# Patroni and pgBackRest: Better together



data egret

Your remote PostgreSQL DBA team

Stefan FERCOT

stefan.fercot@dataegret.com

# 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

Contributing sponsor to PostgreSQL

Open Alliance
For PostgreSQL Education

OSB Open Source Business ALLIANCE
Bundesverband Für digitale Souveränität e.V.
MITGLIED


**EXPERTISE**

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

**DEVELOPMENT**

PostgreSQL Contributors involved in new PostgreSQL feature and extention development.

**CUSTOMISATION**

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

**COMMUNITY**

Recognised as Significant Contributing Sponsor to PostgreSQL.

# Stefan Fercot

- PostgreSQL Expert **@Data Egret**
  - PostgreSQL consulting, support, and training
- pgBackRest contributor and advocate
- Also known as pgstef
- https://pgstef.github.io

# Patroni and pgBackRest

## Working better together

# What this talk is NOT?

- A Patroni cookbook or installation recipe
- A pgBackRest feature tour or deployment best practices session
- A replacement for reading the documentation ;-)

# What happens without integration?

- Lots of GitHub issues…
  - *Restore with `--type=immediate` still replays WALs under Patroni*

> *Despite using `--type=immediate`, the change made after the backup still appears in the restored database — suggesting WAL files are still being applied.*
> *We expect `--type=immediate` to skip all WAL replay, but in this environment (Patroni), it seems WALs are reapplied automatically.*
>
> *The behavior is not observed when Patroni is not involved — restore works as expected in a plain PostgreSQL setup.*

# Kudos

- Julia Gugel
  - https://postgresql.life/post/julia_gugel/
- Federico Campoli
  - https://postgresql.life/post/federico_campoli/

# Today’s agenda

- What are Patroni and pgBackRest?
- How to integrate pgBackRest in a Patroni cluster
  - Rebuilding a standby node and bootstrapping a cluster
  - Performing a point-in-time recovery (PITR)

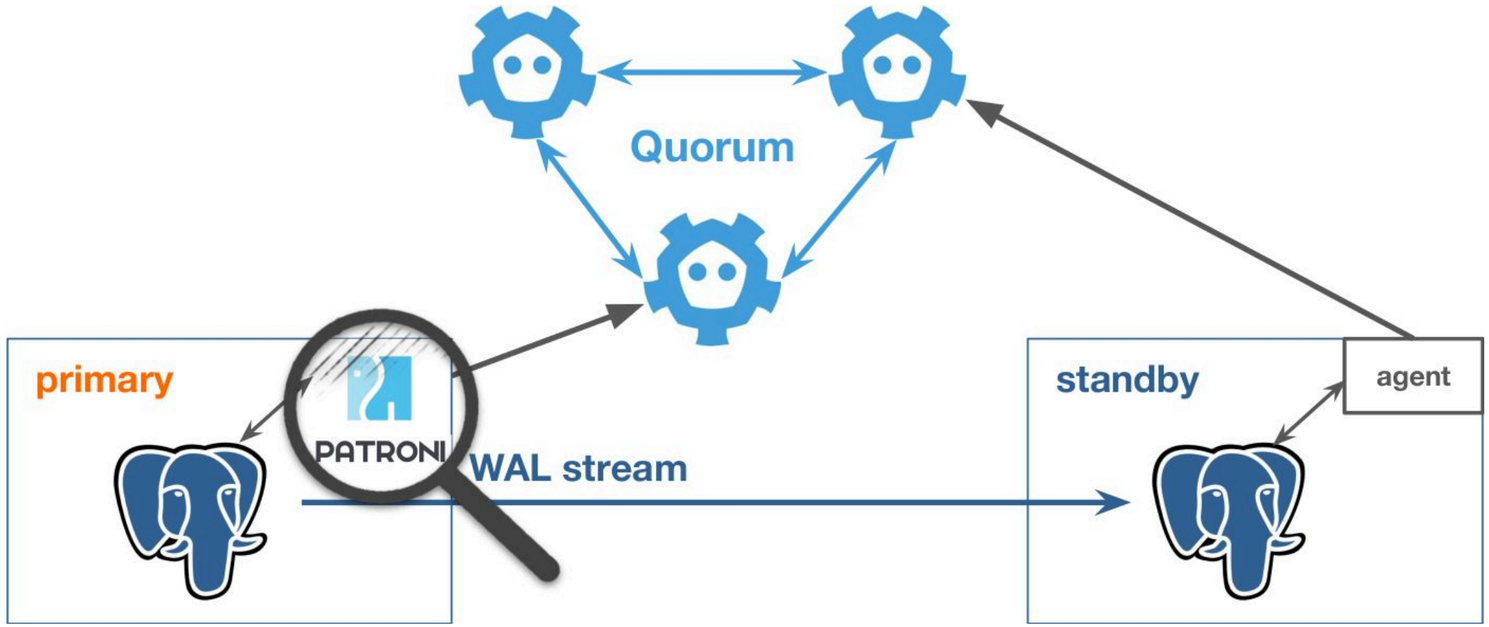
# Patroni

- Template for building PostgreSQL HA solutions in Python
- Depends on a *Distributed Configuration Store*
  - etcd, Consul, or ZooKeeper
- Works on physical servers, VMs, or containers
- Supports manual switchovers and automatic failovers

# General architecture

- Each PostgreSQL instance is managed exclusively by Patroni
- Requires two separate server clusters
  - DCS
  - Patroni/PostgreSQL nodes
- Best practice: run DCS on nodes separate from Patroni/PostgreSQL

# What is Patroni, really?




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

# Patroni configuration management

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

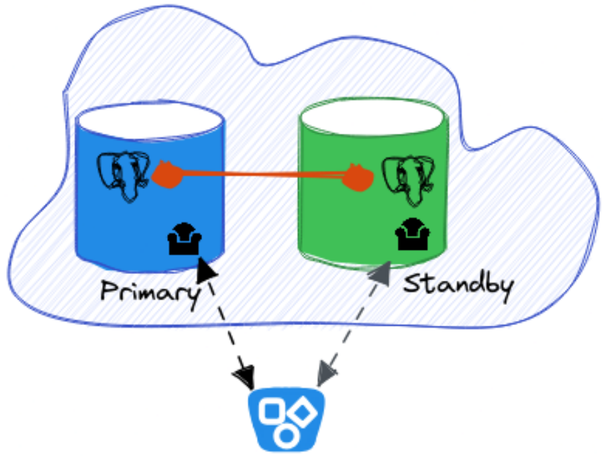
- Global dynamic configuration
  - Stored in the DCS and applied across all cluster nodes
  - Modified using `patronictl edit-config` or the REST API
- Local configuration file ( `/etc/patroni/config.yml` )
  - Takes precedence over dynamic configuration

# pgBackRest

- Reliable backup and restore solution for PostgreSQL
- Supports local or remote operation (via SSH or TLS)
- Parallel and asynchronous operations
- Works with S3, Azure, GCS, NFS, and other storage backends
- Offers multiple compression methods ( `gz` , `bz2` , `lz4` , `zst` )
- Supports client-side encryption ( `aes-256-cbc` )

# Integrating with a standby server

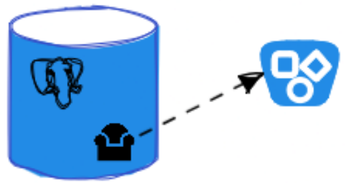
- The backup repository must be reachable from both nodes
- Enables taking backups from the standby while archiving from the primary



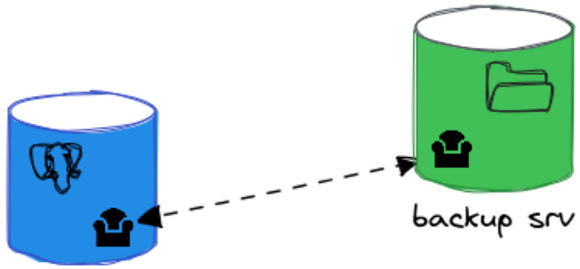
# Where to store your backups

> *Choosing the right location for your pgBackRest backups is an important architectural decision.*

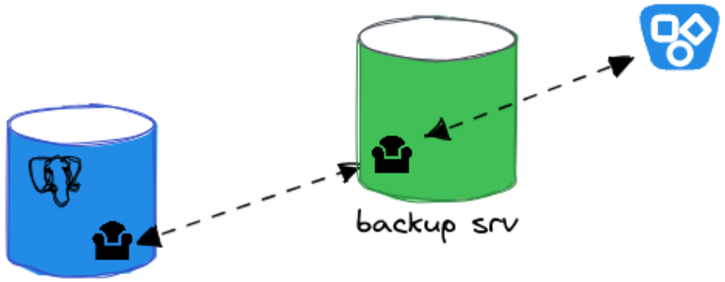
# Direct access from the PostgreSQL host



# Using a dedicated backup host



# Combining backup host and cloud storage



# Patroni + pgBackRest

# Keep some things in mind before you start

> *Before integrating pgBackRest with Patroni, plan your topology carefully.*

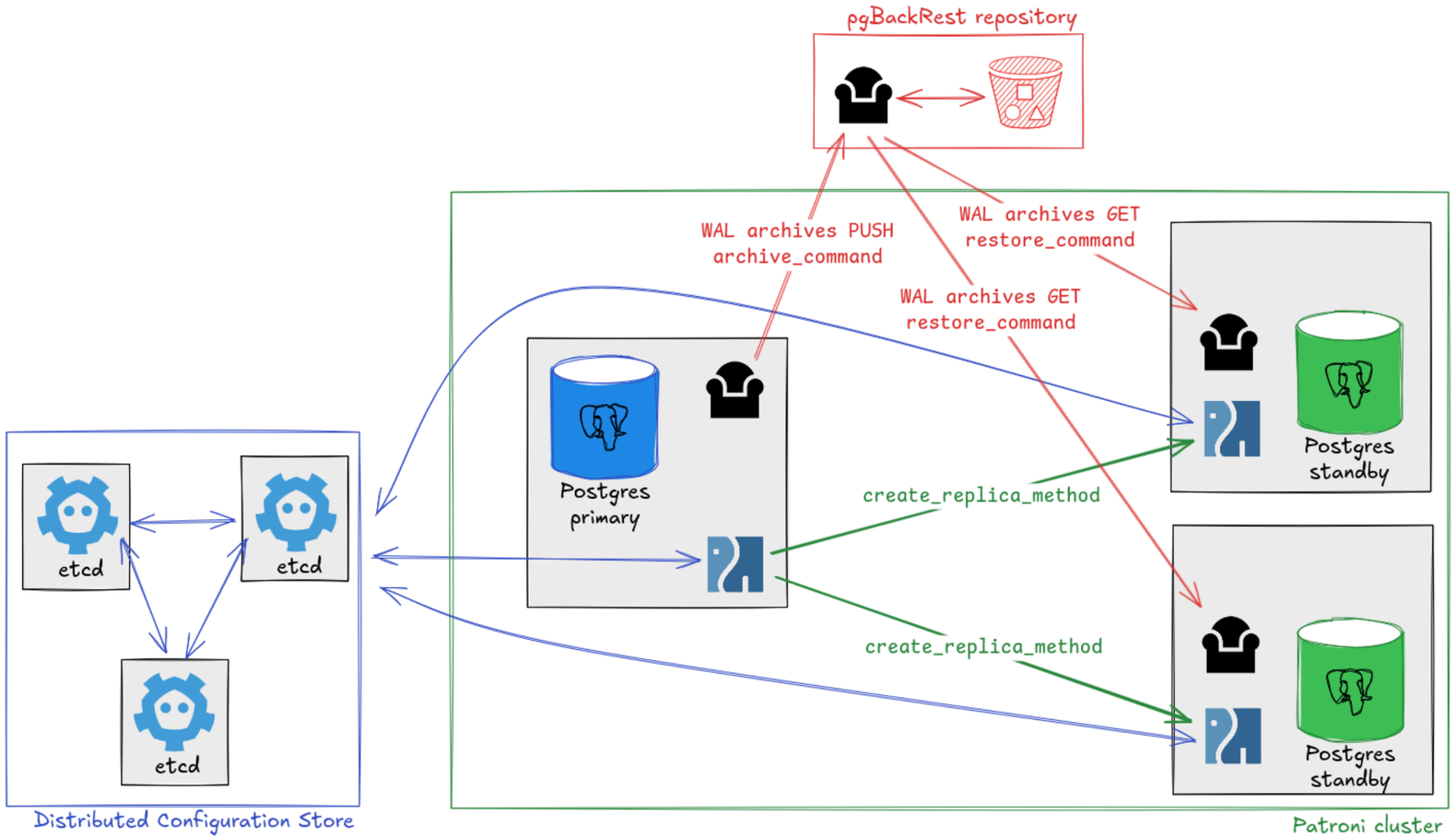
- Decide where your backups will be stored
  - The repository must be reachable from all Patroni nodes
- Choose from which node to take backups
  - Typically the primary, but standby backups are also possible
- Using a dedicated `repo-host` simplifies coordination

# Gotcha: version mismatch strikes again

- pgBackRest requires its version to match
  - Across the PostgreSQL instance and the dedicated *repository host*
- Common error message:

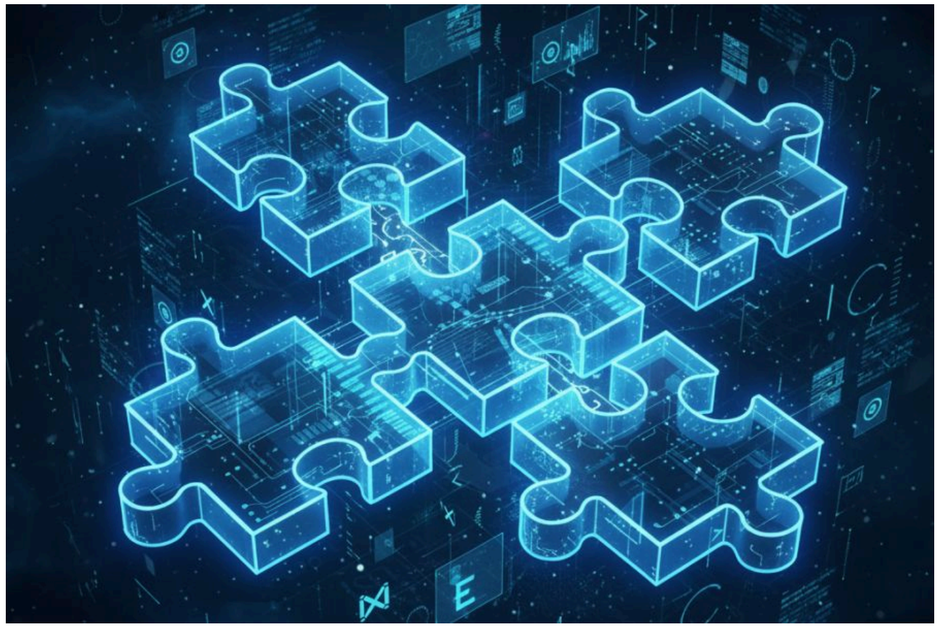
```
WARN: repo1: [ProtocolError] expected value '2.54.2' for greeting key 'version' but got '2.56.0'
  HINT: is the same version of pgBackRest installed on the local and remote host?
```

# How Patroni and pgBackRest work together



# Integration in action: putting it all together

*Now that we understand the architecture, let's see how to apply it in practice.*



# Rebuilding standby nodes from backup

- Use pgBackRest as a `create_replica_method` in Patroni
- Trigger the rebuild process with `patronictl reinit`
- Validate success through logs and cluster status

# Updating dynamic configuration

- `archive_command` and `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
```

# Updating the configuration file

```
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`:
  - Tells pgBackRest not to generate recovery settings
  - Allows Patroni to handle recovery configuration

# Rebuilding the failing standby

- Example of an unhealthy cluster:

```
$ 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 0000000100000000000000F2 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 0000000100000000000000F2 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
```

# Did it work?

- Always look at the 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
```

# Creating a new cluster from a backup

- Define a custom bootstrap method to replace `initdb`
- Force a cluster re-initialization

# Update the Patroni configuration

- Add to the `bootstrap:` section

```
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
```

# Clean nodes

- Stop Patroni (e.g., `systemctl stop patroni`)
- Clean or move the content of the data directory
- Start Patroni on the first node (e.g., `systemctl start patroni`)
  - If a previous cluster exists, the node will appear in a `stopped` state

```
$ 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 |
+--------+------------------+---------+---------+----+-----------+
```

# Remove the old cluster from the DCS

- Delete the old cluster information from the DCS to allow a clean 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 |           |
+--------+-------------------+--------+---------+----+-----------+
```

# Check Patroni logs for initialization

```
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
```

# Start the other nodes

- Start Patroni on the other nodes and look at the 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 point: a great way to test your backups

- Creating a new cluster from a backup
  - is also a **simple way to verify backups** on test systems
- In practice:
  - Restore onto a separate test cluster
  - Set up **read-only access** to the repository
  - Turn off `archive_mode` (i.e. with `--archive-mode=off`)

> *A restore test is the only real proof that your backups work.*

# Performing PITR with Patroni

- How to trigger PITR safely?
  - Use the custom bootstrap method!

# Identify your recovery target

```
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
```

# Adjust the bootstrap method

- Modify the Patroni configuration to include the desired recovery target

```
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
```

## Follow the cluster reset steps

- Stop Patroni
- Clean the data directory
- Start Patroni
- Remove the old cluster from the DCS

# Check the logs

- The logs should confirm the recovery to the specified LSN

```
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 keeps databases running, pgBackRest brings them back.*

- Keep your configuration simple, explicit, and versioned
- Automation without observability is blind faith
- Always test your recovery procedures, documentation is not enough

# Questions?

contact@dataegret.com

Feedback:

# Bonus appendix: Patroni `config.yml`

A working example from a test cluster, split into readable parts

*(Because everyone loves a working config example)*

# Part 1: cluster identity and 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
```

# Part 2: bootstrap base configuration

```
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
```

# Part 3: HBA rules and custom bootstrap method

```
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
```

# Part 4: local PG settings and replica creation method

```
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'
```

# Part 5: watchdog and tags

```
watchdog:
  mode: required
  device: /dev/watchdog
  safety_margin: 5

tags:
  nofailover: false
  noloadbalance: false
  clonefrom: false
  nosync: false
```