Vue d'ensemble sur la sauvegarde, l‘archivage et la réplication PostgreSQL

Que faire, que ne pas faire, les pièges à éviter

Stéphane Kanschine
DBA à mes heures perdues
@carxwol
stephane@hexack.fr

Agenda

- Sauvegarde logique (alias SQL)
- Sauvegarde binaire
- PITR & réplication avec archivage
- Réplication en flux binaire
- Réplication logique

- Idées fausses et pièges
- Ce qui serait pas mal à priori

Quelques assertions

- Vos données ont suffisament de valeur pour que vous n‘ayez pas envie de les perdre.
- Vous avez (à minima) 2 serveurs digne de ce noms disponibles avec à peu près la même quantité de CPU/RAM/disque et selon les moyens
  - ECC memory
  - Contrôleur raid avec batterie / SAN
  - Un UPS fonctionnel
- Ils doivent être les plus éloignés possible l‘un de l‘autre
- Vous connaissez vos objectifs DIMA et PDMA

Lois naturelles omnipotentes

- Gravité
- La vitesse de la lumière
- La loi de l‘emmerdement maximum *
  - Les emmerdes n‘arrivent pas qu‘aux autres
  - Contrairement à l‘éclair et plutôt comme les bombardiers, ça vole (souvent) en escadron
  - Demandez aux gars de Gitlab !

Gardez Einstein en tête

- „Deux choses sont infinies : l‘Univers et la bêtise humain. Mais en ce qui concerne l‘Univers, j‘e n‘en ai pas encore acquis la certitude.“ **

Évolution

- Les options et les outils ont évolué avec les années
- Ça n‘est pas complètement **et** bien mis en valeur dans la documentation
- Consciencieux, vous voulez le faire de la façon la plus récente dans la documentation
- Ce qui signifie que avez avez lu ~ 50 pages A4 dans 2 chapitres...
- Vous devriez quand même la lire en intégralité !

Les options avec la 7.0

Logical backup

La note qui fait sourire

Prev

Chapter 33. Database Recovery

This section needs to be written. Volunteers?

Prev
Debugging Messages

Sauvegarde logique alias „SQL-“

- `pg_dump[all]` se connecte à votre BDD comme n‘importe quel client et vous fournit un instantanée de vos données
  - Vous pouvez restaurez votre BDD dans l‘état qu‘elle avait au moment de la sauvegarde
- Permet de sauvegarder des instances entières (`pg_dumpall`), par BDD, par tables
- Peu fournir un format texte (SQL) ou un format custom (format interne)

Format texte de pg_dump

- Pur SQL
- Restauration UNIX compliant ♥ (cat | psql)
- Utilise `COPY` pour la performance
- Peu servir pour porter des BDD….

- Peut être lu (et corrigé) par un humain

Format custom de pg_dump

- `pg_dump -Fc`
- Restauration avec `pg_restore` (à travers `psql` ou directement dans une BDD)
- Peut restaurer des tables au choix
- Compressé par défaut

Format directory de pg_dump

- `pg_dump -Fd`
- Peut sauvegarder (et restaurer) en paralèlle (`-j X`)
- Restauration avec `pg_restore` (à travers `psql` ou directement dans un BDD)
- Can restore single tables
- Compressed by default

Pensez fort à pg_dumpall !

- `pg_dump` lit depuis les **base de données**
- Donc, les objets globaux ne sont pas sauvégardés !
  - Roles
  - Tablespaces
- Donc, à chaque fois que vous utilisez pg_dump, faites un `pg_dumpall --globals-only` par la même occasion !

DIMA/PDMA de la sauvegarde logique

- DIMA
  - Entre minutes et jours
  - Bêtement dépendante de la taille de votre BDD
- PDMA
  - La date de votre dernière sauvegarde
  - Dans le pire des cas, celle d‘avant* !

Avantages et inconvénients

+ la sauvegarde est lisible par des humains (ou peut l'être), schema & roles peuvent aller dans votre VCS
+ ça peut être lu par les versions plus récentes de PG
+ ça peut sauvegarder et restaurer des entités unique si besoin
+ révélera des corruptions que vous ignoriez (quand lancé avec with data checksums)*

- peut seulement sauvegarder et restaurer un point unique dans le temps
- plutôt lent
- DIMA/PDMA pas terrible...

Depuis pg_dump

- 7.1 a ajouté le WAL
- 8.0 a ajouté la possibilité de faire
  - des sauvegardes à chaud
  - PITR
- 9.1 a ajouté `pg_basebackup`
  - Intégration de la méthode „gift-wrapping“
- 9.2 a permis `pg_basebackup` de récupérer les données des WAL

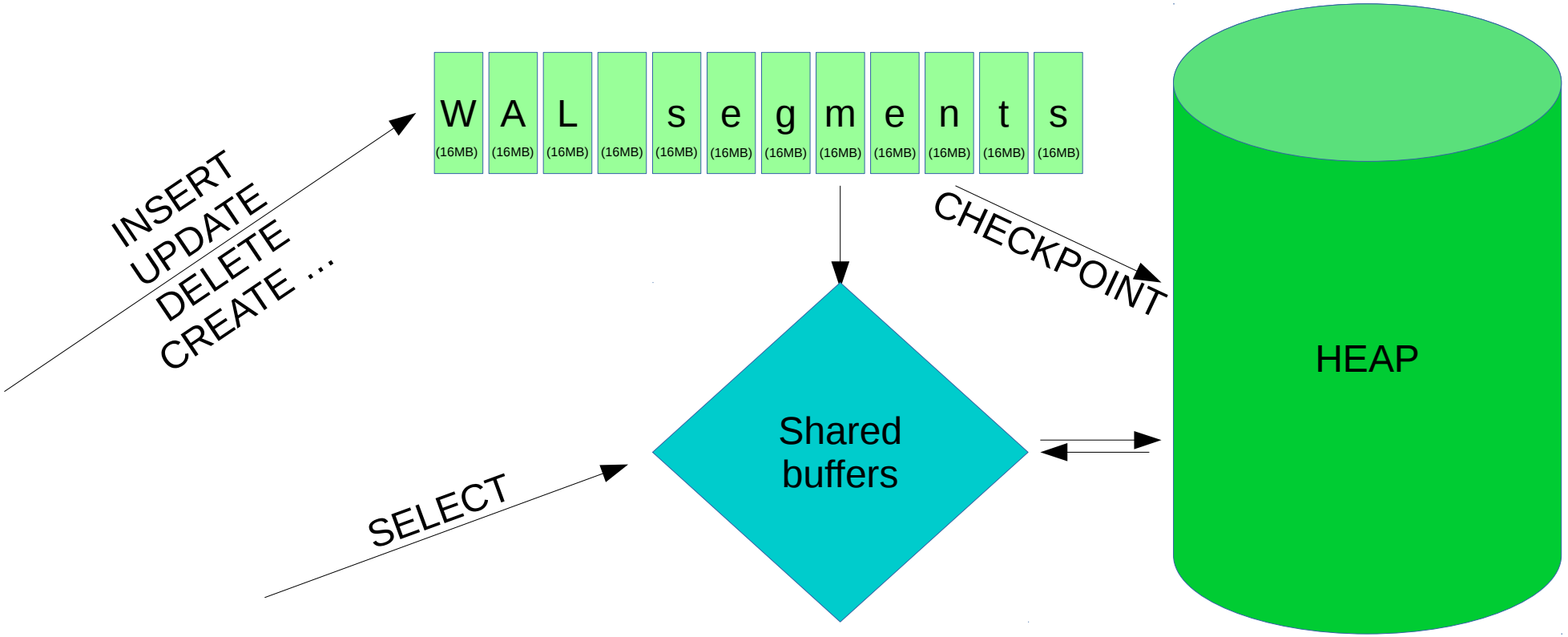
Sauvegarde binaire, à chaud

- Ce n‘est pas aussi simple ;-)
- Il faut connaître quelques particularités avant
- Tout ce qui est binaire tourne autour des WAL

Qu‘est ce que le WAL

- Le Write Ahead Log (WAL) est grossomodo le journal de bord de la BDD
- Les autres SGDB appellent ça „redolog“
  - certains ont même un „undolog“, mais Postgre n‘en a pas besoin
- Tout changement est d‘abord écrit dans les WAL
- À un `CHECKPOINT` (qui peut être propagé!), le contenu est écrit dans le HEAP, en créant des nouvelles versions d‘enregistrements

WAL (très simplifié)



WAL organisation

- Les WAL sont une succession de fichier, de 16Mo chacun, appelés segments
- Comme un anneau, ils sont renommés et écrasés
- C‘est stocké dans `$PGDATA/pg_xlog` (10.0 ff: „`pg_wal`“**)
- La taille totale est déterminé par `wal_min_size` et `wal_max_size` (par défaut : 1GB/2GB)
- Il y a des limites systèmes !

The archiver

- Les WAL segments sont écrit par le processus „wal writer“
- Les WAL segments sont lus et appliqués sur le HEAP par le processus „checkpointer“
- Entre temps, ils sont passés par le processus d‘archivage
  - quand `archive_mode != ,off‘`
  - ce qui est à priori ce que vous voulez !

Instantané binaire

- Préparez votre BDD :
  - `pg_start_backup()`
- Faites un instantané
  - Plusieurs options seront vu un peu plus tard !
- „Libérez“ le HEAP
  - `pg_stop_backup()`

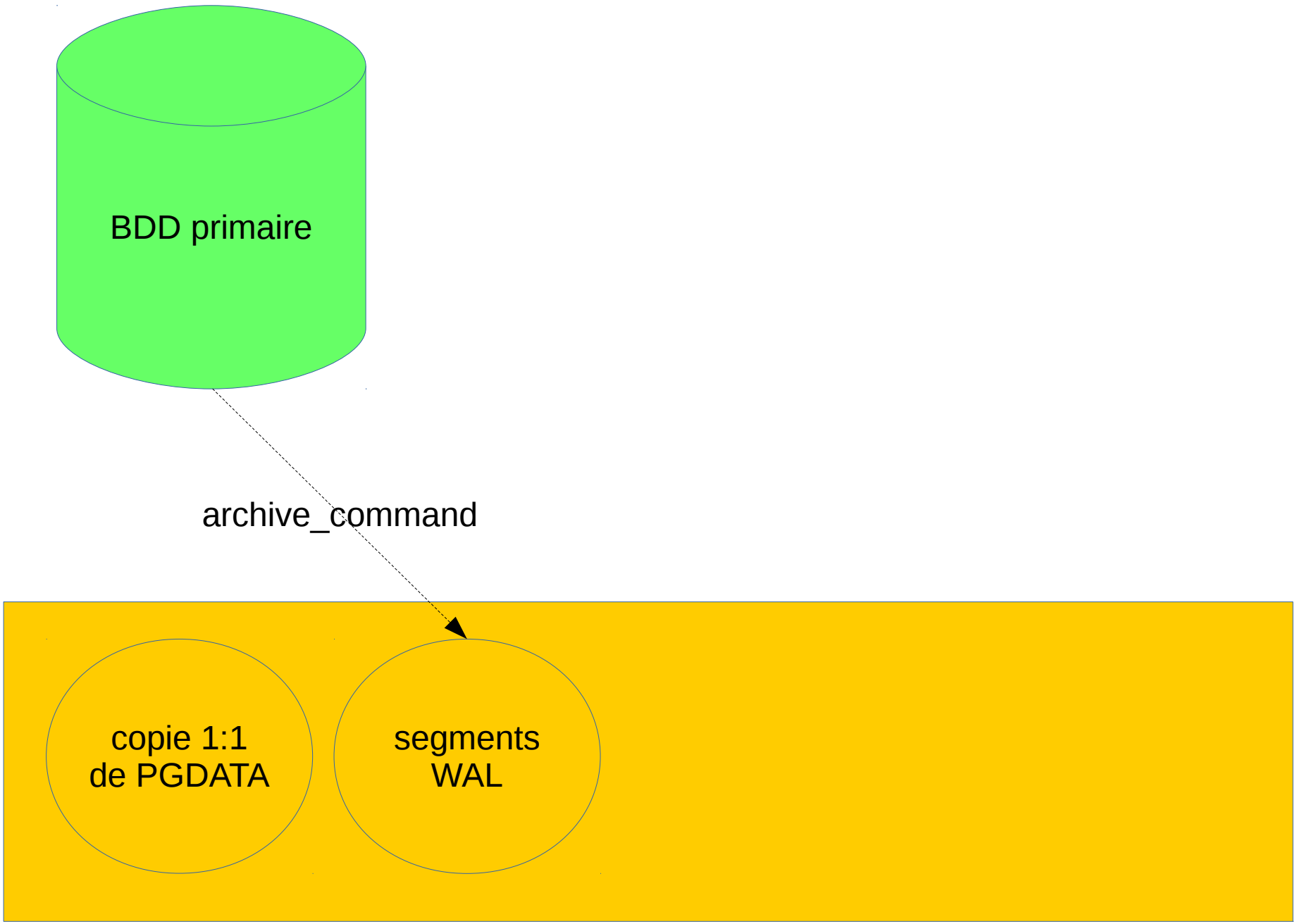
BDD primaire

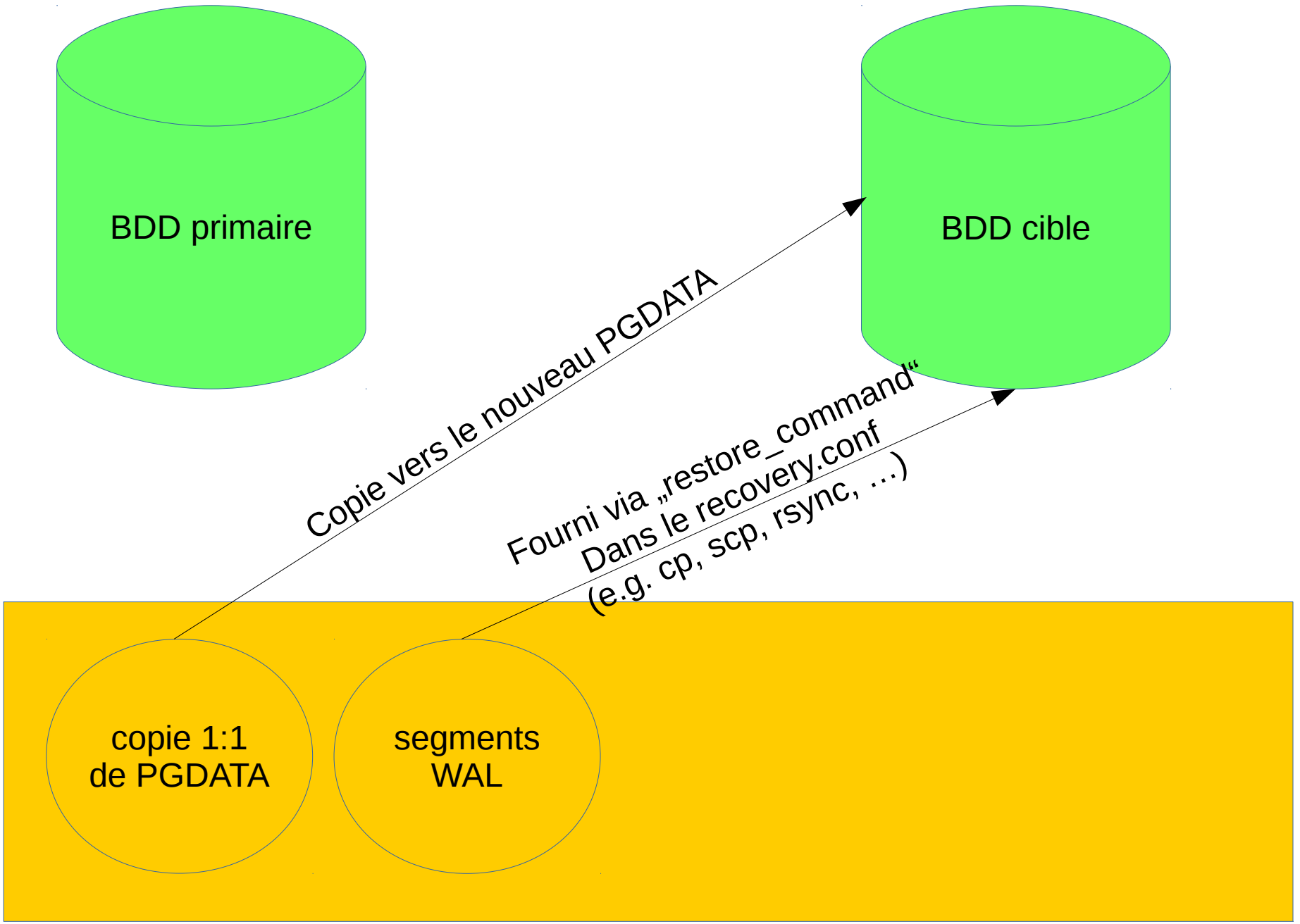
snapshot

copie 1:1
de PGDATA

Ai-je besoin de plus ?

- Bien sûr !
- Tous les segments de WAL depuis le pg_start_backup()!
- Par chance ils sont encore là hein ?
  - S‘il y a eu beaucoup d‘activité dans votre base après pg_start_backup(), ils ont peut être été déjà recyclés !*


BDD primaire
archive_command
copie 1:1
de PGDATA
segments
WAL


BDD primaire
BDD cible
Copie vers le nouveau PGDATA
Fourni via „restore_command“
Dans le recovery.conf
(e.g. cp, scp, rsync, …)
copie 1:1
de PGDATA
segments
WAL

DIMA/PDMA

- DIMA
  - entre minutes et jours
  - fonction de la taille et de l‘activité pendant la sauvegarde
- PDMA
  - le fin de votre dernière sauvegarde
  - ou celle d‘avant !*

Avantages et inconvénients

+ copie 1:1 de votre BDD
+ ceinture/bretelle
+ rapide
+ DIMA fine

- peut seulement sauvegarder et restaurer un point unique dans le temps
- peut seulement sauvegarder et restaurer des instances entières
- PDMA pas terrible

Les options pour faire cet instantané

- LVM / filesystem snapshot
- rsync
- pg_basebackup

Les options pour faire cet instantané

- ~~LVM / filesystem snapshot~~
- ~~rsync~~
- pg_basebackup

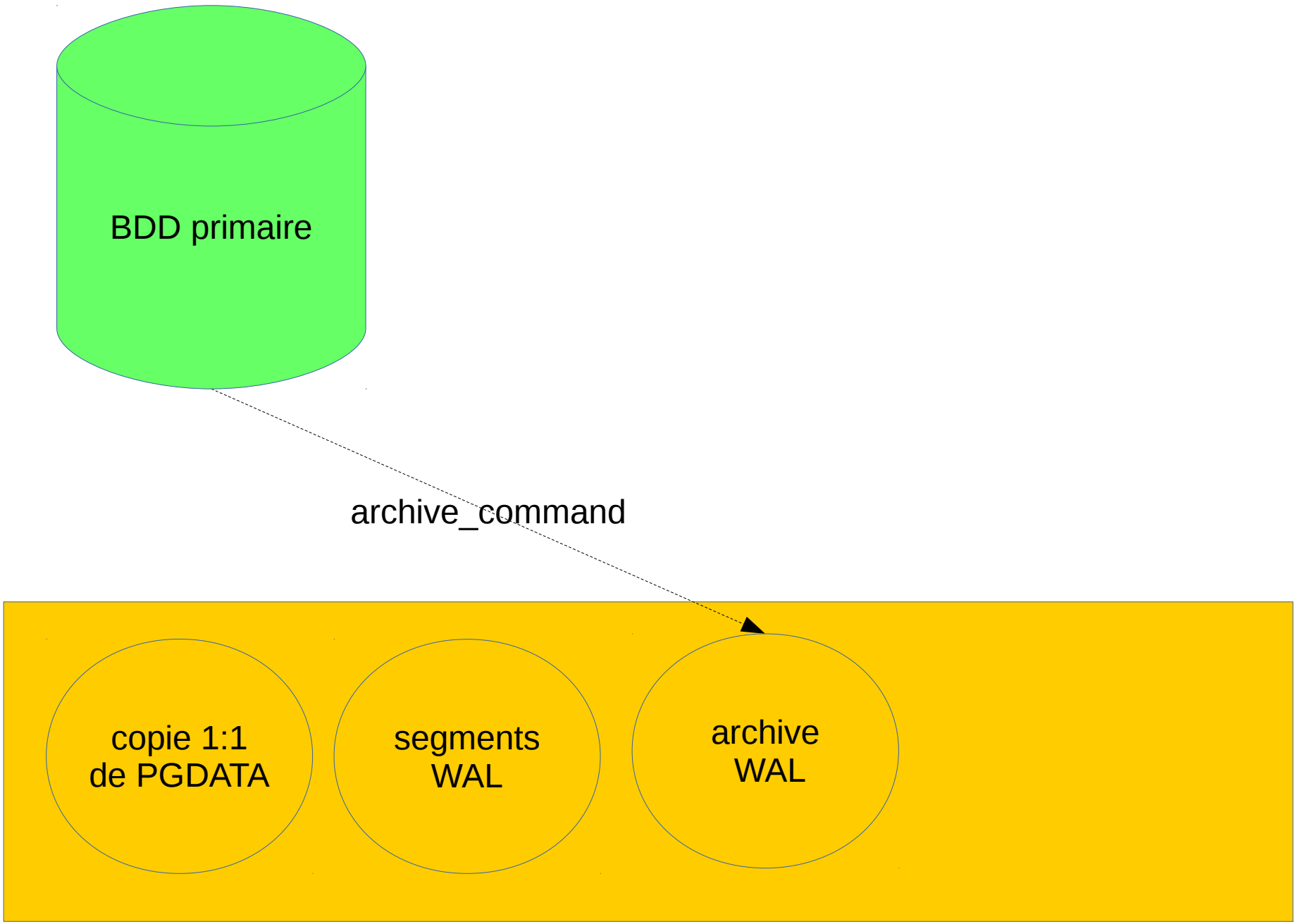
Les options pour les segments WAL

- archive_command (postgresql.conf)
- pg_basebackup
  - Avec l‘option --xlog-method=[fetch|stream]
  - -X [s|f]

UTILISEZ LES DEUX !

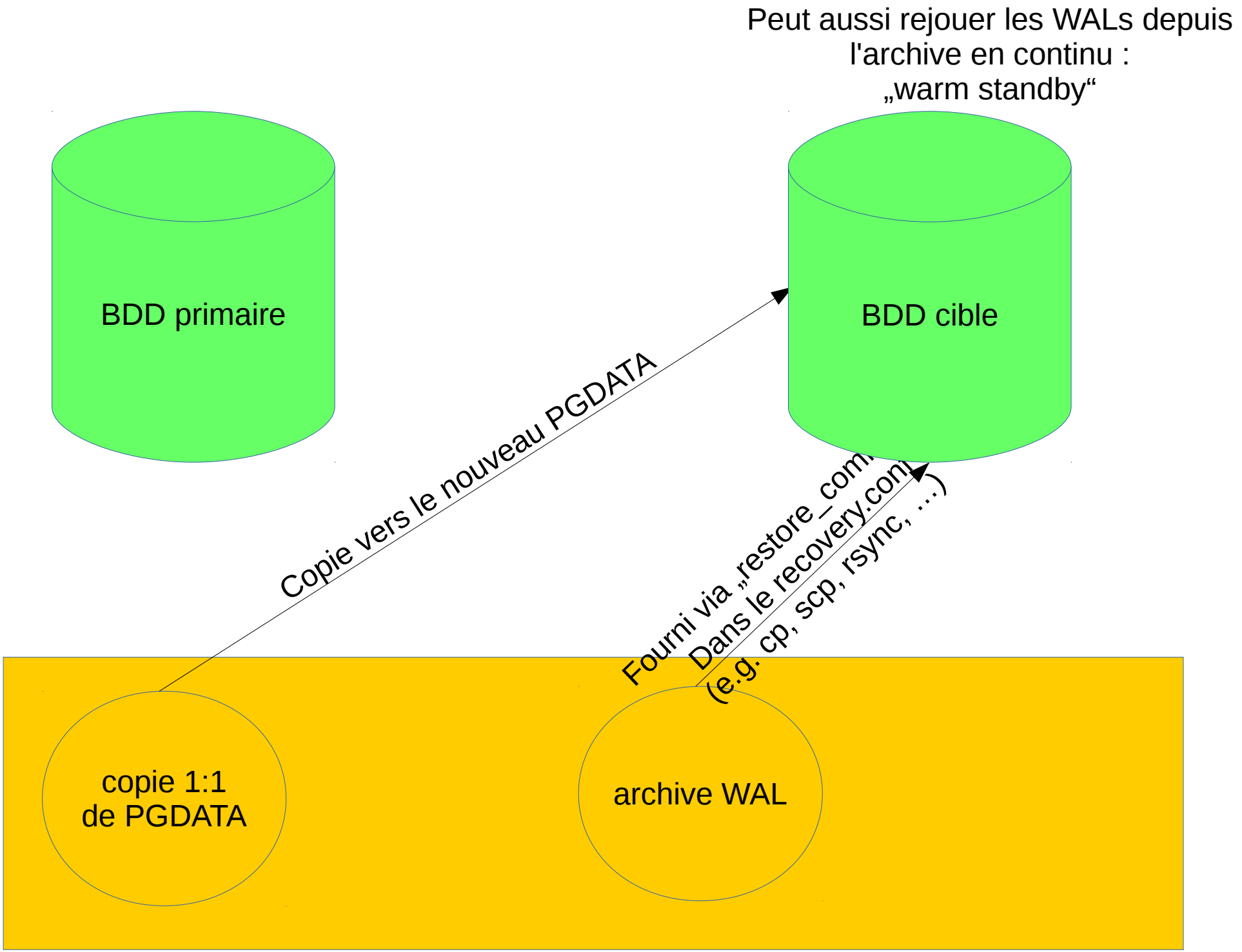
Pourquoi les deux ?

- En fait, prenez l'habitude des deux
- Quand vous archivez les WAL, vous pouvez (probablement) vous y fier
- Mais pg_basebackup avec -X est pratique pour initier un nouveau serveur secondaire (on va y venir)


BDD primaire
archive_command
copie 1:1
de PGDATA
segments
WAL
archive
WAL

Pourquoi je voudrais archiver les WAL ?

- Les segments de WAL, avec l'instantané du HEAP, vous permet de restaurer votre BDD à l'instant de votre choix dans le temps
  - e.g., le moment juste avant que vous oubliez le WHERE dans la requête „DELETE FROM customers“ ;-) **
- C'est la restauration dans le temps, alias Point In Time Recovery („PITR“)
- Forcément, vous aurez besoin de deux choses :
  - un instantané binaire du HEAP
  - tous les segments WAL entre votre instantané et votre erreur


Peut aussi rejouer les WALs depuis
l'archive en continu :
„warm standby“
BDD primaire
BDD cible
Copie vers le nouveau PGDATA
Fourni via „restore_com
Dans le recovery.conf
(e.g. cp, scp, rsync, …)
copie 1:1
de PGDATA
archive WAL

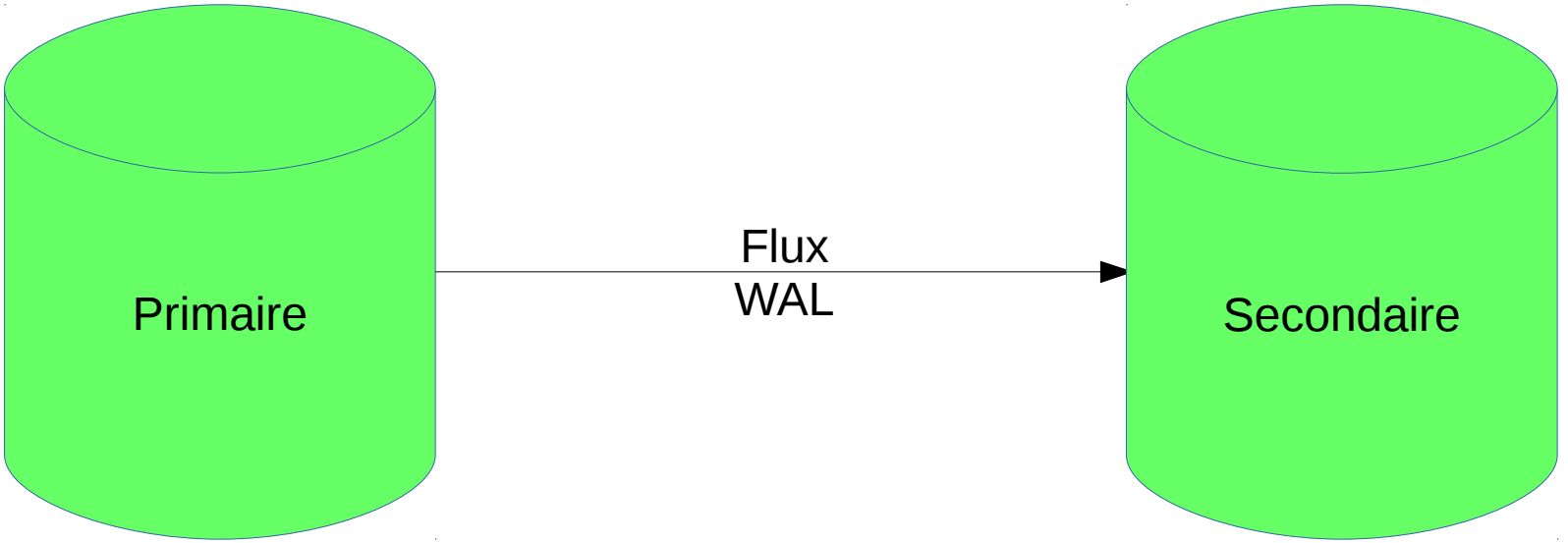
DIMA/PDMA

- DIMA
  - minutes à heures (cold standby)
  - secondes (warm standby)
- PDMA
  - le dernier segment de WAL archivé

- warm standby = „poor man's replication“

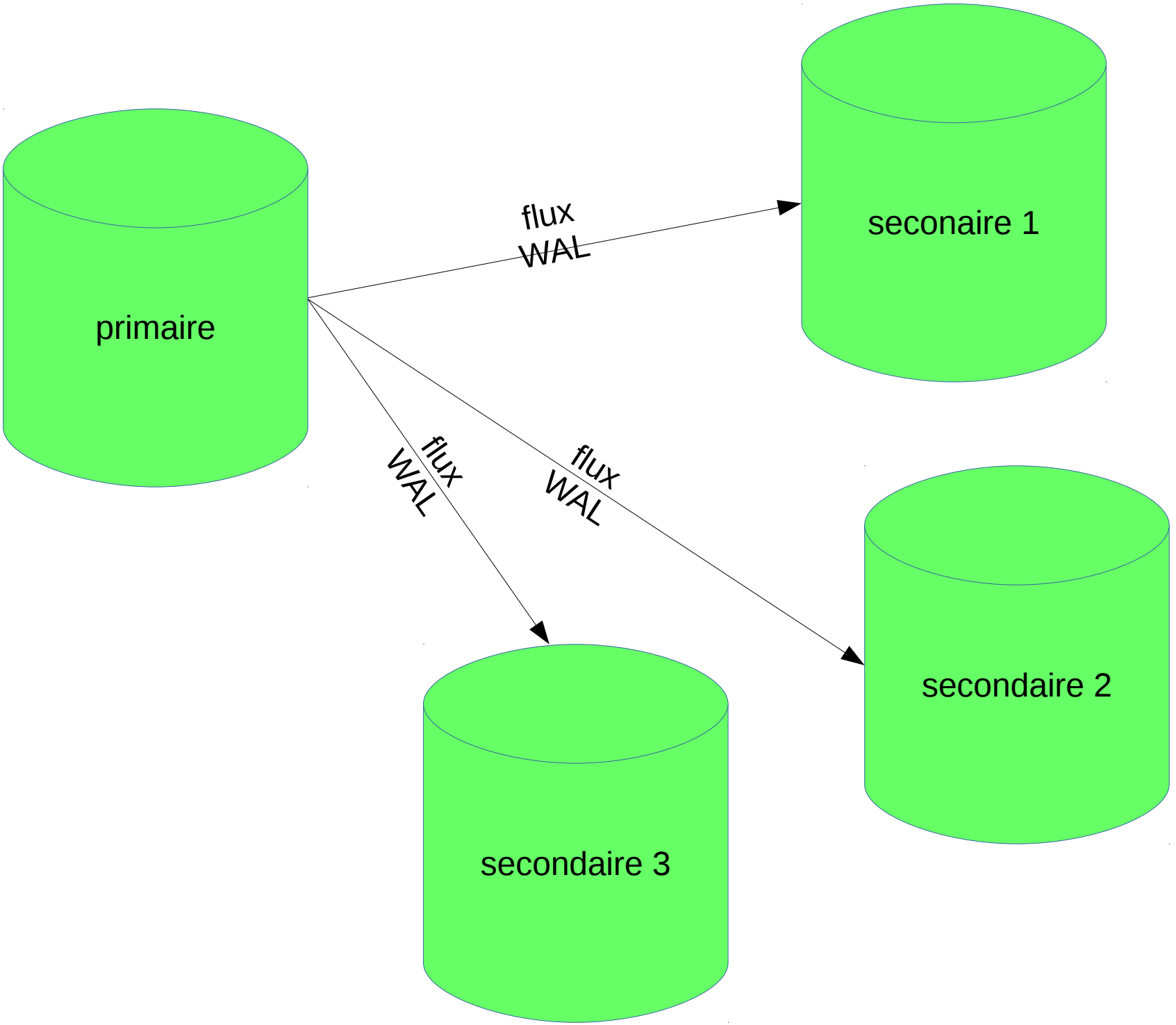
Réplication binaire en flux

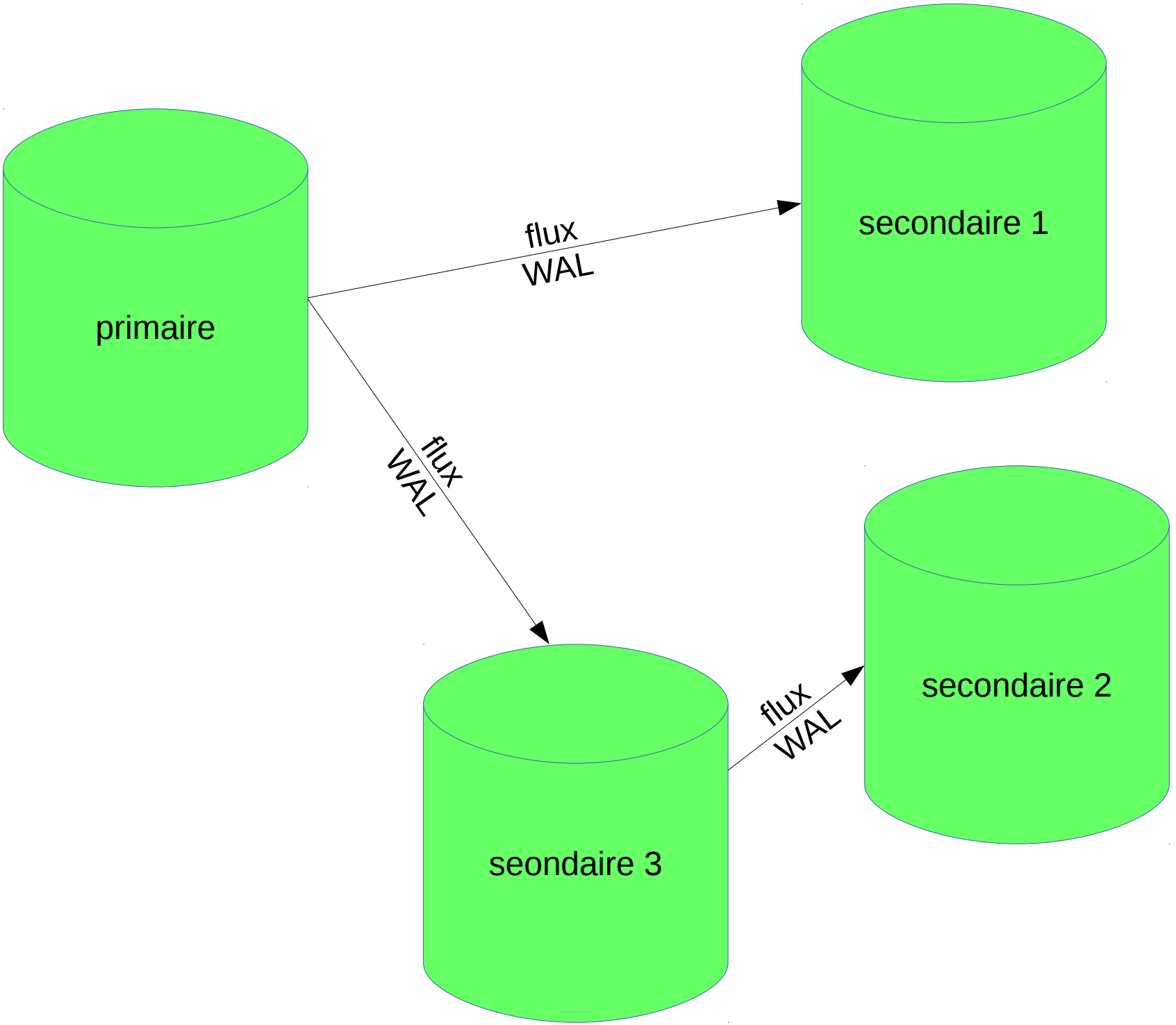
- Le flux (streaming) binaire est comme le warm standby server vu précedemment
- Mais avec les segments de WAL envoyés directement via le réseau
- Les transactions sont rejouées immédiatement
  - i.e., „ASAP“


Primaire
Flux
WAL
Secondaire

Parlons options !

- la réplication en flux peut-être synchrone ou asynchrone
  - choix par transaction !
  - choix entre remote_write & remote_apply
- peut utiliser des replication slots
- peut être en cascade
- les secondaires peuvent servir des requêtes de lecture
  - Vous pouvez les utiliser pour vos sauvegardes
- la réplication sur les secondaire peut être mis en pause (pas de demi wal transferé si l'arrêt du primaire est volontaire) **


primaire
flux
WAL
seconaire 1
flux
WAL
flux
WAL
secondaire 3
secondaire 2


primaire
flux
WAL
secondaire 1
flux
WAL
seondaire 3
flux
WAL
secondaire 2

Les pièges de la réplication synchrones

- Vous avez N secondaires synchrones
- Assurez vous de toujours avoir un total de N+1 secondaires
  - Si ça descend à N-1, votre BDD fonctionnera toujours
  - mais ne finira aucune transaction avant le retour à N ! *
- La latence réseau / le délai entre deux machines devient un problème !
  - donc choisissez prudemment (you can!) quelles transactions devraient être synchrones
  - et où placez vos secondaires synchrones

Avantages et inconvénients

+ Copie 1:1 de votre BDD, à chaud
+ Fiable & éprouvé
+ DIMA et PDMA très bonnes
+ très flexible

- fonctionne sur des cluster entiers uniquement
- conséquences en cas de perte de la connexion réseau

Donc, avec la réplication, ...

- Je n'ai plus besoin d'archiver les WAL, c'est ça ?

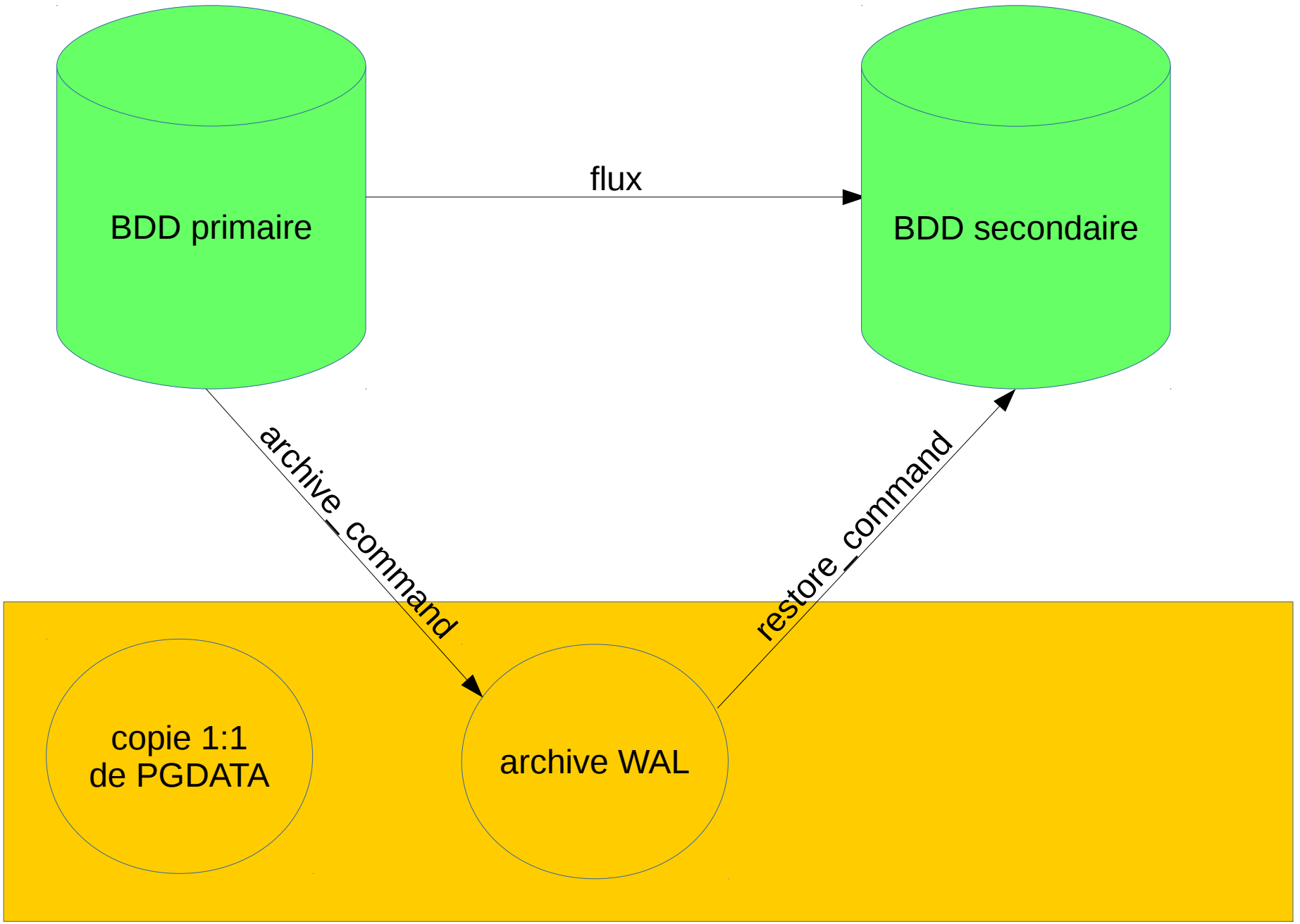
NAN mais ça va pas ?!?

Il faut qu'on parle...

- La réplication ne remplace pas les sauvegardes
- Et pendant qu'on y est : **
- Le RAID ne remplace does not replace les sauvegardes
- Un SAN ne remplace does not replace les sauvegardes
- „Le Cloud“ ne remplace does not replace les sauvegardes **

Assemblons le tout

- Vous voulez archivez vos WAL
- Vous voulez avoir (un/des) réplicat(s) secondaire(s)
  - peut-être plus qu'un
  - peut-être un synchrone
  - peut-être un en retard
  - peut-être en cascade
- DIMA : minimaliste
- PDMA :
  - la plus possible (secondaire synchrone)
  - la plus faisable (secondaire asynchrone)
- Protège des erreurs humaines (la DIMA augmente de toute évidence...)
- Autoriser les requêtes en lecture sur le(s) secondaire(s)


BDD primaire
flux
BDD secondaire
archive_command
restore_command
copie 1:1
de PGDATA
archive WAL

Avantages et inconvénients

+ tout est répliqué
+ tous les WAL sont archivés

- les versions majeures doivent être les mêmes

Configurez postgresql.conf

- `wal_level = replica # or logical already`
- `archive_mode = on   # always to cascade`
- `archive_command = /your/archive_script.sh %p %f`
- `max_wal_senders = 10 # or more`
- `max_replication_slots = 10 # or more`
- `synchronous_commit = local # for now`
- `synchronous_standby_names = '' | <set>`
- `hot_standby = on`
- `log_collector = on`

Régler vos archivages de WAL

- Ne réinventez pas ! **
  - Utilisez pgbarman, pgbackrest, WAL-E, …
  - Suivez leurs instructions
- Consacrez le temps économisé à penser redondance, persistence et sécurités des données
- Votre serveur de BDD n'est pas un bon choix pour cette archivage **
- Au sein du même centre de données est un mauvais choix (à moins que vous mirorriez)

/your/archive_script.sh

- La complexité n'a pas sa place dans archive_command
- Le script peut-être changé sans envoyer le signal HUP à votre BDD
- Le but du script : d'une manière où d'une autre mettre %p ($1) dans votre archivage WAL archive en tant que %f ($2)
- rsync n'est pas un mauvais choix, bien que :
  - assurez vous que %f n'existe pas dans votre archivage avant de commencez à envoyer
  - appelez sync à distance (ou montez votre archivage de façon synchronisé) après l'envoi
  - rsync à tendance à renvoyer de code de retour > 127, filtrez les
- Assurez vous que ça ne retourne pas, jamais des code de retours = 0 sans qu'il ait fait son boulot
  - À moins que vous soyez encore en train de le régler
  - „set -e“ etc.
  - Les erreurs finiront dans les logs Postgres (vu qu'on configure log_collector on)

Résumons

- Vous êtes probablement en train d'écrire dans le pagecache de votre système d'exploitation, et potentiellement asynchrone par là dessus (NFS) !
- Vos sauvegardes ne sont pas sûres tant qu'elle n'a pas été copié vers un stockage persistence dans un endroit sûr *
- Vos archives des segments WAL ne sont pas sûrs tant qu'ils n'ont pas été copié vers un stockage persistence dans un endroit sûr *
- Vous devrez faire des compromis, mais gardez en tête les conséquences sur votre DIMA **

Activez l'archivage

- Regardez le
- Postgres ne jettera pas des segments de WAL qu'il ne peut pas archiver
  - Votre PGDATA peut arriver à court d'espace disque !
- Les replication slots peuvent subir les mêmes conséquences, gardez ça en tête

Essayez une sauvegarde complète

- Vu que vous utilisez un outil**, vous êtes sur les starting blocks (droits, réplication permission, préparation, ...)
- Par exemple, faites un
  - `barman backup all`

Faites votre premier réplicat

- Ajoutez une ligne „replication“ dans le fichier `pg_hba.conf de votre serveur primaire`
- Préparez le nouveau répertoire PGDATA
  - e.g. sur Debian & dérivés, faites un `pg_createcluster` et `rm -rf` le résultat (vraiment)
  - Assurez vous que les fichiers `postgresql.conf` et autres réflète ceux de votre serveur primaire
- Lancez

  ```
  pg_basebackup -X stream -h <master> -U <user> -R -D <new_pgdata>
  ```
- Ajoutez une restore_command au fichier `recovery.conf résultant`
  - Qui récuperera les segments WAL de votre archivage
- Démarrez le secondaire, profitez du spectacle, corrigez, recommencez

Regardons les solutions

- e.g.
  - repmgr
  - PAF
  - pglookout
  - ...

Réplication logique

- Dans un passé pas si lointain...
-

Réplication logique

- Arrive dans le coeur de la 10.0
- Déjà disponible avec par exemple pglogical
- Si vous avez quelques Mo d'espace de sauvegarde disponible, configurez déjà
  - `wal_level = logical`
- Permet par exemple
  - mises à jour indolores avec peu d'indisponibilité
  - sharding
  - la collecte de données de différentes BDD dans une seul entrepôt de données
  - multi-maître
  - …

Quand ce sera dans le coeur

- Ce n'est pas encore clair, mais ça ressemblera un peu plus à

Famous last words

- Ne réinventez pas la roue !
- Testez votre procédure de sauvegarde !
- Testez votre procédure de restauration !!! **
- Supervisez vos logs et vos latences !
- Assurez vous que vos configurations sont synchronisées !
- Assurez vous que toute votre équipe comprend vos procédures de sauvegarde et restauration ! **
- En cas de problème *
  - keep calm and follow your procedures **

Merci de votre attention !