Data Community Conference Switzerland 2024

PostgreSQL in Kubernetes: Operation - Day 2

Borys Neselovskyi

www.data-community.c

About me

follow me:

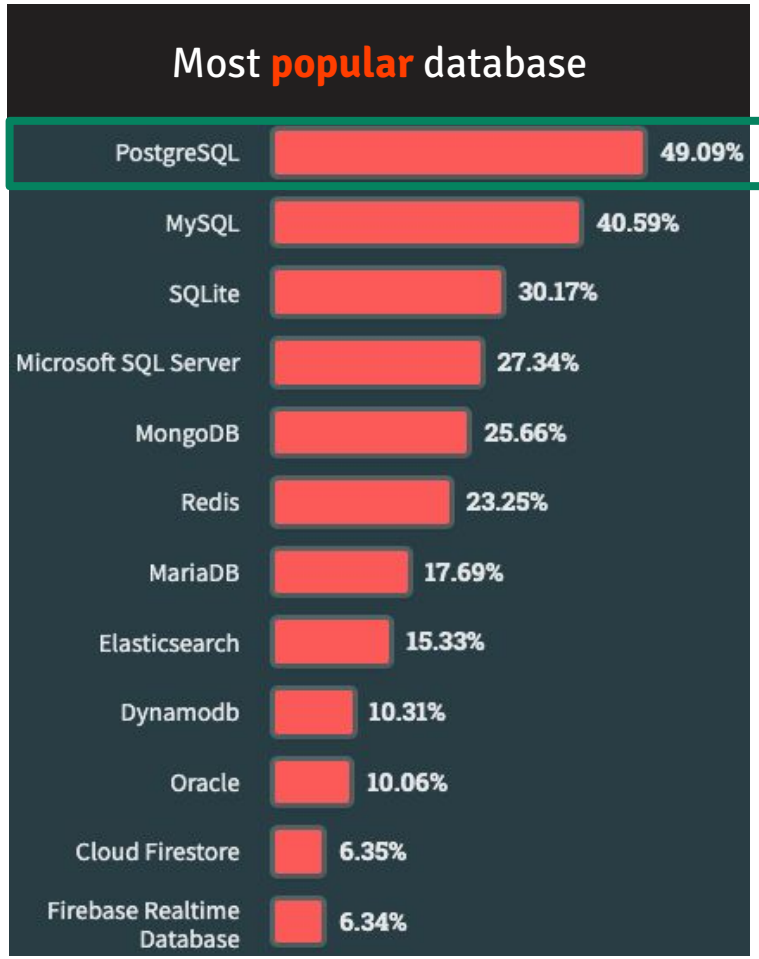
@bneselov

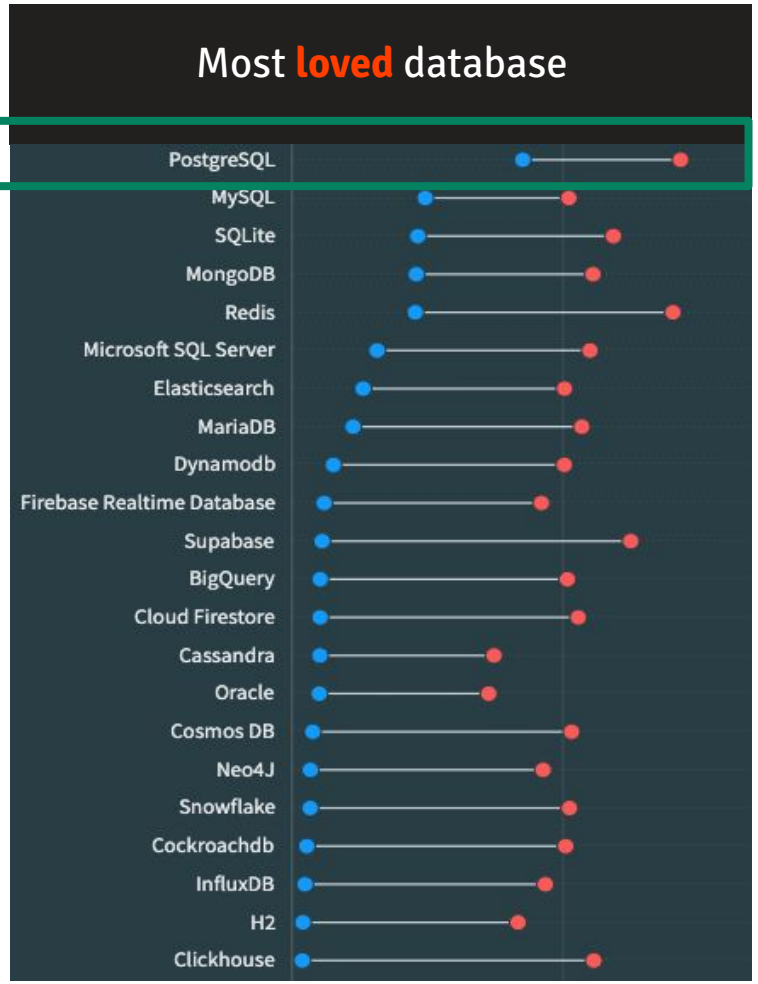
https://www.linkedin.com/in/neselovskyi

- Senior Solution Engineer @ EDB
- Oracle Ace Alumni
- Member of DOAG Community
  - Leading the Middleware Stream
- Oracle Certified Professional
- Postgres Certified Professional
- Database/DevOps/Cloud Expert

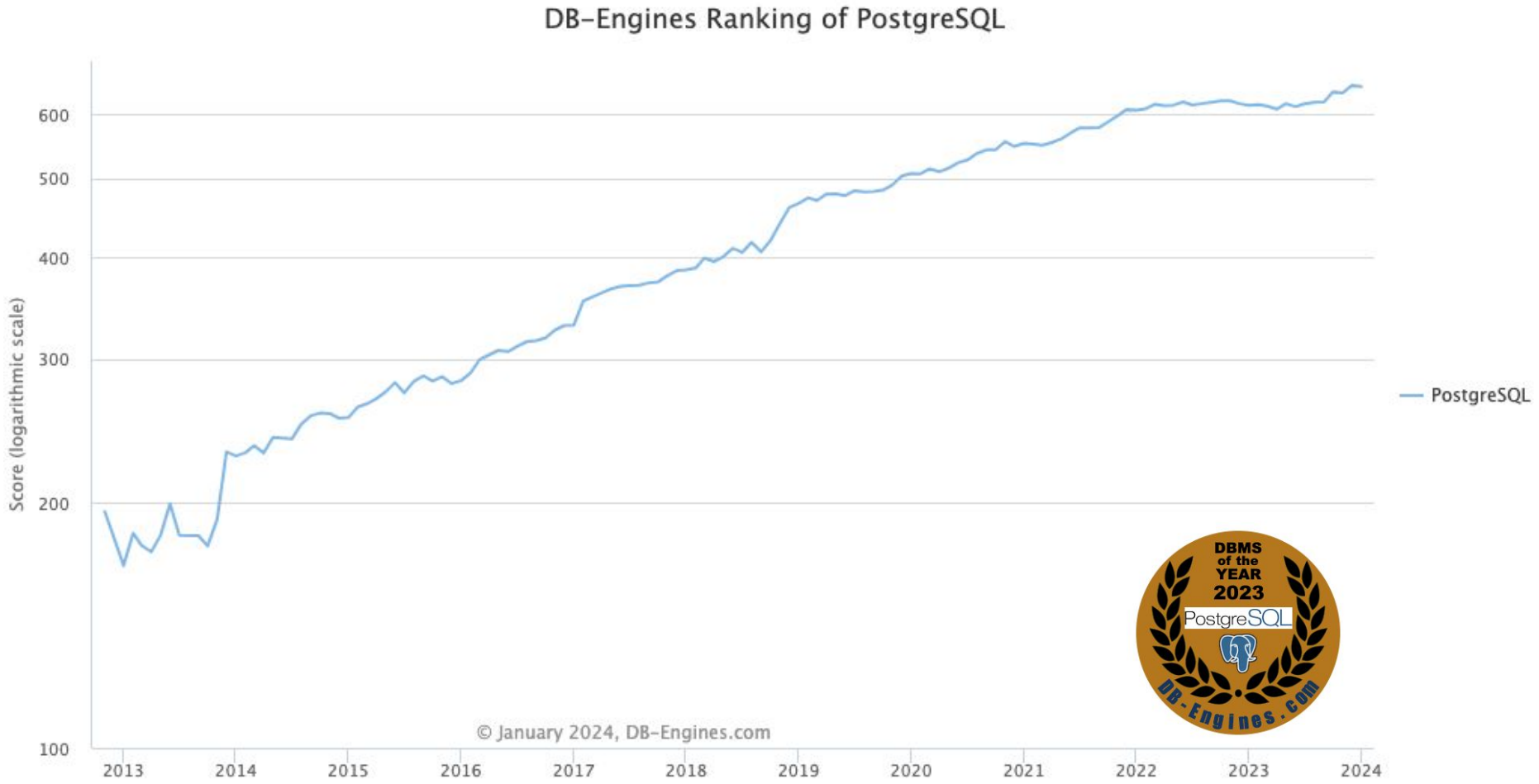
PostgreSQL Is Winning

The most admired, desired & used database - Source: Stack Overflow Developer Survey, 2023





DBMS of the Year

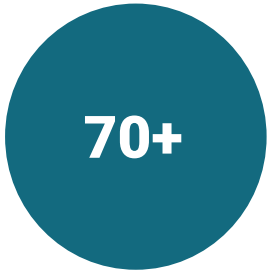


https://db-engines.com/en/blog_post/106

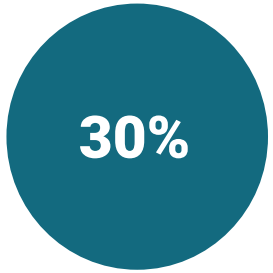
EDB - The Heartbeat of Postgres

Built over 20 years, EDB is the world's largest software, support, and services company focused exclusively on PostgreSQL

F500

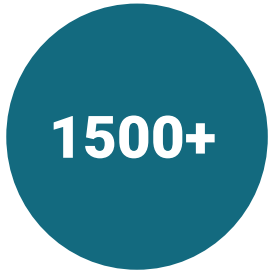


Postgres code contributed



Source of Innovation

Global customers



Customer requirements influence the future direction of Postgres

Dedicated PG engineers

Unparalleled expertise

Postgres core team members

Central source of community influence and expertise

Postgres Operator in Nutshell

CloudNativePG/EDB Postgres for Kubernetes

CloudNativePG 

- Kubernetes operator for PostgreSQL
- “Level 5”, Production ready
- Day 1 & 2 operations of a Postgres database
- Open source (May 2022)
  - Originally created by EDB
  - Apache License 2.0
  - Vendor neutral openly governed
- Extends the K8s controller
  - Status of the `Cluster`
  - “no Patroni, No statefulsets”
- Immutable application containers
- Fully declarative

EDB CloudNativePG 

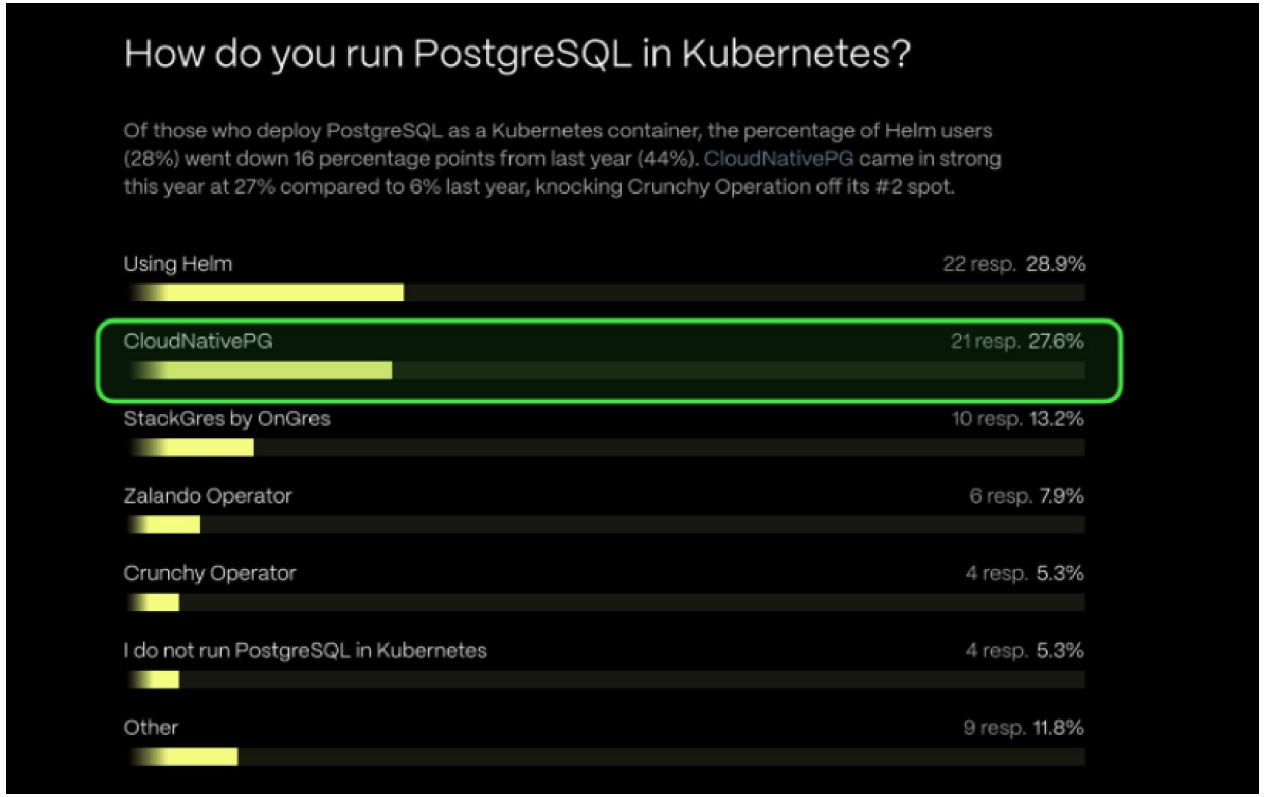
- All Features of CloudNativePG

  +

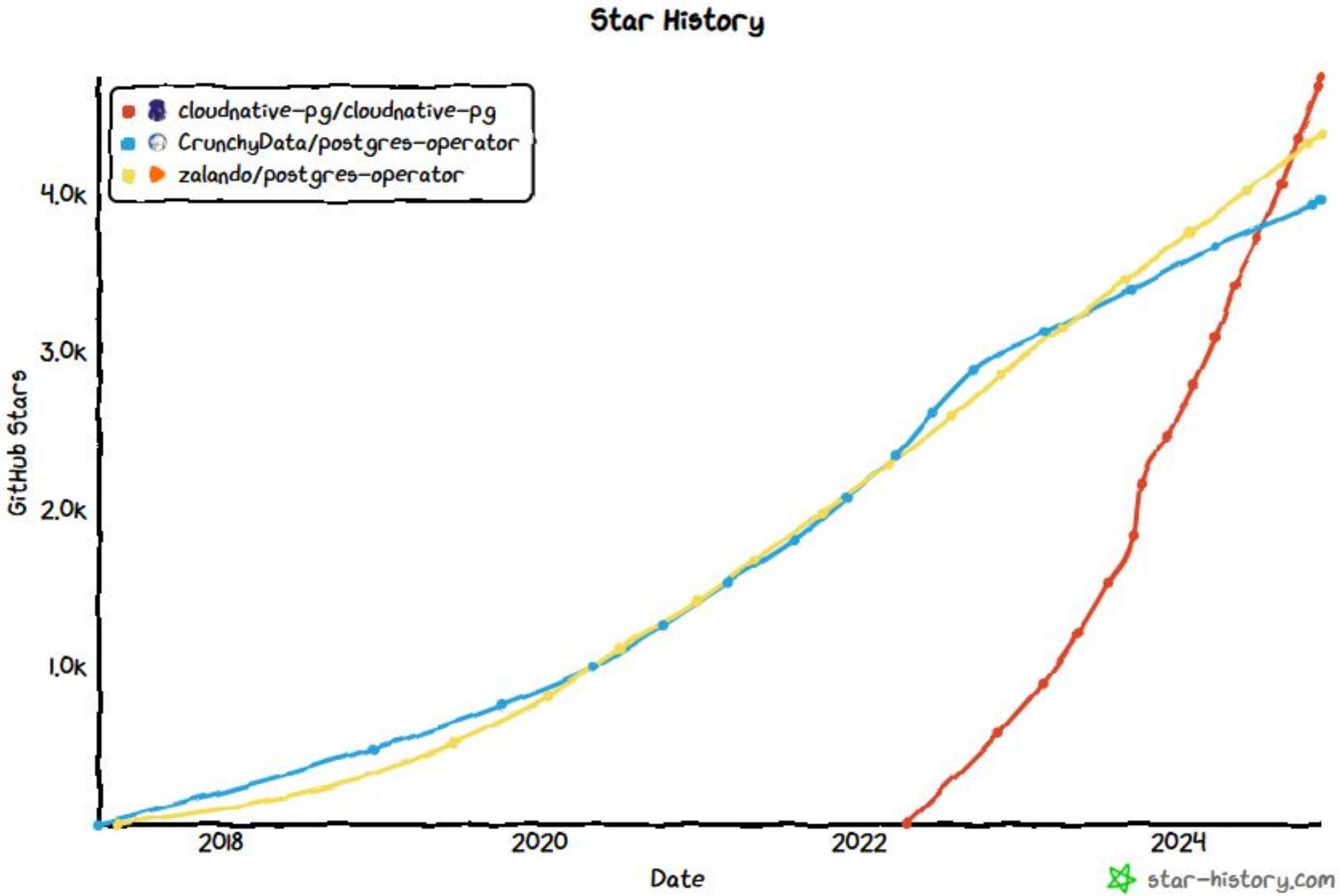
- Provides Long Term Support
- Access to EDB Postgres Advanced (TDE + Oracle Compatibility layer + Security features)
- Red Hat OpenShift compatibility
- Kubernetes level backup integration
  - Generic external backup interface
- **Coming soon**: Hybrid Control Plane (Preview) - centralized management and automation solution, built on Kubernetes, for EDB Postgres AI

CloudNativePG: The Most Popular Postgres Operator in 2023

- A production ready operator originally created by EDB and open sourced in May 2022
- Jumped from 6.1% of shares in 2022 (3rd position) to 27.6% in this year's survey. This makes it the most popular operator, surpassing Stackgres (13.2%) and Zalando (7.9%)
- 25 million downloads
- 4000+ Stars on the Github



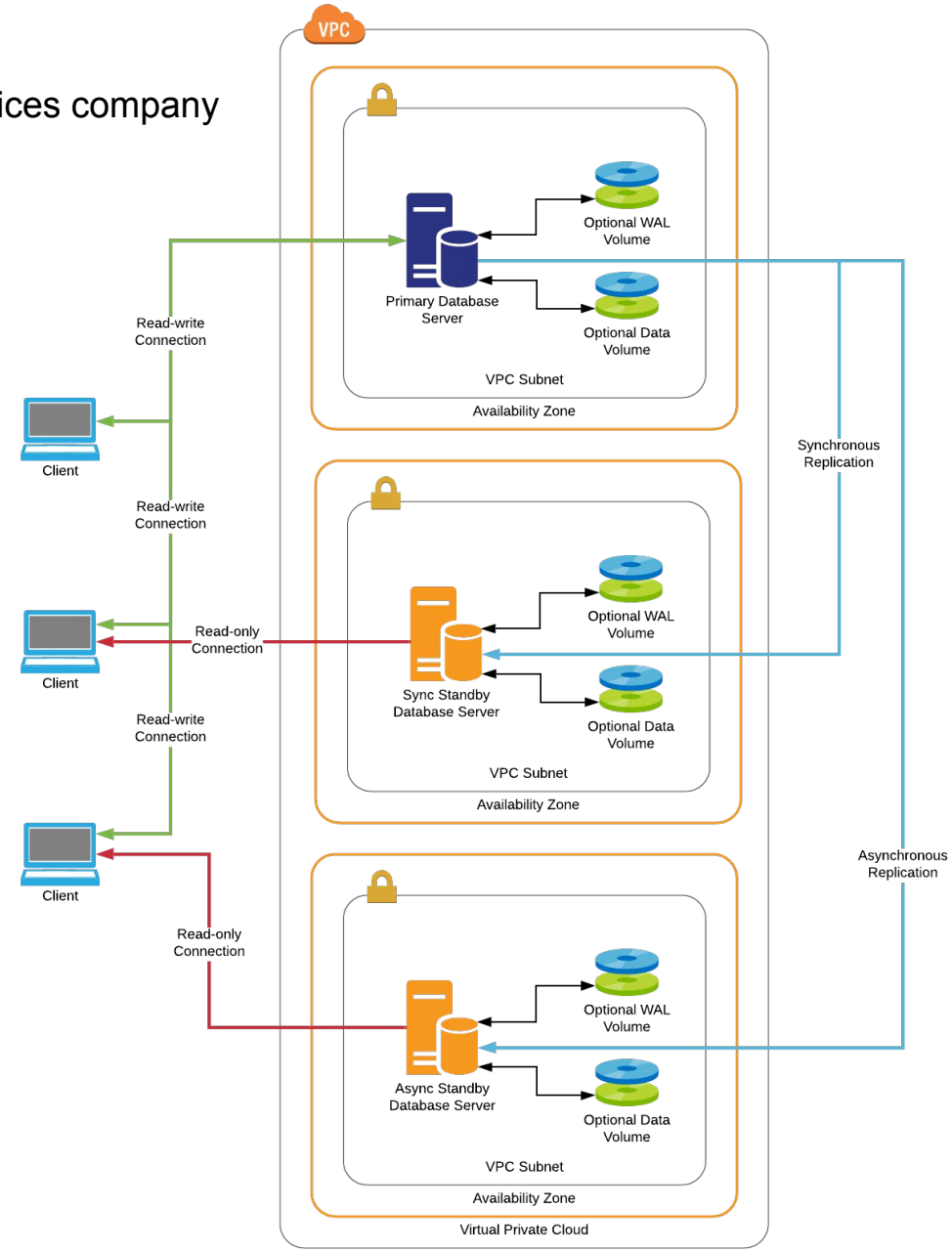
Source: https://www.timescale.com/state-of-postgres/2023



Imperative vs Declarative

Built over 20 years, EDB is the world's largest software, support, and services company focused exclusively on PostgreSQL

- Create and configure VMs
- Create a PostgreSQL 13 instance
- Configure for replication
- Clone a second one
- Set it as a replica
- Clone a third one
- Set it as a replica
- Configure networking
- Configure security
- etc.



Convention over configuration

Declarative - simple to install, simple to maintain

There's a PostgreSQL 15 cluster with 2 replicas:

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: myapp-db
spec:
  instances: 3
  imageName: quay.io/enterprisedb/postgresql:15.2

  storage:
    size: 10Gi
```

Features

- Self-healing through failover and automated recreation of replicas
- Capacity management with scale up/down capabilities
- Planned switchovers for scheduled maintenance
- Read-only and read-write Kubernetes services definitions
- Rolling updates for Postgres minor versions and operator upgrades
- Continuous backup and point-in-time recovery
- Connection Pooling with PgBouncer
- Integrated metrics exporter out of the box
- PostgreSQL replication across multiple Kubernetes clusters
- Separate volume for WAL files
- … and much more

Installing CloudNativePG

- Via the official Kubernetes manifest
- Via a custom Kubernetes manifest generated by the plug-in
- Via the official Helm Chart

```
kubectl apply -f \
https://raw.githubusercontent.com/cloudnative-pg/cloudnative-pg/main/releases/cnpg-1.19.0.yaml
```

Deploy a 3-node HA Postgres Cluster

- Install the latest PostgreSQL 15 minor
- Create a 3-node PostgreSQL 15 cluster
  - A primary, two standby (one synchronous)
  - Use mTLS authentication for the replicas
  - Use replication slots
- Resources:
  - RAM: 4 GB
  - CPU: 8 cores
  - Storage: 40GB for PGDATA, 10GB for WALs
- A user and a database for the application
- A reliable and consistent way to access the primary via network

myapp-db.yaml

```
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: myapp-db
spec:
  resources:
    requests:
      memory: "4Gi"
      cpu: 8
    limits:
      memory: "4Gi"
      cpu: 8
        …
```

postgresql.conf

```
instances: 3
minSyncReplicas: 1
maxSyncReplicas: 1
replicationSlots:
  highAvailability:
    enabled: true

storage:
  size: 40Gi
walStorage:
  size: 10Gi

postgresql:
  parameters:
    shared_buffers: "1GB"
```

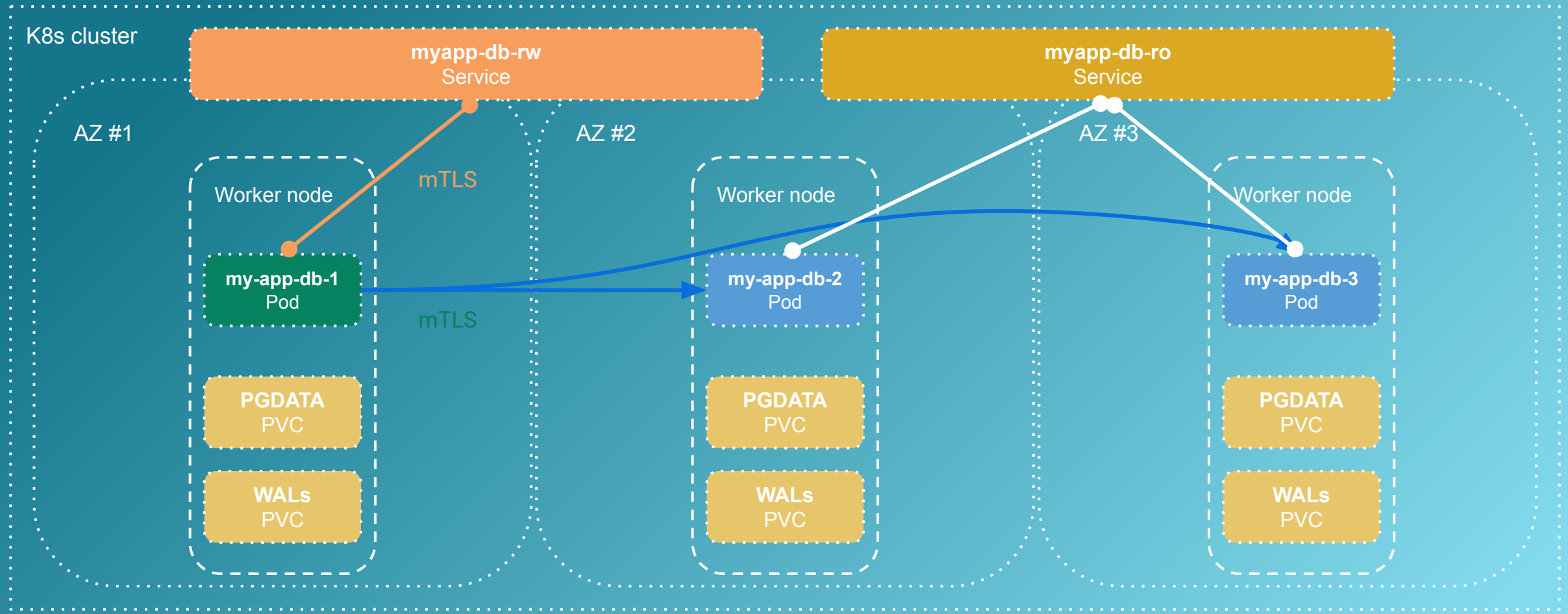
replication

storage

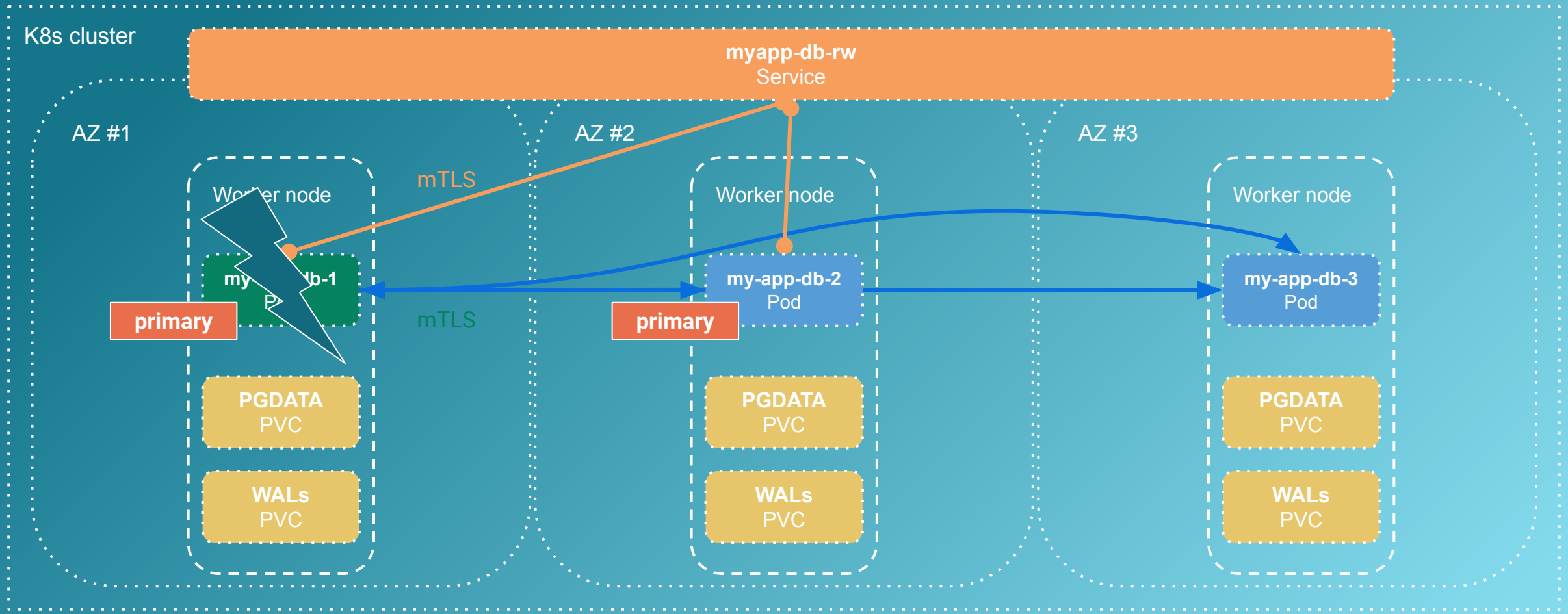
How to deploy the PostgreSQL Cluster

```
kubectl apply -f myapp-db.yaml
```

This is what happens under the hood

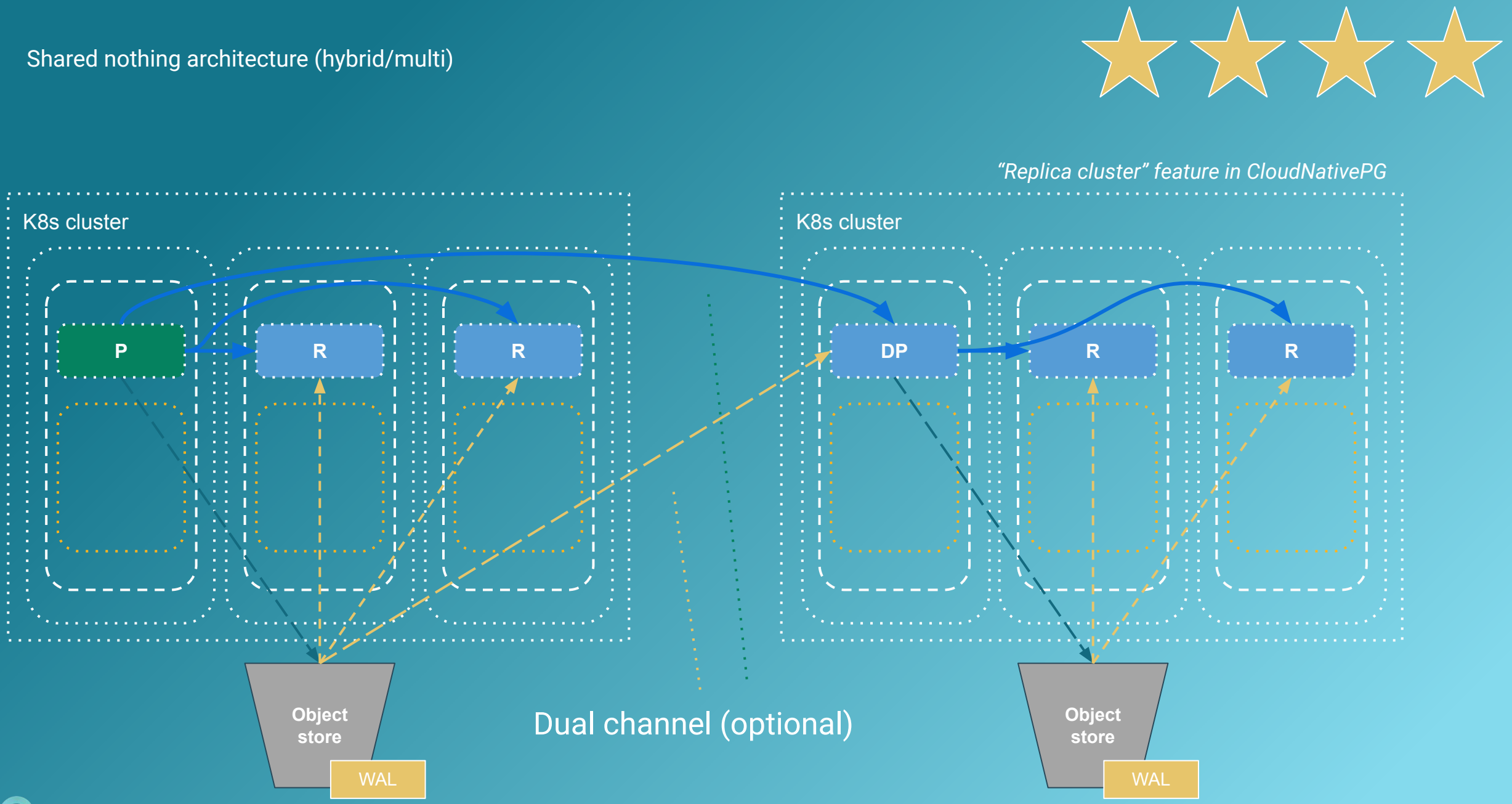


Automated failover

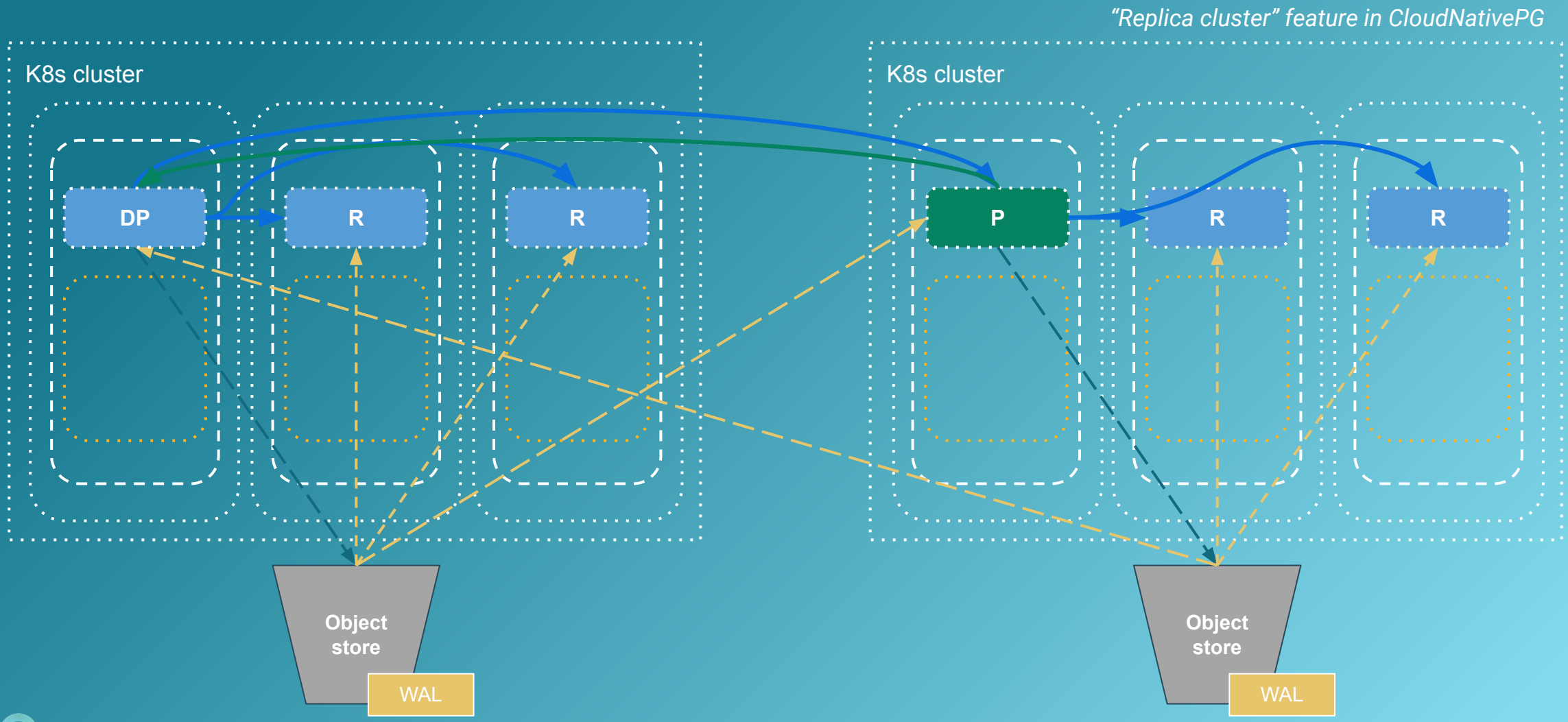


Deployments across Kubernetes clusters

Shared nothing architecture (hybrid/multi)



Shared nothing architecture (hybrid/multi)



Advanced Security

Advanced Security

Password policy management

DBA managed password profiles, compatible with Oracle profiles

Audit compliance

Track and analyze database activities and user connections

Virtual private databases

Fine grained access control limits user views

EDB/SQL protect

SQL firewall, screens queries for common attack profiles

Data redaction

Protect sensitive information for GDPR, PCI and HIPAA compliance

Code protection

Protects sensitive IP, algorithms or financial policies

Transparent Data Encryption (TDE)

- Transparent Data Encryption (TDE) is a feature of EDB Postgres Advanced Server and EDB Postgres Extended Server that prevents unauthorized viewing of data in operating system files on the database server and on backup storage
- Data encryption and decryption is managed by the database and does not require application changes or updated client drivers
- EDB Postgres Advanced Server and EDB Postgres Extended Server provide hooks to key management that is external to the database allowing for simple passphrase encrypt/decrypt or integration with enterprise key management solutions, with initial support for:
    - Amazon AWS Key Management Service (KMS)
    - Google Cloud - Cloud Kay Management Service
    - Microsoft Azure Key Vault
    - HashiCorp Vault (KMIP Secrets Engine and Transit Secrets Engine)
    - Thales CipherTrust Manager
- Data will be unintelligible for unauthorized users if stolen or misplaced

Main capabilities

Storage management

- Storage is the most critical component for a database
- Direct support for Persistent Volume Claims (PVC)
  - We deliberately do not use Statefulsets
- The PVC storing the PGDATA is central to CloudNativePG
  - Our motto is: “PGDATA is worth a 1000 pods”
- Storage agnostic
- Freedom of choice
  - Local storage
  - Network storage
- Automated generation of PVC
- Support for PVC templates
  - Storage classes

Scheduling Postgres instances with CloudNativePG

- Entirely declarative!
- Affinity section in the `Cluster` specification
  - pod affinity/anti-affinity
  - node selectors
  - tolerations against taints placed on nodes

Bootstrap

- Create a new cluster from scratch
  - “initdb”: named after the standard “initdb” process in PostgreSQL that initializes an instance
  - This method can be used to migrate another database (“import”) or upgrade it
    - Uses pg_dump and pg_restore with some intelligence we’ve added
- Create a new cluster from an existing one:
  - Directly (“pg_basebackup”), using physical streaming replication
  - Indirectly (“recovery”), from an object store
    - To the end of the WAL
      - Can be used to start independent replica clusters in continuous recovery
    - Using PITR

EDB

Leverage a New Way to Import an Existing Postgres Database to Kubernetes

Gabriele Bartolini

August 15, 2022

Are you thinking about moving your PostgreSQL databases to Kubernetes but wondering how to do it? What about importing databases from RDS for PostgreSQL or another database as a service?

Release 1.16 of the CloudNativePG open source operator introduces a new feature which makes it easier to import inside Kubernetes an existing Postgres database, from any location as long as it can be reached via the network.

The same feature also enables major version upgrades of PostgreSQL, as well as migrating to CloudNativePG any existing PostgreSQL database that you are already running inside Kubernetes with a different operator—or without one, using a pure statefulset based deployment.

This feature enhances the *initdb* bootstrap, by introducing a new subsection called import. Such a section is evaluated only if it is not empty, after the cluster has been initialized from scratch, and it defines which data to import from an existing Postgres instance. Such a Postgres instance can be running in a virtual machine, or on bare metal, or even as a service—like Amazon RDS. The important thing is that objects can be exported using logical backup from the source, and subsequently imported in the target instance.

Blog article

Rolling updates

- Update of a deployment with ~zero downtime
  - Standby servers are updated first
  - Then the primary:
    - supervised / unsupervised
    - switchover / restart
- When they are triggered:
  - Security update of Postgres images
  - Minor update of PostgreSQL
  - Configuration changes when restart is required
  - Update of the operator
    - Unless in-place upgrade is enabled

Backup and Recovery - Part 1

- Continuous physical backup on “backup object stores”
  - Scheduled and on-demand base backups
  - Continuous WAL archiving (including parallel)
  - From primary or a standby
  - Support for recovery window retention policies (e.g. 30 days)
- Recovery means creating a new cluster starting from a “recovery object store”
  - Then pull WAL files (including in parallel) and replay them
  - Full (End of the WAL) or PITR
- Both rely on Barman Cloud technology
  - AWS S3
  - Azure Storage compatible
  - Google Cloud Storage
  - MinIO

Backup and Recovery - Part 2

- WAL management
  - Object store

- Physical Base backups
  - Object store
  - Kubernetes level backup integration (Velero/OADP, Veem Kasten K10, generic interface)
  - Kubernetes Volume Snapshots

Kubernetes Volume Snapshot: major advantages

- Transparent support for:
  - Incremental backup and recovery at block level
  - Differential backup and recovery at block level
  - Based on copy on write
- Leverage the storage class to manage the snapshots, including:
  - Data mobility across network (availability zones, Kubernetes clusters, regions)
  - Relay files on a secondary location in a different region, or any subsequent one
  - Encryption
- Enhances Very Large Databases (VLDB) adoption

Backup & Recovery via Snapshots: some numbers

Let's now talk about some initial benchmarks I have performed on volume snapshots using 3 `r5.4xlarge` nodes on AWS EKS with the `gp3` storage class. I have defined 4 different database size categories (tiny, small, medium, and large), as follows:

Cluster name	Database size	pgbench init scale	PGDATA volume size	WAL volume size	pgbench init duration
tiny	4.5 GB	300	8 GB	1 GB	67s
small	44 GB	3,000	80 GB	10 GB	10m 50s
medium	438 GB	3,0000	800 GB	100 GB	3h 15m 34s
large	4,381 GB	300,000	8,000 GB	200 GB	32h 47m 47s

The table below shows the results of both backup and recovery for each of them.

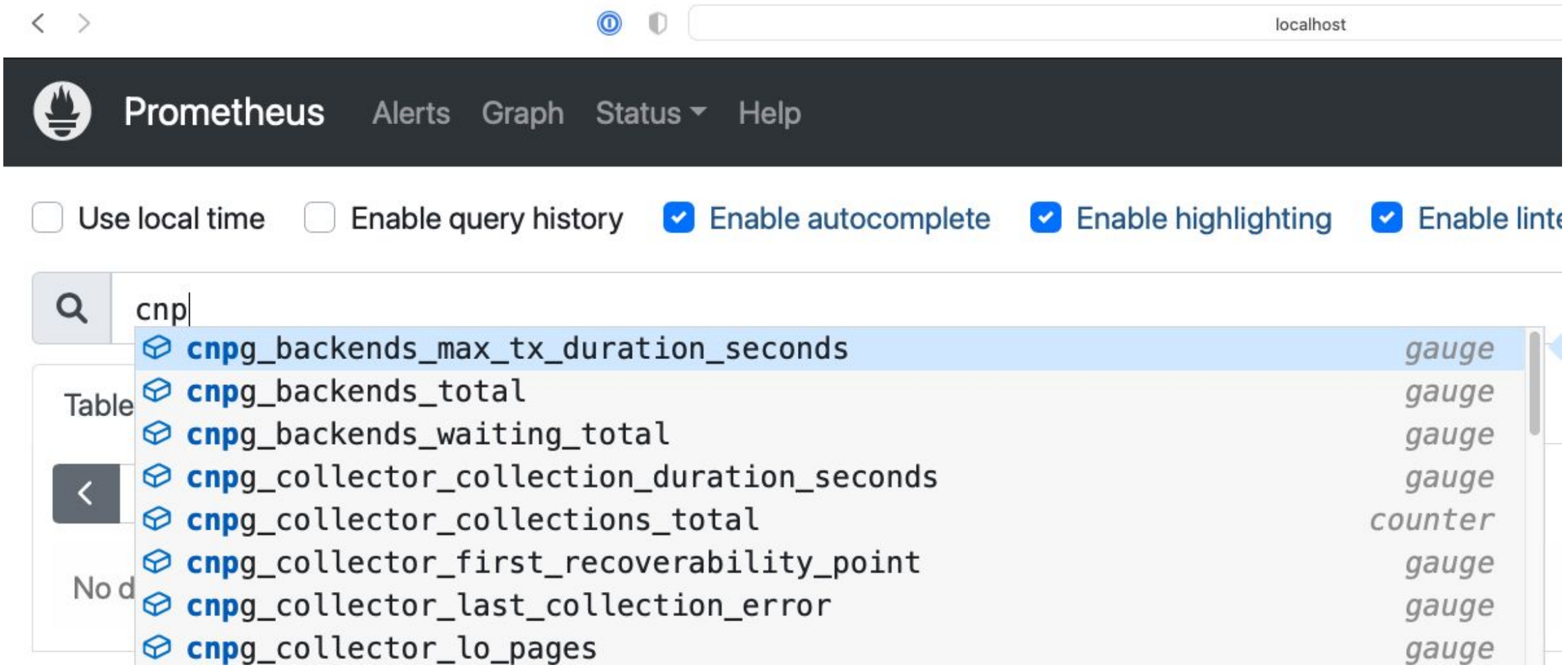
Cluster name	1st backup duration	2nd backup duration after 1hr of pgbench	Full recovery time
tiny	2m 43s	4m 16s	31s
small	20m 38s	16m 45s	27s
medium	2h 42m	2h 34m	48s
large	3h 54m 6s	2h 3s	2m 2s

https://www.enterprisedb.com/postgresql-disaster-recovery-with-kubernetes-volume-snapshots-using-cloudnativepg

Native Prometheus exporter for monitoring

- Built-in metrics at the operator level
- Built-in metrics at the Postgres instance level
  - Customizable metrics at the Postgres instance level
    - Via ConfigMap(s) and/or Secret(s)
    - Syntax compatible with the PostgreSQL Prometheus Exporter
    - Auto-discovery of databases
  - Queries are:
    - transactionally atomic and read-only
    - executed with the pg_monitor role
    - executed with application_name set to cnp_metrics_exporter
- Support for pg_stat_statements and auto_explain

Prometheus



Grafana Dashboard

Logging

- All components directly log to standard output in JSON format
- Each entry has the following structure:
  - level: log level (info, notice, ...)
  - ts: the timestamp (epoch with microseconds)
  - logger: the type of the record (e.g. postgres or pg_controldata)
  - msg: the type of the record (e.g. postgres or pg_controldata)
  - record: the actual record (with structure that varies depending on the msg type)
- Seamless integration with many log management stacks in Kubernetes
- Support for PGAudit
  - EDB Audit as well for EDB Postgres for Kubernetes

The “cnpg” plugin for kubectl

- The official CLI for CloudNativePG
  - Available also as RPM or Deb package
- Extends the ‘kubectl’ command:
  - Customize the installation of the operator
  - Status of a cluster
  - Perform a manual switchover (promote a standby) or a restart of a node
  - Issue TLS certificates for client authentication
  - Declare start and stop of a Kubernetes node maintenance
  - Destroy a cluster and all its PVC
  - Fence a cluster or a set of the instances
  - Hibernate a cluster
  - Generate jobs for benchmarking via pgbench and fio
  - Issue a new backup
  - Run pgadmin
  - Run pgbench

```
Name:               cluster-example
Namespace:          default
System ID:          7100921006673293335
PostgreSQL Image:   ghcr.io/cloudnative-pg/postgresql:14.3
Primary instance:   cluster-example-2
Status:             Cluster in healthy state
Instances:          3
Ready instances:    3
Current Write LSN:  0/C000060 (Timeline: 4 - WAL File: 00000004000000000000000C)

Certificates Status
Certificate Name             Expiration Date               Days Left Until Expiration
----------------             ---------------               --------------------------
cluster-example-replication  2022-08-21 13:15:00 +0000 UTC  89.95
cluster-example-server       2022-08-21 13:15:00 +0000 UTC  89.95
cluster-example-ca           2022-08-21 13:15:00 +0000 UTC  89.95

Continuous Backup status
First Point of Recoverability:  2022-05-23T13:37:08Z
Working WAL archiving:          OK
WALs waiting to be archived:    0
Last Archived WAL:              00000004000000000000000B   @   2022-05-23T13:42:09.37537Z
Last Failed WAL:                -

Streaming Replication status
Name              Sent LSN   Write LSN  Flush LSN  Replay LSN  Write Lag  Flush Lag  Replay Lag  State      Sync State  Sync Priority
----              --------   ---------  ---------  ----------  ---------  ---------  ----------  -----      ----------  -------------
cluster-example-3  0/C000060  0/C000060  0/C000060  0/C000060   00:00:00   00:00:00   00:00:00    streaming  async       0
cluster-example-1  0/C000060  0/C000060  0/C000060  0/C000060   00:00:00   00:00:00   00:00:00    streaming  async       0

Instances status
Name              Database Size  Current LSN  Replication role  Status  QoS         Manager Version
----              -------------  -----------  ----------------  ------  ---         ---------------
cluster-example-3  33 MB          0/C000060    Standby (async)   OK      BestEffort  1.15.0
cluster-example-2  33 MB          0/C000060    Primary           OK      BestEffort  1.15.0
cluster-example-1  33 MB          0/C000060    Standby (async)   OK      BestEffort  1.15.0
```

Connection pooling with PgBouncer

- Managed by the “Pooler” Custom Resource Definition
- Directly mapped to a service of a given Postgres cluster
- Deploys multiple instances of PgBouncer for High Availability
- Supports Pod templates, very important for Pod scheduling rules
- Transparent support for password authentication
- Connects to PostgreSQL via a standard user through a TLS certificate
- Supports configuration for most of PgBouncer options
- Automated integration with Prometheus
- JSON log in standard output

Demo

CloudNativePG Website	cloudnative-pg.io
GitHub project	github.com/cloudnative-pg/cloudnative-pg
PostgreSQL	postgresql.org
EnterpriseDB	enterprisedb.com

Scan the QR-Code & pre-register for our 2025

Postgres on Kubernetes “Hands-On” Workshop

@CloudNativePG

@EDBPostgres

Thank you