Recovery as an operating model

A decision and closure contract from activation through service return

Jeleel Muibi · HybridOps.Tech · Version 1.1 · August 2026

Abstract

A recovery pipeline can create capacity, restore data and redirect traffic while leaving the wider recovery event unresolved. Complete recovery requires an authorised action, a ready target, a usable workload, explicit writer and service authority, a defined return path, and proof of the terminal state of temporary recovery capacity.

This paper introduces a recovery operating contract that treats those decisions as one governed lifecycle. Recovery begins with an explicit authority and source decision, passes through target readiness and workload verification, records any change in service or writer authority, requires a defined return path, and closes when residual infrastructure and service state are known. Pipeline steps become evidence inside that wider lifecycle.

The model is exercised through two non-production PostgreSQL paths: restoration of a three-node HA service into temporary Google Cloud capacity with a controlled return on premises, and a managed Cloud SQL standby with explicit promotion and failback. The paths use different recovery mechanisms while following the same decision, verification and closure contract.

The operational question is:

What proves that a recovery event is complete across decision, service return and closure?

1. Recovery risk sits between the mechanisms

Most organisations already have backups, replication, infrastructure definitions, provider recovery features and runbooks. The difficult failures are often in the handoffs between them.

A recovery event has to answer:

- which service and data source are authoritative;
- which recovery point or replica is acceptable;
- whether the destination has the required identity, network, secrets, quota and capacity;
- whether an existing writer must be fenced before promotion;
- who is authorised to restore, promote, cut traffic or release capacity;
- which checks establish a usable service rather than a running resource;
- what state must survive for rollback or failback; and
- what proves that temporary recovery resources reached their intended terminal state.

Automation makes these decisions executable rather than leaving them implicit.

Established approaches and the remaining gap

NIST contingency-planning guidance places recovery procedures inside a wider cycle of planning, testing, recovery and maintenance [1]. AWS Well-Architected guidance connects disaster recovery to tested recovery procedures, configuration consistency and automation [2]. Microsoft similarly treats business continuity as a wider concern spanning recovery, failover, failback and operational readiness [3].

These approaches establish that recovery is broader than backup availability or one infrastructure workflow. The remaining implementation problem is how to make that wider operating contract executable across heterogeneous mechanisms: separate authority to act from technical readiness, gate transitions on required preconditions, verify the workload independently of resource creation, preserve the reverse path and record the residual state left behind.

The innovation introduced here is an executable recovery lifecycle in which those boundaries are part of one operation.

2. Define the recovery contract before execution

A recovery operation resolves four records before it treats execution as safe.

Record	Question
Recovery declaration	What service outcome, dependencies and checks are required?
Resolved recovery plan	Which source, target, policy and implementation will satisfy it?
Decision and health gates	Is the authority present, is the target ready and is the resulting workload usable?
Runtime and closure record	What ran, what authority changed, what remains active and what is the return state?

The declaration includes the full service outcome beyond provider resources: it may require source assessment, data restoration, writer fencing, routing, application checks, continued backup and a later return operation.

The operating cycle can be summarised as:

```
declare recovery outcome
        |
        v
verify source + target readiness
        |
        v
authorise transition
        |
        v
restore / promote / cut over
        |
        v
verify workload + service route
        |
        v
operate recovery state
        |
        v
return, retain or close
        |
        v
verify residual state
```

3. Separate authority to act from ability to act

Readiness and authority answer different questions.

A target may be technically ready before an authorised recovery decision exists. A recovery may be authorised while required network, secrets, backup state or quota are still unavailable. Promotion and traffic cutover therefore require an explicit authority transition in addition to technical capability.

The operating contract treats restore, promotion, cutover, failback and destructive closure as named transitions. Where a human or policy gate is required, that decision belongs in the lifecycle and its record.

A useful pre-execution gate establishes at least:

1. source and recovery-point authority;
2. target identity and readiness;
3. required credentials, secrets and network paths;
4. upstream backup/replica state;
5. any writer-fencing requirement; and
6. authority for the requested transition.

A failed required condition stops dependent work before partial infrastructure is mistaken for a recovered service.

4. Verify service recovery independently of infrastructure

A provider resource can exist before the service is usable. Recovery therefore separates resource state from workload state.

For a database path, independent verification may include:

- database startup and cluster membership;
- leader/writer identity;
- replica state and lag;
- selected data-integrity checks;
- application or client reachability; and
- active service routing.

The required checks are declared before the event. A recovered VM, pod, managed database or load balancer becomes a successful recovery result when the service outcome required by the contract is verified.

5. Treat return and closure as part of recovery

Once the service is running in the recovery location, the team still needs to decide whether it will remain there, fail back or move to another steady state. The return path therefore belongs to the recovery contract.

It should define:

1. the source of truth for data at return time;
2. any fresh recovery-side backup or replication requirement;
3. readiness conditions for the returning target;
4. authority for changing service location or writer role;
5. rollback conditions during the overlap window; and
6. the terminal state expected for temporary recovery resources.

Closure records one of three explicit outcomes for each recovery resource: removal, return to a prepared standby state, or intentional retention with an owner and purpose.

6. Two implemented recovery paths

The first HybridOps path exercises a three-node on-premises PostgreSQL service restored into temporary Google Cloud capacity and then returned on premises. The second uses a managed Cloud SQL standby with separate source assessment, standby establishment, promotion and failback operations.

The mechanisms differ, but the contract remains the same: source authority, target readiness, authority-changing transition, workload proof, service location, return and closure are all explicit.

A controlled drill on 31 March 2026 recorded the self-managed path. The managed path was exercised separately. The relevant evidence is summarised below.

Evidence boundary	Self-managed HA path	Managed standby path
Source state	Selected backup and source assessment recorded	External source/standby relationship assessed
Recovery target	Temporary GCP HA capacity	Cloud SQL standby
Authority transition	Restore and route change	Explicit promotion
Workload proof	Row-count/checksum continuity, leader and replicas	Standby/promotion state and service checks
Return	Fresh recovery-side backup and on-premises restore	Isolated failback lane
Final service position	Returned on-premises DNS state	Final route recorded after failback
Closure	Temporary recovery resources handled explicitly	Recovery/return state recorded separately

These are completed non-production recovery exercises. Their measured timings provide repeatable drill evidence; production RTO/RPO requires repeated production-representative measurement against the organisation's service objectives.

7. Cost and residual state

Recovery can create temporary spend that outlives the incident. Cloud compute, storage, network paths or managed services may remain active after service has returned.

Cost is therefore attached to the same closure model rather than treated as a separate housekeeping exercise. Runtime estimates provide operational context, provider billing remains authoritative, and retained recovery resources remain visible as continuing cost-bearing components.

The useful control is a terminal resource statement: every resource in the recovery scope is removed, returned to its prepared state, or intentionally retained with an owner and purpose.

Service safety and recovery policy remain primary decision authorities, while cost visibility prevents temporary recovery capacity from becoming unowned steady-state consumption.

8. Failure conditions and evaluation

The model should be challenged through controlled failure as well as successful drills.

Useful tests include:

- unavailable or unacceptable recovery source;
- missing target readiness prerequisite;
- attempted promotion while writer authority is ambiguous;
- infrastructure success with failed workload health;
- successful restore with incorrect service routing;
- failed recovery-side backup before failback;
- failed return verification while recovery service remains active; and
- residual cost-bearing resources after technical service return.

Elapsed time is recorded as operational evidence. Repeated production-representative measurements can then be assessed against formal RTO and RPO objectives.

9. HybridOps implementation boundary

HybridOps is the open-source implementation used to exercise this operating model. It governs the surrounding lifecycle: declared recovery outcome, dependency order, decision/readiness gates, workload verification, service-route transitions, return sequencing and residual-state evidence.

PostgreSQL, Patroni, Cloud SQL, backup systems, DNS and infrastructure providers retain authority for the mechanisms they own. The product innovation is the recovery contract that joins those systems into one governed decision, verification, return and closure lifecycle.

10. Practitioner review question

In a recovery event you would be accountable for, what evidence must exist before you would call the service recovered, and what additional evidence must exist before you would call the event closed?

A failure case that exposes a missing decision, verification, return or closure condition is a useful result.

References

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.
4. HybridOps Core, open-source implementation.
5. HybridOps documentation, PostgreSQL HA DR Cycle showcase.
6. HybridOps documentation, Managed PostgreSQL DR with Cloud SQL showcase.
7. HybridOps documentation, Recovery runbooks.
8. HybridOps demonstration, PostgreSQL HA recovery and failback.
9. HybridOps demonstration, Managed PostgreSQL standby.
10. FinOps Foundation, Usage Optimization.