

Patroni + pgBackRest : le duo gagnant



data egret

Your remote PostgreSQL DBA team

Stefan FERCOT

stefan.fercot@dataegret.com

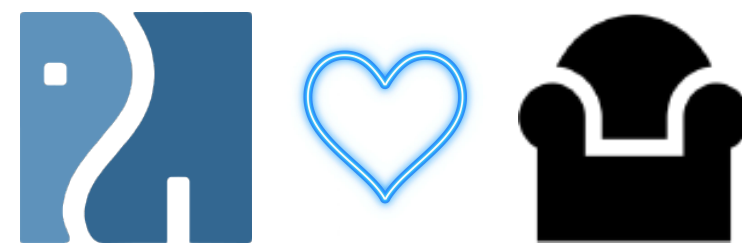
Stefan Fercot

- Consultant / Expert PostgreSQL @Data Egret
 - Conseil, support et formation PostgreSQL
- Fan & contributeur de pgBackRest
- aka. pgstef
- <https://pgstef.github.io>



Patroni et pgBackRest

Le duo gagnant



Ce que cette présentation n'est pas

- Un *cookbook* Patroni ou un guide d'installation pas à pas
- Une présentation exhaustive des fonctionnalités de pgBackRest
- Un substitut à la lecture de la documentation ;-)

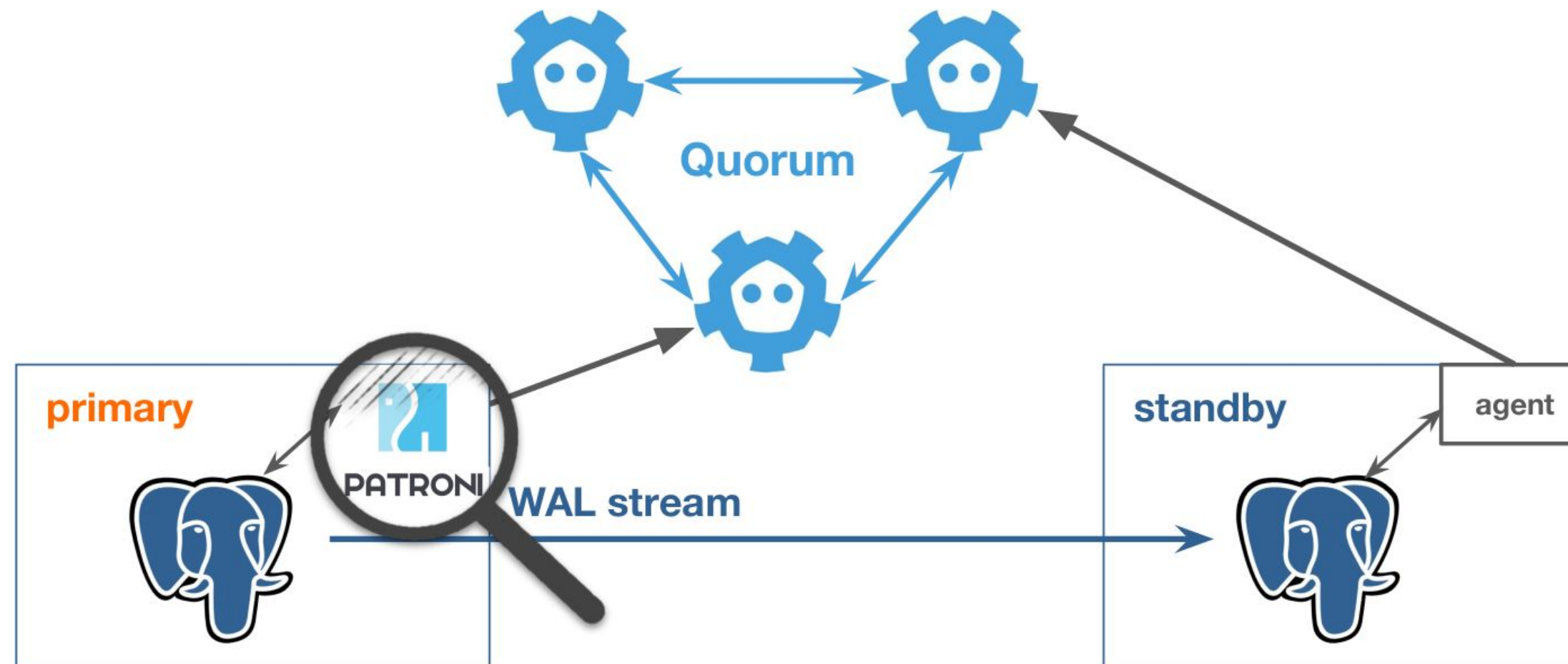
Programme du jour

- Que sont Patroni et pgBackRest ?
- Comment intégrer pgBackRest dans un cluster Patroni
 - Reconstruction d'un nœud standby et bootstrap d'un cluster
 - Réalisation d'une restauration *Point-In-Time* (PITR)

Patroni

- Framework Python pour construire des solutions HA PostgreSQL
- S'appuie sur un *Distributed Configuration Store* (DCS)
 - etcd, Consul ou ZooKeeper
- Chaque instance PostgreSQL est **exclusivement** pilotée par Patroni

Qu'est-ce que Patroni, vraiment ?



- Illustration par *Polina Bungina, PGConf.DE 2025*

Gestion de la configuration Patroni

https://patroni.readthedocs.io/en/latest/patroni_configuration.html

- Configuration dynamique globale
 - Stockée dans le DCS et appliquée à l'ensemble du cluster
 - Modifiable via `patronictl edit-config` ou l'API REST
- Fichier de configuration local (`/etc/patroni/config.yml`)
 - Prioritaire sur la configuration dynamique

pgBackRest

- Solution fiable de sauvegarde et de restauration pour PostgreSQL
- Fonctionne en local ou à distance (via SSH ou TLS)
- Opérations parallèles et asynchrones
- Compatible avec S3, Azure, GCS, NFS et d'autres backends de stockage
- Propose plusieurs méthodes de compression (`gz` , `bz2` , `lz4` , `zst`)
- Prend en charge le chiffrement côté client (`aes-256-cbc`)

Patroni + pgBackRest

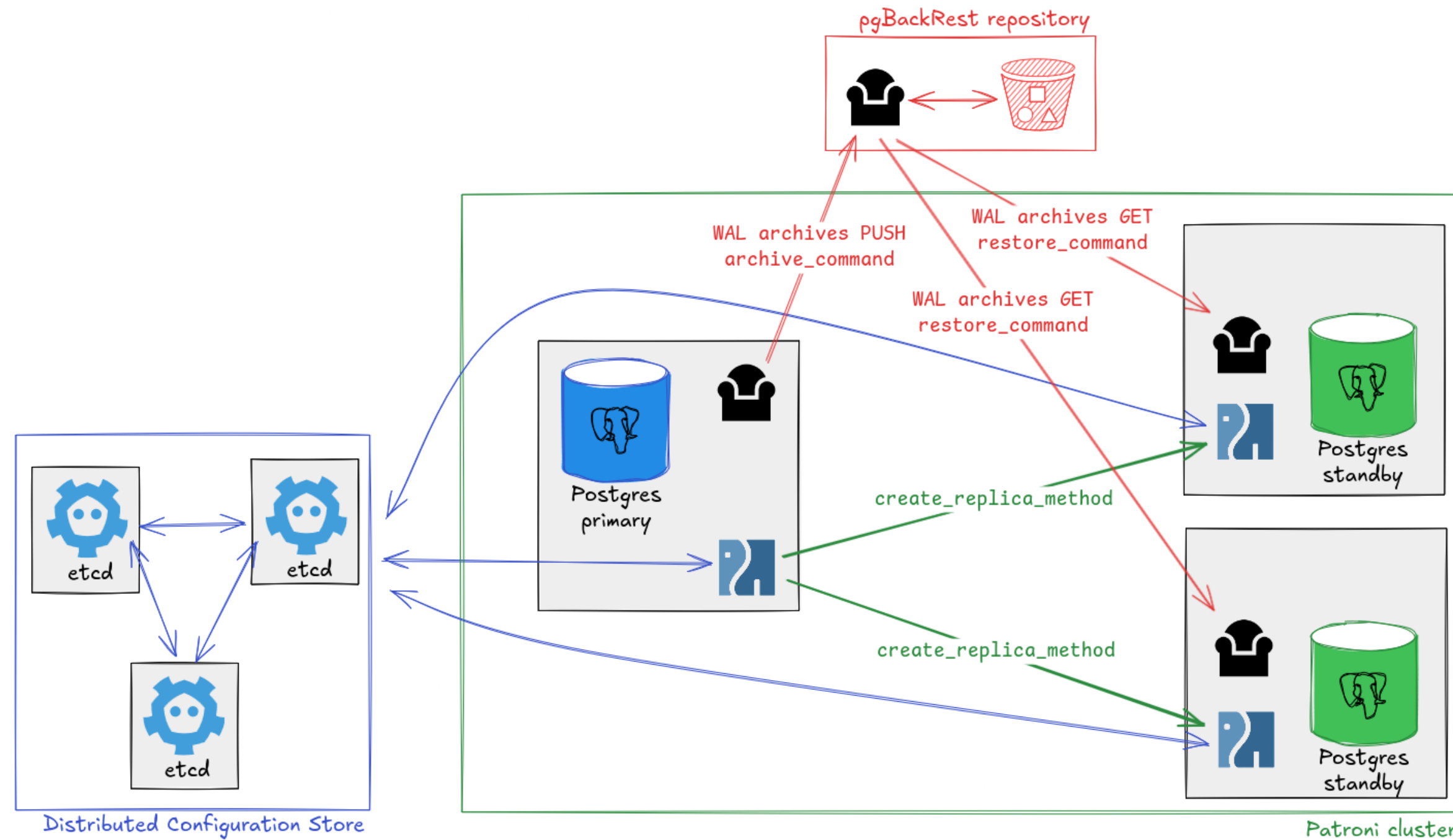


Quelques points importants avant de commencer

Avant d'intégrer pgBackRest avec Patroni, il est essentiel de bien concevoir votre topologie.

- Décider où seront stockées les sauvegardes
 - Le dépôt (*repository*) doit être accessible depuis tous les nœuds Patroni
- Choisir depuis quel nœud effectuer les sauvegardes
 - En général le primaire
 - Les sauvegardes depuis un standby sont également possibles
- L'utilisation d'un `repo-host` dédié simplifie grandement la coordination

Comment Patroni et pgBackRest fonctionnent ensemble



Intégration en pratique : assembler toutes les pièces

*Nous avons maintenant la vision d'ensemble,
voyons comment l'appliquer concrètement.*



Reconstruction des nœuds standby à partir d'une sauvegarde

- Utiliser pgBackRest comme `create_replica_method` dans Patroni
- Déclencher le processus de reconstruction avec `patronictl reinit`
- Contrôler le résultat à l'aide des logs et de l'état du cluster

Mise à jour de la configuration dynamique

- `archive_command` et `restore_command`

```
postgresql:  
  parameters:  
    archive_command: pgbackrest --stanza=demo archive-push %p  
    archive_mode: 'on'
```

```
recovery_conf:  
  restore_command: pgbackrest --stanza=demo archive-get %f %p
```

Mise à jour du fichier de configuration

```
postgresql:
  listen: 0.0.0.0:5432
  ...
  create_replica_methods:
    - pgbackrest
    - basebackup
  pgbackrest:
    command: pgbackrest restore --stanza=demo --delta --type=preserve
    keep_data: True
    no_params: True
  basebackup:
    checkpoint: 'fast'
```

- `--type=preserve` :
 - Indique à pgBackRest de ne pas générer la configuration de restauration
 - Permet à Patroni de gérer lui-même la configuration de recovery

Reconstruction d'un standby défaillant

- Exemple de cluster dans un état dégradé :

```
$ patronictl -c /etc/patroni/config.yml topology
+ Cluster: demo (7558759450904440237) -----+-----+-----+
| Member | Host                | Role   | State                | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg1     | 192.168.121.82      | Leader | running              | 1  |           |
| + pg2   | 192.168.121.90      | Replica | in archive recovery | 1  | 2476      |
+-----+-----+-----+-----+-----+-----+
```

```
LOG:  waiting for WAL to become available at 0/F2000078
P00   INFO: archive-get command begin 2.56.0: ...
P00   INFO: unable to find 000000010000000000000000F2 in the archive
P00   INFO: archive-get command end: completed successfully
LOG:  started streaming WAL from primary at 0/F2000000 on timeline 1
FATAL:  could not receive data from WAL stream: ERROR:  requested WAL segment 000000010000000000000000F2 has already been removed
LOG:  waiting for WAL to become available at 0/F2000078
```

patronictl reinit

<https://patroni.readthedocs.io/en/latest/patronictl.html#patronictl-reinit>

```
$ patronictl -c /etc/patroni/config.yml reinit demo pg2
+ Cluster: demo (7558759450904440237) -----+-----+-----+
| Member | Host                | Role   | State               | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg1     | 192.168.121.82      | Leader | running             | 1  |          |
| pg2     | 192.168.121.90      | Replica | in archive recovery | 1  |      2476 |
+-----+-----+-----+-----+-----+-----+
Are you sure you want to reinitialize members pg2? [y/N]: y
Success: reinitialize for member pg2
```

Est-ce que cela a fonctionné ?

- Toujours commencer par consulter les logs !

```
[12236] LOG:  database system is shut down
INFO: Leaving data directory uncleaned
P00 INFO: restore command begin 2.56.0: ...
P00 INFO: restore command end: completed successfully
INFO: replica has been created using pgbackrest
INFO: bootstrapped from leader 'pg1'
...
[12586] LOG:  starting backup recovery with redo LSN 1/3F0000D8, checkpoint LSN 1/3F000130, on timeline ID 1
[12586] LOG:  completed backup recovery with redo LSN 1/3F0000D8 and end LSN 1/3F0001D0
[12586] LOG:  consistent recovery state reached at 1/3F0001D0
...
[12753] LOG:  started streaming WAL from primary at 1/8C000000 on timeline 1
```

Création d'un nouveau cluster à partir d'une sauvegarde

- Définir une méthode de bootstrap personnalisée pour remplacer `initdb`
- Forcer la réinitialisation complète du cluster

Mise à jour de la configuration Patroni

- Ajouter dans la section `bootstrap:`

```
method: pgbackrest
pgbackrest:
  command: pgbackrest restore --stanza=demo
  keep_existing_recovery_conf: False
  no_params: True
  recovery_conf:
    recovery_target_action: promote
    recovery_target_timeline: latest
    restore_command: pgbackrest --stanza=demo archive-get %f %p
```

Nettoyage des nœuds

- Arrêter Patroni (ex. `systemctl stop patroni`)
- Nettoyer ou déplacer le contenu du répertoire de données
- Démarrer Patroni sur le premier nœud (ex. `systemctl start patroni`)
 - Si un cluster existait précédemment, le nœud apparaît en état `stopped`

```
$ patronictl -c /etc/patroni/config.yml list
+ Cluster: demo (7558759450904440237) -----+-----+-----+
| Member | Host                | Role    | State  | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg1     | 192.168.121.82      | Replica | stopped |    | unknown   |
+-----+-----+-----+-----+-----+-----+
```

Suppression de l'ancien cluster dans le DCS

- Nettoyer le DCS pour permettre un nouveau bootstrap

```
$ patronictl -c /etc/patroni/config.yml remove demo
+ Cluster: demo (7558759450904440237) -----+-----+-----+
| Member | Host                | Role    | State  | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg1     | 192.168.121.82      | Replica | stopped |    | unknown    |
+-----+-----+-----+-----+-----+-----+
Please confirm the cluster name to remove: demo
You are about to remove all information in DCS for demo, please type: "Yes I am aware": Yes I am aware
```

```
$ patronictl -c /etc/patroni/config.yml list
+ Cluster: demo (7558759450904440237) -----+-----+-----+
| Member | Host                | Role    | State  | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg1     | 192.168.121.82      | Leader  | running | 2  |           |
+-----+-----+-----+-----+-----+-----+
```

Vérifier les logs Patroni lors de l'initialisation

```
INFO: trying to bootstrap a new cluster
INFO: Running custom bootstrap script: pgbackrest restore --stanza=demo
P00 INFO: restore command begin 2.56.0: ...
P00 INFO: restore command end: completed successfully
[13289] LOG: starting backup recovery with redo LSN 1/C00000D8, checkpoint LSN 1/C0000130, on timeline ID 1
[13289] LOG: starting archive recovery
[13289] LOG: completed backup recovery with redo LSN 1/C00000D8 and end LSN 1/C00001D0
[13289] LOG: consistent recovery state reached at 1/C00001D0
[13289] LOG: selected new timeline ID: 2
[13289] LOG: archive recovery complete
[13282] LOG: database system is ready to accept connections
INFO: establishing a new patroni heartbeat connection to postgres
INFO: initialized a new cluster
```

Démarrer les autres nœuds

- Activer Patroni et vérifier les logs

```
INFO: trying to bootstrap from leader 'pg1'
P00 INFO: restore command begin 2.56.0: ...
P00 INFO: restore command end: completed successfully
INFO: replica has been created using pgbackrest
INFO: bootstrapped from leader 'pg1'
[12904] LOG:  consistent recovery state reached at 1/C00001D0
[12897] LOG:  database system is ready to accept read-only connections
[12923] LOG:  started streaming WAL from primary at 1/C2000000 on timeline 2
```

```
$ patronictl -c /etc/patroni/config.yml list
+ Cluster: demo (7558759450904440237) -----+-----+-----+
| Member | Host                | Role    | State        | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pg1    | 192.168.121.82      | Leader  | running      | 2  |           |
| pg2    | 192.168.121.90      | Replica | streaming    | 2  |          0 |
+-----+-----+-----+-----+-----+-----+
```



Bonus : un excellent moyen de tester vos sauvegardes

- Créer un nouveau cluster à partir d'une sauvegarde
 - est également une **méthode simple et efficace pour valider vos backups**
- En pratique :
 - Restaurer sur un cluster de test dédié
 - Mettre en place un accès **en lecture seule** au dépôt
 - Désactiver `archive_mode` (par exemple avec `--archive-mode=off`)

Un test de restauration est la seule véritable preuve que vos sauvegardes fonctionnent.

Réaliser un PITR avec Patroni

- Comment déclencher un PITR de manière sûre ?
 - Utiliser la méthode de bootstrap personnalisée !

Identifier la cible de restauration

```
importantdb=# SELECT pg_current_wal_lsn();  
pg_current_wal_lsn  
-----  
1/C2006E10
```

```
importantdb=# DELETE FROM important_table;  
DELETE 1
```

```
importantdb=# SELECT pg_switch_wal();  
pg_switch_wal  
-----  
1/C2006F08
```

Ajuster la méthode de bootstrap

- Modifier la configuration Patroni afin d'inclure la cible de restauration souhaitée

```
method: pgbackrest
pgbackrest:
  command: pgbackrest restore --stanza=demo --type=lsn --target="1/C2006E10"
  keep_existing_recovery_conf: True
  no_params: True
  recovery_conf:
    recovery_target_action: promote
    recovery_target_timeline: latest
    restore_command: pgbackrest --stanza=demo archive-get %f %p
```

Suivre les étapes de réinitialisation du cluster

- Arrêter Patroni
- Nettoyer le répertoire de données
- Démarrer Patroni
- Supprimer l'ancien cluster du DCS

Vérifier les logs

- Les logs doivent confirmer la restauration au LSN spécifié

```
INFO: trying to bootstrap a new cluster
INFO: Running custom bootstrap script: pgbackrest restore --stanza=demo --type=lsn --target="1/C2006E10"
P00  INFO: restore command begin 2.56.0: ...
P00  INFO: restore command end: completed successfully
[13462] LOG:  starting point-in-time recovery to WAL location (LSN) "1/C2006E10"
[13462] LOG:  recovery stopping after WAL location (LSN) "1/C2006E10"
[13462] LOG:  selected new timeline ID: 3
[13462] LOG:  archive recovery complete
[13455] LOG:  database system is ready to accept connections
INFO: establishing a new patroni heartbeat connection to postgres
INFO: initialized a new cluster
```

Conclusion

Patroni assure le fonctionnement des bases de données, pgBackRest les ramène à la vie.

- Conservez une configuration simple, explicite et versionnée
- L'automatisation sans observabilité relève de la foi aveugle
- Testez toujours vos procédures de restauration : la documentation ne suffit pas



Questions ?

contact@dataegret.com



Annexe bonus : `config.yml` Patroni

Un exemple fonctionnel issu d'un cluster de test, découpé en parties lisibles
(Parce que tout le monde aime un exemple de configuration qui fonctionne)

Partie 1 : identité du cluster et etcd

```
scope: demo
namespace: /db/
name: pg1

restapi:
  listen: 0.0.0.0:8008
  connect_address: pg1:8008
  authentication:
    username: patroni
    password: mySupeSecretPassword

etcd3:
  hosts:
    - etcd1:2379
    - etcd2:2379
    - etcd3:2379
```

Partie 2 : configuration de base du bootstrap

```
bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 1048576
    postgresql:
      use_pg_rewind: false
      use_slots: false

    parameters:
      archive_mode: "on"
      archive_command: "pgbackrest --stanza=demo archive-push %p"
    recovery_conf:
      restore_command: "pgbackrest --stanza=demo archive-get %f %p"

initdb:
- encoding: UTF8
- data-checksums
- auth-local: peer
- auth-host: scram-sha-256
```

Partie 3 : règles HBA et bootstrap personnalisé

```
pg_hba:  
- host replication replicator 0.0.0.0/0 scram-sha-256  
- host all all 0.0.0.0/0 scram-sha-256  
  
method: pgbackrest  
pgbackrest:  
  command: pgbackrest restore --stanza=demo  
  keep_existing_recovery_conf: false  
  no_params: true  
  recovery_conf:  
    recovery_target_action: promote  
    recovery_target_timeline: latest  
    restore_command: pgbackrest --stanza=demo archive-get %f %p
```

Partie 4 : PostgreSQL local et création des replicas

```
postgresql:
  listen: 0.0.0.0:5432
  connect_address: pg1:5432
  data_dir: /var/lib/postgresql/18/main
  bin_dir: /usr/lib/postgresql/18/bin
  pgpass: /tmp/pgpass0
  authentication:
    replication:
      username: replicator
      password: confidential
    superuser:
      username: postgres
      password: my-super-password
  create_replica_methods:
    - pgbackrest
    - basebackup
  pgbackrest:
    command: pgbackrest restore --stanza=demo --delta --type=preserve
    keep_data: true
    no_params: true
  basebackup:
    checkpoint: 'fast'
```

Partie 5 : watchdog et tags

```
watchdog:  
  mode: required  
  device: /dev/watchdog  
  safety_margin: 5  
  
tags:  
  nofailover: false  
  noloadbalance: false  
  clonefrom: false  
  nosync: false
```

SECURING YOUR POSTGRESQL DATABASE AVAILABILITY AND HIGH PERFORMANCE

- **Migration**
- Performance **audit**
- **Cloud Cost** management
- **Backup & restore**
- **Architecture** review
- **DataOps/CDC** projects
- **Consulting** for data science and analytics teams
- **PostgreSQL Courses** for DBA and Developers



MITGLIED



EXPERTISE

Senior DBA with 10+ years of experience in PostgreSQL administration.



DEVELOPMENT

PostgreSQL Contributors involved in new PostgreSQL feature and extension development.



CUSTOMISATION

Flexible approach and dedicated team focused on success of your project.



COMMUNITY

Recognised as Significant Contributing Sponsor to PostgreSQL.