



# Postgres on Kubernetes Workshop Copenhagen

27 May | Copenhagen, Denmark

# Agenda

| Start | End   | Session  |
|-------|-------|--|
| 13:00 | 13:30 | Registration & Welcome   |
| 13:30 | 13:45 | Red Hat OpenShift & EDB Partnership (Red Hat - Michael Bang)                   |
| 13:45 | 14:00 | Introduction to Postgres marketplace and EDB (EDB - Søren Boss Jacobsen)       |
| 14:00 | 14:30 | CNPG Operator Reference Architecture and Functionalities (EDB - Sergio Romera) |
| 14:30 | 16:30 | Interactive session & demo (EDB - Michael Willer & Niels van Noort)            |
| 16:30 | 17:00 | What more?   |
| 17:00 | 18:00 | Drinks and pizza   |



# CREW for today's workshop

Red Hat team



Michael Bang

EDB team



Søren Boss Jacobsen



Sergio Romera



Michael Willer



Niels van Noort



# EDB Partnership with EDB

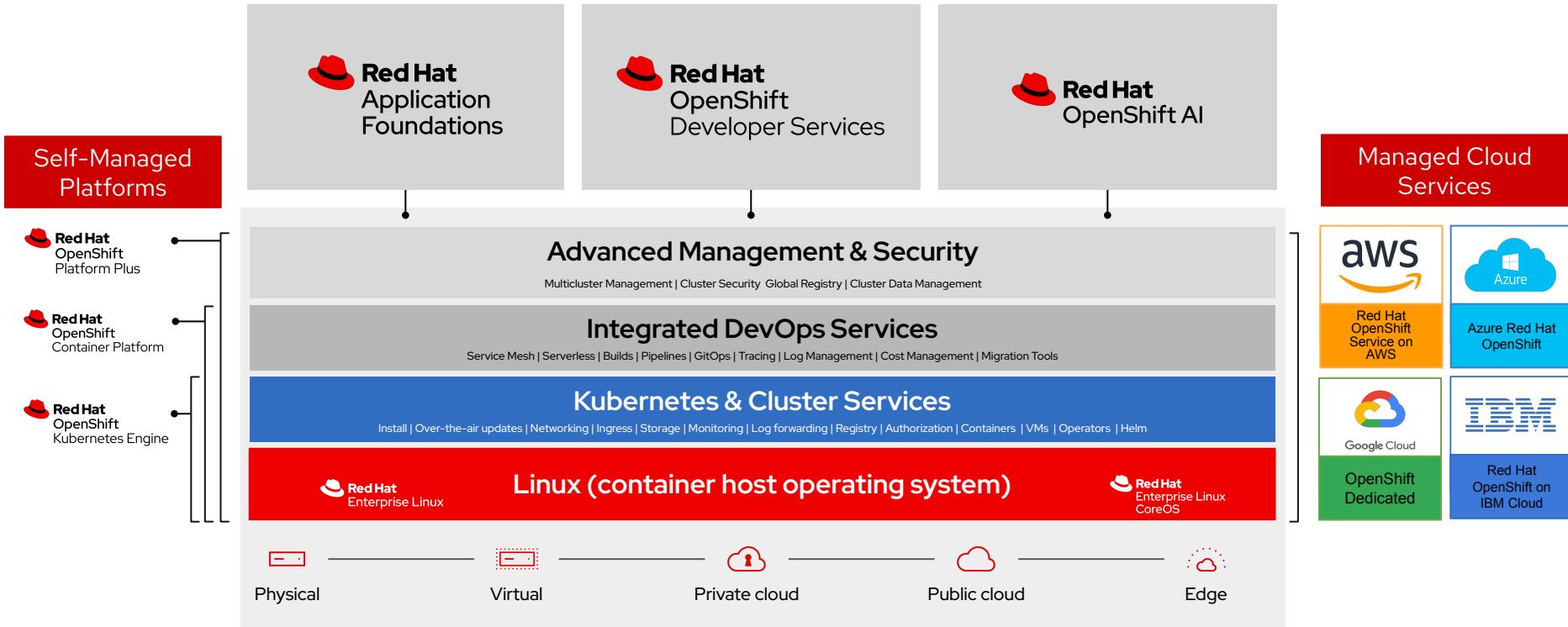


# Red Hat OpenShift with EDB

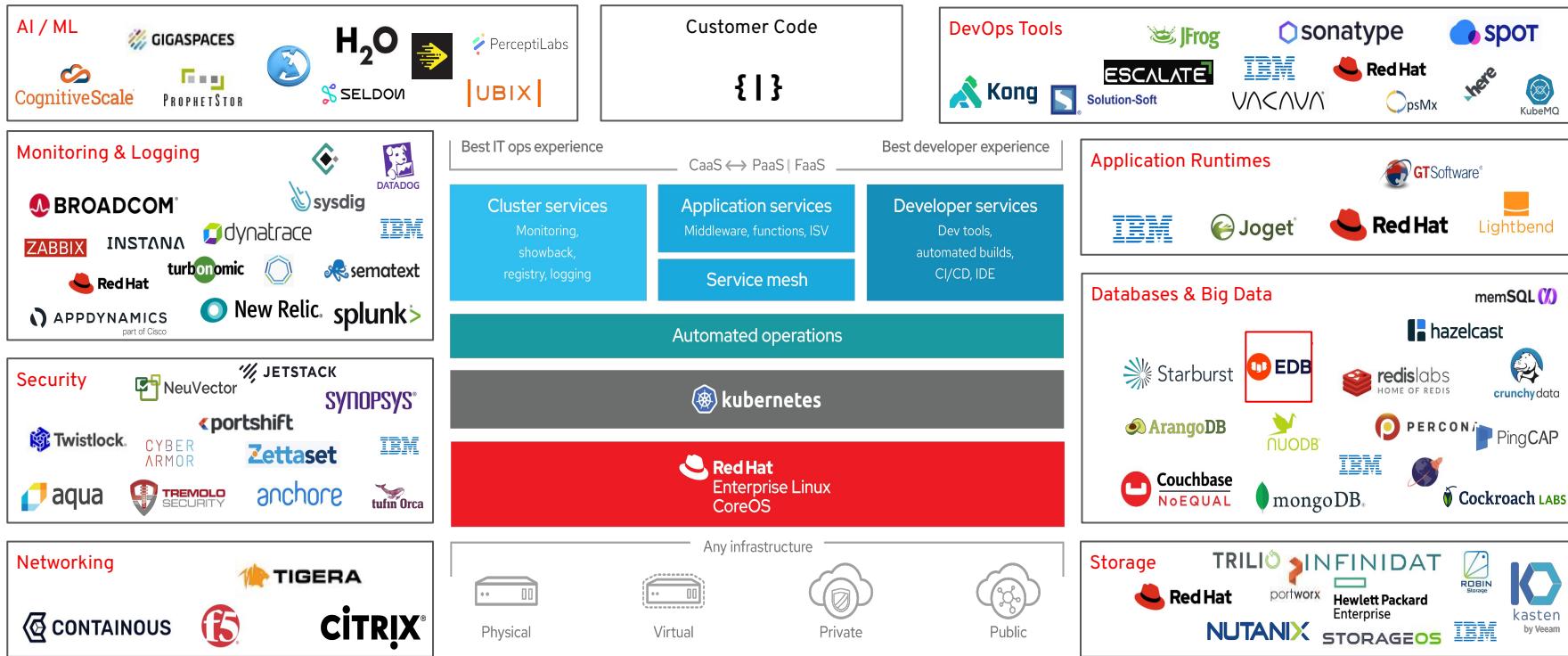
Michael Bang  
Associate Principal Solution  
Architect



# Hybrid Cloud Application Platform



# Red Hat open hybrid cloud platform with ISV ecosystem



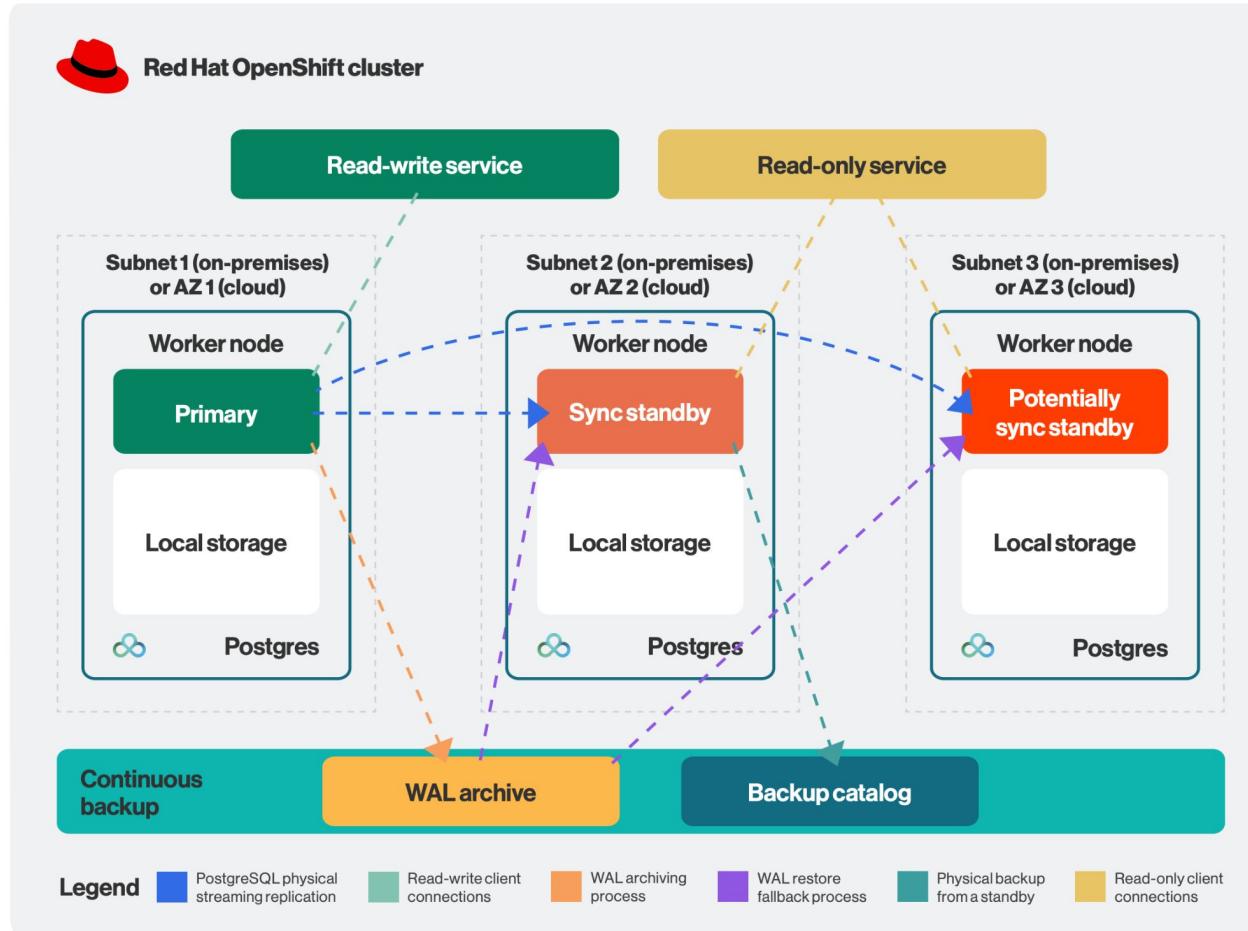
# Why Red Hat OpenShift for EDB: operator certification

The screenshot shows the Red Hat Ecosystem Catalog interface. At the top, there's a navigation bar with the Red Hat logo, 'Red Hat Ecosystem Catalog', and links for 'Solutions', 'Products', 'Artifacts', and 'Partners'. A search bar is also present. Below the navigation, the URL 'Home > Software > All software results > Containerized applications' is visible. The main content area features the 'EDB Postgres for Kubernetes' operator. It includes the EDB logo, a 'Certified' badge, and a brief description: 'PostgreSQL Operator for mission critical databases in Openshift Container Platform'. Below this, there are tabs for 'Overview', 'Resources', 'Certifications' (which is highlighted with a red border), 'Deploy & use', and 'FAQs'. Under the 'Certifications' tab, there's a section titled 'Learn about Red Hat Certification and Partner Validation'. The 'Certified components' section lists the 'EDB Postgres for Kubernetes (formerly Cloud Native PostgreSQL)' operator, showing its container image details: 'enterpriseedb/cloud-native-postgresql', version '1.25.1', and architecture 'amd64'. It also indicates it was published last month.

EDB Postgres for Kubernetes is a certified Level 5 Operator for Red Hat OpenShift

- ▶ This is designed to streamline Day 2 operations of PostgreSQL databases
- ▶ Enhanced Database Management
- ▶ Supports point-in-time recovery (PITR)
- ▶ Ensures robust data protection and recovery options
- ▶ Integration with business continuity solutions such as Red Hat OpenShift API for Data Protection (OADP) and Veeam Kasten, Trilio, Portworx Backup, IBM Fusion, and others

# Why Red Hat OpenShift for EDB : reference architecture



# EDB on OpenShift use cases

- ▶ Cloud-Native Database Deployment
- ▶ Database as a Service (DBaaS)
- ▶ High Availability and Disaster Recovery (HA & DR)
- ▶ DevOps and Continuous Integration/Continuous Deployment (CI/CD)
- ▶ Microservices and Application Modernization
- ▶ Move from VMWare to OpenShift
- ▶ Data Security and Compliance (using **TDE** and Advanced Security provided by EPAS)
- ▶ Hybrid and Multi-Cloud Deployments
- ▶ Multi-Tenant Applications (isolation)

# Euro Information

## Company profile

Euro-Information is the fintech company of the Crédit Mutuel group. Euro-Information manages the IT systems of 16 federations of Crédit Mutuel as well as those of CIC and of all the financial, insurance, property, consumer credit, private banking, financing, telephony and technological subsidiaries.

## Problem

- Fast database deployment
- Adopt a supported and secure Open Source platform
- Onprem DBaaS
- Align to in-house RDBMS standardization

## Solution

- Use Postgres capabilities to build and maintain local applications
- Use Red Hat OpenShift platform to accelerate the provisioning of databases and applications

## Results

- Applications running with PostgreSQL databases in a centralized environment
- Massive reduction of TCO of database service operations



- Red Hat OpenShift
- EDB Postgres for Kubernetes
- PostgreSQL
- EPAS

- EDB considerably reduces IT costs associated with database maintenance.
- 280 cores: Enterprise Plan + Production Support

## Summary

Use Case

On prem DBaaS ([in Production](#))

Workload

Transactional

Application Name

All internal Postgres applications

EDB Tools of Interest

PostgreSQL and EDB Postgres for Kubernetes



# La Poste

## Company profile

La Poste is a postal service company in France, operating in Metropolitan France, the five French overseas departments and regions and the overseas collectivity of Saint Pierre and Miquelon. Under bilateral agreements, La Poste also has responsibility for mail services in Monaco through La Poste Monaco and in Andorra alongside the Spanish company Correos.

## Problem

- Provide a database HA solution for Ansible Automation Platform (AAP)
- Database must be in HA and DR

## Solution

- Use EDB Postgres for Kubernetes to provide a HA and DR solution for PostgreSQL databases
- Deploy in 2 OpenShift clusters our operator

## Results

- La Poste developer can use their internal 'La Post Service Portal' to provision more than 64 backends.
- Reduce risk deploying EDB solutions.



- Red Hat OpenShift
- EDB Postgres for Kubernetes
- PostgreSQL

- EDB considerably reduces IT costs associated with database maintenance.
- 12 Cores: Standard Plan + Premium Support

## Summary

Use Case

On prem DBaaS with HA and DR  
[\(in Production\)](#)

Workload

Transactional

Application Name

Portail XaaS

EDB Tools of Interest

PostgreSQL and EDB Postgres for Kubernetes

# Airbus

## Company profile

Airbus SE is a European aerospace corporation. The company's primary business is the design and manufacturing of commercial aircraft but it also has separate defence and space and helicopter divisions.

## Problem

- Flexibility
- Cost reduction
- New managed service in OpenShift

## Solution

- EDB Postgres for Kubernetes with EPAS. Depending of the applications needs, EPAS and/or TDE will be activated



- Red Hat OpenShift
- EDB Postgres for Kubernetes
- EPAS
- TDE



- Improve database deployment speed
- Reduce DB support
- Cost reduction

## Summary

|                       |   |
|-----------------------|---|
| Use Case              | On prem DBaaS ( <a href="#">Production</a> )                |
| Workload              | Transactional   |
| Application Name      | All VMWare PostgreSQL databases                             |
| EDB Tools of Interest | EDB Postgres Advanced Server with Oracle and TDE (optional) |

## Results

- POC done
- Decision is taken
- Number of cores not yet communicated

# Banque de France

## Company profile

The Banque de France is France's central bank. A two-hundred-year-old institution, privately-owned when it was founded on January 18, 1800 under the Consulate by General Bonaparte, it became state-owned on January 1, 1946 when it was nationalized by General de Gaulle.

## Problem

- Fast database deployment
- Provide containerized Postgres DBaaS

## Solution

- Use OpenShift to provide this service with the operator
- Fast deployment and with Open Source database

## Results

- OpenShift based PostgreSQL cluster deployments expand the internal offering alongside traditional VM based database cluster deployments



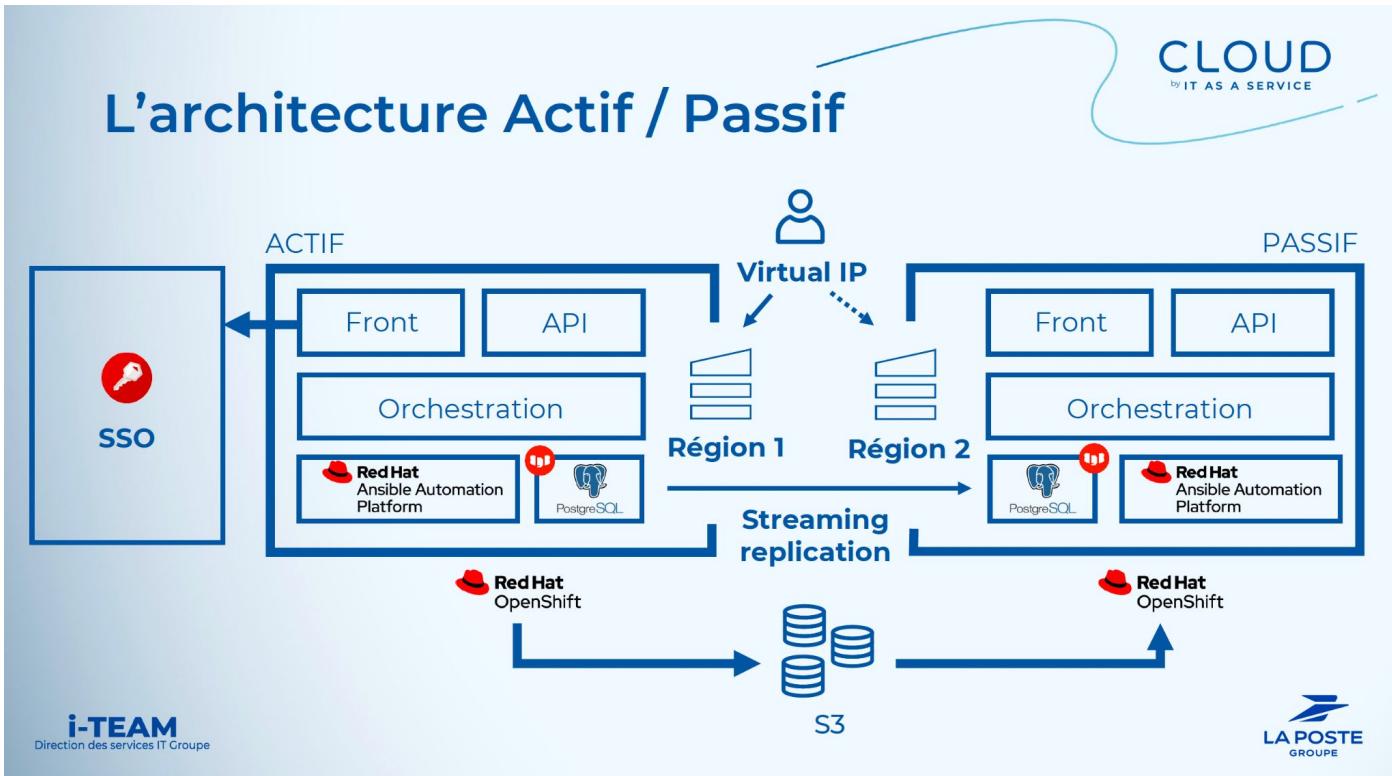
- Red Hat OpenShift
- CloudNativePG
- PostgreSQL

- 100 cores
- Subscription plan:
  - Community360 plan + Production Support

## Summary

|                       |  |
|-----------------------|--|
| Use Case              | OnPrem DBaaS ( <a href="#">in production</a> ) |
| Workload              | Transactional                                  |
| Application Name      | Multiple applications                          |
| EDB Tools of Interest | PostgreSQL, CloudNativePG                      |

# La Poste Architecture



# Thank you

Red Hat is the world's leading provider of enterprise open source software solutions.

Award-winning support, training, and consulting services make

Red Hat a trusted adviser to the Fortune 500.



[linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)



[youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)



[facebook.com/redhatinc](https://www.facebook.com/redhatinc)



[twitter.com/RedHat](https://twitter.com/RedHat)



# Introduction to Postgres marketplace and EDB



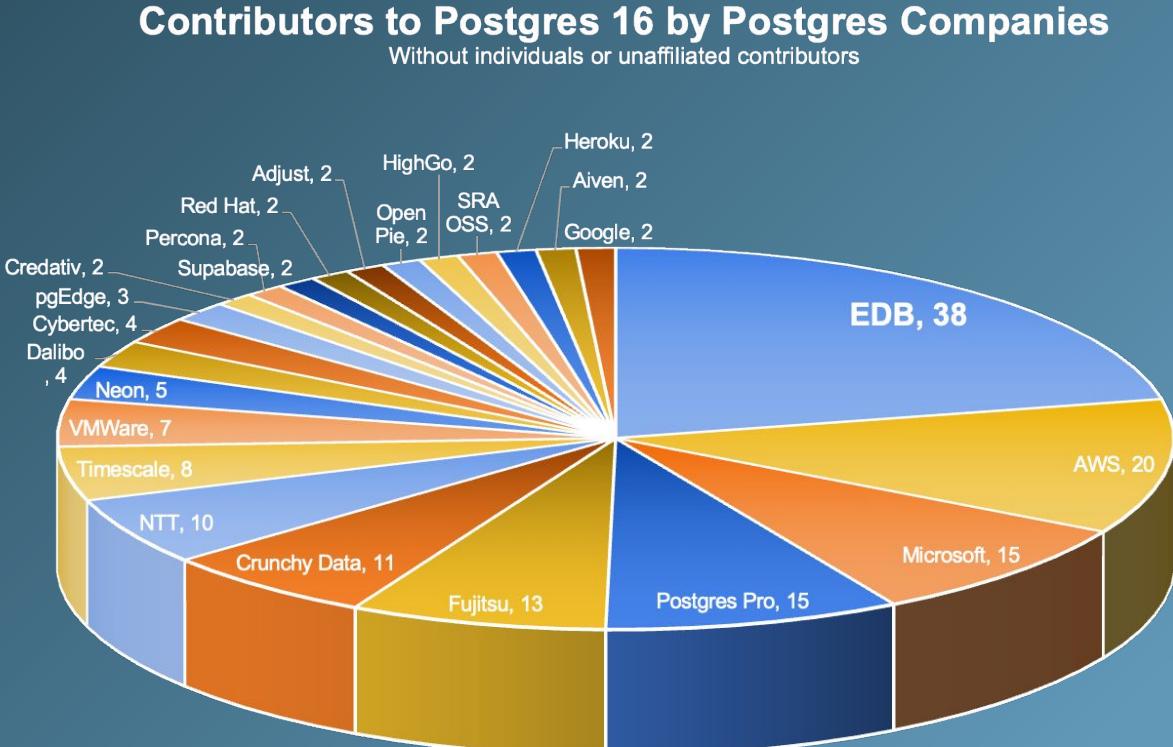
# 20+ years of Postgres innovation & adoption

- Number one contributor to Postgres, fastest-growing and most loved Database in the world
  - 2 Core Team members, 7 Committers, 9 Major Contributors, 10 Contributors, #1 site for desktop downloads
- Over 700 employees in more than 30 countries
- EDB Postgres AI
  - The industry's first platform that can be deployed as cloud, software or physical appliance
  - Secure, compliant and enterprise grade performance guaranteed



# Large Developer Community

- 407 + contributors to Postgres 16
- 111 companies actively contributing to Postgres 16



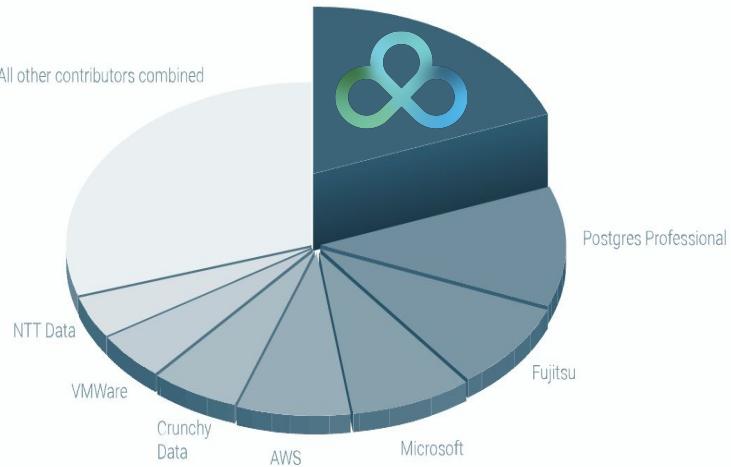
## 20+ years of innovation

- 760+ employees, 300 dedicated to Postgres
- 79 countries

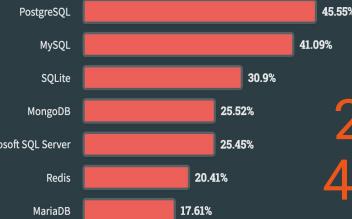
## #1 contributor to Postgres

- Present in the Core Team
- 7 Committers
- 9 Major Contributors

30% + of Postgres Code Contributed in 2024



Postgres has won the database race



2023:  
45.5%



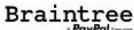
2024:  
48.7%

Stack Overflow Survey 2023/2024

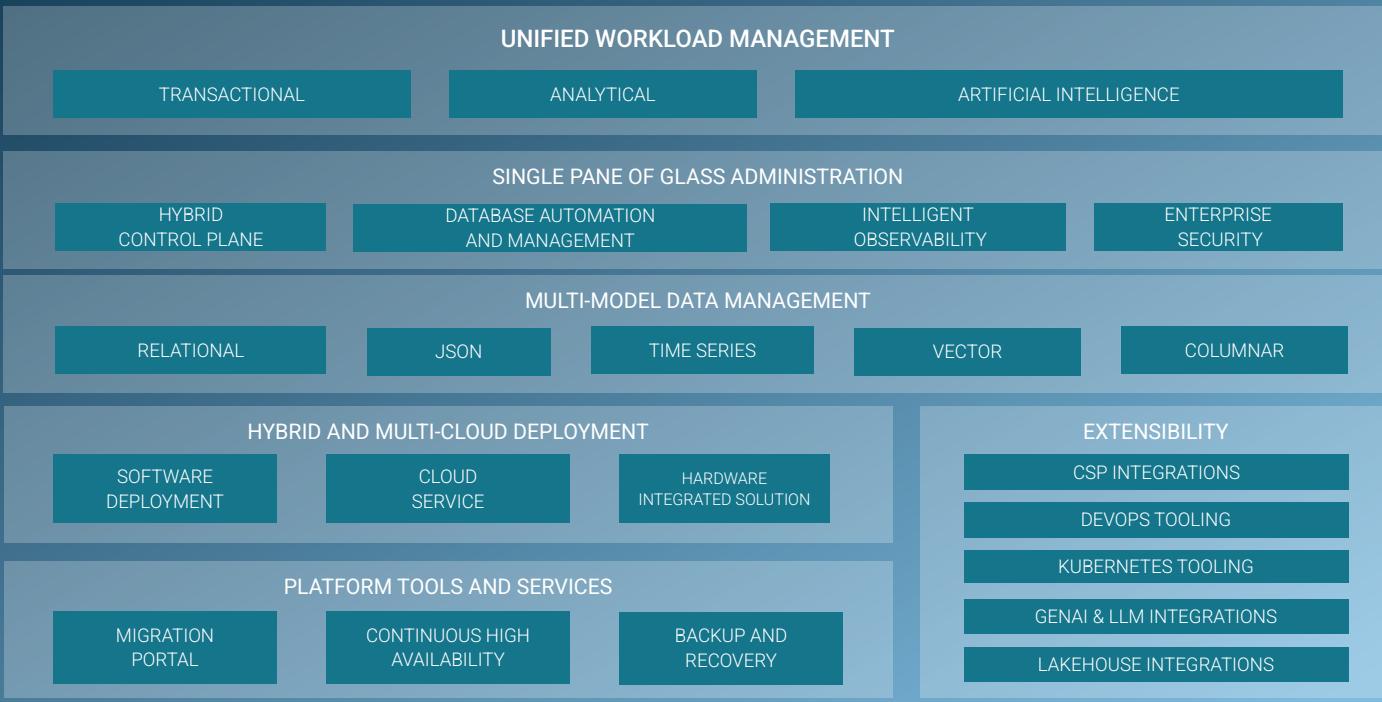
BANKING FINANCIAL

TECHNOLOGY

TELCO



# EDB POSTGRES AI PLATFORM





# CNPG Operator: Reference Architecture and functionalities

# Kubernetes timeline

- 2014, June: Google open sources Kubernetes
- 2015, July: Version 1.0 is released
- 2015, July: Google and Linux Foundation start the CNCF
- 2016, November: The operator pattern is introduced in a blog post
- 2018, August: The Community takes the lead
- 2019, April: Version 1.14 introduces **Local Persistent Volumes**
- 2019, August: EDB team starts the Kubernetes initiative
- 2020, June: we publish this blog about benchmarking local PVs on bare metal
- 2020, June: Data on Kubernetes Community founded
- 2021, February: EDB Cloud Native Postgres (CNP) 1.0 released
- 2022, May: **EDB donates CNP** and open sources it under CloudNativePG
- 2025, January: CloudNativePG was recognized as an official **#CNCF** project



# Enabling the same PostgreSQL everywhere

From self-managed to fully managed DBaaS in the Cloud

- Same applications
- Faster innovation
- Performance and scalability
- Stability, security and control
- Seamless integration
- Obsolescence



Private

Hybrid

Multi-cloud

Public



Bare Metal



Virtual Machines



Containers



# A kubernetes operator for Postgres



Kubernetes adoption is rising and it is already the de facto **standard** orchestration tool



PostgreSQL clusters “management the kubernetes way” enables many cloud native usage patterns, e.g. spinning up, disposable clusters during tests, one cluster per microservice and one database per cluster

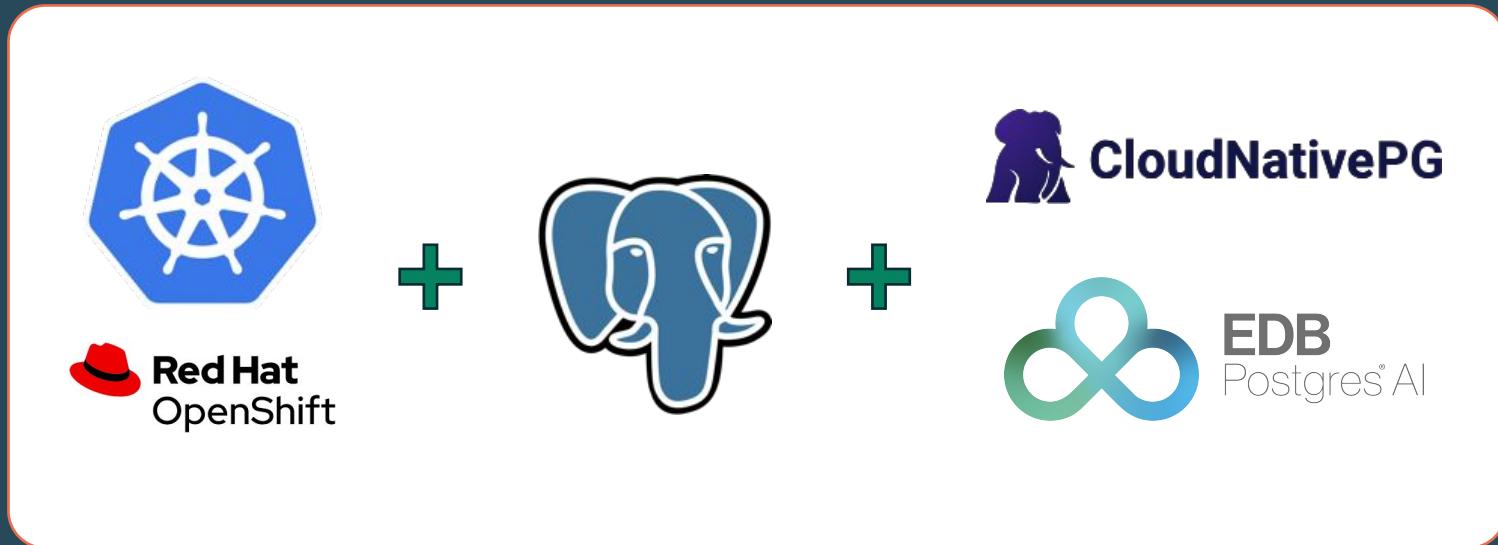


CNP tries to encode years of experience managing PostgreSQL clusters into **an Operator which should automate all the known tasks** a user could be willing to do

Our PostgreSQL operator must simulate the work of a DBA



# Win Technology



## Autopilot

It automates the steps that a human operator would do to deploy and to manage a Postgres database inside Kubernetes, including automated failover.



## Security

A grayscale photograph of a security guard from behind. The guard is wearing a light-colored jacket with the word "SECURITY" printed in large, bold, white capital letters on the back. He is holding a dark walkie-talkie up to his ear with his right hand. The background is a plain, light-colored wall.

CloudNativePG is secured by default.





It doesn't rely on statefulsets and uses its own way to manage persistent volume claims where the PGDATA is stored.

## Data persistence



## Designed for Kubernetes

It's entirely declarative, and directly integrates with the Kubernetes API server to update the state of the cluster — for this reason, it does not require an external failover management tool.



# Features

| Deployment          | Administration           | Backup & Recovery | Monitoring                  | Security     | High Availability      |
|---------------------|--------------------------|-------------------|-----------------------------|--------------|------------------------|
| Kubernetes operator | Single node              | Backup            | Prometheus                  | TDE          | Switchover             |
| Kubernetes plugin   | Cluster (Multi node)     | Recovery          | Grafana dashboards          | Certificates | Failover               |
| EDB Postgres (EPAS) | PostgreSQL configuration | PITR              | Postgres Enterprise Manager | Data masking | Scale out / scale down |
| PostGIS             | Logging                  | Volume Snapshots  |                             |              | Minor rolling updates  |
|                     | Pooling                  |                   |                             |              | *Major updates         |



# Decision-making for choosing the deployment platform



# When to choose Kubernetes over VMs?

**01** | Cloud Native Applications that already run in Kubernetes

**02** | Scalable, replicated databases

**03** | Applications requiring automated failover and self-healing

**04** | Teams skilled in Kubernetes who want a unified infrastructure



# Advantage of deploying Postgres Databases in Kubernetes

## Automation & Orchestration

- 01 |
- Self-healing
  - Automated scaling
  - Rolling updates

## Self-healing

- 02 |
- Best resource utilization
  - Dynamic Resource allocation

## Rolling updates

- 03 |
- Cloud-agnostic
  - Consistent deployment

## Service discovery & networking

- 04 |
- Built-it service discovery
  - Load Balancing

## Automated backups and disaster recovery

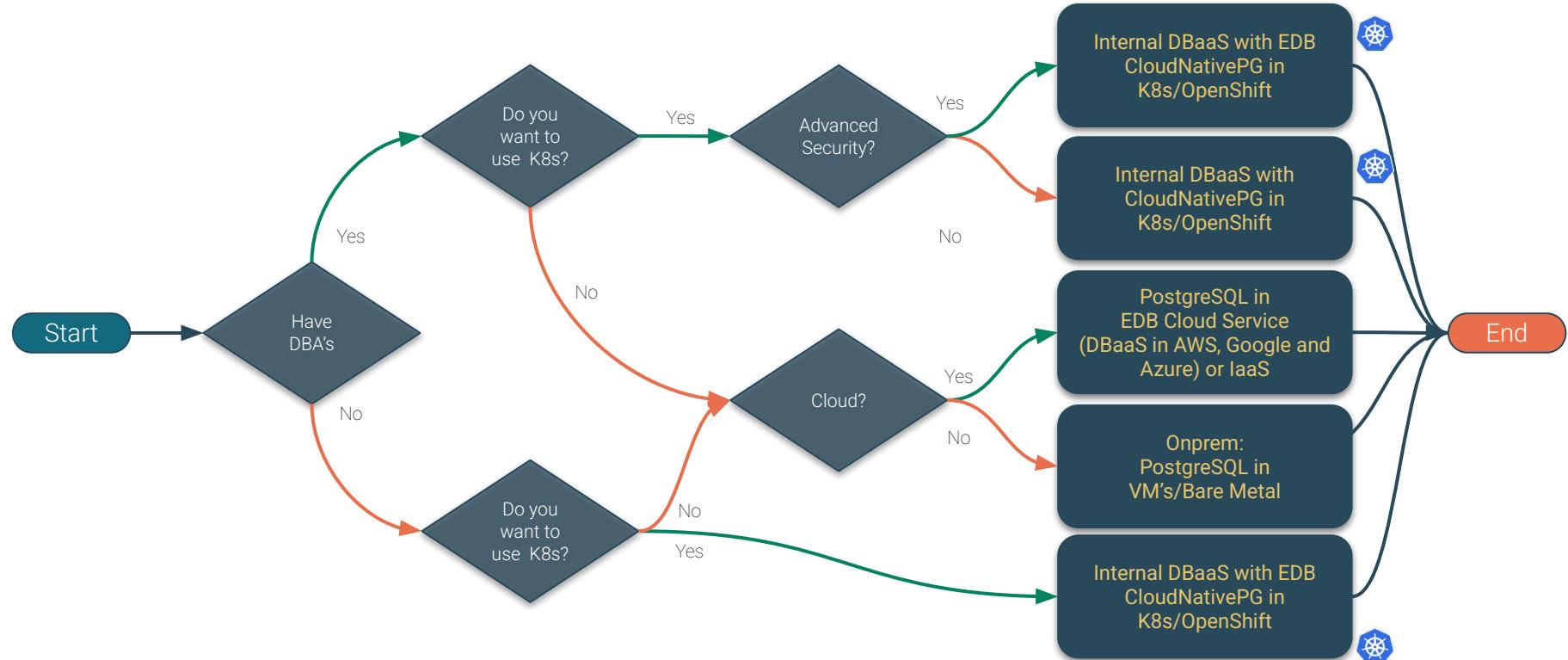
- 05 |
- Automated backups
  - Multi-region failover

## Security & access control

- 06 |
- RBAC
  - Secret management



# Decision-making for choosing the deployment platform

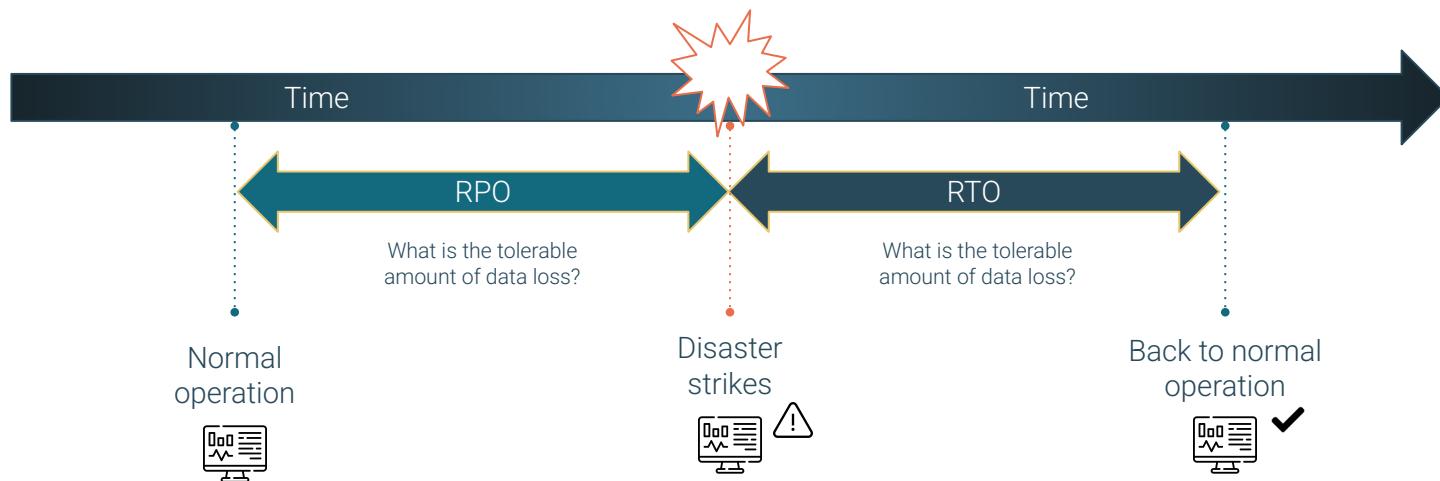


# Architectures



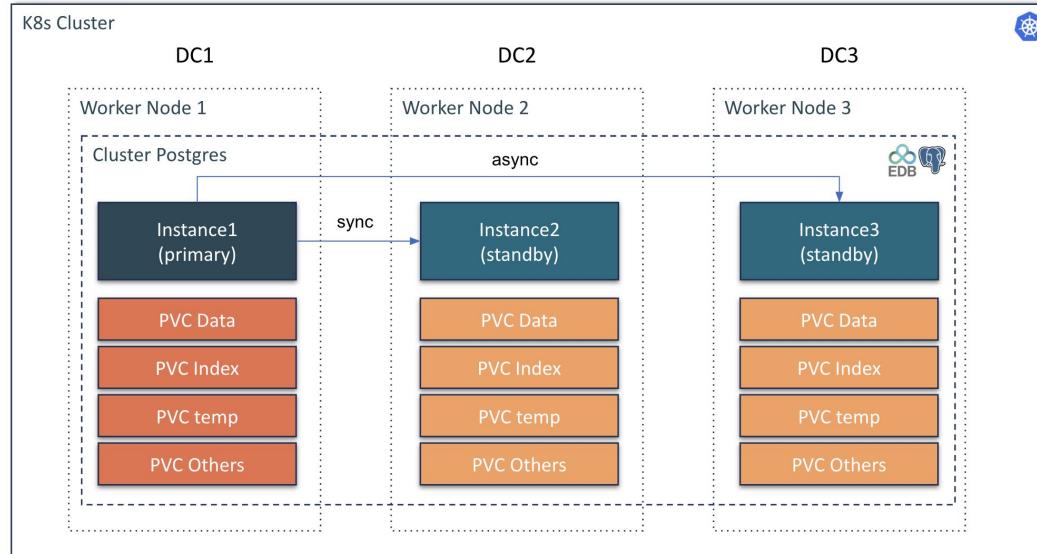
# Concepts

- Recovery Point Objective (**RPO**) and Recovery Time Objective (**RTO**) are key concepts in disaster recovery and business continuity planning, particularly related to data loss and system downtime.



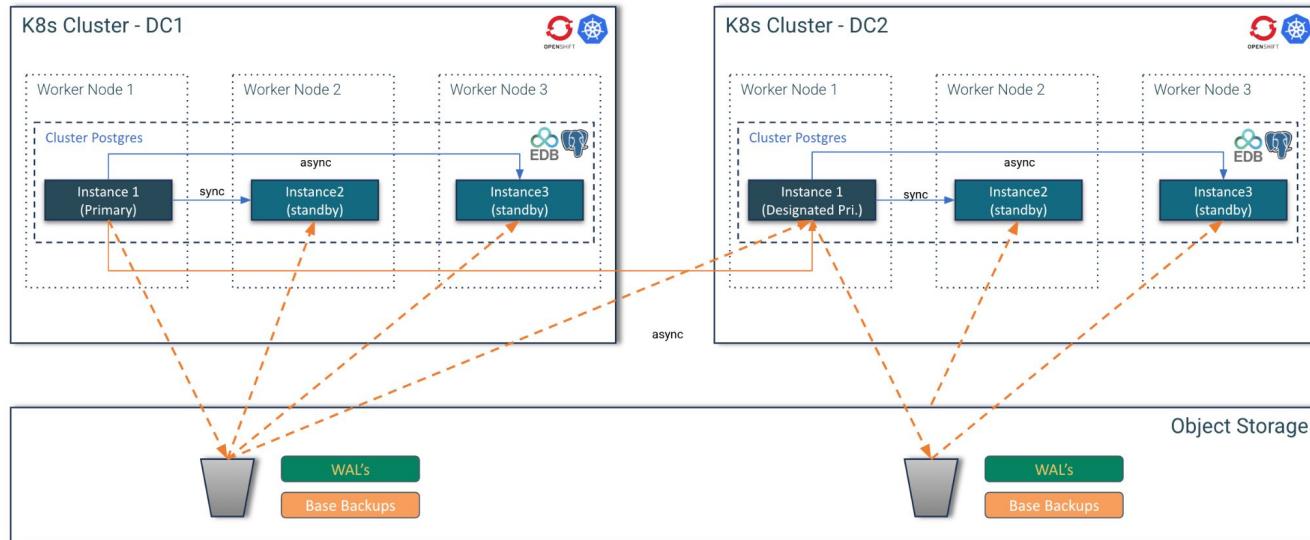
# Red Hat Recommendation

Red Hat recommend stretched clusters ONLY when latencies don't exceed 5 milliseconds (ms) round-trip time (RTT) between the nodes in different locations, with a maximum RTT of 10 ms.

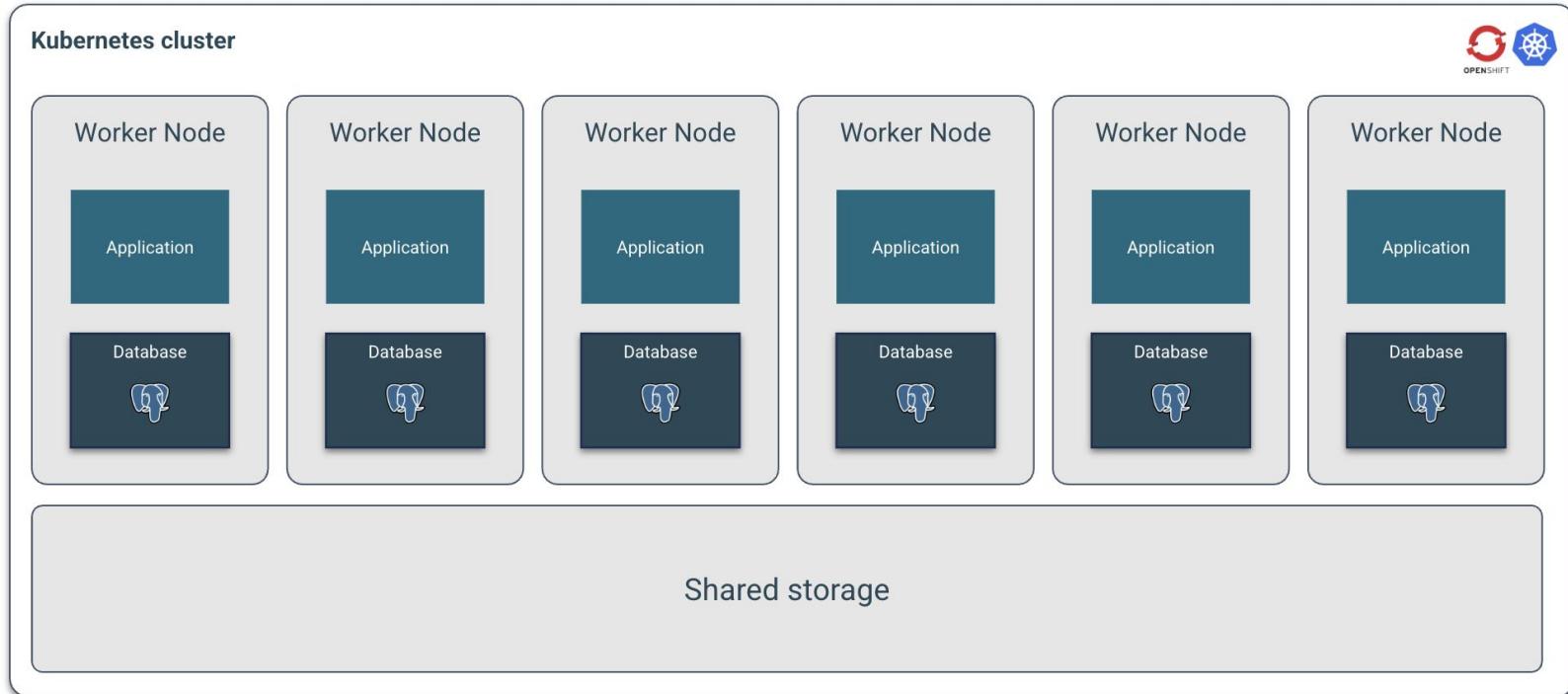


# Two separate single data center Kubernetes clusters

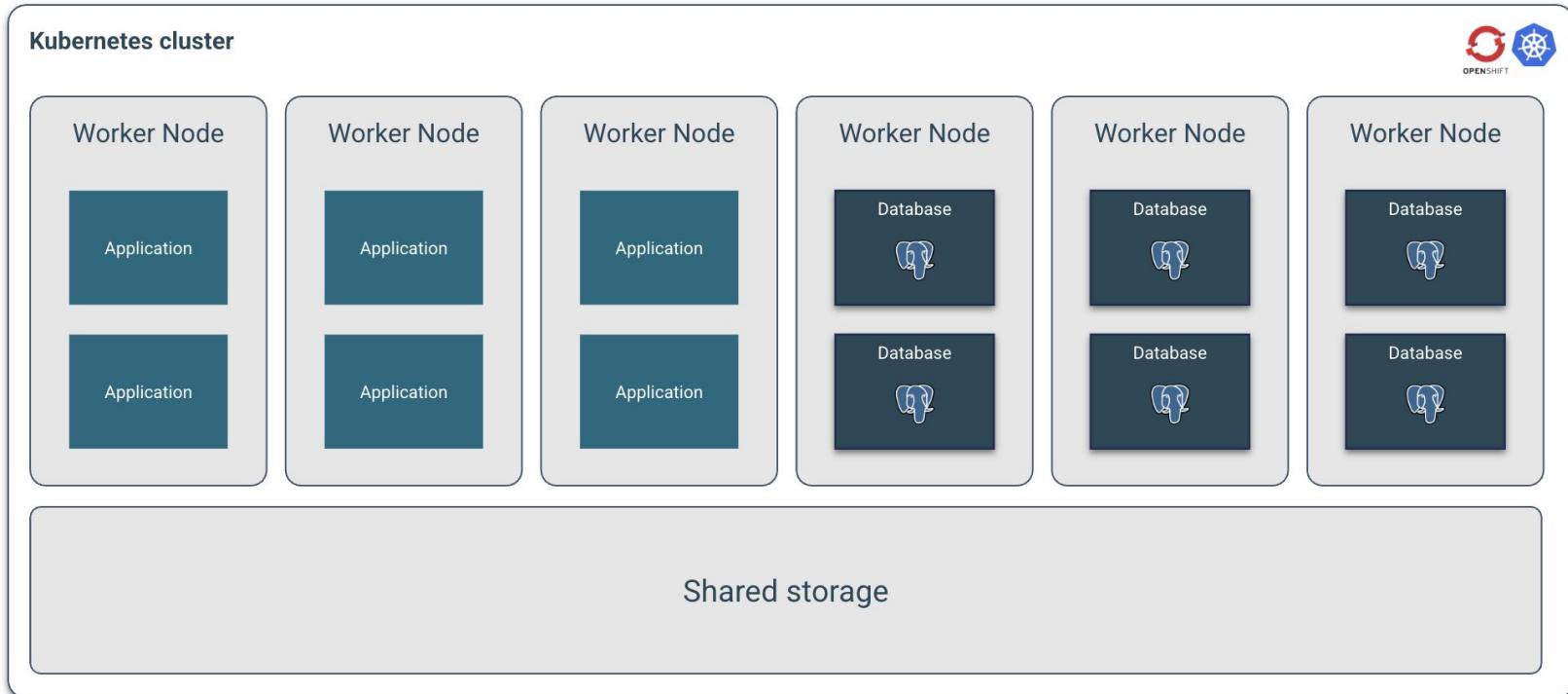
In case you cannot go beyond two data centers and you end up with two separate Kubernetes clusters, don't despair.



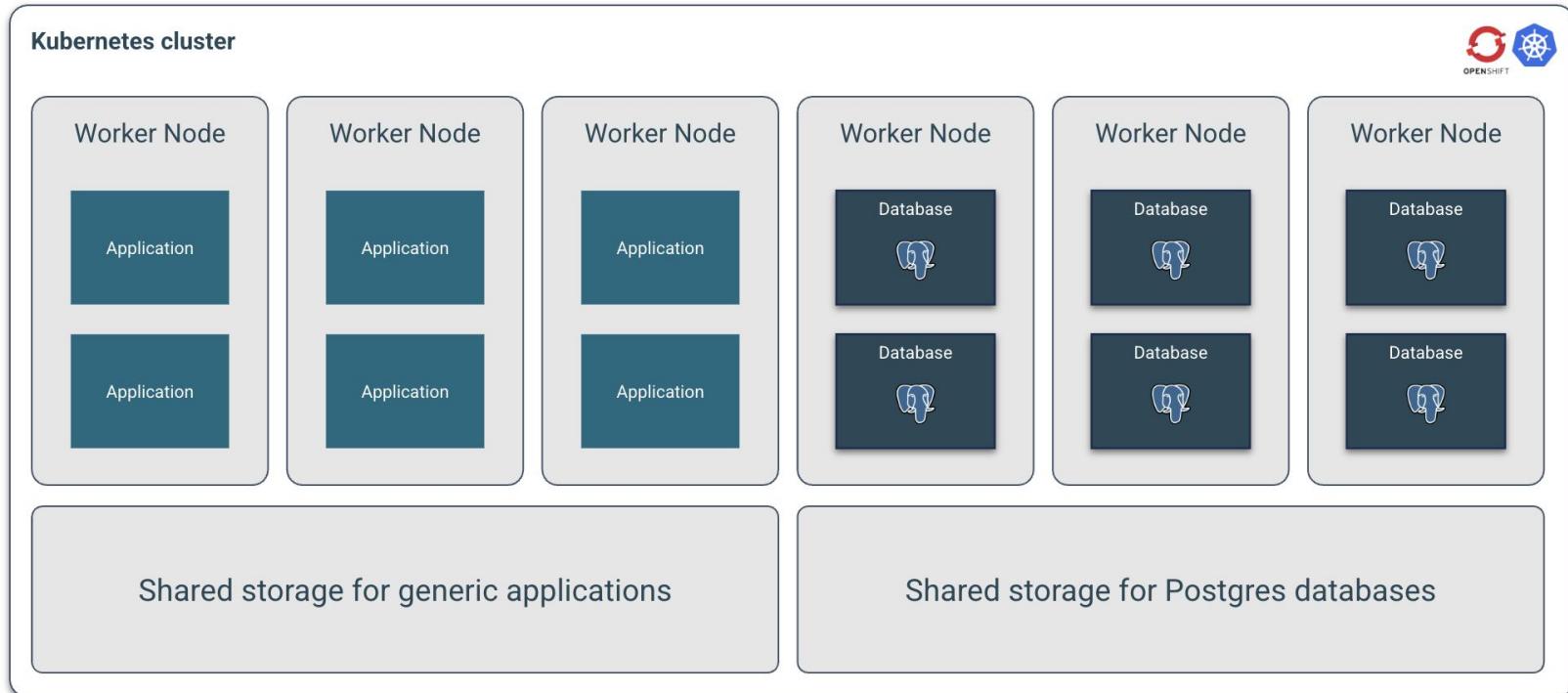
# Shared workload, shared storage



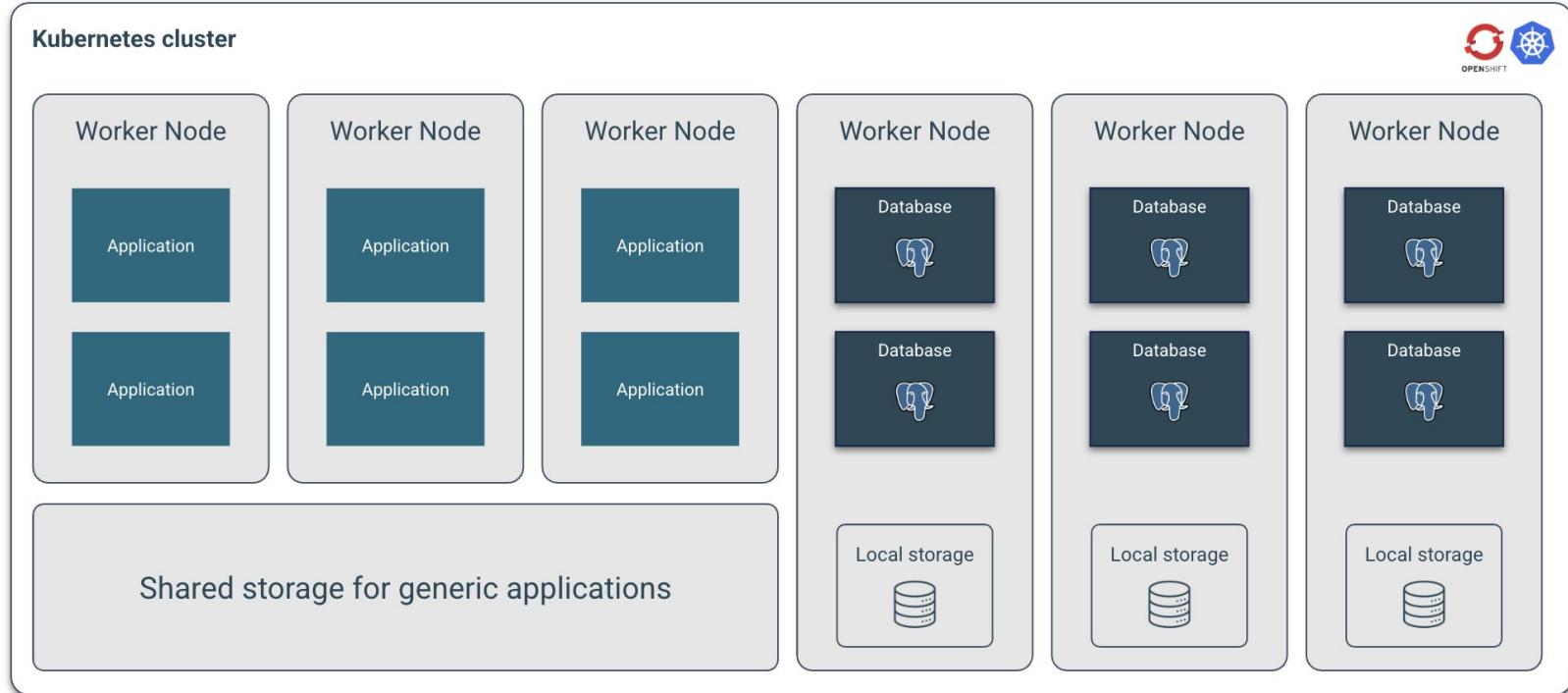
# Shared workload, shared storage



# Shared workload, shared storage



# Shared workloads, local storage



# Recommended architectures

<https://www.cncf.io/blog/2023/09/29/recommended-architectures-for-postgresql-in-kubernetes/>



## Recommended architectures for PostgreSQL in Kubernetes

By Gabriele Bartolini

September 29, 2023

Member post by Gabriele Bartolini, VP of Cloud Native at EDB

"You can run databases on Kubernetes because it's fundamentally the same as running a database on a VM", [tweeted Kelsey Hightower just a few months ago](#). Quite the opposite from what the former Google engineer and advocate said back in 2018 on Twitter: "Kubernetes supports stateful workloads; I don't."



Kelsey Hightower @kelseyhightower

You can run databases on Kubernetes because it's fundamentally the same as running a database on a VM. The biggest challenge is understanding that rubbing Kubernetes on Postgres won't turn it into Cloud SQL.

Truth is that I agree with him now as much as I agreed with him back then. At that time, the holistic offering of storage capabilities in Kubernetes was still immature (local persistent volumes would become GA only the year after), the operator pattern – which in the meantime has proven to be crucial for stateful applications like databases – was yet to become widely accepted, and the [Data on Kubernetes Community](#) was more than two years away (second half of 2020).

Nowadays, the situation is completely different. And I am sure that many people who've worked hard in the last few years to bring stateful workloads in Kubernetes agree with me that Kelsey's recent powerful words will contribute to reversing the public perception and facilitate our mission – provided we keep doing great.



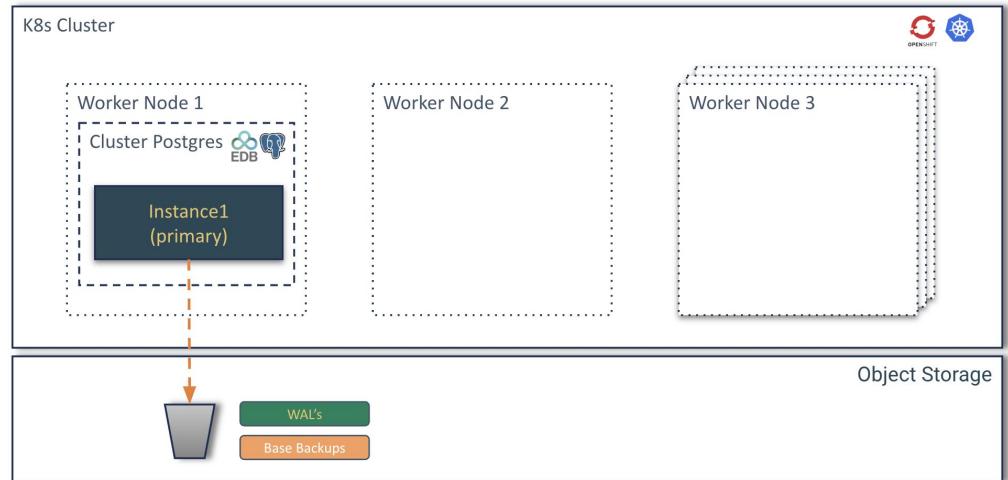
# Use Cases



# Use case 1 architecture

A single database is the simplest setup, involving one instance of a database server.

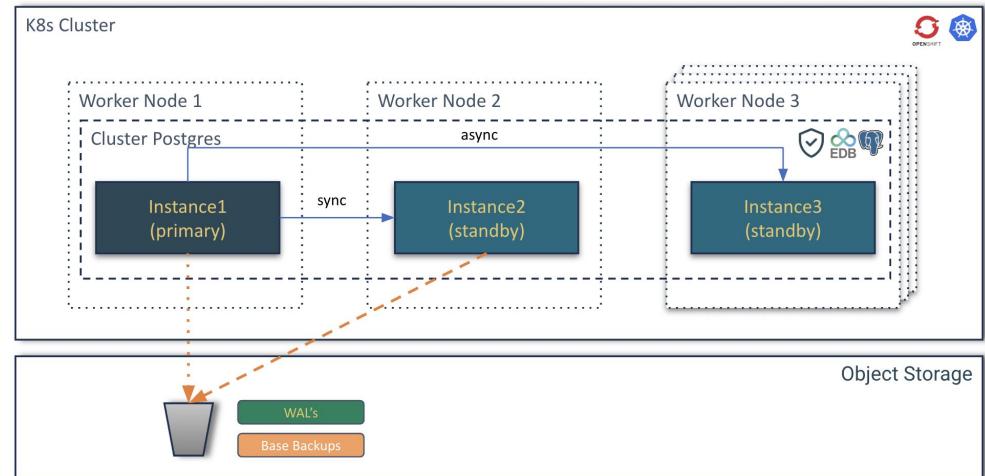
- Development and testing environments
- Small applications with low traffic
- Non-critical data analysis
- Applications with high tolerance for downtime
- Cost-sensitive projects



# Use case 2 architecture

An HA database setup aims to minimize downtime by having redundant components. If one component fails, another takes over automatically or with minimal intervention. This usually involves techniques like clustering, replication, or mirroring within the same data center or availability zone.

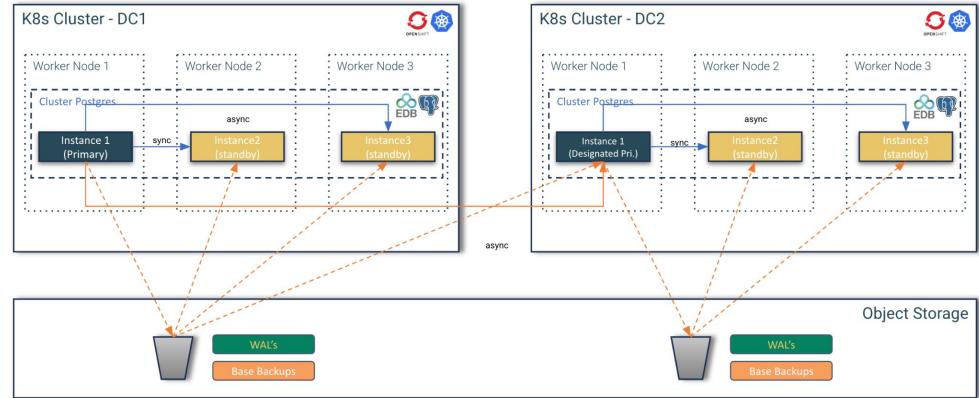
- Business critical Applications
- Applications with stringent SLAs
- Real-time systems
- Improving user experience
- Minimizing planned downtime



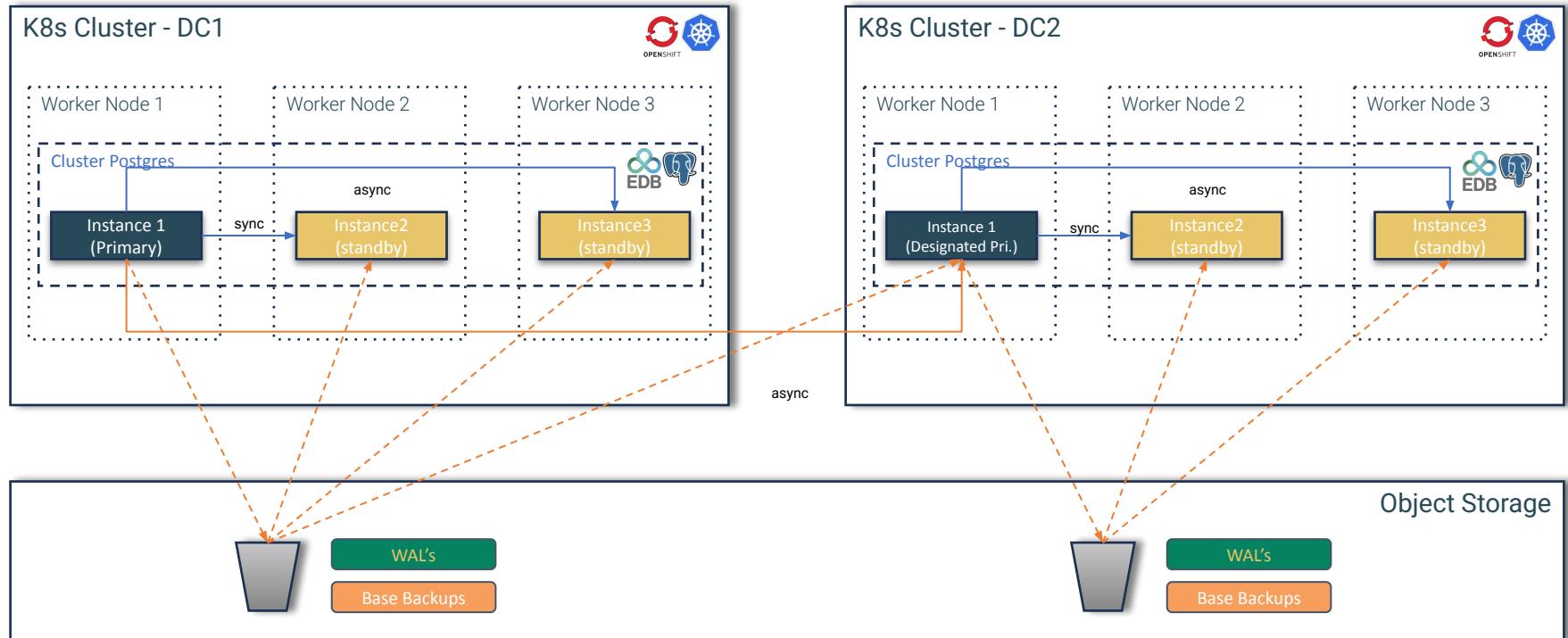
# Use case 3 architecture

A DR database setup focuses on protecting data and ensuring business continuity in the event of a large-scale disaster affecting an entire data center or region (e.g., natural disasters, power outages, cyberattacks). This typically involves replicating data to a geographically separate location.

- Regulatory compliance
- Protecting against catastrophic data loss
- Ensuring business continuity for mission-critical systems



# Use case 3 architecture





# Interactive session

# It's time to go hands-on!



Hand-on documentation



Download this presentation

<https://bit.ly/3SM4GdB>



# Links:

## Openshift Console:

<https://console-openshift-console.apps.cluster-52d72.dynamic.redhatworkshops.io/>

## Users:

name: user4..user40  
password: nLUHTLaOiHHxa1kS

## Devspaces url:

<https://devspaces.apps.cluster-52d72.dynamic.redhatworkshops.io/>

## Short url to Devspaces:

<http://red.ht/edb-cph25>

## URL for devfile (use to create devspace):

<https://github.com/michael-bang/devspace-edb>

## Minio:

UI: <https://minio-ui-minio.apps.cluster-52d72.dynamic.redhatworkshops.io>  
API: <https://minio-api-minio.apps.cluster-52d72.dynamic.redhatworkshops.io>

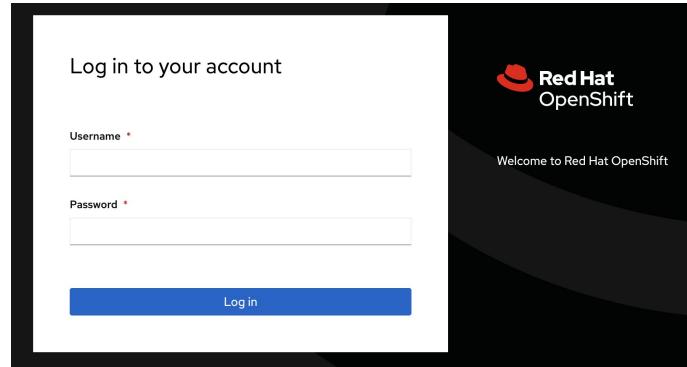
User: minio  
Password: edb-workshop



Open the following URL in your browser:

 <https://red.ht/edb-cph25>

<https://red.ht/edb-cph25>



Username and password provided to you

## Authorize Access

openshift-operators-client is requesting permission to access your account (user)

### Requested permissions

user:full

Full read/write access with all of your permissions

Includes any access you have to escalating resources like secrets

You will be redirected to <https://devspaces.apps.cluster-52d72.dynamic.redhatworkshops.io/oauth/callback>

[Allow selected permissions](#)

[Deny](#)

Press “Allow selected permissions”

## Select a Sample

Select a sample to create your first workspace.

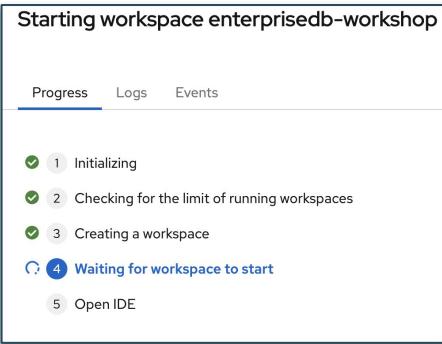
workshop x 1 item



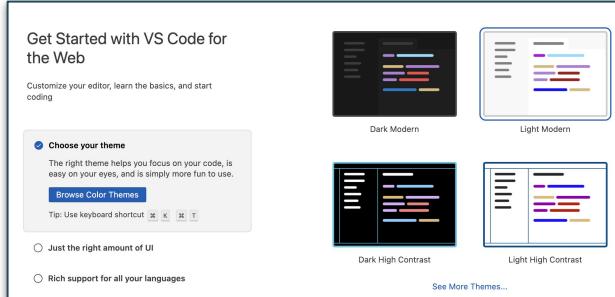
Postgres on OpenShift  
Workshop  
CloudNativePG on OpenShift  
Workshop

In the Select a Sample section search for “Workshop” and click on the tile

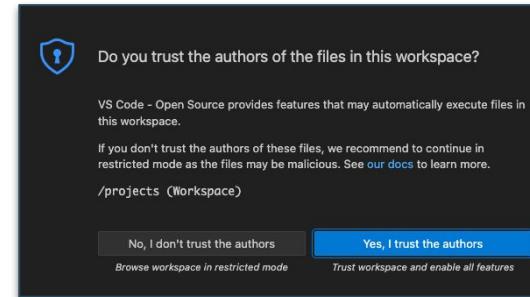
## Environment information



Your workshop is loading ...



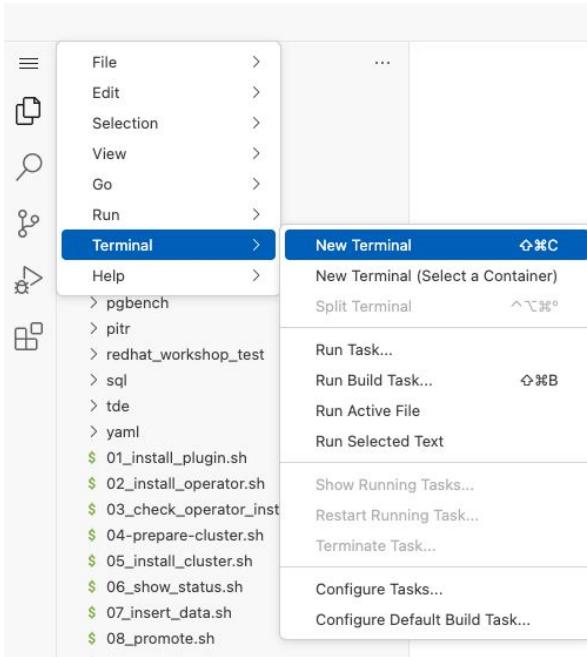
Select your theme



And trust the authors

```
$ config.sh ×  
workshop > $ config.sh  
1  #!/bin/bash  
2  
3  ./commands.sh  
4  
5  # Variables to be replaced  
6  export id=<user>                      # your name or id
```

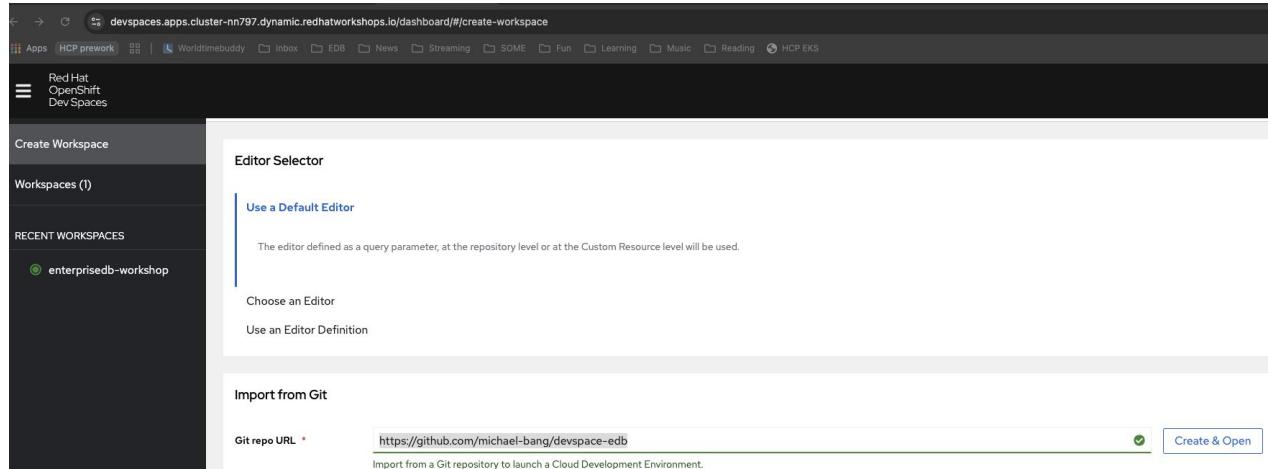
Update your id with the user name you have been assigned



Open a new terminal

# OpenShift Access

Go to DEV Spaces URL and login with your user



Paste the devspace github URL, and click “Create and Open”

Import from Git

Git repo URL \*



Create & Open

Import from a Git repository to launch a Cloud Development Environment.



# Setting up the environment

Use the MinIO URL, username and password to open the MinIO console

The screenshot shows the MinIO Object Store interface. On the left, there's a sidebar with a 'User' section and three items: 'Object Browser', 'Access Keys', and 'Documentation'. The 'Access Keys' item is selected, indicated by a blue background. The main area is titled 'Access Keys' and contains a search bar labeled 'Search Access Keys'. Below the search bar is a table with two columns: 'Select Access Key' and 'Expiry'. There is one row in the table with a checkbox next to the access key '36sVrrwdHFSLTo7ZSbtV' and the expiry status 'no-expiry'.

| Select Access Key                             | Expiry    |
|---|-----------|
| <input type="checkbox"/> 36sVrrwdHFSLTo7ZSbtV | no-expiry |



# Use case

# The environment



# Features shown during the demo

- Kubernetes plugin install
- Check the CloudNativePG operator status
- Postgres cluster install
- Insert data in the cluster
- Failover
- Backup
- Recovery
- Scale out/down
- Fencing
- Hibernation
- Monitoring
- Rolling updates (minor and major)

Deployment

High Availability

Administration

Monitoring

Backup and Recovery

Last CloudNativePG tested version is 1.25

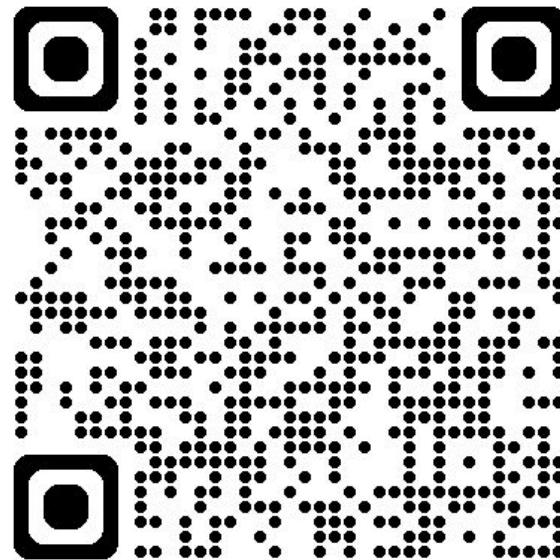


This demo is in



<https://github.com/sergioenterprisedb/edb-postgres-for-kubernetes-in-openshift>

<http://bit.ly/4duKxm7>



# Use case Plug-in installation



# The “cnp” 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
  - Start pgadmin

**NOT NEEDED DURING WORKSHOP**  
**For illustrative purposes.**



```
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: 000000040000000000000000C)
```

NOT NEEDED DURING WORKSHOP  
For illustrative purposes.

#### 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          |



# Install CNPG plugin

NOT NEEDED DURING WORKSHOP  
For illustrative purposes.

- In the web terminal run the script 01\_install\_plugin.sh:

```
./01_install_plugin.sh
```

- Call the help for the CNPG Plugin, run:

```
kubectl-cnp help
```

## Try it for yourself

05:00



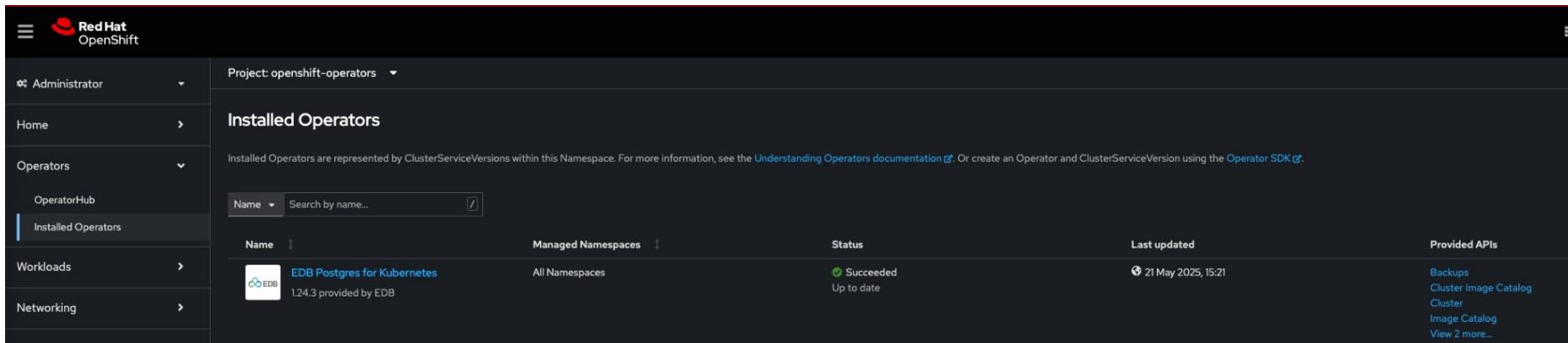
# Use case Operator installation



# Check the installed CNPG Operator in the console

- In the OpenShift console navigate to:
  - -> Operators
  - -> Installed Operators
  - -> Klick on the Operator installed in your namespace, for example: user1:

**NOT NEEDED DURING WORKSHOP**  
For illustrative purposes.



Project: openshift-operators

## Installed Operators

Installed Operators are represented by ClusterServiceVersions within this Namespace. For more information, see the [Understanding Operators documentation](#). Or create an Operator and ClusterServiceVersion using the [Operator SDK](#).

| Name  | Managed Namespaces | Status   | Last updated       | Provided APIs  |
|---|--------------------|--|--------------------|--|
| EDB Postgres for Kubernetes<br>1.24.3 provided by EDB | All Namespaces     | <span>Green circle icon</span> Succeeded<br>Up to date | 21 May 2025, 15:21 | <a href="#">Backups</a><br><a href="#">Cluster Image Catalog</a><br><a href="#">Cluster</a><br><a href="#">Image Catalog</a><br><a href="#">View 2 more...</a> |



# Discover the features of the Operator in the OpenShift environment

NOT NEEDED DURING WORKSHOP  
For illustrative purposes.

The screenshot shows the Red Hat OpenShift web interface. The left sidebar is titled "Administrator" and includes sections for Home, Operators (OperatorHub), Installed Operators, Workloads, Networking, Storage, Builds, Compute, User Management, and Administration. The main content area is titled "Project: openshift-operators". It shows the "Installed Operators" section with "EDB Postgres for Kubernetes" selected. The "Operator details" page displays various tabs: Details (selected), YAML, Subscription, Events, All instances, Backups, Cluster Image Catalog, Cluster, Image Catalog, Pooler, and Scheduled Backups. The "Provided APIs" section lists several services: Backups, Cluster Image Catalog, Cluster, Image Catalog, Pooler, and Scheduled Backups, each with a "Create instance" button. To the right, detailed information about the operator is shown, including its provider (EDB), creation date (21 May 2025, 15:21), links to EDB Postgres for Kubernetes documentation and GitHub, and maintainers' names and emails (Jonathan Gonzalez V., Jonathan Battato, Niccolo Fei, Gabriele Bartolini).

Project: openshift-operators

Installed Operators > Operator details

EDB Postgres for Kubernetes

124.3 provided by EDB

Actions

Administrator

Home

Operators

OperatorHub

Installed Operators

Workloads

Networking

Storage

Builds

Compute

User Management

Administration

Details YAML Subscription Events All instances Backups Cluster Image Catalog Cluster Image Catalog Pooler Scheduled Backups

Provided APIs

Backups PostgreSQL backup (physical base backup)

Create instance

Cluster Image Catalog A cluster-wide catalog of PostgreSQL operand images

Create instance

Cluster PostgreSQL cluster (primary/standby architecture)

Create instance

Image Catalog A catalog of PostgreSQL operand images

Create instance

Pooler Pooler for a Postgres Cluster (with PgBouncer)

Create instance

Scheduled Backups Backup scheduler for a given Postgres cluster

Create instance

Description

Main features:

Provider EDB

Created at 21 May 2025, 15:21

Links EDB Postgres for Kubernetes <https://www.enterprisedb.com/products/postgresql-on-kubernetes-ha-clusters-k8s-containers-scalable>

Documentation [https://www.enterprisedb.com/docs/postgres\\_for\\_kubernetes/latest/](https://www.enterprisedb.com/docs/postgres_for_kubernetes/latest/)

Maintainers

Jonathan Gonzalez V. [jonathan.gonzalez@enterprisedb.com](mailto:jonathan.gonzalez@enterprisedb.com)

Jonathan Battato [jonathan.battato@enterprisedb.com](mailto:jonathan.battato@enterprisedb.com)

Niccolo Fei [niccolo.fei@enterprisedb.com](mailto:niccolo.fei@enterprisedb.com)

Gabriele Bartolini [gabriele.bartolini@enterprisedb.com](mailto:gabriele.bartolini@enterprisedb.com)



# Check the installed CNPG Operator in the web terminal

- In the web terminal check the installation of the operator:

```
./02_install_operator.sh (will require admin privs on Openshift)
```

```
./03_check_operator_installed.sh
```



**NOT NEEDED DURING WORKSHOP**  
For illustrative purposes.

# Operator Installation demonstration

- Check the installed CNP Operator in the console
- Discover the features of the Operator in the OpenShift environment
- Check the installed CNP Operator in the web terminal



NOT NEEDED DURING WORKSHOP  
For illustrative purposes.

# Use case

## Create the postgres cluster



# Bootstrap - Two ways of creating a cluster

- Create a new cluster from scratch
  - “initdb”: named after the standard “initdb” process in PostgreSQL that initializes an instance
- Create a new cluster from an existing one:
  - Directly (“pg\_basebackup”), using physical streaming replication
  - Directly (logical backup/restore) using pg\_dump and pg\_restore
  - 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



# Configure and Install the Postgres cluster

- Prepare for cluster-creation (ensure minio secrets are in place)

```
./04-prepare-cluster
```

- Create a new 3-node cluster by running

```
./05_install_cluster.sh
```

- Check the status of the cluster (using the CNP plugin):

```
./06_show_status.sh
```



# Create table test with 1000 rows

- Once cluster is running ... (minimum the primary) run the script:

```
./07_insert_data.sh
```

Try it for yourself

10:00



# Use case

## Promote & Upgrade the postgres cluster



# 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



# Check the cluster status

- In terminal **1**: (prepare a terminal for status - and one to run the admin-commands):
  - Run the command  
./06\_show\_status.sh
  - Review the output:
    - check Postgres version: "PostgreSQL Image: quay.io/enterprisedb/postgresql:**16.2**"
    - check "Continuous Backup status": "**Not configured**"
  - Check the updated cluster configuration - file cluster-example-upgrade.yaml  
less ./yaml/cluster-sample-upgrade.yaml
    - Check Postgres version: "imageName: quay.io/enterprisedb/postgresql:**16.4**"
    - Check the Backup section



# Run the Promote and Upgrade

- With this step we will:
  - Promote node-2 to become the primary
  - Run the postgres minor update from the version 16.2 to 16.4
  - We will configure the WAL files backup to the S3 storage
- In the web terminal **2**:
  - Check the upgrade status:  
`./06_show_status.sh`
- In the terminal **1**:
  - Run the script:  
`./08_promote.sh`
  - Run the script:  
`./09_upgrade.sh`

## Try it for yourself

05:00



# Use case Backup & Restore



# 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 |
|---------------|---------------|--------------------|--------------------|-----------------|-----------------------|
| <i>tiny</i>   | 4.5 GB        | 300                | 8 GB               | 1 GB            | 67s                   |
| <i>small</i>  | 44 GB         | 3,000              | 80 GB              | 10 GB           | 10m 50s               |
| <i>medium</i> | 438 GB        | 3,0000             | 800 GB             | 100 GB          | 3h 15m 34s            |
| <i>large</i>  | 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 |
|--|---------------|---------------------|--|--------------------|
|  | <i>tiny</i>   | 2m 43s              |  | 4m 16s             |
|  | <i>small</i>  | 20m 38s             |  | 16m 45s            |
|  | <i>medium</i> | 2h 42m              |  | 2h 34m             |
|  | <i>large</i>  | 3h 54m 6s           |  | 2m 2s              |

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



# Create the full backup

- With this step we will:
  - Create the full backup of the postgres cluster in the MinIO storage:

- In the web terminal 1:

- Run the script:

```
cd /projects/workshop  
./10_backup_cluster.sh
```

- Check the backup status:

```
cd /projects/workshop  
./11_backup_describe.sh
```



# Check Backup in the Openshift Console

- Navigate to:
  - -> Operators
  - -> Installed Operators
  - -> Press on the Operator installed in your namespace, for example: user1:
  - -> Go to the Backup section and show the created backup:

The screenshot shows two panels of the OpenShift web interface. The left panel displays the 'OperatorHub' section under the 'Operators' menu, with a red arrow pointing to the 'Installed Operators' link. The right panel shows the 'EDB Postgres for Kubernetes' operator details page, also with a red arrow pointing to the 'Backups' tab. Both panels include navigation links for Details, YAML, Subscription, Events, All instances, and Backups.

**Left Panel (OperatorHub):**

- Project: openshift-operators
- OperatorHub
- Installed Operators
- Workloads
- Networking
- Storage
- Builds
- Compute
- User Management
- Administration

**Right Panel (Operator details):**

- Project: openshift-operators
- Installed Operators > Operator details
- EDB Postgres for Kubernetes
- Provider: EDB
- Created at: 21 May 2025, 15:21
- Links: EDB Postgres for Kubernetes | Documentation | GitHub | Issues | Conferences | Available
- Documentation: <https://www.edb-postgres.com>
- Maintainers:
  - Jean-Baptiste V (jv@edb-postgres.com)
  - Nicolas F (nicolas@edb-postgres.com)
  - Gabriel Barcelo (gbarcelo@edb-postgres.com)
- Workloads
- Networking
- Storage
- Builds

**Backups Tab:**

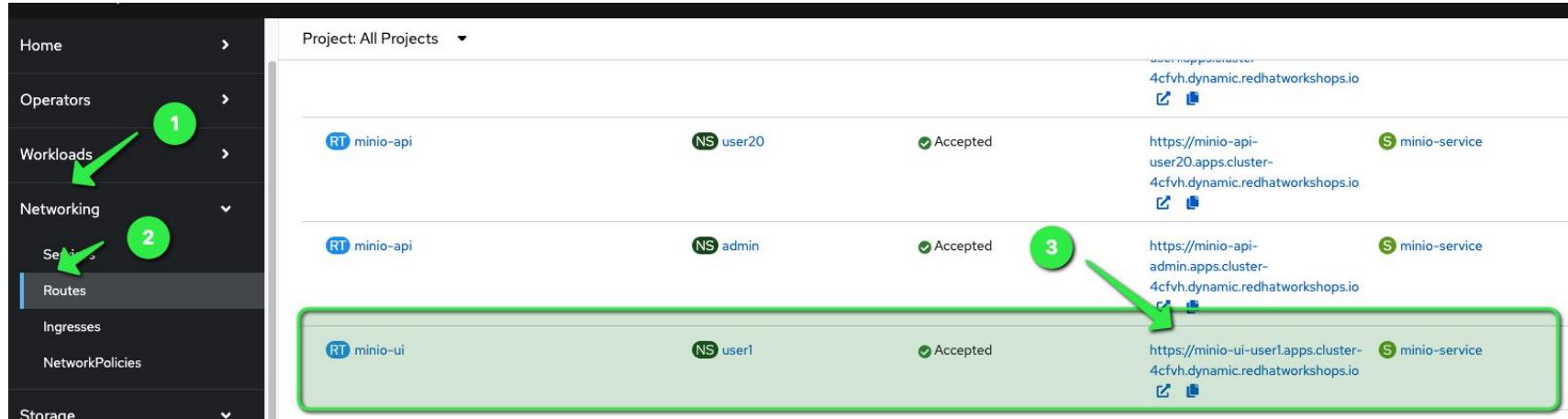
Show operands in:  All namespaces  Current namespace only

| Name                      | Kind   | Namespace         | Status         |
|---------------------------|--------|-------------------|----------------|
| backup-test               | Backup | NS edb-eMEA-user1 | Phase: Pending |
| cluster-user1-backup-test | Backup | NS edb-eMEA-user1 | Phase: Pending |



# Check Backup in MinIO UI

- In the Openshift console navigate to Networking -> Routes
- Search for route minio-ui for your user and press the button with url:



# Check Backup in MinIO UI

- Connect as user **admin** with the password: **password**
- The page will appear:

The screenshot shows the MinIO Object Browser interface. On the left, a sidebar menu includes 'User' sections for 'Object Browser', 'Access Keys', 'Documentation', and 'Administrator' sections for 'Buckets', 'Policies', and 'Identity'. The 'Object Browser' item is highlighted with a blue background. The main content area is titled 'Object Browser' with a back arrow. It displays a search bar with the placeholder 'Start typing to filter objects in the bucket'. Below the search bar, a table lists objects in the 'cnp' bucket. The table has columns for 'Name' and 'Last Modified'. One object, 'cluster-example', is listed under the 'Name' column.

| Name            | Last Modified |
|-----------------|---------------|
| cluster-example |               |

← Object Browser

Start typing to filter objects in the bucket

cnp

Created on: Mon, Apr 14 2025 18:54:07 (GMT+2) Access: PRIVATE 63.6 MiB - 6 Objects

Name Last Modified

cluster-example



# Restore the database from the backup

- With this step we will:
  - Create the new cluster cluster-restore
  - Restore the full backup created in the previous step in the new cluster:
- In the terminal 1:
  - Run the restore:  
`./12_restore_cluster.sh`
  - Check the creation status:  
`kubectl get pods -w # after creation stop the execution with <ctrl>+c`
  - Check the table test in the cluster-restore, run the script:  
`oc exec -it cluster-restore-user<X>-1 - psql -U postgres -c "\d test"`
  - Delete the cluster-restore-user<x> to avoid resource problems during the workshop:  
`oc delete cluster cluster-restore-user<X>`



# Backup demonstration

- Create the full backup
- Check Backup in the Openshift Console
- Check Backup in MinIO UI
- Restore the database from the backup

Try it for yourself

15:00



# Use case: Failover



# Run failover test

- With this step we will:
  - Delete the primary database of the cluster cluster-example
  - Check the cluster status in the another terminal window
- In the web terminal 1:
  - Run the script:  
`./13_failover.sh`
- In the web terminal **2**:
  - Check the failover cluster status:  
`./06_show_status.sh`

Try it for yourself

05:00



# Use case Scale-out and scale-down



# Scale-out the postgres cluster

- With this step we will:
  - Add the 1 standby to the cluster
- In the web terminal 1:
  - Run the script:  
`./14_scale_out.sh` (using `-replicas=X...` another way would be to update the YAML)
- In the web terminal **2**:
  - Check the cluster status:  
`./06_show_status.sh`



# Scale-down the postgres cluster

- With this step we will:
  - Remove 2 standby pods from the cluster
- In the web terminal 1:
  - Run the script:  
`./15_scale_down.sh`
- In the web terminal **2**:
  - Check the cluster status:  
`./06_show_status.sh`

Try it for yourself

05:00



# Use Case Fencing



# Stop postgres process on the pod

- In the web terminal 1:
  - Run the script:

```
./30_fencing_on.sh
```

- In the web terminal **2**:
  - Check the cluster status:

```
./06_show_status.sh
```



# Start the postgres process on the pod

- In the terminal 1:
  - Run the script:  
./31\_fencing\_off.sh
- In the terminal **2**:
  - Check the cluster status:  
./06\_show\_status.sh

## Try it for yourself

05:00



# Use case Hibernation



# Stop the postgres cluster

- In the terminal 1:
  - Run the script:

```
./32_hibernation_on.sh
```

- In the terminal **2**:
  - Check the cluster status:

```
./06_show_status.sh
```



# Start the postgres cluster

- In the terminal 1:
  - Run the script:

./33\_hibernation\_off.sh

- In the terminal **2**:
  - Check the cluster status:

./06\_show\_status.sh

## Try it for yourself

05:00



# Use case Monitoring



# 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

< >

① ⚡

localhost



Prometheus

Alerts

Graph

Status ▾

Help

Use local time

Enable query history

Enable autocomplete

Enable highlighting

Enable linting

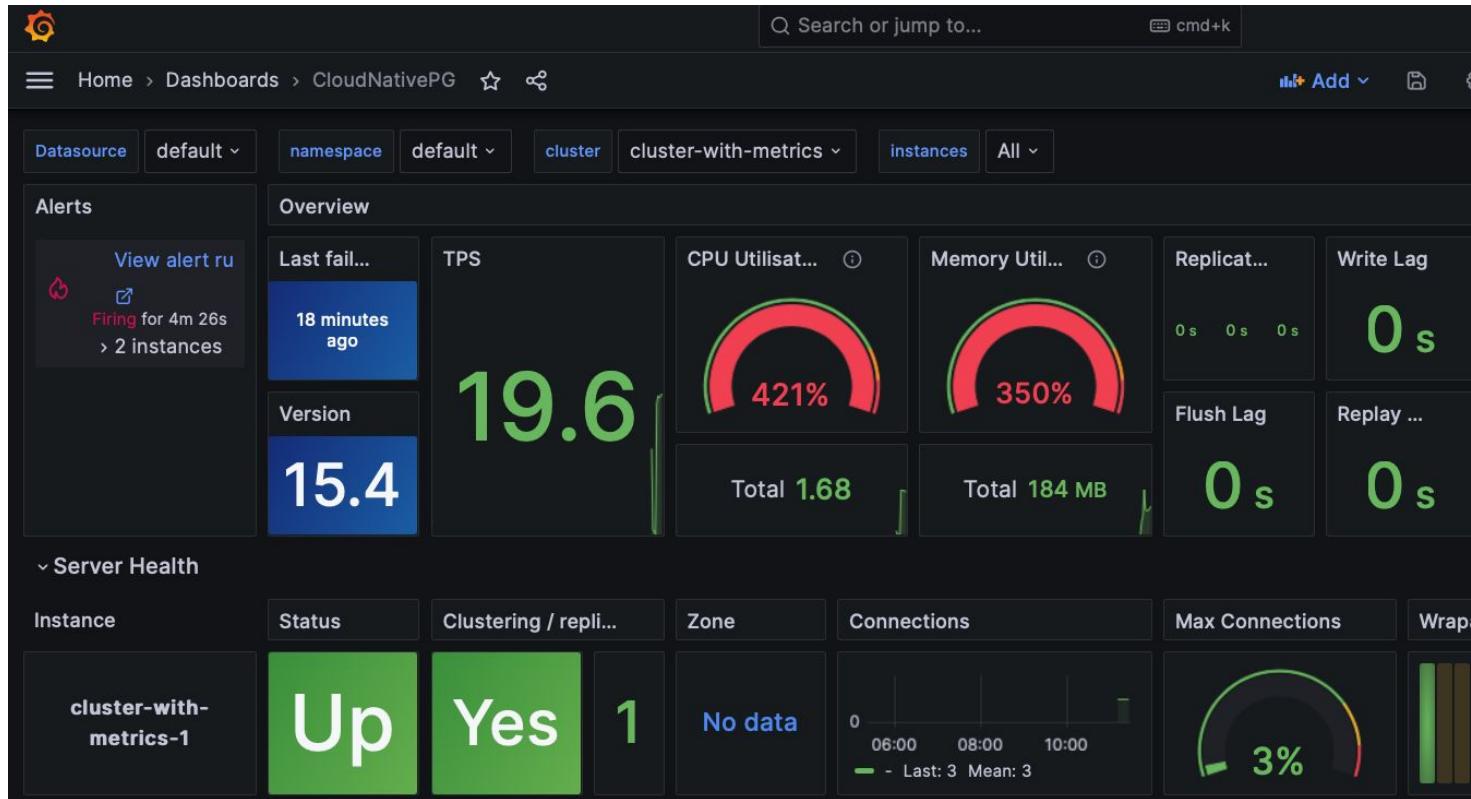


cnp|

|       |   |         |
|-------|---|---------|
| Table | <input type="checkbox"/> <a href="#">cnp_pg_backends_max_tx_duration_seconds</a>      | gauge   |
|       | <input type="checkbox"/> <a href="#">cnp_pg_backends_total</a>                        | gauge   |
|       | <input type="checkbox"/> <a href="#">cnp_pg_backends_waiting_total</a>                | gauge   |
|       | <input type="checkbox"/> <a href="#">cnp_pg_collector_collection_duration_seconds</a> | gauge   |
|       | <input type="checkbox"/> <a href="#">cnp_pg_collector_collections_total</a>           | counter |
| No d  | <input type="checkbox"/> <a href="#">cnp_pg_collector_first_recoverability_point</a>  | gauge   |
|       | <input type="checkbox"/> <a href="#">cnp_pg_collector_last_collection_error</a>       | gauge   |
|       | <input type="checkbox"/> <a href="#">cnp_pg_collector_lo_pages</a>                    | gauge   |



# Grafana Dashboard



# Step 20: Setup monitoring

- In the web terminal 1:
  - Configure Grafana:

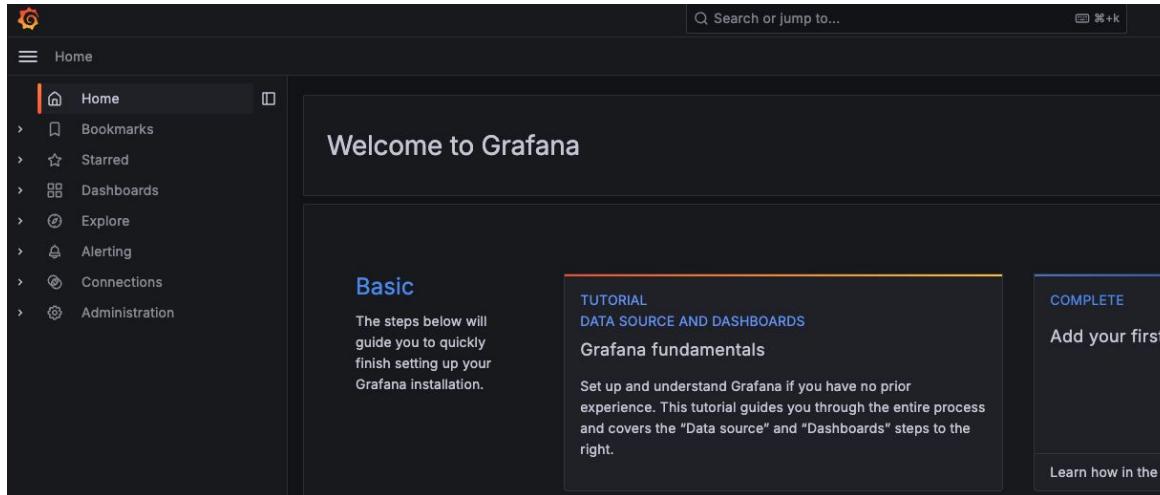
```
cd /home/user/edb-workshop
./monitoring.sh
```
  - In the Openshift console navigate to Networking, then to Routes
    - Search for the route "grafana" for your user, then click on location:

The screenshot shows the Red Hat OpenShift console interface. On the left, there is a navigation sidebar with various project and cluster management options. The 'Networking' section is expanded, showing 'Workloads' (with a green arrow pointing to it), 'Services' (with a green arrow pointing to it), and 'Routes'. Under 'Routes', a search bar is present with the query 'grafana'. Below the search bar, a table lists four routes. The first route, for namespace 'user3', is highlighted with a green box and has a green arrow pointing to its 'location' column, which contains the URL 'https://grafana-user3.apps.cluster-4cfvh.dynamic.redhatworkshops.io'. The other three routes are for namespaces 'user30', 'admin', and 'user4', each with a different URL.

| Name       | Namespace | Status   | Location   | Service         |
|------------|-----------|----------|--|-----------------|
| RT grafana | NS user3  | Accepted | https://grafana-user3.apps.cluster-4cfvh.dynamic.redhatworkshops.io  | grafana-service |
| RT grafana | NS user30 | Accepted | https://grafana-user30.apps.cluster-4cfvh.dynamic.redhatworkshops.io | grafana-service |
| RT grafana | NS admin  | Accepted | https://grafana-admin.apps.cluster-4cfvh.dynamic.redhatworkshops.io  | grafana-service |
| RT grafana | NS user4  | Accepted | https://grafana-user4.apps.cluster-4cfvh.dynamic.redhatworkshops.io  | grafana-service |

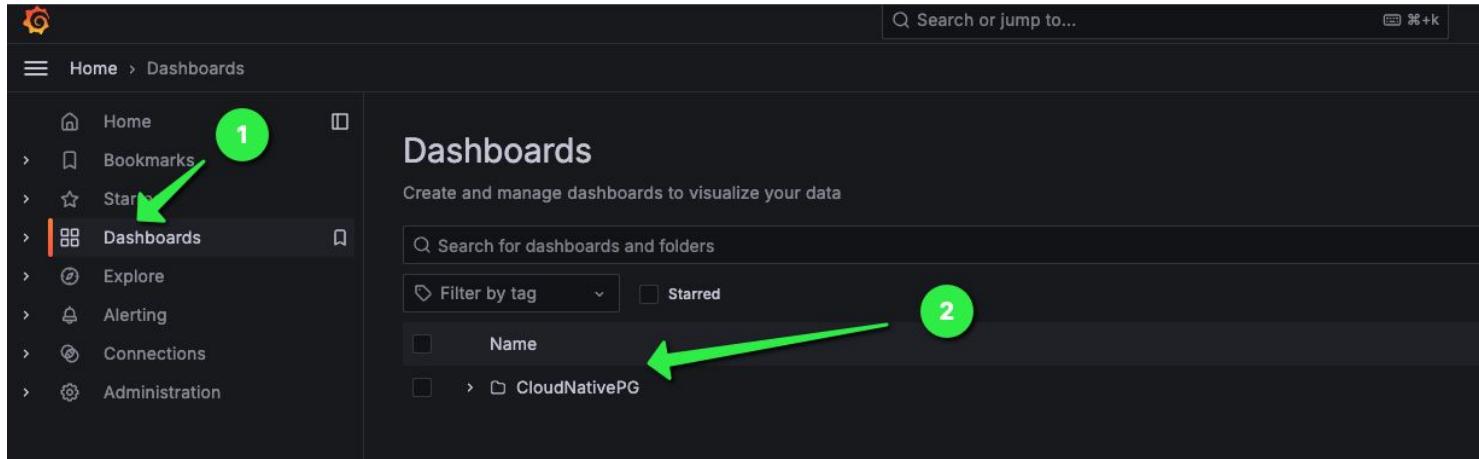
# Step 21: Access the Grafana page

- Grafana interface appears
- Connect as user **admin** with the password: **grafana**
- The grafana page will appear



# Step 22: Navigate to CloudNativePG Dashboard

- Go to Dashboards and click on CloudNativePG:



# Step 23: Explore CNPG Dashboard

- Explore the CNPG Dashboard:



# Monitoring demonstration

- Step 9: Create the full backup
- Step 2: Discover the features of the Operator in the OpenShift environment
- Step 3: Check the installed CNPG Operator in the web terminal

Try it for yourself

10:00



# Use case Major version Upgrade



# Delete cluster restore and upgrade cluster

- In the web terminal 1:
  - Delete the cluster cluster-restore:

kubectl delete cluster cluster-restore-user<x>

- In the web terminal 1:
  - ./20\_upgrade\_major\_version.sh



What more?  
(some additional features from EDB)

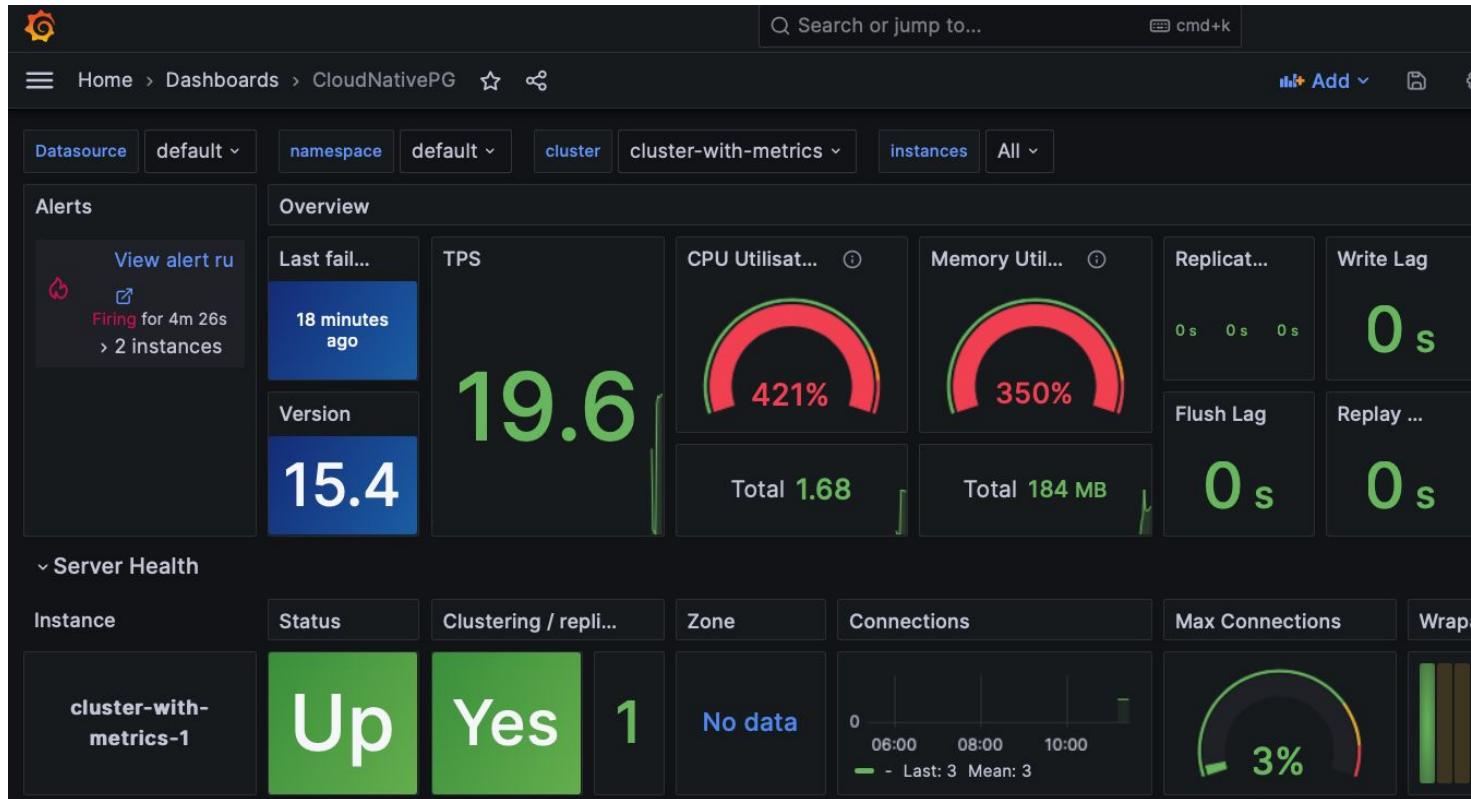


# What we didn't show you today ....

- PgBouncer (Pooler) integration
  - Create a PgBouncer deployment and automatically configure to the cluster.
- Monitoring using Prometheus and Grafana
  - Exporting to OpenMetrics (Prometheus)



# Grafana Dashboard



# 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

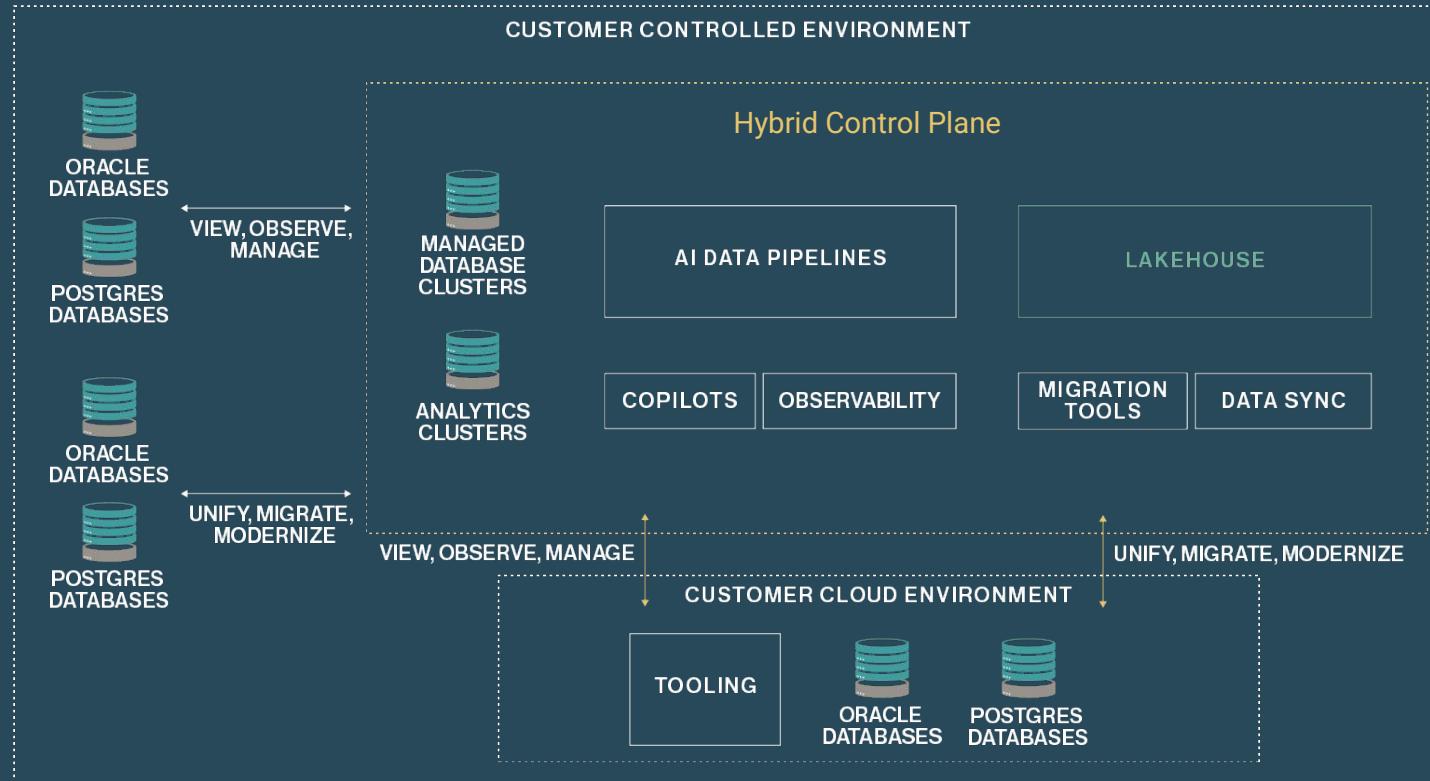


# Transparent Data Encryption (EDB-only features)

- 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

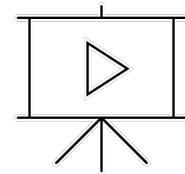


# Hybrid Control Plane at a glance



# Hybrid Control Plane

LIVE DEMO



# Something Big Is Coming

June 17, 2025 | 11 AM CEST

Meet the Future of EDB Postgres® AI



On June 17, we reveal the next stage of EDB Postgres AI and show you how enterprises are using it to build secure, scalable, sovereign AI foundations.

Join us and our partners Red Hat & SuperMicro to explore what's driving that and how EDB Postgres AI helps you act on it.





Share Your Opinion &  
Win a LEGO  
LEGO Technic 42207  
Ferrari!

Scan the QR Code & complete the survey  
to let us know your feedback!



By accepting this prize, you confirm that the prize conforms to  
your employer's internal rules, policies, and codes of conduct.



<https://wheelofnames.com/>



By accepting this prize, you confirm that the prize conforms to your employer's internal rules, policies, and codes of conduct.



Thank you for participating in the  
Postgres on Kubernetes Workshop

Please pick up your certificate :-)



# Drinks & Pizza

