EDB POSTGRES AI

Summer Academy

Stateful on Kubernetes: Running Postgres the Right Way

01 02 03 04

Raphaël Chir - Senior Sales Engineer
Leonardo Cecchi - Principal Engineer

60 MINUTES

AGENDA

Keep your questions coming in the chat — we answer live at the end.

- Welcome
- Approaches from Stateless to Stateful
- Why CloudNativePG Operator ?
- HA/DR Architectures
- CNPG-I
- What's new ?
- **CNPG in action** — Trailer
- Wrap-up, community & Q&A

TIMEBOXED 45 minutes of content + a generous Q&A buffer — if a section runs over, the buffer absorbs it.

EDB POSTGRES AI

PostgreSQL

Major contributor to the core database engine, release after release.
30% of PostgreSQL code contributed every year
2/7 PostgreSQL Core Team members · 7 committers · 40+ contributors

CloudNativePG

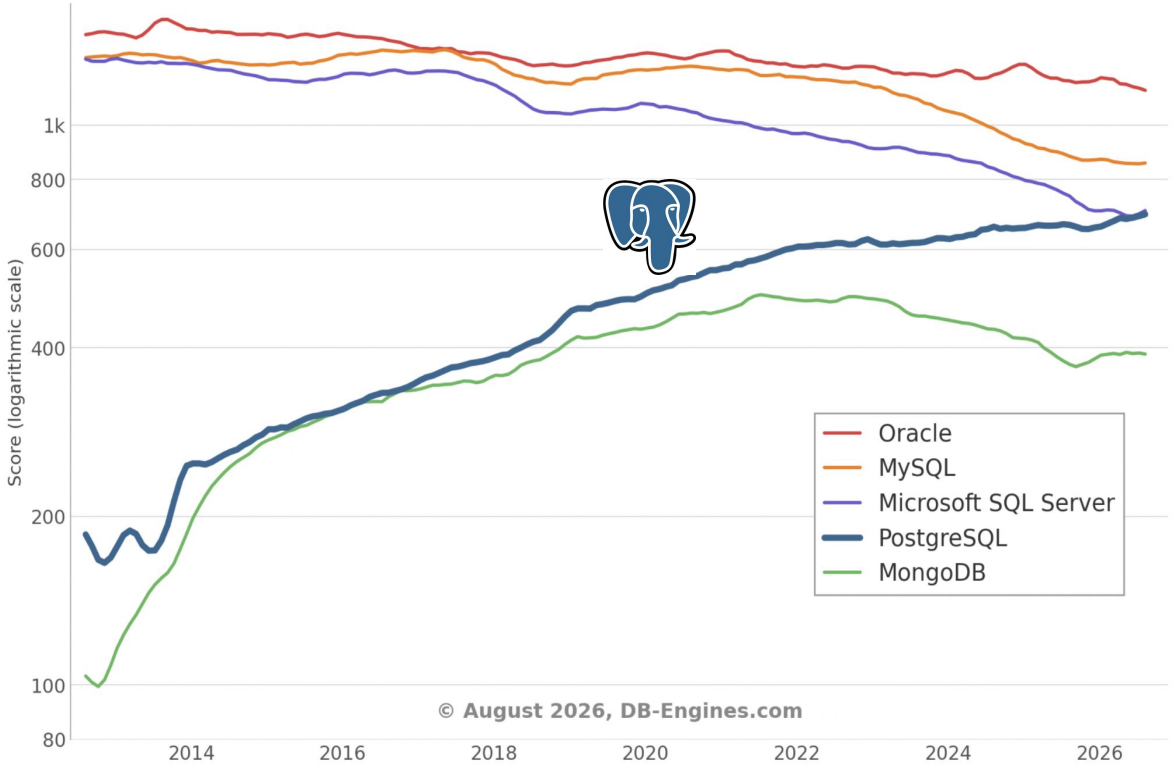
Kubernetes operator for PostgreSQL
Originally created **by EDB**
CNCF Sandbox project
Maintained **by the CNPG Community and EDB**

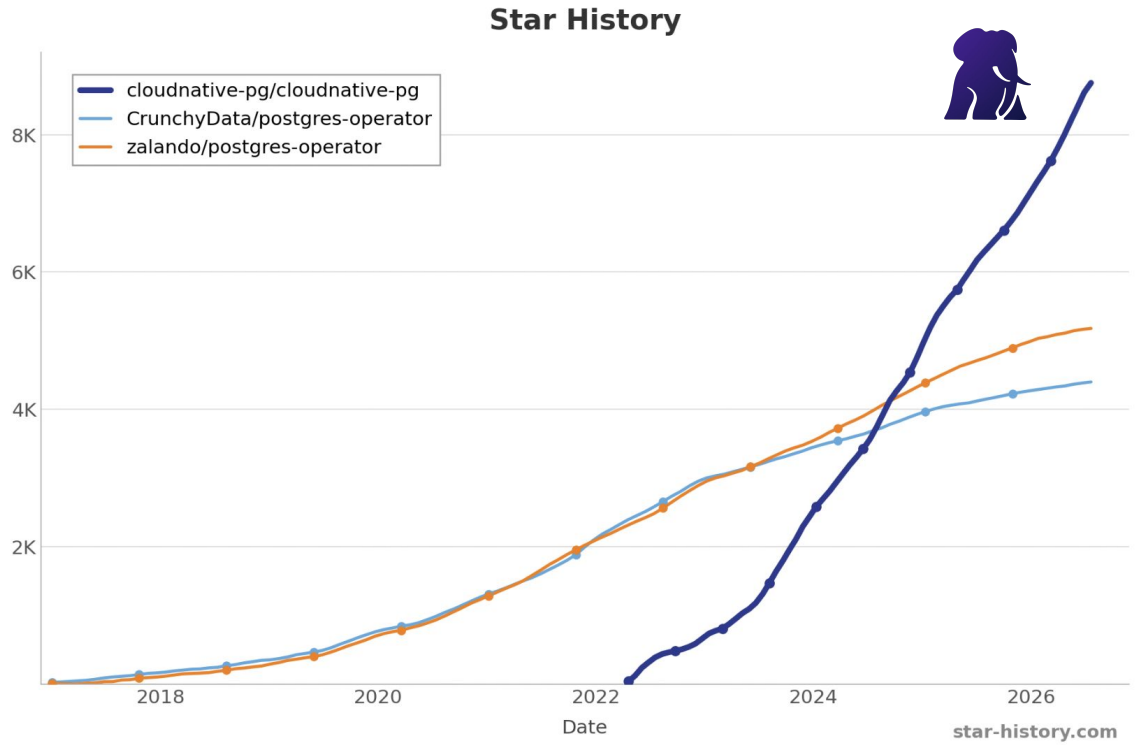
CLOUD NATIVE COMPUTING FOUNDATION

EDB The creator of **CloudNativePG**, leading **PostgreSQL** contributor.

POSTGRESQL IS BOOMING

The database everyone wants to run everywhere — including on **Kubernetes.**





CONTEXT The question is no longer “should we run Postgres ?” — it is “how do we run it on the same platform as everything else?” For most teams, that platform is Kubernetes.

APPROACHES FROM STATELESS TO STATEFUL

K8S the Cloud Native platform

Kubernetes is designed to be resilient, scalable and cloud agnostic

Self-healing — Failed containers restart and reschedule automatically

Declarative configuration — Desired state in YAML, reconciled forever

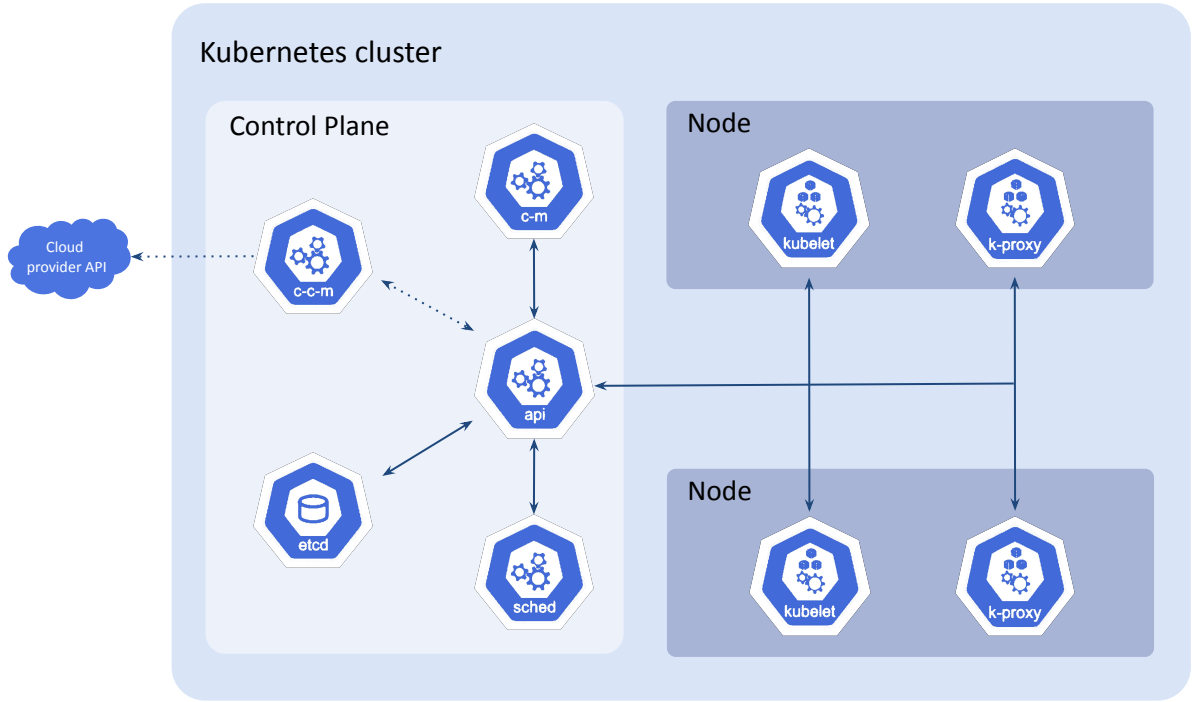
Rolling updates & rollbacks with minimal downtime

Portability across public, private and hybrid clouds

One platform for apps and infrastructure (GitOps-ready)

Decoupled architecture – Each Kubernetes resource has a single responsibility and interacts with other resources through references.

Scalability - Each nodes and pods can scale in/out and scale up/down



Kubernetes applies the principles of Cloud Native Architecture

Stateless vs Stateful

Kubernetes was designed for stateless workloads — databases are anything but.

Deployments are designed for stateless application

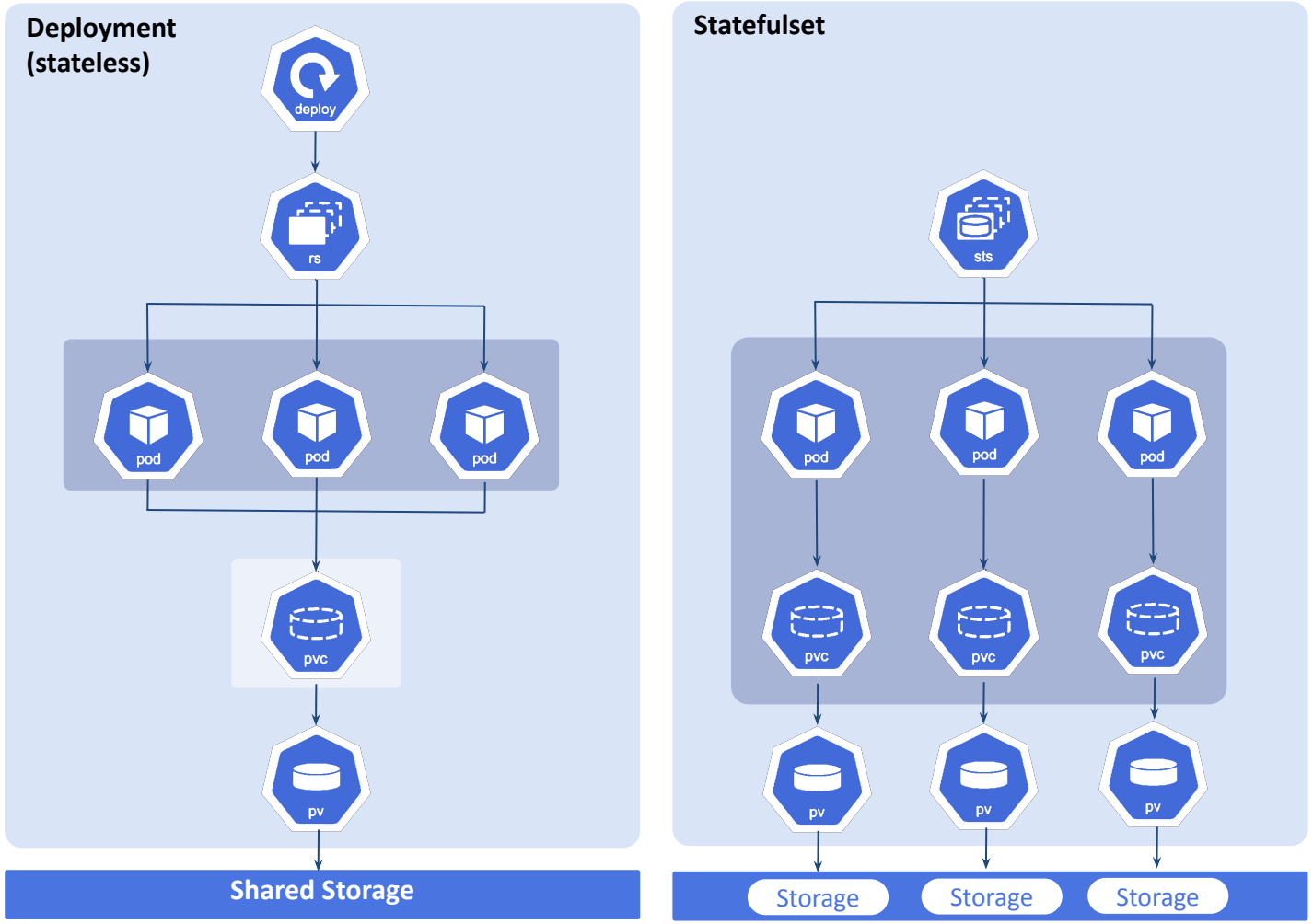
- It doesn't allocate a group of PVs for each pods
- A PVC is here considered as a decoupled component (as Secret, ConfigMap).

Statefulset object is stateful

- Stable identity (persistent Pod name and DNS).
- Dedicated persistent storage (PVCs group / Pod)
- Ordered lifecycle management (deterministic Pod creation, deletion, and rolling updates).

A Statefulset is just **solving K8S needs, NOT database needs** at all !

- What about database instance roles, automatic failover, HA/DR, database performances ?
- You need to build it yourself and that's why CNPG comes into the game !



THE GAP IS CLOSING — BUT NOT BY ITSELF

Kubernetes has matured into a production platform for stateful workloads.

2018	TODAY
Stateless workloads	**Stateful workloads**
Limited storage solutions	**Local Storage support, Enterprise-grade CSI storage**
Manual operations	**Kubernetes Operators**
Limited observability	**Mature monitoring ecosystem (Prometheus)**
Experimental for stateful workload	**Production platform for mission-critical data**

TAKEAWAY - The missing piece is **operational knowledge** — and that is exactly **what an operator encodes**.

WHY CLOUDNATIVEPG OPERATOR ?

WHY CLOUDNATIVEPG?

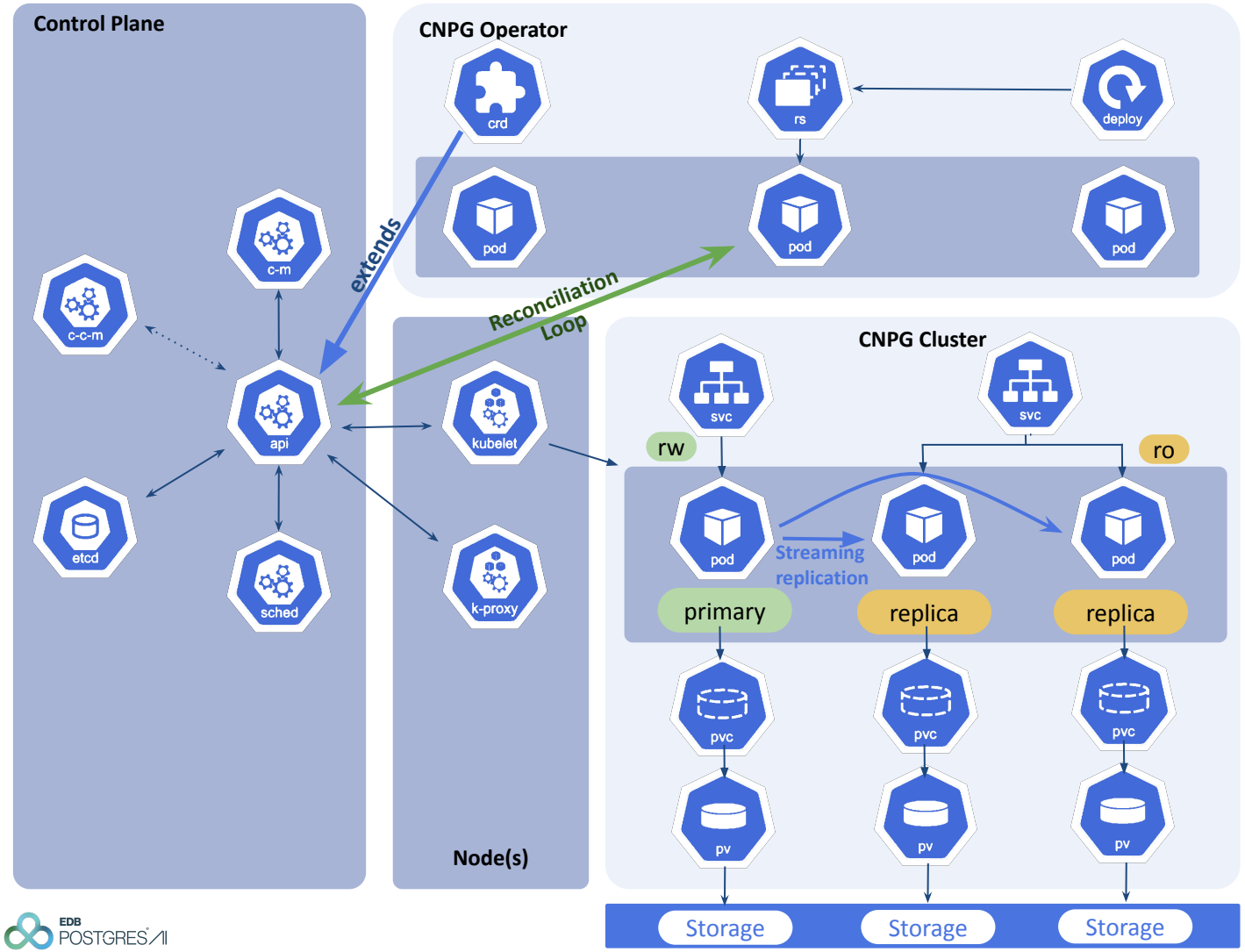
It fills exactly the gaps between what Kubernetes does and what PostgreSQL needs.

KUBERNETES	POSTGRESQL	CLOUDNATIVEPG
Runs Pods	Needs a primary	**Manages primary election**
Creates Volumes	Needs durable data	**Manages storage lifecycle**
Restarts containers	Needs failover and switchover	**Performs automatic failover, switchover for maintenance operations**
Exposes Services	Needs client routing	**Services are transparently following the instances roles**
Schedules workloads	Needs backups	**Manages backup & recovery**

KEY IDEA - CloudNativePG orchestrates PostgreSQL, no StatefulSets, no Patroni, no etcd !

CNPG Operator under the hood overview

CNPG Operator managed the state of PostgreSQL instances



CNPG Operator provides

- **CRD** that extends kube-apiserver
- **Pods** managed by a **deployment** : in charge of managing and controlling your CNPG clusters through a **reconciliation loop** with kube-apiserver

CNPG Cluster

- Fully deployed and controlled by the CNPG operator
- Primary/Replicas clusters role with their own storage
- 3 Services K8S object : rw, ro and r (read everywhere) entry points plugged to the instance pods
- Secrets, ConfigMaps, to secure your connexion and tuned your clusters

THE “TRUST IT IN PRODUCTION” CHECKLIST

Six capabilities you would otherwise build and maintain yourself.

BACKUPS & PITR

Continuous WAL archiving + scheduled base backups via the **Barman Cloud Plugin.**

Recovery is a **bootstrap method** — test it like a feature.

MONITORING

Native Prometheus exporter on every instance with **PodMonitor** support.

Ready-made **Grafana dashboards**

TLS EVERYWHERE

Operator-managed certificates **for client and replication traffic**

Rotated automatically, cert-manager friendly.

MINOR UPGRADES

Minor Postgres updates with **automated switchover** — near-zero downtime.

MAJOR UPGRADES

Declarative in-place pg_upgrade: change the image tag, the operator orchestrates shutdown, upgrade, rollback safety.

DAY-2 CONTROLS

Scheduled switchovers for maintenance operation
Automatic Failover (quorum based)
Hibernation, Fencing
CNPG Operator upgrades
Tools to administrate your PG clusters

GITOPS - Everything on this slide is declarative — it lives in Git, not in a runbook.

HA / DR ARCHITECTURE

HA CNPG cluster

A CloudNativePG cluster spread across three availability zones of a region

Features

– Primary sends WALs to replicas - streaming replication (sync or async), known as physical replication.

– Primary archives WALs to Object storage

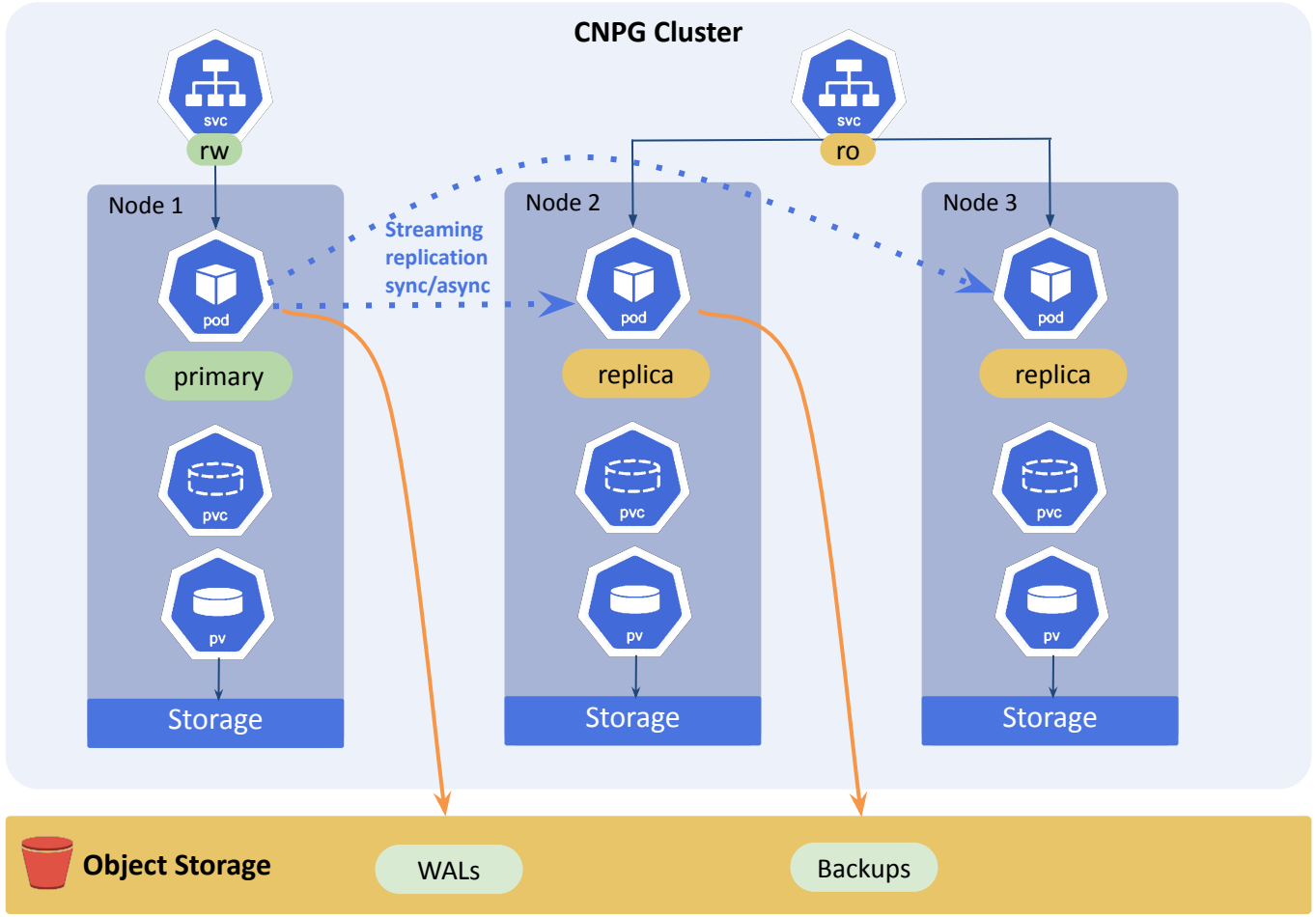
– Backups are by default performed by Replicas to Object storage

– Services always follow the role of Postgres instances

– Promote a replica to primary role updates all the flows accordingly

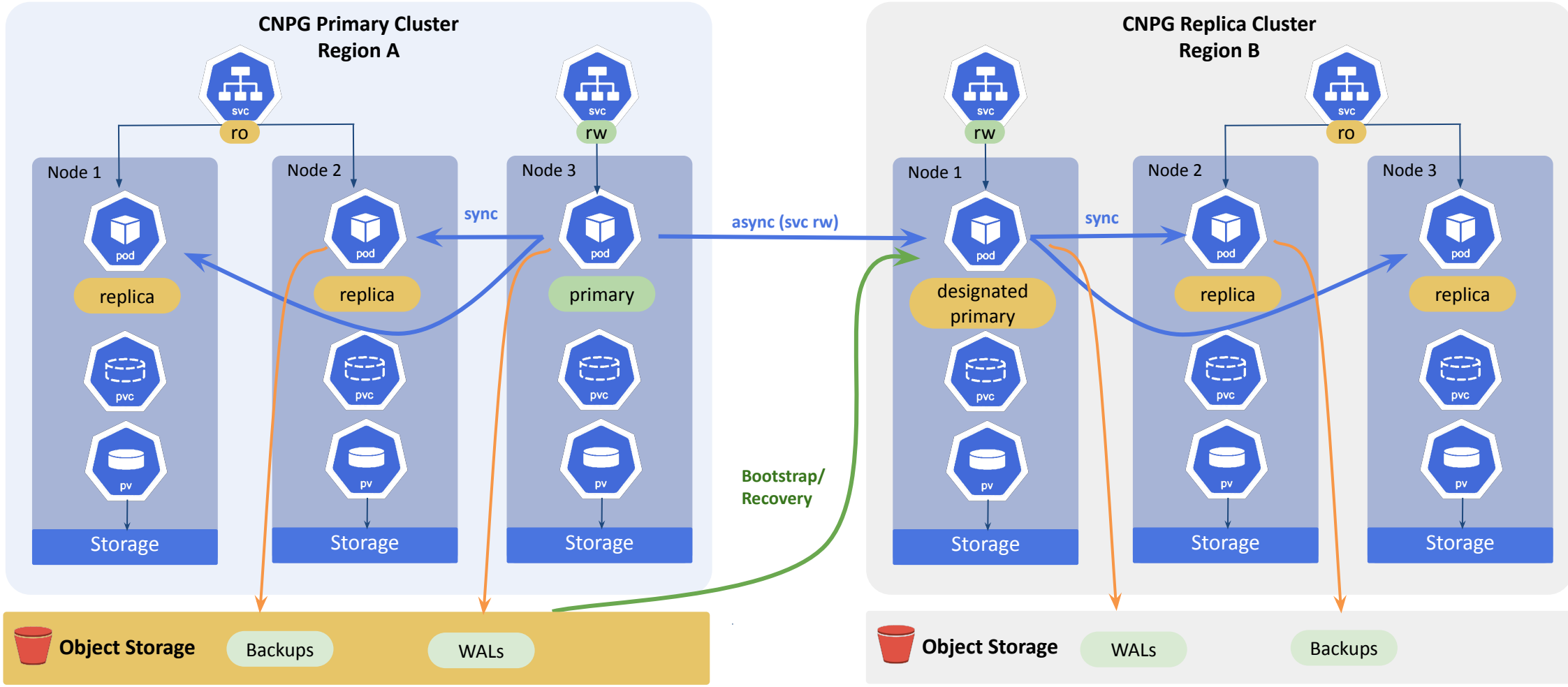
– Automatic failover in case of primary instance failure

For Dev & test, low-traffic apps, cost-sensitive projects you should probably set up just a single instance and still gets continuous backup to object storage.



HA/DR CNPG cluster

Demotion token is generated during a planned switchover to prevent data loss and sync the process of demotion/promotion.



CNPG-I

CNPG-I — WHY THE FUTURE IS PLUGIN-FIRST

The same trajectory Kubernetes itself followed with CSI and CNI.

CloudNativePG

BEFORE CNPG-I

Supporting every PostgreSQL use case would have slowed the operator's development.

- Fork the project to add custom behavior, or
- Extend the upstream codebase on top
- Both = maintenance debt, slower upgrades, delayed features

WITH CNPG-I

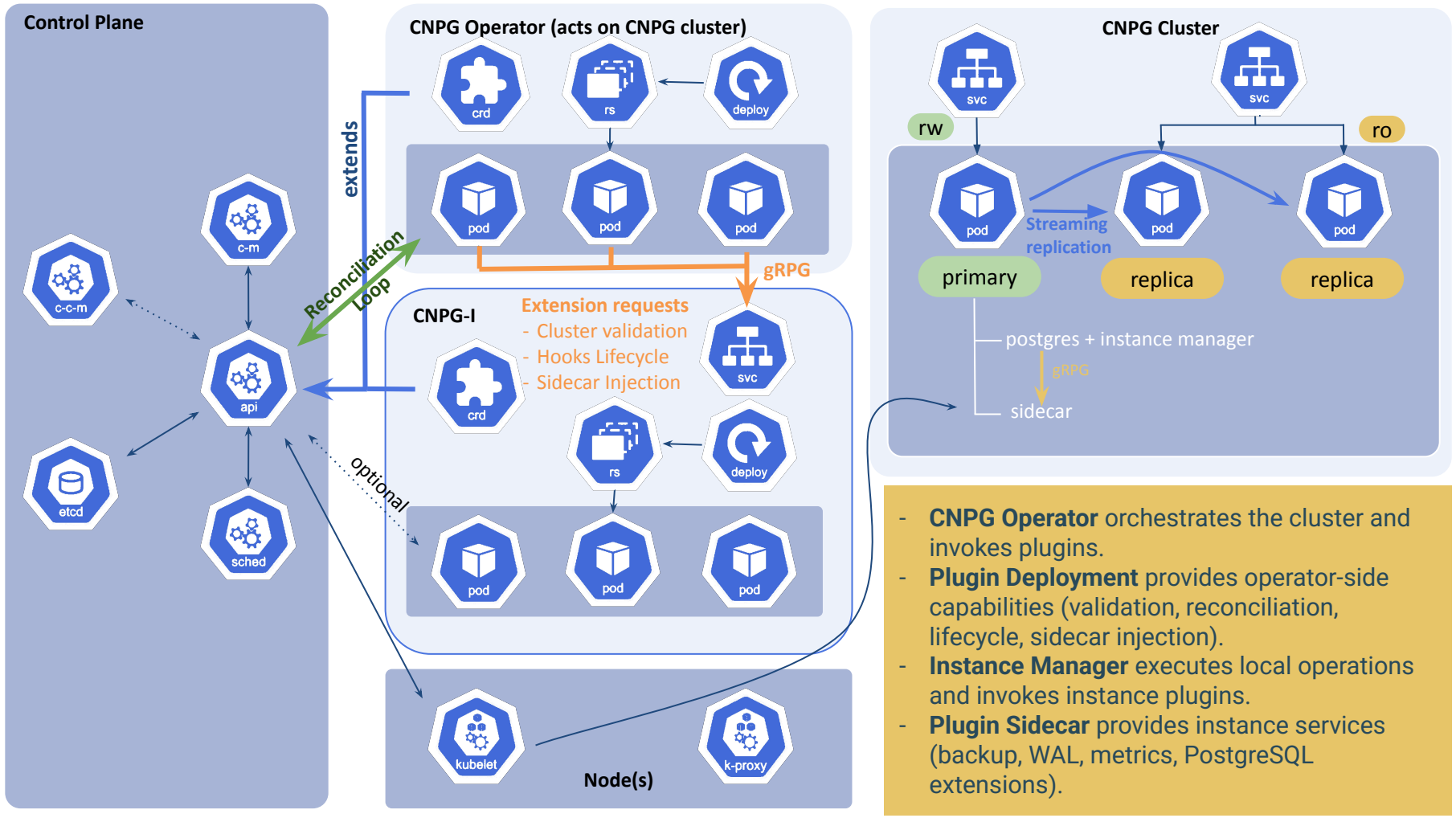
Contributors can extend CNPG without touching its source code, easing maintenance and empowering the project and its community.

- A stable gRPC integration point at key lifecycle moments
- Backups, recovery, sub-resource reconciliation — without touching the core
- The operator becomes a pure, maintainable control plane

NOTES: Kubernetes moved storage out of the core with CSI. CloudNativePG is doing the same for Postgres operations.

HOW CNPG-I WORKS

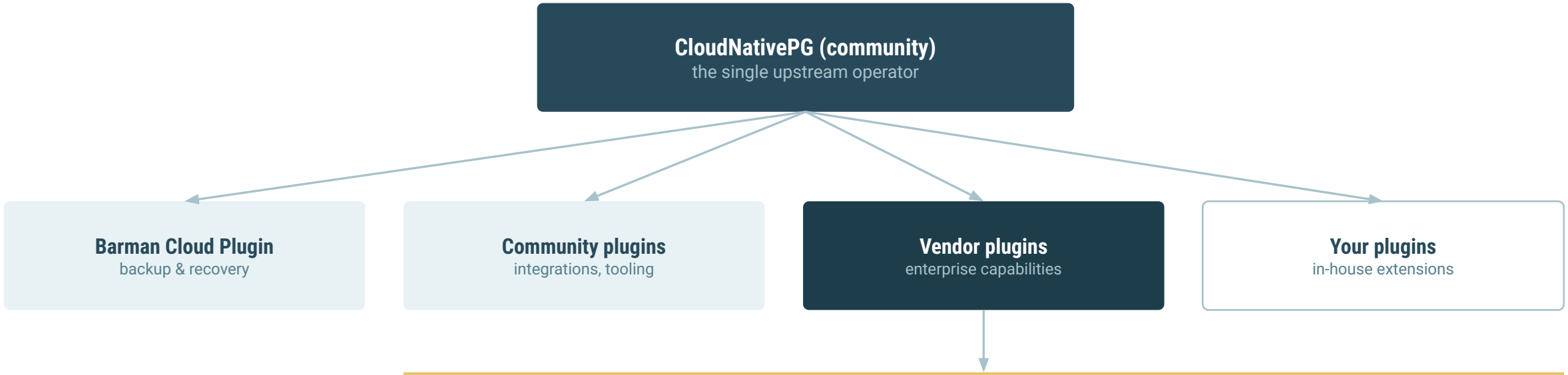
gRPC + mandatory mTLS — inspired by the Container Storage Interface.



- **CNPG Operator** orchestrates the cluster and invokes plugins.
- **Plugin Deployment** provides operator-side capabilities (validation, reconciliation, lifecycle, sidecar injection).
- **Instance Manager** executes local operations and invokes instance plugins.
- **Plugin Sidecar** provides instance services (backup, WAL, metrics, PostgreSQL extensions).

ONE COMMUNITY OPERATOR — EXTENSIONS AS PLUGINS

What plugin-first means for the ecosystem, including commercial vendors.



NEW - The proprietary **EDB CNPG operator** fork will be retired and replaced by **CNPG-I plugins** on top of the community operator.

Enterprise capabilities (TDE, advanced distributions, geo-distributed Global Cluster with PGD) arrive as plugins.

BENEFITS - One codebase to trust, one community to strengthen: no divergence between “community” and “enterprise” Postgres on Kubernetes

WHAT'S NEW?

WHAT'S NEW IN CLOUDNATIVEPG 1.30

Released June 2026 — Kubernetes 1.34–1.36, PostgreSQL 14–18.

01 PRIMARY LEASE

A Kubernetes Lease serializes primary promotion — a mutex the instance manager must hold before acting as primary. Configurable via .spec.primaryLease.

02 DATABASEROLE CRD + TLS CLIENT CERTS

Roles as standalone GitOps objects — with operator-generated, auto-renewed client certificates for password-free cert authentication.

03 POOLER IMAGE CATALOGS

PgBouncer images managed centrally via ImageCatalog — update the catalog, every Pooler rolls out automatically.

04 SECURITY HARDENING

Three advisories fixed: search_path pinning, authenticated operator→instance calls (mTLS fingerprint), SCRAM-SHA-256 password encoding.

HEADS-UP Native (in-tree) Barman Cloud support will be removed in 1.31 — migrate to the Barman Cloud Plugin now. Which brings us to CNPG-I…

A COMMUNITY PROJECT ON A CNCF TRAJECTORY

Vendor-neutral governance, verified adopters, and an incubation application in progress.

CLOUD NATIVE COMPUTING FOUNDATION

~9,000

GitHub stars — fastest-growing Postgres operator

225+

contributors · 700+ forks · public ADOPTERS.md

5

PostgreSQL major versions supported (14 → 18)

2025

accepted into the CNCF Sandbox (January)

2022

Open-sourced by EDB

JAN 2025

CNCF Sandbox

NOV 2025

Incubation application (cncf/toc #1961)

IN PROGRESS

2026

Governance restructuring: Steering Committee, non-EDB maintainer pathway, one vote per org

Why it matters to you: incubation-level due diligence means audited governance, a security response process and adopter interviews — signals you can reuse in your own production-readiness review.

EDB POSTGRES AI


EDB
POSTGRES®AI

KEY TAKEAWAYS

01 THE GAP IS OPERATIONAL, NOT TECHNICAL

Kubernetes can run stateful workloads — what a database needs is encoded DBA knowledge. That is what an operator is.

02 FAILURE SCENARIOS ARE COVERED BY DESIGN

Primary Lease, quorum failover, self-fencing isolation checks — deterministic behavior instead of 3 a.m. runbooks.

03 PLUGIN-FIRST IS THE CSI MOMENT FOR POSTGRES

Backups already moved out of the core; in-tree Barman goes away in 1.31. Extensions — community or commercial — are CNPG-I plugins.

04 THE PROJECT IS MATURING IN THE OPEN

CNCF Sandbox → incubation application, vendor-neutral governance, public adopters — signals for your own production review.

BOTTOM LINE Kubernetes-native Postgres you can actually trust in production — and a community you can join today.

JOIN THE CLOUDNATIVEPG COMMUNITY

A CNCF project is only as strong as its community — here is how to plug in (pun intended).

SLACK

#cloudnativepg-users on the CNCF Slack — maintainers answer daily

GITHUB

github.com/cloudnative-pg
issues, roadmap, CNPG-I protocol & examples

DOCS 1.30

cloudnative-pg.io/docs
architecture, failure modes, plugin guide

ADOPTERS.MD

Running CNPG in production? Add your organization — it directly supports the CNCF incubation

COMMUNITY MEETINGS

Public schedule on the CNCF calendar — everyone welcome

THIS WEBINAR

Replay + slides + demo repository shared by email after the session

THANK YOU

QUESTIONS?

NEXT SUMMER ACADEMY SESSION

STOP MANAGING SILOS: A MODERN ANALYTICS ARCHITECTURE BUILT ON POSTGRES

14:00-15:00 UTC+2
27TH AUGUST

FREE TRAINING

EDB ESSENTIALS FOR CLOUDNATIVEPG

EDB POSTGRES AI