Recovery is an operating model, not a pipeline

A governed lifecycle for recovery, service return, failback and closure

Jeleel Muibi · HybridOps.Tech
Technical paper · Version 1.0
First published July 2026 · revised 8 August 2026

Abstract

A recovery pipeline can provision infrastructure, restore data and redirect traffic. Those actions are necessary, but they do not establish whether the recovery decision was authorised, the target was ready, the restored service is usable or the organisation has a controlled route back.

HybridOps treats recovery as an executable operating model. The requested outcome, dependencies and required checks are declared before execution. The resolved plan identifies the target and implementation. Decision and readiness gates separate authority to act from technical ability to act, while workload checks and environment-scoped records connect the request, executed transition and observed result.

PostgreSQL recovery is used here as a demanding systems case. It requires more than recreating compute: source authority, recovery-point selection, writer fencing, data restoration, service health, routing, failback and residual cost must remain coherent across cloud and on-premises environments. The recorded drills cover a three-node HA restore and return, including row-count and checksum continuity, and a separate managed-standby promotion and failback.

The central review question is whether governing the decision, forward path, verification, reverse path and residual state as one operation closes a real gap between recovery automation and operational control.

1. The difficult part begins at the decision boundary

Most organisations already have backups, infrastructure definitions, provider services, runbooks and automation jobs. The operational risk lies in the gaps between them. During a recovery event, a team must still determine:

- which service and data source are authoritative;
- whether the selected backup or replica satisfies the recovery objective;
- whether identity, network, quota, secrets and target capacity are ready;
- whether another writer must be fenced before promotion;
- who can authorise destructive or traffic-changing actions;
- which checks define a usable recovered service; and
- how failback, teardown and continuing cost will be handled.

NIST places recovery procedures inside a wider cycle of planning, testing and maintenance. AWS guidance joins tested recovery, configuration drift and automation rather than reducing disaster recovery to backup availability. Microsoft likewise treats failover, failback, runbooks and escalation as parts of business continuity. The design problem is therefore not whether recovery should be automated. It is how automation operates inside explicit authority, readiness and verification boundaries.

2. An executable recovery contract

HybridOps separates four records that are often collapsed into one pipeline:

Record	Question answered
Declaration	What outcome, dependencies and checks are required?
Resolved plan	Which policy, implementation and target will satisfy it?
Decision and health gates	Is execution allowed, and is its result usable?
Runtime record	What ran, what changed and what state remains?

In the public implementation, capability declarations are composed into a directed dependency graph and an invalid plan is rejected before execution. The same resolved graph governs readiness, deployment, access, rebuild and teardown, so lifecycle completion is not appended as cleanup.

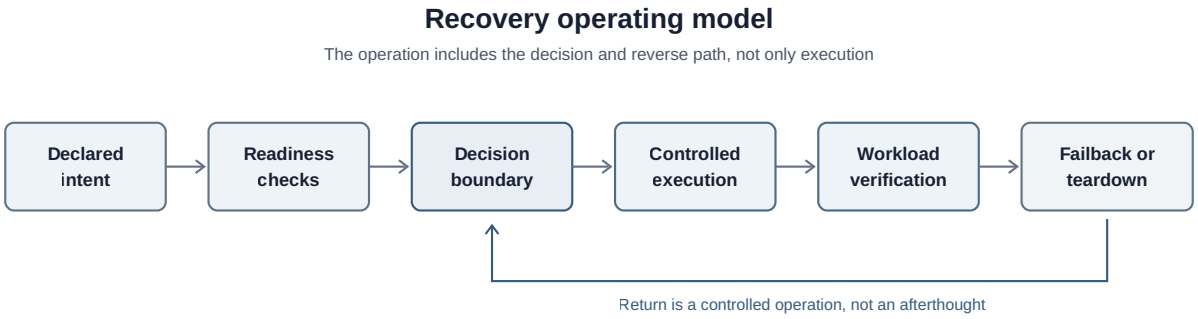

Figure 1. Recovery is governed as a complete operating cycle rather than a one-way deployment path.

The underlying provider and workload mechanisms remain inspectable. The contract does not disguise them; it provides a stable operating boundary above them. An implementation or target can change without changing the meaning of the requested capability.

3. What a controlled run must establish

A zero exit status is not sufficient proof of recovery. A controlled run must establish four conditions.

Before execution, target access, credentials, required secrets, dependency state, network paths and policy must be known. A required failure must stop its dependants before partial infrastructure is mistaken for a recovery system.

At the decision boundary, the requested action and its impact must be visible. Restore, promotion and traffic cutover cannot be inferred from a routine convergence event. Where a human gate is required, it belongs in the declared operation and its record.

After execution, infrastructure existence must be distinguished from workload health. Database state, application continuity and service routing require their own checks.

After stability, the reverse path and residual state must be explicit. Failback, retained capacity, teardown and cost are operating decisions, not cleanup notes.

4. Two implemented PostgreSQL recovery paths

The public implementation contains two PostgreSQL recovery models. One restores a self-managed HA cluster in Google Cloud and returns the service on premises. The other establishes a managed Cloud SQL standby with controlled promotion and failback. Their database architectures differ; their control boundaries do not.

4.1 Self-managed HA recovery and return

The self-managed route begins with a three-node on-premises PostgreSQL cluster. The recovery blueprint prepares isolated cloud compute, restores the selected backup into a new HA cluster, tests cluster and application state, and records the resulting service route. Before failback, the cloud primary produces a fresh backup. The paired return path rebuilds the on-premises target from that state and changes the service route only after the returned cluster is ready.

A controlled recovery drill conducted on 31 March 2026 recorded a three-node cloud restore, a fresh recovery-side backup, row-count and checksum continuity, restoration of the on-premises HA target, one leader with two streaming replicas, and final DNS state pointing to the returned service.

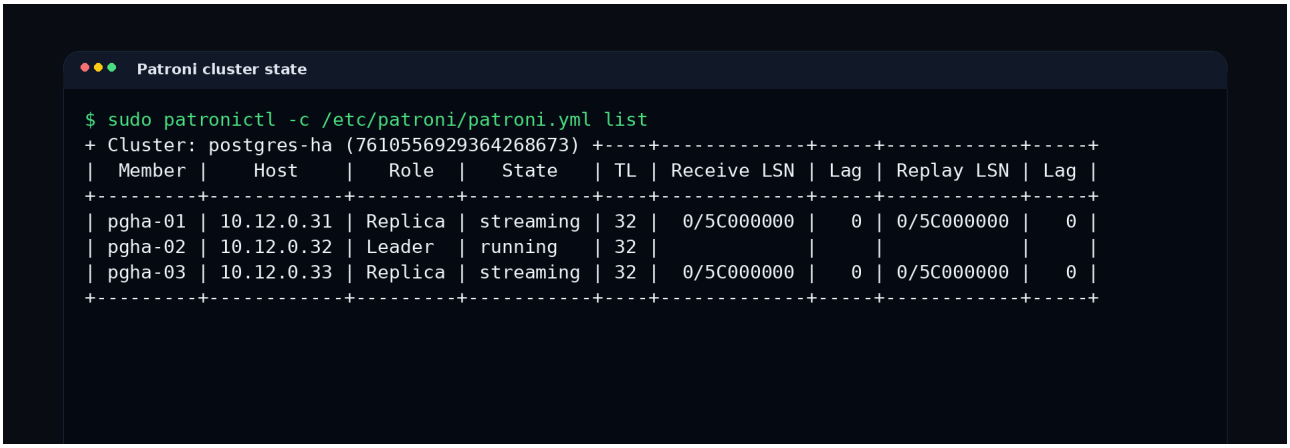

Figure 2. The returned PostgreSQL cluster records one leader, two streaming replicas and zero replication lag at capture time.

The backup, cloud restore, recovery-side backup, on-premises restore and route changes have separate run records. This preserves which recovery point was used and which checks supported each transition instead of reporting the whole cycle as one opaque success.

4.2 Managed standby, promotion and failback

The managed route first assesses the on-premises source and establishes an external-replica relationship with Cloud SQL. Standby readiness, promotion and failback are distinct operations. Promotion and return use explicit gates so routine infrastructure convergence cannot silently change the authoritative writer.

The recorded drill established the standby, exercised controlled promotion, returned service through an isolated on-premises failback lane and confirmed the final DNS position. Source assessment, standby state, promotion, failback and routing are retained independently.

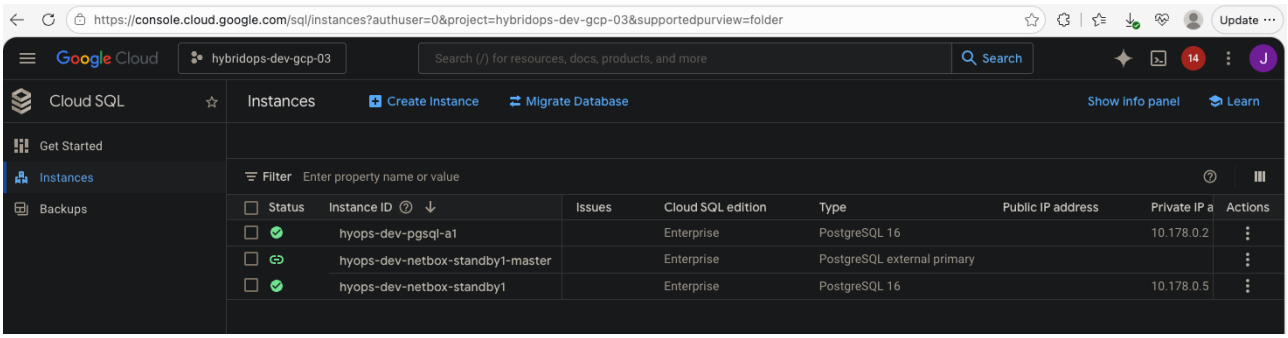

Figure 3. Managed recovery lane showing the external primary relationship and ready Cloud SQL standby.

4.3 What the records prove

The captures and records describe completed non-production drills. They do not claim that historical recovery resources remain active or establish a production recovery time objective (RTO) or recovery point objective (RPO). The durable result is the dated relationship between the selected source, executed transition, workload check, service route and reverse operation.

That separation exposes meaningful failure modes. Compute may exist while access is unavailable. A database may start from the wrong recovery point. A restore may pass while traffic still resolves to the old endpoint. Treating source assessment, infrastructure, restoration, health, routing and return as separate contracts keeps those conditions visible.

5. The contribution is the operating boundary

Backups, replication, failover and infrastructure automation provide the established mechanisms. HybridOps owns the wider lifecycle decision: whether recovery may begin, whether the result counts as service return, whether authority may move and whether the reverse path is complete. That decision is enforced through the following operating boundaries:

- capability intent is separated from its implementation;
- dependency order is computed and validated before execution;
- policy and environment bindings are explicit;
- destructive and authority-changing transitions can be gated;
- workload health is distinct from resource creation; and
- teardown and retained state are first-class outcomes.

For platform teams, this creates a boundary between the service outcome and the mechanisms maintained by individual teams. For SRE and incident leads, it separates a signal from authority to act. For MSP and consulting teams, it allows customer policy to vary without changing the operating sequence. For a small team, it gives a second operator a declared path and a dated record to inspect.

6. Controlled evaluation

The model should be evaluated through a controlled drill. The drill should check whether:

- a deliberately missing prerequisite stops dependent work;
- the recovery decision and affected stages are visible before action;
- restoration and service health are verified independently;
- the active service location is unambiguous after cutover;

- a second operator can follow the recorded path; and
- failback or teardown leaves a known residual state.

Elapsed time can be recorded, but a single fast run is not evidence of an RTO. Recovery objectives remain workload and business decisions. The purpose of the drill is to establish whether the runtime exposes the facts needed to measure and improve them.

7. Request for technical review

1. Is the distinction between a recovery pipeline and a recovery operating model technically and operationally sound?
2. Do the decision, readiness, workload-verification and reverse-path gates expose enough evidence to govern a recovery transition?
3. Which failure injection would most usefully test the model next?

A short technical assessment or correction is sufficient.

References

Operational guidance

1. NIST, SP 800-34 Rev. 1: Contingency Planning Guide for Federal Information Systems.
2. AWS, Well-Architected Framework: Plan for Disaster Recovery.
3. Microsoft, Business continuity, high availability and disaster recovery.

Implementation and drill records

1. HybridOps Core.
2. PostgreSQL HA recovery-cycle case.
3. Managed PostgreSQL recovery with Cloud SQL.
4. HybridOps recovery runbooks.
5. PostgreSQL HA recovery and failback demonstration.
6. Managed PostgreSQL standby demonstration.