

PERCONA LIVE AMSTERDAM

Schrödinger's Backup

It's only good until you try to restore it.

Neil Armitage

Senior Database Reliability Engineer



Many decades of finding out the hard way



Neil Armitage

Senior Database Reliability Engineer — Anaplan

MySQL & Oracle DBA

Over three decades of hands-on DBA work before "reliability engineer" was a job title.

Continuent — Tungsten Clustering

Assisting customers deploying complex clustering solutions

Skyscanner, VMware & beyond

DevOps and platform leadership roles at scale, 100M+ customers a month.

Anaplan, today

Running MySQL, PXC, Redis, RabbitMQ & Pulsar on Kubernetes, 24/7.

DISCLAIMER

A few caveats before we start



Steps have been simplified

The real runbook is longer and messier — detail has been cut to fit the time slot.



Details have been obscured

Names, hostnames and figures are altered or redacted for confidentiality.



Details may be wrong

The presenter is elderly; corrections are welcome, gently and in writing.

Why should we care about backups



You have a backup. Do you have a backup you can restore?

Until you attempt a restore, your backup exists in a kind of superposition — it might be perfectly fine, or it might be silently corrupt, incomplete, or built on assumptions about cluster state that no longer hold.



Backup ran
successfully



Backup fails to restore

Both true until you open the box.

Redundancy assumptions fail in ways DR plans don't rehearse



March 2026: drone strikes hit AWS's ME-CENTRAL-1 and ME-SOUTH-1 regions, taking multiple availability zones offline at the same time.

2 of 3



UAE availability zones
impacted at once

109



AWS services disrupted
at peak

Months

estimated time to fully
restore the region



now H1 2027

"Highly available" and "recoverable" are not the same guarantee — zone redundancy is not a tested restore path.

Plans sometimes must change

1

The planned DR region was also in the UAE — the fallback shared the same blast radius

2

Building a new environment in what was rapidly becoming a war zone was not a real option

3

Data residency constraints put lawyers and management straight into the recovery path



But we, as a team, were comfortable

1

We had backup on NFS which had been tested and the testing had been documented

2

We had tested and documented the DR restore process within the last 24hours

3

We also knew how long it would take



A green checkmark tells you almost nothing



"Backup job succeeded"

- The process ran without error
- A file exists in object storage
- The job scheduler is happy

What it doesn't tell you



- Whether the data is internally consistent
- Whether it restores on a clean node
- Whether GTID / wsrep position is sane
- Whether schema drift crept in unnoticed

What a successful backup job won't surface



xtrabackup consistency

A completed xtrabackup run can still leave an inconsistent snapshot if the prepare step wasn't validated.



GTID / wsrep position drift

The backup's recorded position can silently diverge from what the cluster expects on restore.



Incomplete SST vs. successful job

A State Snapshot Transfer can report success while leaving a donor node in a bad state.

Failure modes that only show up mid-recovery



Bootstrap sequencing errors

Nodes racing to bootstrap in the wrong order, splitting the cluster before it ever forms.



Wrong node picked as bootstrap source

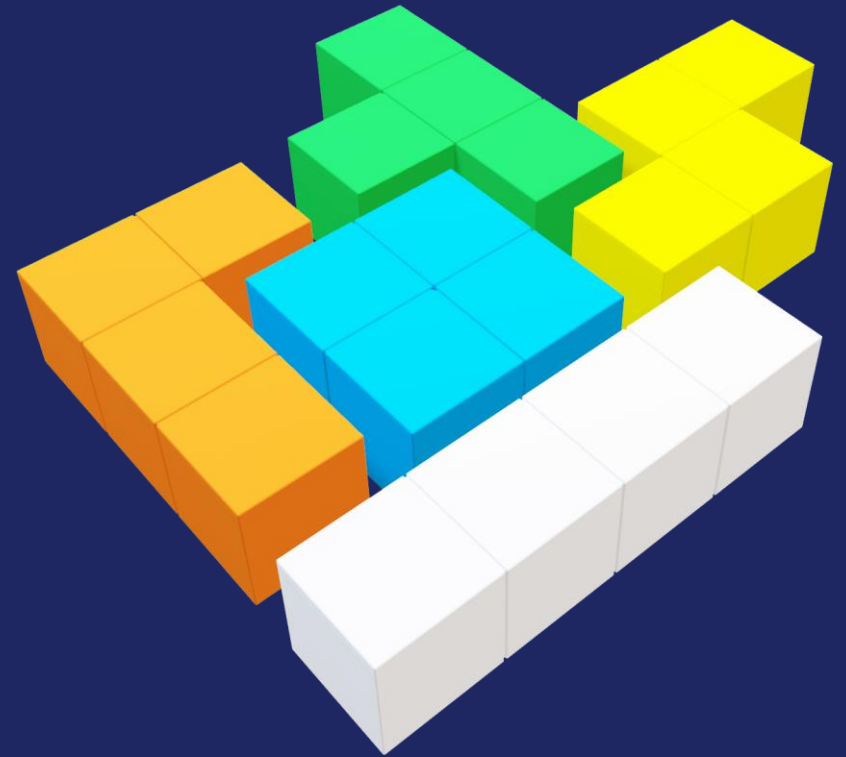
Bootstrapping from a node with a stale or lagging position instead of the most advanced one.



Patching

Has some change been introduced which affects backups?

Architectural Overview



Three key layers of the Anaplan architecture

Customer data

Models, transactional data and customer configuration

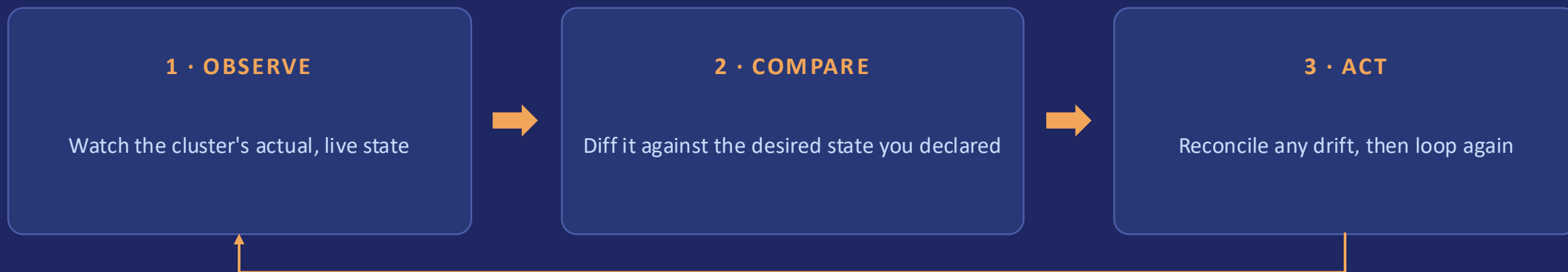
Application

Anaplan services, APIs and the calculation engine

Compute

Kubernetes and the storage everything runs on

Why we use Kubernetes Operators



You describe the end state once in a Custom Resource (a CR). The operator's controller runs this loop forever to keep the live cluster matching it.

In effect, an operator is a human sysadmin's runbook packaged as code — it reacts to drift and failures the same way an expert would, but in seconds and without being paged.

One spec, a self-running database cluster

PROVISION

Creates the pods, storage and networking from your spec

KEEP HEALTHY

Configures replication and fails over when a node dies

BACK UP

Schedules backups and knows which node to restore from

A database operator (CloudNativePG, Vitess, MongoDB Operator...) turns a few lines of YAML into a replicated, monitored cluster — no hand-run failover or backup scripts.

But the operator only does what its CRD tells it to — "we run it with an operator" is not the same as "backups exist and the restore has been tested."

One CRD becomes a whole Postgres cluster

```
# kubectl apply -f cluster.yaml
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: app-db
spec:
  instances: 3
  storage:
    size: 20Gi
  backup:
    schedule: "0 2 * * *"
```



The operator creates

- A StatefulSet running 3 Pods — 1 primary, 2 replicas
- 3 PersistentVolumeClaims · 20 Gi each
- Read-write and read-only Services
- A Secret with generated credentials
- A nightly backup job → object storage (02:00)

Current Services Deployed



Operators



Helm

The fleet these operators run

70+

Kubernetes clusters

across private cloud (OpenStack) plus AWS, GCP & Azure

aws



Azure



openstack.

1,200



Percona XtraDB

MySQL / Galera clusters

600



Redis

in-memory data clusters

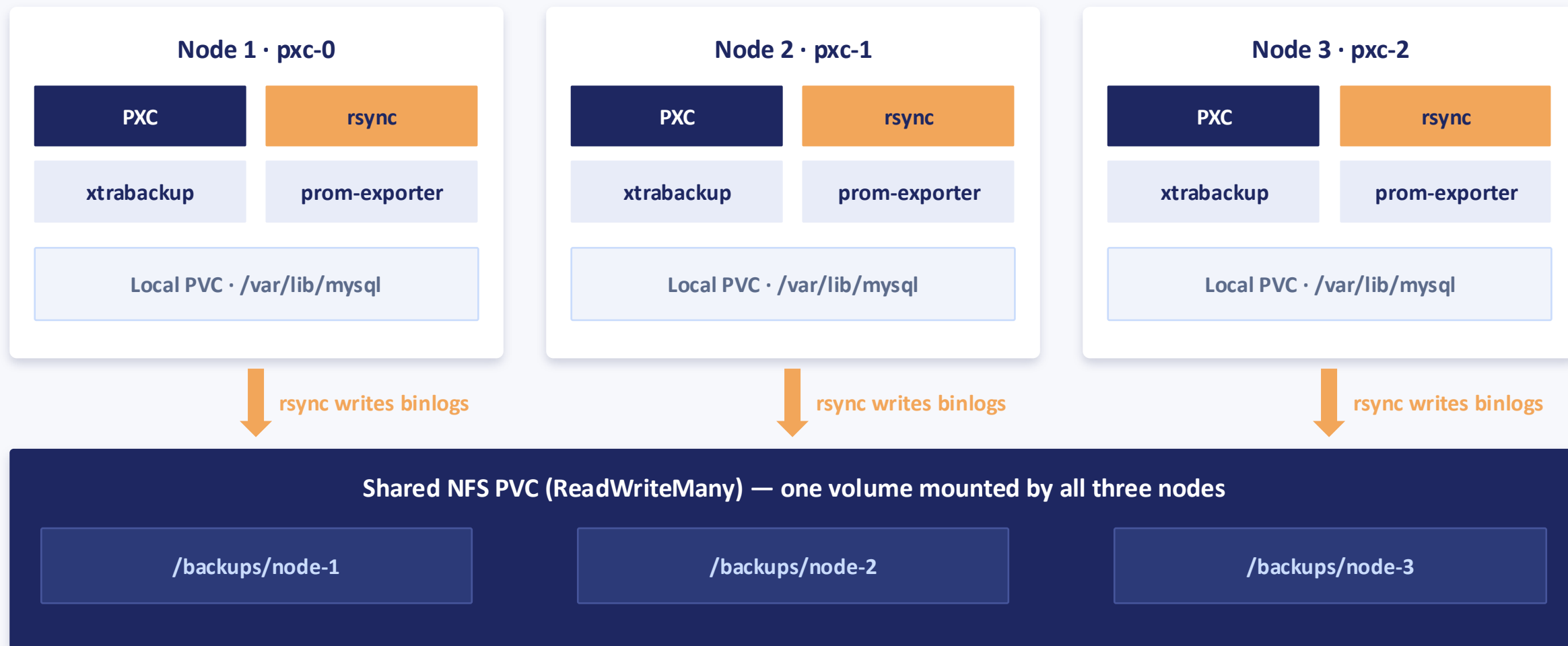
300



RabbitMQ

message broker clusters

What does a PXC Cluster consist of?



Shared NFS replicates to DR



The Testing Process



Two ways to answer, “have we tested this?”



Manual restore drills

- Depends on someone remembering to run it
- Steps drift depending on who's doing it
- Result lives in someone's head, not a record
- Not scalable in modern environment



Automated restore testing

- Runs on a schedule — no one has to ask
- Same steps every time, versioned like code
- Result is a pass/fail you can point to and report on
- EKS isn't the same as GKE, AKS or OpenStack

Every PXC cluster ships with a default backup schedule

```
$ kubectl get -n my-app xtradbbackupschedule \
  --selector=xtradb.anaplan.com/default-for=my-xtradb
```

NAME	TARGET	SCHEDULE	READY	RECONCILED	AGE
my-xtradb-2d6d2	my-xtradb	43 1/12 * * *	True	True	2s

can be customised
via a CRD

XtraDBBackupSchedule

A namespaced CRD — one default per cluster, found by the default-for label

43 1/12 * * *

Fires at 43 minutes past the hour, every 12 hours from 01:00

Ready and Reconciled

The operator has accepted the schedule and owns it from here

From cron trigger to a backup on the shared NFS volume



The container streams the backup straight onto the shared NFS PVC

Writes to /backups/node-N

Same volume every node shares

Replicated to DR ≥ 15 min later

A manual restore starts with one hand-written CRD

```
apiVersion: xtradb.anaplan.com/v1alpha1
kind: XtradbRestore
metadata:
  name: my-xtradb-restore
spec:
  target: my-xtradb
  pointInTime: 2026-09-08T01:39:14Z
  ttlSecondsAfterComplete: 3600
```

target

The cluster the restore runs against

pointInTime

Binlog replay stops at this UTC timestamp

ttlSecondsAfterComplete

The finished Job is cleaned up an hour later

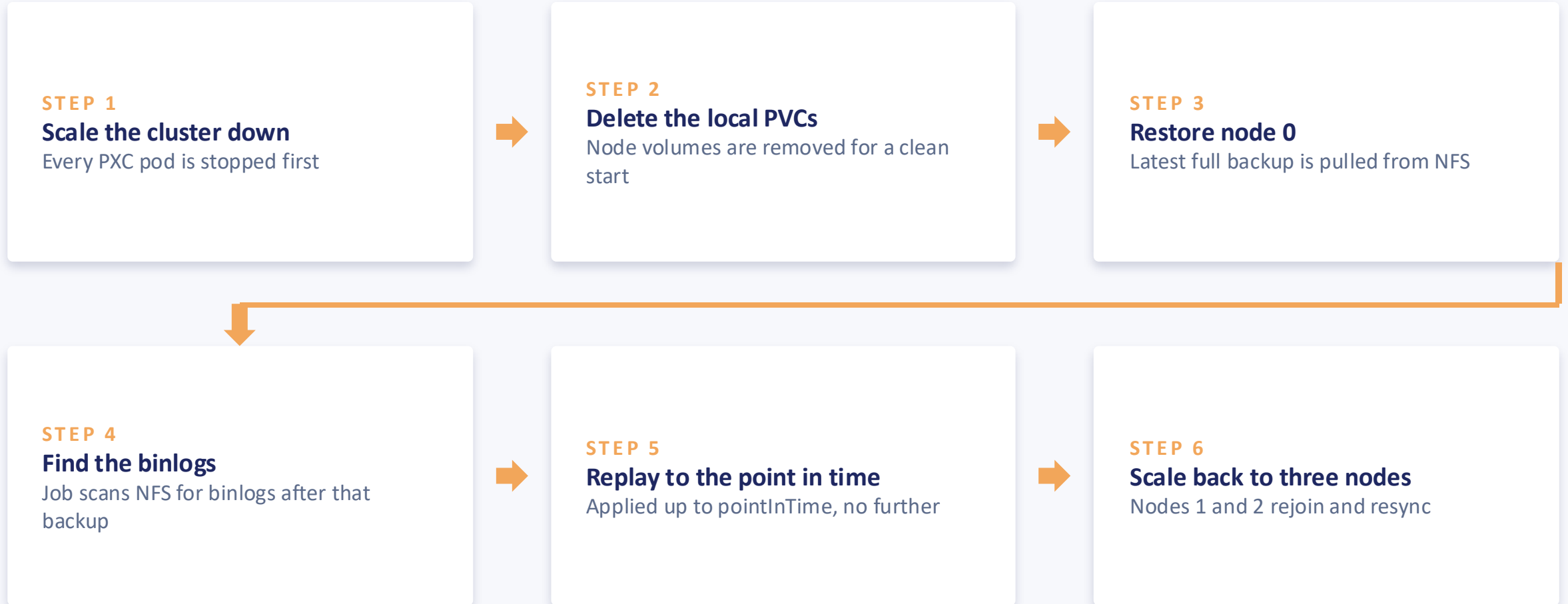
The operator picks the CRD up and creates a restore Job

No console, no runbook

Declarative and reviewable

Any point in time on NFS

What the restore Job actually does



Three pieces, wired into one pipeline

BACKUP

The operator provisions a default backup schedule

RESTORE

The operator exposes a restore CRD

RESTORE TEST

The operator provisions a single node MySQL – restores the backup and checks it's ok

Each piece already exists somewhere in your tooling — the backup job, a restore script, a health check. An automated test is just wiring them together, so they run unattended.

Run in isolation, backup and restore each look fine. Chained together on a schedule, they become a test that can actually fail.

A pipeline that proves the backup, not just runs it



The loop runs on a schedule, the same way an operator's reconcile loop does — except instead of keeping a cluster alive, it keeps proving the backup still works.

A test that only runs when someone remembers isn't a test. Wire it into CI and let the schedule do the remembering.

Operator e2e/smoke tests include testing the restore tests.

The same loop, pointed at your DR region

- 1 Same pipeline, different region — provision a throwaway cluster in the DR region and restore the latest backup into it
- 2 Validate, don't assume — check GTID / wsrep position, row counts and app health before the region is the only one left
- 3 Run it on a schedule — so “we have a DR plan” becomes “we tested it last Tuesday”

A DR region you have never restored into is just another backup in superposition.

The same pattern, extended past PXC

1

Postgres next — the same provision → restore → validate loop against Postgres clusters

2

ProxySQL Testing

3

Data querying and extracting

"Restore to the same box" isn't a real test

Restoring in place

- Same network, same secrets, same DNS
- Cluster config already primed and cached
- Hides exactly the failure modes that matter



A real restore test

- Fresh cluster, no leftover state
- Forces you to reproduce bootstrap, TLS, routing
- Surfaces the gotchas before an outage does
- Metrics and Alerts
- You sleep a lot better.



How often should you actually test this?



Restore testing that happens once, after being asked, is a one-off. Restore testing on a schedule, with an owner, is a practice.

Cadence

Scheduled, not heroic — quarterly at minimum for critical clusters.

Ownership

One team or engineer accountable for the drill actually happening.

Evidence

A record of what was restored, when, and what broke — not just a green tick.

Alerts feed into routine support queues.

A practical restore-testing checklist

-  Restore to a fresh, isolated cluster — never the same box
-  Verify GTID / wsrep position matches expectations before declaring success
-  Simulate quorum loss and confirm your bootstrap runbook actually works
-  Confirm ProxySQL / routing layer picks up the restored topology
-  Pin and verify every image tag used in the restore path
-  Record the drill — date, what broke, what you fixed

Wrap Up



Collapse the uncertainty before an incident does it for you

1

A backup job succeeding is not the same claim as a restore working.

2

Quorum loss and multi-zone failures are not edge cases anymore — test for them.

3

A disposable, repeatable restore-testing process beats a heroic one-off every time.

Ending Thoughts

1

In-House Developed, Open source and contribute

2

Data Sovereignty, where is your data and is it portable

3

Have we moved from disposable servers to disposable data centers

THANK YOU

Questions?

Schrödinger's Backup — open the box before production does it for you.

Neil Armitage

Senior Database Reliability Engineer

neil@statefulops.com