# pgBackRest in HA setups: Deployment patterns that work



data egret

Your remote PostgreSQL DBA team

Stefan FERCOT

stefan.fercot@dataegret.com

# Securing your PostgreSQL database availability and high performance


MIGRATION
POSTGRESQL SUPPORT
CLOUD COST MANAGEMENT

## data egret

Your PostgreSQL DBA team

### EXPERTISE

- DBA with 10+ years of experience in PostgreSQL administration.
- Significant Contributors

### EDUCATION

Co-founders of Open Alliance for PostgreSQL Education

Open Alliance
For PostgreSQL Education

### OPEN SOURCE ADVOCATES

Accompanying you on your journey to Open Source PostgreSQL

OSBA Mitglied

### COMMUNITY

Recognised as Significant Contributing Sponsor to PostgreSQL.

COURSES
FOR DBA
AND DEVELOPERS

Scan
to explore
curriculum
→

# Stefan Fercot

- PostgreSQL Expert **@Data Egret**
  - PostgreSQL consulting, support, and training
- Significant contributor to the PostgreSQL Project
- Vice-Treasurer, PostgreSQL Europe
- pgBackRest fan, advocate & sponsor
- Also known as pgstef
- https://pgstef.github.io

# Designing backup systems in HA environments

*So, what are we going to talk about?*

# pgBackRest

- Reliable backup and restore tool for PostgreSQL
- MIT licensed
- Supports local and remote operation (SSH / TLS)
- Works with S3, Azure, GCS, NFS, and more
- Parallel and asynchronous operations
  - See pgBackRest magic you'll wish you knew sooner

> *Flexible, powerful… and easy to underuse*

# Are you using pgBackRest?

- Widely used in production
- Consider sponsoring the project ❤️

https://github.com/sponsors/dwsteele

# Agenda

> *How do we design systems that survive failure?*

- pgBackRest is flexible…
  - …and that flexibility creates complexity
- Designing backup architectures
  - HA <> Backup <> Disaster Recovery
  - Deployment patterns in HA setups
  - Strengths and weaknesses
- Towards better designs
  - Decoupling execution from storage

# pgBackRest key building blocks

- Repository (local / remote / object storage)
- WAL archives (the backbone)
- Backup sets (full / diff / incr)
- Stanza (scope)

# Critical design levers

Everything comes down to three questions:

- *Where is the repository?*
- *How do WAL archives get there?*
- *Who runs the backups?*

# Pattern 1 - classic repo host

Dedicated repository host (storage + execution)

# Configuration example
# Repository host

```
[global]
repo1-path=/srv/pgbackrest/repo1
repo1-retention-full=2

[demo]
pg1-host=pg1
pg1-path=/var/lib/postgresql/18/demo
```

# Configuration example PostgreSQL host

```
[global]
repo1-host=repository
repo1-host-user=pgbackrest

[demo]
pg1-path=/var/lib/postgresql/18/demo
```

Requires an SSH connection between the hosts

# Strengths

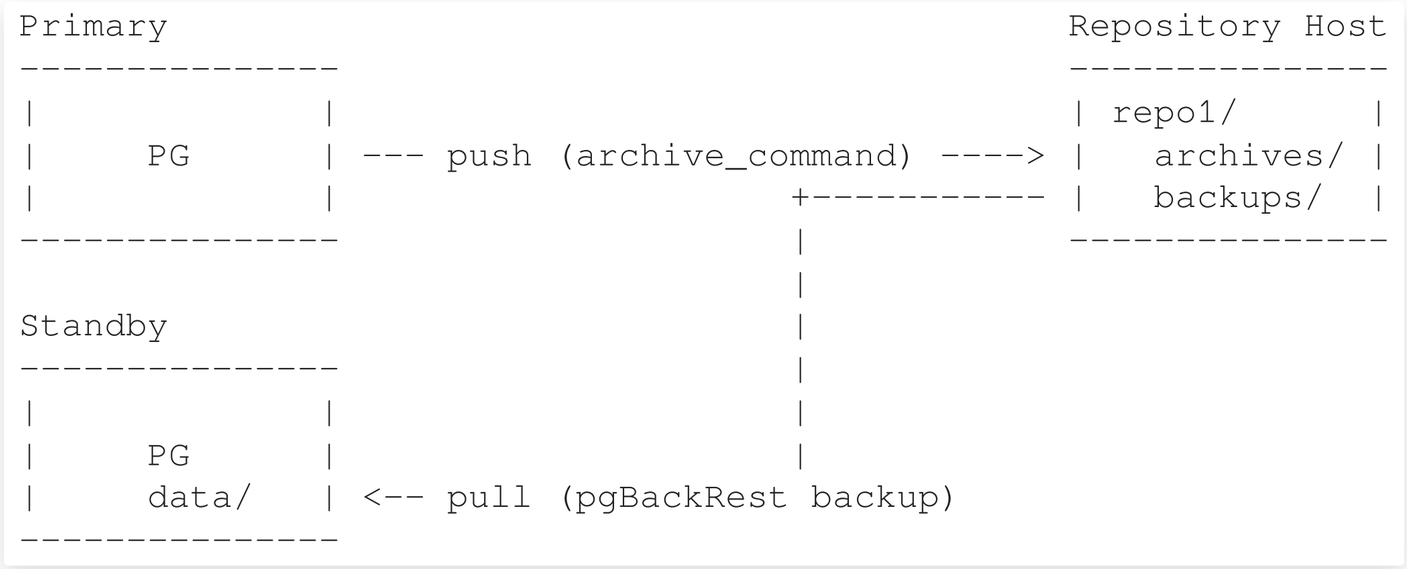
- Good isolation
- Simple mental model

# Weaknesses

- pgBackRest versions must match
- Single point of failure (repo host)
  - No restore possible
  - WAL archiving **stops**

# In HA setups

## Adds the possibility to take backups from a standby



# Configuration adjustment
# Repository host

```
[demo]
backup-standby=y
pg1-host=pg1
pg1-path=/var/lib/postgresql/18/demo
pg2-host=pg2
pg2-path=/var/lib/postgresql/18/demo
```

```
P00   INFO: wait for replay on the standby to reach 0/7000028
P00   INFO: replay on the standby reached 0/7000028
```

# Strengths

- Reduces load on primary
- pgBackRest auto-selects where to run the backup
  - `backup-standby=n/y/prefer`
- Best balance for many production setups
  - Common in serious HA environments

# Gotcha

- Backup from standby is **not independent**
- WAL still comes from primary
- If repo host dies, WAL archiving is blocked

# Pattern 2 - cloud storage backend

- Backups stored directly in object storage (S3/Azure/GCS)
- PostgreSQL hosts push WAL archives and backups directly to cloud

# Configuration example PostgreSQL host

```
[global]
repo1-type=s3
repo1-s3-endpoint=s3endpoint
repo1-s3-region=eu-central-1
repo1-s3-bucket=pgbackrest
repo1-path=/repo1
repo1-bundle=y
repo1-retention-full=2

[demo]
pg1-path=/var/lib/postgresql/18/demo
```

# Strengths

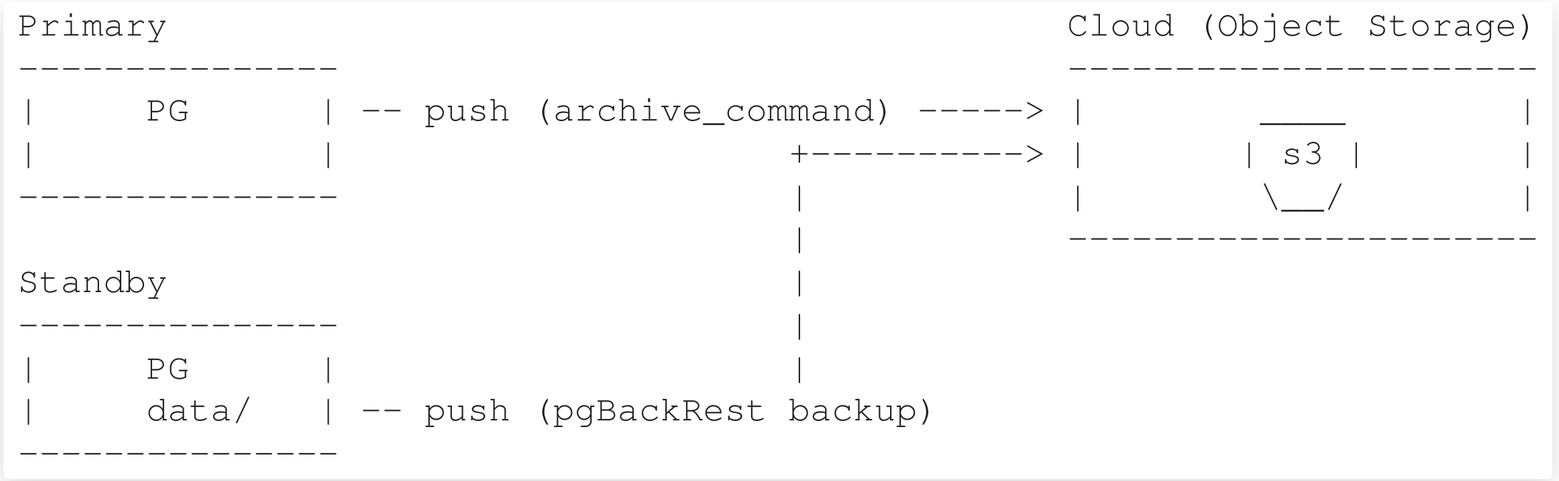
- Durability
- Off-site by design

# Weaknesses

- Restore speed
- Cost unpredictability
- External dependency

> *“S3 is not a backup strategy — it’s a storage backend”*

# In HA setups


Primary
PG
-- push (archive_command) ----->
Cloud (Object Storage)
s3
Standby
PG
data/
-- push (pgBackRest backup)

# Configuration adjustment
# Standby server

```
[demo]
backup-standby=y
pg1-host=pg1
pg1-path=/var/lib/postgresql/18/demo
pg2-path=/var/lib/postgresql/18/demo
```

```
P00   INFO: wait for replay on the standby to reach 0/D000028
P00   INFO: replay on the standby reached 0/D000028
```

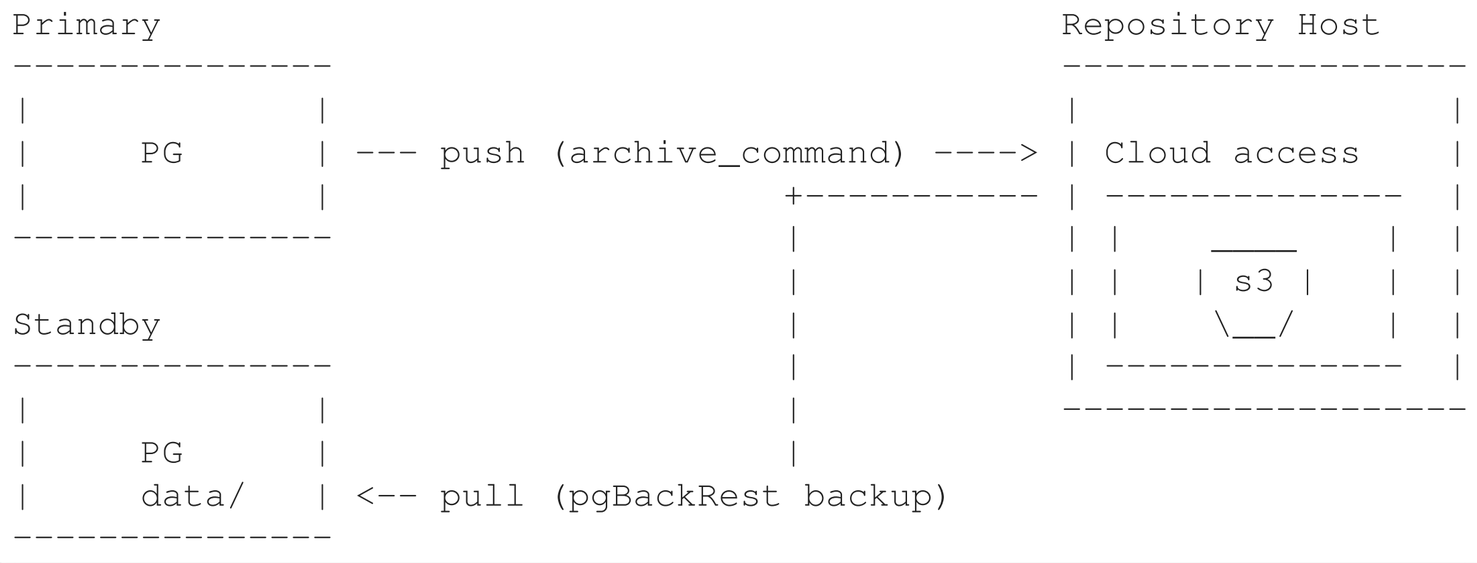
Requires an SSH connection between primary and standby(s)

# Gotcha

- *Where to run the backup?*
  - Single standby = easy
  - Multiple standbys = tricky
  - **Solution**: introduce repo host for orchestration

# Pattern 3 - hybrid

Combine repo host, standby server(s) and cloud storage backend



# Configuration example
# Repository host

```
[global]
repo1-type=s3
repo1-s3-endpoint=s3endpoint
repo1-s3-region=eu-central-1
repo1-s3-bucket=pgbackrest
repo1-path=/repo1
repo1-bundle=y
repo1-retention-full=2

[demo]
backup-standby=prefer
pg1-host=pg1
pg1-path=/var/lib/postgresql/18/demo
pg2-host=pg2
pg2-path=/var/lib/postgresql/18/demo
```

# Configuration example PostgreSQL host

```
[global]
repo1-host=repository
repo1-host-user=pgbackrest

[demo]
pg1-path=/var/lib/postgresql/18/demo
```

# Strengths

- Centralized access to cloud storage
- Centralized backup orchestration

# Weaknesses

- Repo host is still a bottleneck / SPOF

# Decoupling execution from storage

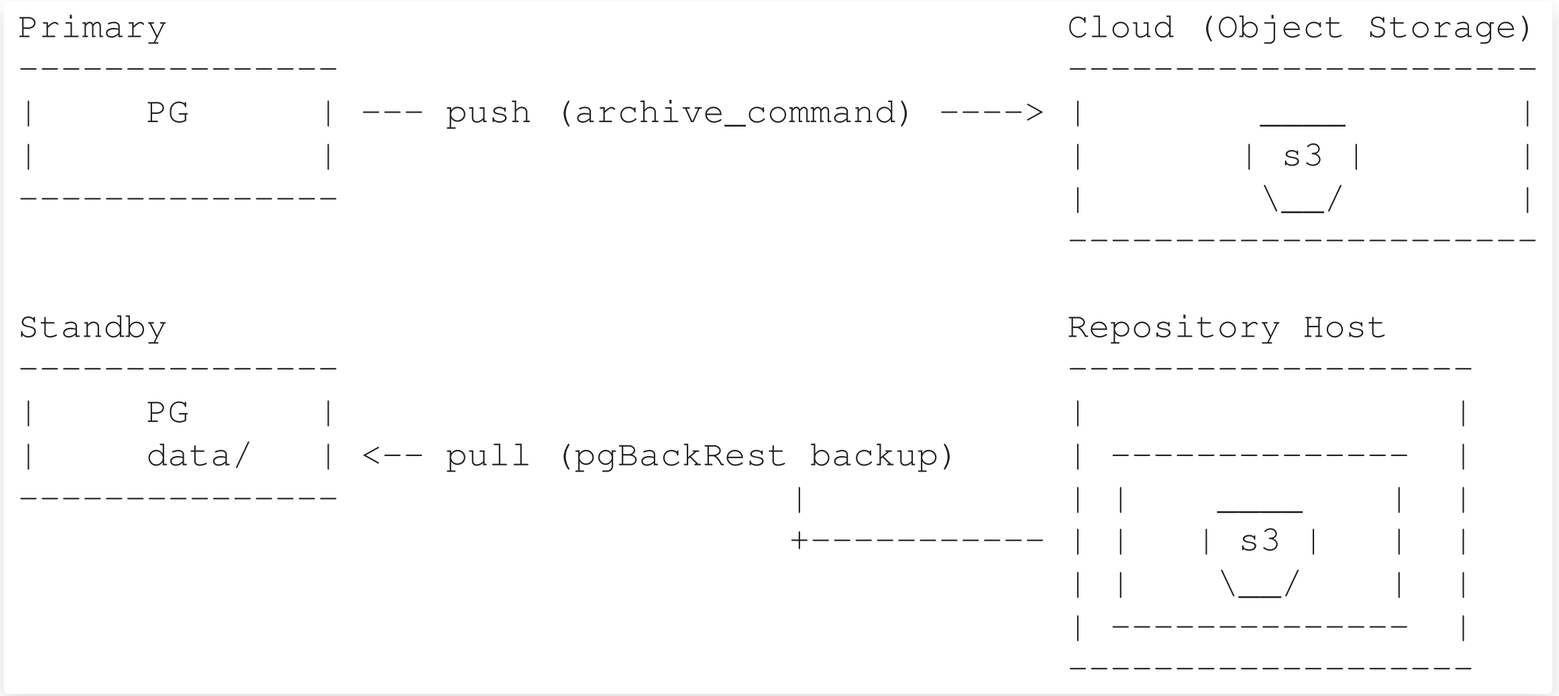
*The undocumented magical solution…*

# The problem

- With repo host
  - Bottleneck / SPOF
  - WAL archiving depends on it
- Without repo host
  - No coordination for backups

# The solution

- Separate execution from storage
  - PG pushes WAL archives directly to cloud
  - The repo host orchestrates backups and pushes them to the cloud



# Why this works

- Scales with cluster
- Removes repo host SPOF for WAL archiving
- Keeps backup orchestration centralized

# Extra benefit

- Version coupling reduced
  - Only required for backup execution
  - WAL archiving is independent
- If repo host dies:
  - WAL archiving continues
  - Recovery still possible
  - Using older backups = *slower* recovery
- Still depends on cloud availability
  - Consider multi-repo

# Configuration example
# Repository host

```
[global]
repo1-type=s3
repo1-s3-endpoint=s3endpoint
repo1-s3-region=eu-central-1
repo1-s3-bucket=pgbackrest
repo1-path=/repo1
repo1-bundle=y
repo1-retention-full=2

[demo]
backup-standby=prefer
pg1-host=pg1
pg1-path=/var/lib/postgresql/18/demo
pg2-host=pg2
pg2-path=/var/lib/postgresql/18/demo
```

# Configuration example PostgreSQL host

```
[global]
repo1-type=s3
repo1-s3-endpoint=s3endpoint
repo1-s3-region=eu-central-1
repo1-s3-bucket=pgbackrest
repo1-path=/repo1

[demo]
pg1-path=/var/lib/postgresql/18/demo
```

# Comparison table

| Pattern | Storage | WAL flow | Backup execution |
|---|---|---|---|
| Classic repo | Repo host | PG -> repo host | Repo host |
| Cloud-only | Cloud | PG -> cloud | PG node |
| Hybrid | Cloud | PG -> repo host | Repo host |
| Decoupled | Cloud | PG -> cloud | Repo host |

# Possible alternative: use TLS server instead of SSH

- Repo host by default needs an SSH connection to the PG nodes
- Consider using the pgBackRest TLS server mode instead

# Key takeaways

- Design for failure
- Combine patterns, don’t just pick one
- Optimize for restore, not backup
- WAL is everything

# Conclusion

> *“pgBackRest is not just a backup tool — it’s a system design component.”*

- Want to go deeper?
  - Patroni and pgBackRest: better together

# Questions?

contact@dataegret.com

# Bonus: multi-repository setup

## How to combine local and cloud repositories

# Repository host (1)

- `repo1` = local (fast restore)
- `repo2` = cloud (durability / long retention)

```
[global]
repo1-path=/srv/pgbackrest/repo1   # local storage
repo1-retention-full=2
repo1-bundle=y
repo1-block=y

repo2-type=s3                      # cloud storage
repo2-s3-endpoint=s3endpoint
repo2-s3-region=eu-central-1
repo2-s3-bucket=pgbackrest
repo2-path=/repo2
repo2-bundle=y
repo2-block=y
repo2-retention-full=7             # longer retention in cloud
```

# Repository host (2)

```
# general options
process-max=2
log-level-console=info
log-level-file=detail
compress-type=zst
start-fast=y
delta=y

[demo]
backup-standby=prefer
pg1-host=pg1
pg1-path=/var/lib/postgresql/18/demo
pg2-host=pg2
pg2-path=/var/lib/postgresql/18/demo
pg3-host=pg3
pg3-path=/var/lib/postgresql/18/demo
```

# PostgreSQL hosts

```
[global]
repo1-host=repository
repo1-host-user=pgbackrest

repo2-type=s3
repo2-s3-endpoint=s3endpoint
repo2-s3-region=eu-central-1
repo2-s3-bucket=pgbackrest
repo2-path=/repo2

# general options
compress-type=zst
log-level-file=detail
process-max=2
archive-async=y
archive-get-queue-max=1GB

[demo]
pg1-path=/var/lib/postgresql/18/demo
```

WAL archives are pushed to both repositories

# General advice

- Keep `repo1` closer to your PostgreSQL data
- Use external storage as second source of truth for longer storage

# Benefits

- Backup redundancy with flexible retention settings
  - Fast local restores + long-term cloud retention
- Increased resilience with asynchronous archiving
  - WAL archiving continues even if one repository is unavailable

# Drawbacks

- Backups are independent per repository
  - Must be scheduled separately
  - Increased operational complexity