Can a Kubernetes Operator Replace a DBA?

Amarachi Iheanacho · SideroLabs

The Question

What a DBA does

- Failover when a primary dies
- Monitor replication lag
- Manage storage pressure
- Execute version upgrades
- Maintain backup & recovery

→

Can the operator do this?

CloudNativePG watches your PostgreSQL cluster and tries to keep it in the state you declared, automatically, without you lifting a finger.

But how far does that promise actually go?

The Setup

Cluster OS

Talos Linux (via Omni)

Kubernetes

v1.35, 4 nodes on AWS

Operator

CloudNativePG

Database

PostgreSQL 16 → 17

The Experiment, 6 Scenarios

01

Kill the primary pod

Does it failover automatically?

02

Drain the node running primary

Can it reschedule across nodes?

03

Fill the disk to breaking point

Does it detect storage pressure?

04

Pause replication / create lag

Does it notice replicas falling behind?

05

Trigger a major version upgrade

Does it work?

06

Backup + point-in-time recovery

Can you actually restore data?

```
NAME                STATUS   ROLES           AGE   VERSION
ip-172-31-34-49     Ready    <none>          36h   v1.35.3
ip-172-31-37-120    Ready    <none>          36h   v1.35.3
ip-172-31-40-42     Ready    <none>          36h   v1.35.3
ip-172-31-43-138    Ready    control-plane   36h   v1.35.3
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get cluster experiment-db
NAME            AGE   INSTANCES   READY   STATUS                     PRIMARY
experiment-db   36h   3           3       Cluster in healthy state   experiment-db-1
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pods -w
NAME              READY   STATUS    RESTARTS   AGE
experiment-db-1   1/1     Running   0          36h
experiment-db-2   1/1     Running   0          36h
experiment-db-3   1/1     Running   0          36h
```

Scenario 1 · Kill the Primary Pod

What I did

- Deleted experiment-db-1 (the primary pod) with kubectl delete pod
- Started a timer the moment the command ran
- Watched for which replica gets promoted and how fast

What happened

- experiment-db-1 went Terminating, then Completed
- experiment-db-2 was promoted to primary
- A fresh experiment-db-1 pod was created on the same node
- Cluster back to 3/3 healthy. Total time: ~11 seconds

What the operator did

- Detected the primary pod was gone immediately
- Promoted the most up-to-date replica (experiment-db-2) automatically
- Updated internal service routing
- Spun up a new experiment-db-1 to replace the lost instance

What a DBA would have done

- Check each standby's WAL replay position (pg_last_wal_replay_lsn()) to find the most up-to-date replica
- Manually promote it using pg_ctl promote or pg_promote()
- Verify that the connection layer routes to the new primary correctly
- Provision a new standby replica to replace the lost instance

✅ **Result: Fully automated. 11-second failover. Zero human intervention required.**

```
Last login: Thu May 14 14:12:40 on ttys000
amarachiiheanacho@Amarachis-MacBook-Pro ~ % cd documents/conf42-db
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl delete pod experiment-db-1
pod "experiment-db-1" deleted from default namespace
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get cluster experiment-db
NAME            AGE   INSTANCES   READY   STATUS                     PRIMARY
experiment-db   36h   3           3       Cluster in healthy state   experiment-db-2
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pods -o wide
NAME              READY   STATUS    RESTARTS   AGE    IP           NODE                NOMINATED NODE   READINESS GATES
experiment-db-1   1/1     Running   0          3m1s   10.244.2.6   ip-172-31-37-120   <none>           <none>
experiment-db-2   1/1     Running   0          36h    10.244.3.6   ip-172-31-40-42    <none>           <none>
experiment-db-3   1/1     Running   0          36h    10.244.0.6   ip-172-31-34-49    <none>           <none>
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db %
```

```
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pods -w
NAME              READY   STATUS    RESTARTS   AGE
experiment-db-1   1/1     Running   0          36h
experiment-db-2   1/1     Running   0          36h
experiment-db-3   1/1     Running   0          36h
experiment-db-1   1/1     Terminating   0          36h
experiment-db-1   1/1     Terminating   0          36h
experiment-db-1   1/1     Terminating   0          36h
experiment-db-1   0/1     Completed     0          36h
experiment-db-1   0/1     Completed     0          36h
experiment-db-1   0/1     Completed     0          36h
experiment-db-2   1/1     Running       0          36h
experiment-db-1   0/1     Pending       0          0s
experiment-db-1   0/1     Pending       0          0s
experiment-db-1   0/1     Init:0/1      0          0s
experiment-db-1   0/1     PodInitializing   0          1s
experiment-db-1   0/1     Running           0          2s
experiment-db-1   0/1     Running           0          3s
experiment-db-1   0/1     Running           0          11s
experiment-db-1   1/1     Running           0          11s
```

Scenario 2 · Drain the Node Running Primary

What I did

- Ran kubectl get pods -o wide to see which node each pod was on
- Identified that experiment-db-2 (primary) was on ip-172-31-40-42
- Ran kubectl drain ip-172-31-40-42 --ignore-daemonsets --delete-emptydir-data

What happened

- Primary evicted, operator promoted experiment-db-1 to primary ✅
- Replica (experiment-db-2) stuck Pending for 26+ minutes ❌
- Error: PersistentVolume node affinity conflict, PVC tied to drained node
- Recovered only after running kubectl uncordon to bring the node back

What the operator did

- Detected primary was evicted, automatic failover to experiment-db-1
- Cluster stayed available throughout (degraded at 2/3, not down)
- Attempted to reschedule experiment-db-2 on another node failed
- Kept retrying but could not resolve the PVC node affinity constraint

What a DBA would have done

- Know the storage topology before draining any node
- Verify PVC bindings to ensure replicas can reschedule freely
- Planned switchover first, promote a replica explicitly, then drain the former primary with no urgency
- With local storage: plan for replica data to be re-seeded after drain

⚠️ **Result: Partial. Failover was automatic. Self-healing blocked by local storage node affinity.**

```
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pods -o wide
NAME              READY   STATUS    RESTARTS   AGE    IP           NODE                NOMINATED NODE   READINESS GATES
experiment-db-1   1/1     Running   0          3m1s   10.244.2.6   ip-172-31-37-120    <none>           <none>
experiment-db-2   1/1     Running   0          36h    10.244.3.6   ip-172-31-40-42     <none>           <none>
experiment-db-3   1/1     Running   0          36h    10.244.0.6   ip-172-31-34-49     <none>           <none>
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % date && kubectl drain ip-172-31-40-42 --ignore-daemonsets --delete-emptydir-data
Fri May 15 11:11:48 WAT 2026
node/ip-172-31-40-42 cordoned
Warning: ignoring DaemonSet-managed Pods: kube-system/kube-flannel-wfglc, kube-system/kube-proxy-f5g96
evicting pod cnpg-system/cnpg-cloudnative-pg-f768ffdfb-kz2qj
evicting pod default/experiment-db-2
pod/cnpg-cloudnative-pg-f768ffdfb-kz2qj evicted
pod/experiment-db-2 evicted
node/ip-172-31-40-42 drained
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get cluster experiment-db
NAME            AGE   INSTANCES   READY   STATUS                                     PRIMARY
experiment-db   36h   3           2       Waiting for the instances to become active   experiment-db-1
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get cluster experiment-db -w
NAME            AGE   INSTANCES   READY   STATUS                                     PRIMARY
experiment-db   36h   3           2       Waiting for the instances to become active   experiment-db-1
experiment-db   36h   3           2       Waiting for the instances to become active   experiment-db-1
^C%
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
experiment-db-1   1/1     Running   0          29m
experiment-db-2   0/1     Pending   0          23m
experiment-db-3   1/1     Running   0          36h
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl describe pod experiment-db-2 | grep -A 10 Events
Events:
  Type     Reason            Age                From               Message
  ----     ------            ----               ----               -------
  Warning  FailedScheduling  24m                default-scheduler  0/4 nodes are available: 1 node(s) had untolerated taint(s), 1 node(s) were unschedulable, 2 node(s) didn't match PersistentVolume's node affinity.
 no new claims to deallocate, preemption: 0/4 nodes are available: 4 Preemption is not helpful for scheduling.
  Warning  FailedScheduling  14m (x2 over 19m)  default-scheduler  0/4 nodes are available: 1 node(s) had untolerated taint(s), 1 node(s) were unschedulable, 2 node(s) didn't match PersistentVolume's node affinity.
 no new claims to deallocate, preemption: 0/4 nodes are available: 4 Preemption is not helpful for scheduling.
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db %
```

```
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pods -w
NAME              READY   STATUS    RESTARTS   AGE
experiment-db-1   1/1     Running   0          36h
experiment-db-2   1/1     Running   0          36h
experiment-db-3   1/1     Running   0          36h
experiment-db-1   1/1     Terminating   0          36h
experiment-db-1   1/1     Terminating   0          36h
experiment-db-1   1/1     Terminating   0          36h
experiment-db-1   0/1     Completed     0          36h
experiment-db-1   0/1     Completed     0          36h
experiment-db-1   0/1     Completed     0          36h
experiment-db-2   1/1     Running       0          36h
experiment-db-1   0/1     Pending       0          0s
experiment-db-1   0/1     Pending       0          0s
experiment-db-1   0/1     Init:0/1      0          0s
experiment-db-1   0/1     PodInitializing   0          1s
experiment-db-1   0/1     Running           0          2s
experiment-db-1   0/1     Running           0          3s
experiment-db-1   0/1     Running           0          11s
experiment-db-1   1/1     Running           0          11s
experiment-db-2   1/1     Running           0          36h
experiment-db-2   0/1     Completed         0          36h
experiment-db-2   0/1     Running           1 (2s ago)   36h
experiment-db-2   0/1     Running           1 (2s ago)   36h
experiment-db-2   0/1     Terminating       1 (2s ago)   36h
experiment-db-2   0/1     Terminating       1 (3s ago)   36h
experiment-db-2   0/1     Completed         1 (7s ago)   36h
experiment-db-2   0/1     Completed         1 (8s ago)   36h
experiment-db-2   0/1     Completed         1 (8s ago)   36h
experiment-db-1   1/1     Running           0            6m7s
experiment-db-2   0/1     Pending           0            0s
experiment-db-2   0/1     Pending           0            0s
```

```
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl uncordon ip-172-31-40-42
node/ip-172-31-40-42 uncordoned
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pods
kubectl get cluster experiment-db
NAME              READY   STATUS    RESTARTS   AGE
experiment-db-1   1/1     Running   0          33m
experiment-db-2   1/1     Running   0          27m
experiment-db-3   1/1     Running   0          36h
NAME            AGE   INSTANCES   READY   STATUS                  PRIMARY
experiment-db   36h   3           3       Cluster in healthy state   experiment-db-1
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db %
```

```
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pods -w
NAME              READY   STATUS    RESTARTS   AGE
experiment-db-1   1/1     Running   0          36h
experiment-db-2   1/1     Running   0          36h
experiment-db-3   1/1     Running   0          36h
experiment-db-1   1/1     Terminating   0          36h
experiment-db-1   1/1     Terminating   0          36h
experiment-db-1   1/1     Terminating   0          36h
experiment-db-1   0/1     Completed     0          36h
experiment-db-1   0/1     Completed     0          36h
experiment-db-1   0/1     Completed     0          36h
experiment-db-2   1/1     Running       0          36h
experiment-db-1   0/1     Pending       0          0s
experiment-db-1   0/1     Pending       0          0s
experiment-db-1   0/1     Init:0/1      0          0s
experiment-db-1   0/1     PodInitializing   0          1s
experiment-db-1   0/1     Running           0          2s
experiment-db-1   0/1     Running           0          3s
experiment-db-1   0/1     Running           0          11s
experiment-db-1   1/1     Running           0          11s
experiment-db-2   1/1     Running           0          36h
experiment-db-2   0/1     Completed         0          36h
experiment-db-2   0/1     Running           1 (2s ago)   36h
experiment-db-2   0/1     Running           1 (2s ago)   36h
experiment-db-2   0/1     Terminating       1 (2s ago)   36h
experiment-db-2   0/1     Terminating       1 (3s ago)   36h
experiment-db-2   0/1     Completed         1 (7s ago)   36h
experiment-db-2   0/1     Completed         1 (8s ago)   36h
experiment-db-2   0/1     Completed         1 (8s ago)   36h
experiment-db-1   1/1     Running           0            6m7s
experiment-db-2   0/1     Pending           0            0s
experiment-db-2   0/1     Pending           0            0s
experiment-db-2   0/1     Pending           0            26m
experiment-db-2   0/1     Pending           0            26m
experiment-db-2   0/1     Init:0/1          0            26m
experiment-db-2   0/1     PodInitializing   0            27m
experiment-db-2   0/1     Running           0            27m
experiment-db-2   0/1     Running           0            27m
experiment-db-2   0/1     Running           0            27m
experiment-db-2   1/1     Running           0            27m
```

Scenario 3 · Fill the Disk Until it Breaks

What I did

- kubectl exec into experiment-db-1 (primary)
- Wrote 8.4GB of junk data with dd to /var/lib/postgresql/data/
- Wrote another 4GB to push disk toward 90%+ utilisation
- Watched cluster status and pod events throughout

What happened

- At 64%: operator reported Cluster in healthy state, no alert at all
- At ~90%: primary pod crashed, operator triggered automatic failover
- Kubernetes tainted the node disk-pressure:NoSchedule
- PVC pinned to full node, pod stuck Pending. kubectl debug blocked by Talos
- Fix required deleting PVC + force-deleting pod. Operator then created experiment-db-4

What the operator did

- Stayed completely silent through 64% disk fill, no warning issued
- Detected primary crash and triggered failover to experiment-db-2
- Could not exec into the node to clean up, Talos blocks privileged pods
- Self-healed by creating experiment-db-4 on a clean node once PVC was deleted

What a DBA would have done

- Set up disk usage alerts at 70%, 80%, 90% thresholds on every database node
- Investigate and clean up space proactively before hitting a crash
- Document a runbook for disk-full scenarios before they happen in prod

❌ **Result: No warning at 64%. Crash+failover at ~90%. Recovery needed human intervention the operator couldn't provide.**

```
node/ip-172-31-40-42 uncordoned
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pods
kubectl get cluster experiment-db
NAME              READY   STATUS    RESTARTS   AGE
experiment-db-1   1/1     Running   0          33m
experiment-db-2   1/1     Running   0          27m
experiment-db-3   1/1     Running   0          36h
NAME            AGE   INSTANCES   READY   STATUS                     PRIMARY
experiment-db   36h   3           3       Cluster in healthy state   experiment-db-1
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl exec -it experiment-db-1 -- bash
Defaulted container "postgres" out of: postgres, bootstrap-controller (init)
postgres@experiment-db-1:/$ dd if=/dev/zero of=/var/lib/postgresql/data/fillfile bs=1M count=8000
8000+0 records in
8000+0 records out
8388608000 bytes (8.4 GB, 7.8 GiB) copied, 57.8323 s, 145 MB/s
postgres@experiment-db-1:/$ exit
exit
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl exec -it experiment-db-1 -- df -h /var/lib/postgresql/data
Defaulted container "postgres" out of: postgres, bootstrap-controller (init)
Filesystem      Size  Used Avail Use% Mounted on
none             16G   11G  5.9G  64% /var/lib/postgresql/data
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl exec -it experiment-db-1 -- bash -c "dd if=/dev/zero of=/var/lib/postgresql/data/fillfile2 bs=1M count=4000"
Defaulted container "postgres" out of: postgres, bootstrap-controller (init)
E0515 11:59:10.794102   43417 v2.go:129] "Unhandled Error" err="next reader: websocket: close 1006 (abnormal closure): unexpected EOF"
                                                                                                                                    E0515 11:59:10.794126   43417 v2.go:150] "Unhandled Error" err="next reader: web
socket: close 1006 (abnormal closure): unexpected EOF"
                                                  error: error reading from error stream: next reader: websocket: close 1006 (abnormal closure): unexpected EOF
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db %
```

```
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get cluster experiment-db
NAME            AGE   INSTANCES   READY   STATUS                                     PRIMARY
experiment-db   37h   3           2       Waiting for the instances to become active   experiment-db-2
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db %
```

```
experiment-db-2   1/1     Running   0          29m
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pods -w
NAME              READY   STATUS    RESTARTS   AGE
experiment-db-1   1/1     Running   0          49m
experiment-db-2   1/1     Running   0          43m
experiment-db-3   1/1     Running   0          36h
experiment-db-1   0/1     Completed   0          53m
experiment-db-1   0/1     Completed   0          53m
experiment-db-1   0/1     Completed   0          53m
experiment-db-2   1/1     Running     0          47m
experiment-db-1   0/1     Completed   0          53m
experiment-db-1   0/1     Completed   0          53m
experiment-db-1   0/1     Pending     0          0s
experiment-db-1   0/1     Pending     0          0s
```

```
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
experiment-db-1   0/1     Pending   0          5m30s
experiment-db-2   1/1     Running   0          78m
experiment-db-3   1/1     Running   0          37h
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl describe pod experiment-db-1 | grep -A 5 Events
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  117s  default-scheduler  0/4 nodes are available: 2 node(s) didn't match PersistentVolume's node affinity, 2 node(s) had untolerated taint(s). no new claims to deallocate, preemption: 0
/4 nodes are available: 4 Preemption is not helpful for scheduling.
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pvc experiment-db-1 -o jsonpath='{.spec.volumeName}' | xargs kubectl get pv -o jsonpath='{.spec.nodeAffinity}'
{"required":{"nodeSelectorTerms":[{"matchExpressions":[{"key":"kubernetes.io/hostname","operator":"In","values":["ip-172-31-37-120"]}]}]}}%
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get nodes -o custom-columns=NAME:.metadata.name,TAINTS:.spec.taints
NAME              TAINTS
ip-172-31-34-49   <none>
ip-172-31-37-120   [map[effect:NoSchedule key:node.kubernetes.io/disk-pressure timeAdded:2026-05-15T11:32:36Z]]
ip-172-31-40-42   <none>
ip-172-31-43-138   [map[effect:NoSchedule key:node-role.kubernetes.io/control-plane]]
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db %
```

Scenario 4 · Pause Replication / Create Lag

What I did

- kubectl exec into experiment-db-3 (replica)
- Ran SELECT pg_wal_replay_pause(), muted the replica from receiving WAL
- Wrote 1 million rows to the primary: CREATE TABLE lag_test AS SELECT generate_series(1,1000000)
- Queried pg_stat_replication on the primary to measure the lag

What happened

- experiment-db-3 fell 102MB behind the primary silently
- Cluster status: Cluster in healthy state, zero reaction from operator
- pg_stat_replication showed byte_lag: 102339240 on that replica
- Once resumed with pg_wal_replay_resume(), replica caught up to 0 lag almost instantly

What the operator did

- Did not detect the lag at all, no alert, no event, no status change
- Continued reporting the cluster as healthy throughout
- Has no built-in mechanism to monitor or alert on replication lag
- Operator only sees pod health, not PostgreSQL-level replication state

What a DBA would have done

- Monitor pg_stat_replication lag continuously, alert above a defined threshold
- Investigate the cause: network issue? pod resource pressure? intentional pause?
- If lag is too large: rebuild the replica from primary rather than letting it catch up
- Configure wal_keep_size (or use replication slots) to ensure lagging replicas can catch up without WAL being recycled first

❌ **Result: Operator is completely blind to replication lag. 102MB drift went unnoticed. Gap exists at PostgreSQL layer.**

```
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pods

NAME              READY   STATUS    RESTARTS   AGE
experiment-db-2   1/1     Running   0          2d6h
experiment-db-3   1/1     Running   0          3d18h
experiment-db-4   1/1     Running   0          2d4h
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl exec -it experiment-db-3 -- bash
Defaulted container "postgres" out of: postgres, bootstrap-controller (init)
postgres@experiment-db-3:/$ psql -U postgres -c "SELECT pg_wal_replay_pause();"
 pg_wal_replay_pause
---------------------

(1 row)

postgres@experiment-db-3:/$
```

```
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl exec -it experiment-db-2 -- psql -U postgres -c "
CREATE TABLE lag_test AS SELECT generate_series(1,1000000) AS id;
"
Defaulted container "postgres" out of: postgres, bootstrap-controller (init)
SELECT 1000000
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pods -w
NAME              READY   STATUS    RESTARTS   AGE
experiment-db-2   1/1     Running   0          2d6h
experiment-db-3   1/1     Running   0          3d18h
experiment-db-4   1/1     Running   0          2d4h
```

```
amarachiiheanacho@Amarachis-MacBook-Pro ~ % kubectl get cluster experiment-db -w
NAME            AGE     INSTANCES   READY   STATUS                     PRIMARY
experiment-db   3d18h   3           3       Cluster in healthy state   experiment-db-2
```

```
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl exec -it experiment-db-2 -- psql -U postgres -c "
SELECT client_addr, state, sent_lsn, replay_lsn,
(sent_lsn - replay_lsn) AS byte_lag
FROM pg_stat_replication;
"
Defaulted container "postgres" out of: postgres, bootstrap-controller (init)
 client_addr |   state   |  sent_lsn  | replay_lsn | byte_lag
-------------+-----------+------------+------------+-----------
 10.244.2.17 | streaming | 0/1C199308 | 0/1C199308 |         0
 10.244.0.6  | streaming | 0/1C199308 | 0/16000060 | 102339240
(2 rows)

amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl exec -it experiment-db-3 -- psql -U postgres -c "SELECT pg_wal_replay_resume();"
Defaulted container "postgres" out of: postgres, bootstrap-controller (init)
 pg_wal_replay_resume
----------------------

(1 row)

amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl exec -it experiment-db-2 -- psql -U postgres -c "
SELECT client_addr, state, sent_lsn, replay_lsn,
(sent_lsn - replay_lsn) AS byte_lag
FROM pg_stat_replication;
"
Defaulted container "postgres" out of: postgres, bootstrap-controller (init)
 client_addr |   state   |  sent_lsn  | replay_lsn | byte_lag
-------------+-----------+------------+------------+----------
 10.244.2.17 | streaming | 0/1C199308 | 0/1C199308 |        0
 10.244.0.6  | streaming | 0/1C199308 | 0/1C199308 |        0
(2 rows)
```

Scenario 5 · Major Version Upgrade (PG 16 → PG 17)

What I did

- Confirmed cluster was on PostgreSQL 16.13 via SELECT version()
- Ran a single kubectl patch to change imageName to postgresql:17.6
- Opened two watch terminals, one on pods, one on cluster status
- Noted the exact time the patch was applied

What happened

- All 3 pods terminated simultaneously, planned downtime window began
- A major-upgrade job spun up and ran pg_upgrade on the primary's data
- Primary restarted on PostgreSQL 17 in ~47 seconds
- Two fresh replicas (experiment-db-5, experiment-db-6) cloned from upgraded primary
- SELECT version() confirmed PostgreSQL 17.6, zero data loss

What the operator did

- Terminated all instances in a controlled, coordinated shutdown
- Ran pg_upgrade automatically on the primary's persistent volume
- Restarted the primary on the new PostgreSQL 17 image
- Discarded old replicas and re-cloned fresh ones from the upgraded primary

What a DBA would have done

- Plan a maintenance window weeks in advance
- Test the upgrade on a staging cluster first
- Take a full backup before starting
- Run pg_upgrade manually, validate with pg_upgrade --check, then cut over traffic to the new instance

✅ **Result: One kubectl patch command. 47 seconds downtime. A DBA would plan this operation for days.**

```
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl exec -it experiment-db-2 -- psql -U postgres -c "SELECT version();"
Defaulted container "postgres" out of: postgres, bootstrap-controller (init)
                                                 version

--------------------------------------------------------------------------------
------------------------------------------------
 PostgreSQL 16.13 (Debian 16.13-1.pgdg11+1) on x86_64-pc-linux-gnu, compiled by
gcc (Debian 10.2.1-6) 10.2.1 20210110, 64-bit
(1 row)

amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl patch cluster experiment-db --type merge -p \
  '{"spec":{"imageName":"ghcr.io/cloudnative-pg/postgresql:17.6"}}'
cluster.postgresql.cnpg.io/experiment-db patched
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
experiment-db-2   1/1     Running   0          7m14s
experiment-db-5   1/1     Running   0          6m34s
experiment-db-6   1/1     Running   0          6m
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl exec -it experiment-db-2 -- psql -U postgres -c "SELECT version();"
Defaulted container "postgres" out of: postgres, bootstrap-controller (init)
                                                 version

--------------------------------------------------------------------------------
------------------------------------------------
 PostgreSQL 17.6 (Debian 17.6-2.pgdg11+1) on x86_64-pc-linux-gnu, compiled by gc
c (Debian 10.2.1-6) 10.2.1 20210110, 64-bit
(1 row)
```

```
amarachiiheanacho@Amarachis-MacBook-Pro ~ % kubectl get cluster experiment-db -w
NAME            AGE     INSTANCES   READY   STATUS                                  PRIMARY
experiment-db   3d18h   3           3       Cluster in healthy state                experiment-db-2
experiment-db   3d18h   3           3       Cluster in healthy state                experiment-db-2
experiment-db   3d18h   3           3       Cluster in healthy state                experiment-db-2
experiment-db   3d18h   3           3       Upgrading Postgres major version        experiment-db-2
experiment-db   3d18h   3                   Upgrading Postgres major version        experiment-db-2
experiment-db   3d18h   3                   Upgrading Postgres major version        experiment-db-2
experiment-db   3d18h   3                   Upgrading Postgres major version        experiment-db-2
experiment-db   3d18h   3                   Upgrading Postgres major version        experiment-db-2
experiment-db   3d18h   3                   Upgrading Postgres major version        experiment-db-2
experiment-db   3d18h   3                   Upgrading Postgres major version        experiment-db-2
experiment-db   3d18h   3                   Upgrading Postgres major version        experiment-db-2
experiment-db   3d18h   1                   Upgrading Postgres major version        experiment-db-2
experiment-db   3d18h   1                   Upgrading Postgres major version        experiment-db-2
experiment-db   3d18h   1                   Upgrading Postgres major version        experiment-db-2
experiment-db   3d18h   1                   Waiting for the instances to become active   experiment-db-2
experiment-db   3d18h   1                   Waiting for the instances to become active   experiment-db-2
experiment-db   3d18h   1                   Waiting for the instances to become active   experiment-db-2
experiment-db   3d18h   1           1       Waiting for the instances to become active   experiment-db-2
experiment-db   3d18h   1           1       Waiting for the instances to become active   experiment-db-2
experiment-db   3d18h   1           1       Creating a new replica                  experiment-db-2
experiment-db   3d18h   2           1       Creating a new replica                  experiment-db-2
experiment-db   3d18h   2           1       Creating a new replica                  experiment-db-2
experiment-db   3d18h   2           1       Creating a new replica                  experiment-db-2
experiment-db   3d18h   2           1       Waiting for the instances to become active   experiment-db-2
experiment-db   3d18h   2           1       Waiting for the instances to become active   experiment-db-2
experiment-db   3d18h   2           1       Waiting for the instances to become active   experiment-db-2
experiment-db   3d18h   2           2       Waiting for the instances to become active   experiment-db-2
experiment-db   3d18h   2           2       Waiting for the instances to become active   experiment-db-2
experiment-db   3d18h   2           2       Creating a new replica                  experiment-db-2
experiment-db   3d18h   3           2       Creating a new replica                  experiment-db-2
experiment-db   3d18h   3           2       Creating a new replica                  experiment-db-2
experiment-db   3d18h   3           2       Creating a new replica                  experiment-db-2
experiment-db   3d18h   3           2       Waiting for the instances to become active   experiment-db-2
experiment-db   3d18h   3           2       Waiting for the instances to become active   experiment-db-2
experiment-db   3d18h   3           2       Waiting for the instances to become active   experiment-db-2
experiment-db   3d18h   3           3       Waiting for the instances to become active   experiment-db-2
experiment-db   3d18h   3           3       Cluster in healthy state                experiment-db-2
experiment-db   3d18h   3           3       Cluster in healthy state                experiment-db-2
```

Scenario 6 · Backup + Point-in-Time Recovery

What I did

- Patched cluster with S3 backup config (barmanObjectStore)
- Applied kind: Backup, watched status with kubectl get backup -w
- Inserted a row with a timestamp, noted exact time: 2026-05-17 17:09:28
- Dropped the table (DROP TABLE pitr_test), confirmed data gone
- Applied kind: Cluster with recovery bootstrap targeting 17:09:50

What happened

- Backup completed in 8 seconds and uploaded to S3 ✅
- Recovery cluster spun up and began restoring from S3 ✅
- PITR failed, WAL segment 000...026 not found in S3 ❌
- Operator kept restarting the recovery pod in a loop for 1+ hour
- Root cause: WAL archiving was not configured at cluster creation time

What the operator did

- Triggered the base backup and uploaded it to S3 automatically
- Created a new recovery cluster from the backup declaratively
- Got stuck trying to replay WAL, segment was never archived to S3
- Provided no warning that continuous archiving was missing before the failure

What a DBA would have done

- Configure WAL archiving from day one, not as an afterthought
- Verify recovery works by testing a restore before you need it in production
- Monitor backup age and WAL archive lag as part of normal operations
- Maintain a documented RTO/RPO and test against it regularly

⚠️ **Result: Base backup worked. PITR failed, WAL archiving must be enabled at cluster creation. Operator doesn't enforce this.**

```
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % cat <<EOF | kubectl apply -f -
apiVersion: postgresql.cnpg.io/v1
kind: Backup
metadata:
  name: backup-scenario-6
spec:
  cluster:
    name: experiment-db
EOF
backup.postgresql.cnpg.io/backup-scenario-6 created
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get backup backup-scenario-6 -w
NAME                AGE   CLUSTER         METHOD             PHASE       ERROR
backup-scenario-6   8s    experiment-db   barmanObjectStore   completed
```

```
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl exec -it experiment-db-2 -- psql -U postgres -c "
CREATE TABLE pitr_test (id serial, data text, created_at timestamptz default now());
INSERT INTO pitr_test (data) VALUES ('this data must survive');
SELECT * FROM pitr_test;
"
Defaulted container "postgres" out of: postgres, bootstrap-controller (init)
CREATE TABLE
INSERT 0 1
 id |          data          |          created_at
----+------------------------+-------------------------------
  1 | this data must survive | 2026-05-17 17:09:28.042535+00
(1 row)

amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl exec -it experiment-db-2 -- psql -U postgres -c "DROP TABLE pitr_test;"
Defaulted container "postgres" out of: postgres, bootstrap-controller (init)
DROP TABLE
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl exec -it experiment-db-2 -- psql -U postgres -c "\dt pitr_test"
Defaulted container "postgres" out of: postgres, bootstrap-controller (init)
Did not find any relation named "pitr_test".
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % cat <<EOF | kubectl apply -f -
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: experiment-db-recovered
spec:
  instances: 1
  bootstrap:
    recovery:
      source: experiment-db
      recoveryTarget:
        targetTime: "2026-05-17 17:09:50"
  externalClusters:
  - name: experiment-db
    barmanObjectStore:
      destinationPath: s3://cloudnativepg-operator/backups
      s3Credentials:
        accessKeyId:
          name: s3-backup-creds
          key: ACCESS_KEY_ID
        secretAccessKey:
          name: s3-backup-creds
          key: ACCESS_SECRET_KEY
  storage:
    size: 10Gi
EOF
Warning: Native support for Barman Cloud backups and recovery is deprecated and will be completely removed in CloudNativePG 1.30.0. Found usage in: spec.externalClusters.0.barmanObjectStore. Please migrate existing
 clusters to the new Barman Cloud Plugin to ensure a smooth transition.
cluster.postgresql.cnpg.io/experiment-db-recovered created
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get cluster experiment-db-recovered -w
NAME                     AGE   INSTANCES   READY   STATUS              PRIMARY
experiment-db-recovered   7s    1                   Setting up primary
```

```
amarachiiheanacho@Amarachis-MacBook-Pro conf42-db % kubectl get cluster experiment-db-recovered
kubectl logs -l cnpg.io/cluster=experiment-db-recovered --tail=20
NAME                      AGE   INSTANCES   READY   STATUS             PRIMARY
experiment-db-recovered   52m   1                   Setting up primary
Defaulted container "full-recovery" out of: full-recovery, bootstrap-controller (init)
Defaulted container "full-recovery" out of: full-recovery, bootstrap-controller (init)
Defaulted container "full-recovery" out of: full-recovery, bootstrap-controller (init)
Defaulted container "full-recovery" out of: full-recovery, bootstrap-controller (init)
Defaulted container "full-recovery" out of: full-recovery, bootstrap-controller (init)
Defaulted container "full-recovery" out of: full-recovery, bootstrap-controller (init)
Defaulted container "full-recovery" out of: full-recovery, bootstrap-controller (init)
{"level":"info","ts":"2026-05-17T17:13:21.12620163Z","msg":"Starting webserver","logging_pod":"experiment-db-recovered-1-full-recovery","address":"localhost:8010","hasTLS":false}
{"level":"info","ts":"2026-05-17T17:13:21.226863982Z","msg":"Recovering from external cluster","logging_pod":"experiment-db-recovered-1-full-recovery","sourceName":"experiment-db"}
{"level":"info","ts":"2026-05-17T17:13:22.106908639Z","msg":"Target backup found","logging_pod":"experiment-db-recovered-1-full-recovery","backup":{"backup_name":"backup-20260517170723","backup_label":"'START WAL L
OCATION: 0/26000060 (file 000000010000000000000026)\\nCHECKPOINT LOCATION: 0/260000B8\\nBACKUP METHOD: streamed\\nBACKUP FROM: standby\\nSTART TIME: 2026-05-17 17:07:24 UTC\\nLABEL: Barman backup cloud 20260517T170
724\\nSTART TIMELINE: 1\\n'","begin_time":"Sun May 17 17:07:24 2026","end_time":"Sun May 17 17:07:25 2026","begin_time_iso":"2026-05-17T17:07:24.251214+00:00","end_time_iso":"2026-05-17T17:07:25.574116+00:00","Begi
nTime":"2026-05-17T17:07:24.251214Z","EndTime":"2026-05-17T17:07:25.574116Z","begin_wal":"000000010000000000000026","end_wal":"000000010000000000000027","begin_xlog":"0/26000060","end_xlog":"0/27000060","systemid":
"7640902656783786006","backup_id":"20260517T170724","error":"","timeline":1}}
{"level":"error","ts":"2026-05-17T17:13:22.920948986Z","msg":"Error while restoring a backup","logging_pod":"experiment-db-recovered-1-full-recovery","error":"encountered an error while checking the presence of fir
st needed WAL in the archive: object storage or file not found 000000010000000000000026: WAL not found","stacktrace":"github.com/cloudnative-pg/machinery/pkg/log.(*logger).Error\n\tpkg/mod/github.com/cloudnative-pg
/machinery@v0.4.0/pkg/log/log.go:125\ngithub.com/cloudnative-pg/cloudnative-pg/internal/cmd/manager/instance/restore.restoreSubCommand\n\tinternal/cmd/manager/instance/restore/restore.go:79\ngithub.com/cloudnative-
pg/cloudnative-pg/internal/cmd/manager/instance/restore.(*restoreRunnable).Start\n\tinternal/cmd/manager/instance/restore/restore.go:62\nsigs.k8s.io/controller-runtime/pkg/manager.(*runnableGroup).reconcile.func1\n
\tpkg/mod/sigs.k8s.io/controller-runtime@v0.24.0/pkg/manager/runnable_group.go:260"}
{"level":"info","ts":"2026-05-17T17:13:22.921072508Z","msg":"Stopping and waiting for non leader election runnables"}
{"level":"info","ts":"2026-05-17T17:13:22.921085545Z","msg":"Stopping and waiting for leader election runnables"}
{"level":"info","ts":"2026-05-17T17:13:22.921123255Z","msg":"Stopping and waiting for warmup runnables"}
{"level":"info","ts":"2026-05-17T17:13:22.921377626Z","msg":"Webserver exited","logging_pod":"experiment-db-recovered-1-full-recovery","address":"localhost:8010"}
{"level":"info","ts":"2026-05-17T17:13:22.921436596Z","msg":"Stopping and waiting for caches"}
{"level":"info","ts":"2026-05-17T17:13:22.921584178Z","msg":"Stopping and waiting for webhooks"}
{"level":"info","ts":"2026-05-17T17:13:22.921651228Z","msg":"Stopping and waiting for HTTP servers"}
{"level":"info","ts":"2026-05-17T17:13:22.921671011Z","msg":"Wait completed, proceeding to shutdown the manager"}
{"level":"error","ts":"2026-05-17T17:13:22.921696642Z","msg":"restore error","logging_pod":"experiment-db-recovered-1-full-recovery","error":"while restoring cluster: encountered an error while checking the presenc
e of first needed WAL in the archive: object storage or file not found 000000010000000000000026: WAL not found","stacktrace":"github.com/cloudnative-pg/machinery/pkg/log.(*logger).Error\n\tpkg/mod/github.com/cloudn
ative-pg/machinery@v0.4.0/pkg/log/log.go:125\ngithub.com/cloudnative-pg/cloudnative-pg/internal/cmd/manager/instance/restore.NewCmd.func1\n\tinternal/cmd/manager/instance/restore/cmd.go:101\ngithub.com/spf13/cobra.
(*Command).execute\n\tpkg/mod/github.com/spf13/cobra@v1.10.2/command.go:1015\ngithub.com/spf13/cobra.(*Command).ExecuteC\n\tpkg/mod/github.com/spf13/cobra@v1.10.2/command.go:1148\ngithub.com/spf13/cobra.(*Command).
Execute\n\tpkg/mod/github.com/spf13/cobra@v1.10.2/command.go:1071\nmain.main\n\tcmd/manager/main.go:71\nruntime.main\n\t/opt/hostedtoolcache/go/1.26.3/x64/src/runtime/proc.go:290"}
{"level":"info","ts":"2026-05-17T17:14:26.318332105Z","msg":"Starting webserver","logging_pod":"experiment-db-recovered-1-full-recovery","address":"localhost:8010","hasTLS":false}
{"level":"info","ts":"2026-05-17T17:14:26.419179857Z","msg":"Recovering from external cluster","logging_pod":"experiment-db-recovered-1-full-recovery","sourceName":"experiment-db"}
{"level":"info","ts":"2026-05-17T17:14:27.280129051Z","msg":"Target backup found","logging_pod":"experiment-db-recovered-1-full-recovery","backup":{"backup_name":"backup-20260517170723","backup_label":"'START WAL L
OCATION: 0/26000060 (file 000000010000000000000026)\\nCHECKPOINT LOCATION: 0/260000B8\\nBACKUP METHOD: streamed\\nBACKUP FROM: standby\\nSTART TIME: 2026-05-17 17:07:24 UTC\\nLABEL: Barman backup cloud 20260517T170
724\\nSTART TIMELINE: 1\\n'","begin_time":"Sun May 17 17:07:24 2026","end_time":"Sun May 17 17:07:25 2026","begin_time_iso":"2026-05-17T17:07:24.251214+00:00","end_time_iso":"2026-05-17T17:07:25.574116+00:00","Begi
nTime":"2026-05-17T17:07:24.251214Z","EndTime":"2026-05-17T17:07:25.574116Z","begin_wal":"000000010000000000000026","end_wal":"000000010000000000000027","begin_xlog":"0/26000060","end_xlog":"0/27000060","systemid":
```

The Verdict

The operator doesn't replace a DBA.

It replaces the routine execution of DBA work.

✅ Operator excels at

- Pod-level failover (~11 seconds, zero intervention)
- Major version upgrades (~47 seconds, one command)
- Replica provisioning and cloning
- Self-healing after pod failures

❌ Still needs a human for

- Storage topology decisions (use networked storage)
- Replication lag monitoring & alerting
- Disk-full recovery on immutable OS (Talos)
- WAL archiving and backup architecture planning

Thank you!

Socials

- Linkedin: Amarachi Iheanacho
- Instagram: @amaraiheanach0
- Tiktok: @amaraiheanach0