pgbackrest/2/backup/demo/20240604-122646F_20240604-123447I
```

# Sauvegarde incrémentale chaînée - pg\_basebackup

```
$ pgbench -c 4 -n -b simple-update -t 100 pgbench

$ pg_basebackup -D /var/lib/pgsql/17/backups/incr_from_incr_1 \
  --incremental=/var/lib/pgsql/17/backups/incr_1/backup_manifest \
  --checkpoint=fast --wal-method=none --progress
$ du -hs /var/lib/pgsql/17/backups/incr_from_incr_1
16M /var/lib/pgsql/17/backups/incr_from_incr_1

$ pg_basebackup -D /var/lib/pgsql/17/backups/incr_from_full_1 \
  --incremental=/var/lib/pgsql/17/backups/full_1/backup_manifest \
  --checkpoint=fast --wal-method=none --progress
$ du -hs /var/lib/pgsql/17/backups/incr_from_full_1
23M /var/lib/pgsql/17/backups/incr_from_full_1
```

# Sauvegarde incrémentale chaînée - pgBackRest

```
$ pgbackrest --stanza=demo --type=incr --repo=2 backup
$ pgbackrest info --stanza=demo --repo=2
incr backup: 20240604-122646F_20240604-124330I
  timestamp start/stop: 2024-06-04 12:43:30+00 / 2024-06-04 12:43:34+00
  wal start/stop: 00000001000000001000000E6 / 00000001000000001000000E6
  database size: 8.8GB, database backup size: 8.8GB
  repo2: backup size: 6.6MB
  backup reference list: 20240604-122646F, 20240604-122646F_20240604-123447I
```

```
$ pgbackrest --stanza=demo --type=diff --repo=2 backup
$ pgbackrest info --stanza=demo --repo=2
diff backup: 20240604-122646F_20240604-124344D
  timestamp start/stop: 2024-06-04 12:43:44+00 / 2024-06-04 12:43:47+00
  wal start/stop: 00000001000000001000000E8 / 00000001000000001000000E8
  database size: 8.8GB, database backup size: 8.8GB
  repo2: backup size: 12.4MB
  backup reference list: 20240604-122646F
```

# Performance de la restauration

```
$ time pg_combinebackup -o new_full full_1 incr_1 incr_from_incr_1
real    0m9.628s
user    0m1.289s
sys     0m4.944s
```

```
$ time pg_combinebackup -o new_full full_1 incr_from_full_1
real    0m10.136s
user    0m1.659s
sys     0m4.895s
```

```
$ time pgbackrest --stanza=demo restore --repo=2 --set=20240604-122646F_20240604-124330I
real    0m11.512s
user    0m9.352s
sys     0m6.537s
```

```
$ time pgbackrest --stanza=demo restore --repo=2 --set=20240604-122646F_20240604-124344D
real    0m9.464s
user    0m10.635s
sys     0m4.302s
```

# Résumé

- nouvelle fonctionnalité nécessitant de l'orchestration
- réfléchir attentivement à sa politique de rétention
  - utiliser régulièrement `pg_verifybackup`
  - garder une liste des inter-connexions entre les sauvegardes
  - penser à configurer `wal_summarize_keep_time` correctement
  - ne pas trop compliquer les choses !
- tenter régulièrement de restaurer les sauvegardes



# A propos de récupération...

*Schrödinger's Law of Backups*

*The condition/state of any backup is unknown until  
a restore is attempted.*

WEBINAR

## THE PATH TO A SUCCESSFUL POSTGRESQL RECOVERY

20 JUNE, 13:00-14:00CEST

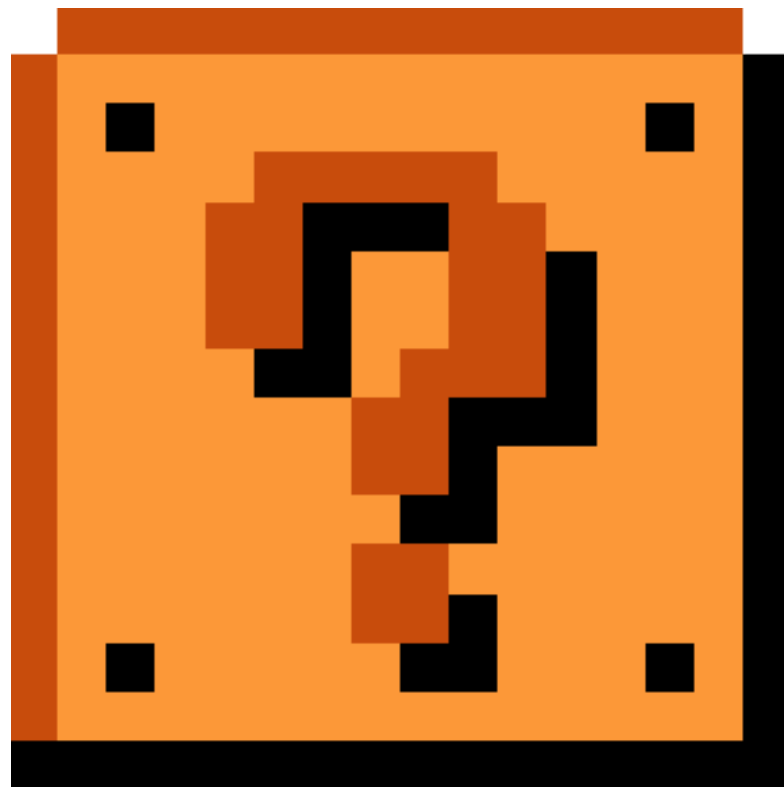
- Step-by-step restore procedure
- Various settings influencing the PostgreSQL recovery
- Practical scenarios using a quick demo setup



**Register!**

To boost your  
confidence in facing  
PostgreSQL recovery  
challenges!

# Questions ?



Merci pour votre attention !