



CNPG on Openshift Workshop Oslo

24 November 2025

Per Bakke - Sales Specialist, Red Hat

Michael Willer - Senior Solutions Engineer, EDB

Niels van Noort - Senior Solutions Engineer, EDB

Søren Boss Jacobsen - Senior Account Executive, EDB



Agenda

Start	End	Session
13:00	13:30	Registration & Welcome
13:30	13:45	Red Hat OpenShift & EDB Partnership (Red Hat - Per Bakke)
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 - Michael Willer)
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 food



EDB Partnership with Redhat



Red Hat OpenShift with EDB

Michael Bang

Associate Principal Solution Architect

Red Hat is a Leader in the 2024 Gartner® Magic Quadrant™: Container Management

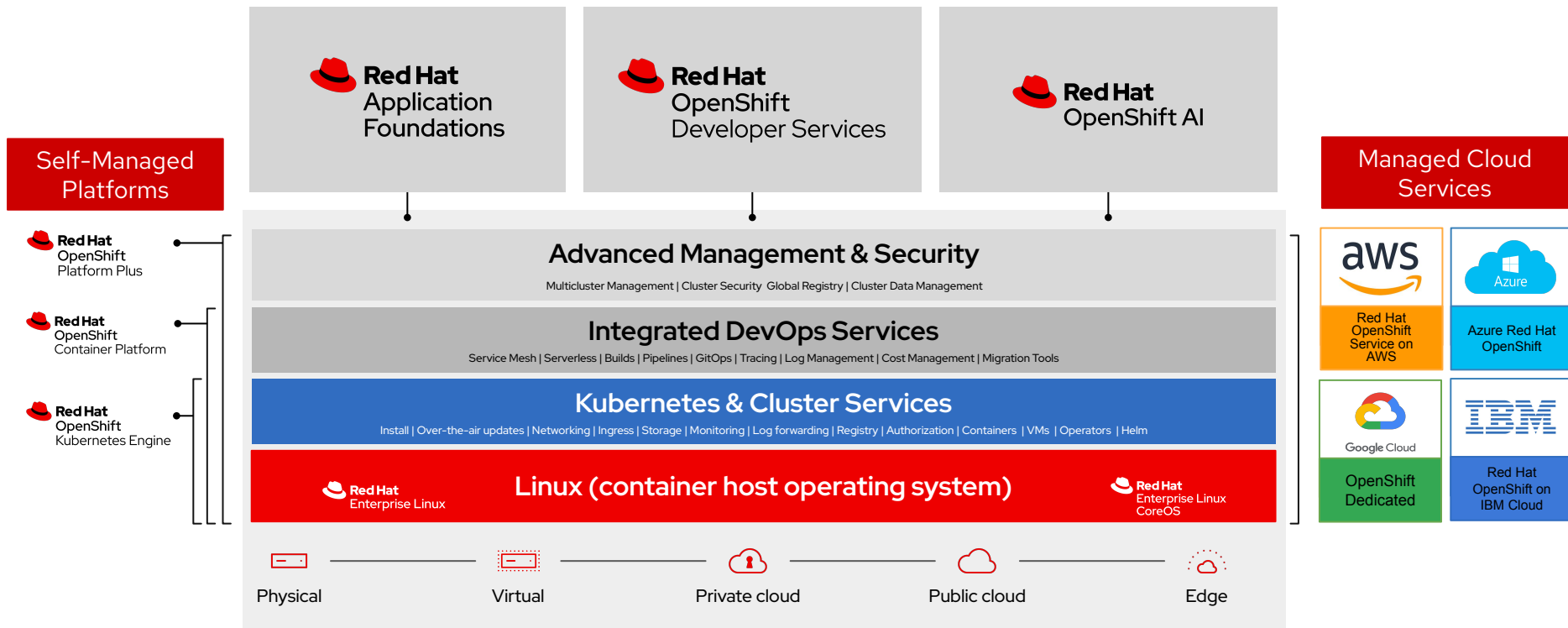
Figure 1: Magic Quadrant for Container Management



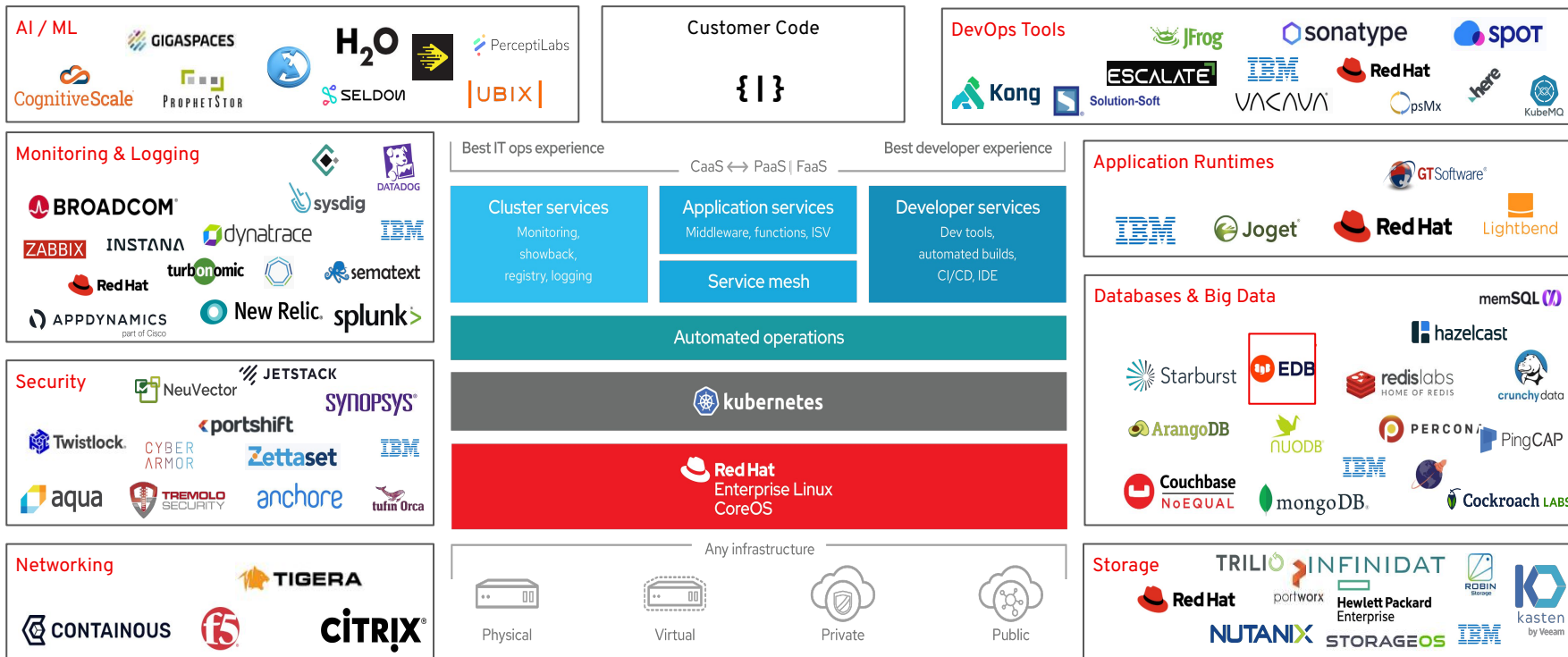
Gartner

Source: Gartner, "Magic Quadrant for Container Management," September 2024.

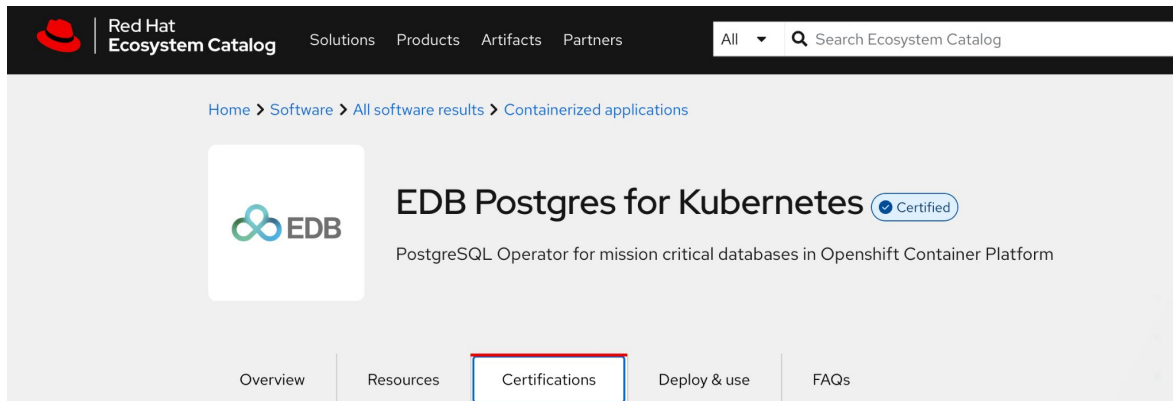
Hybrid Cloud Application Platform



Red Hat open hybrid cloud platform with ISV ecosystem




Why Red Hat OpenShift for EDB: operator certification



Red Hat Ecosystem Catalog Solutions Products Artifacts Partners All Search Ecosystem Catalog

Home > Software > All software results > Containerized applications

 **EDB Postgres for Kubernetes** Certified

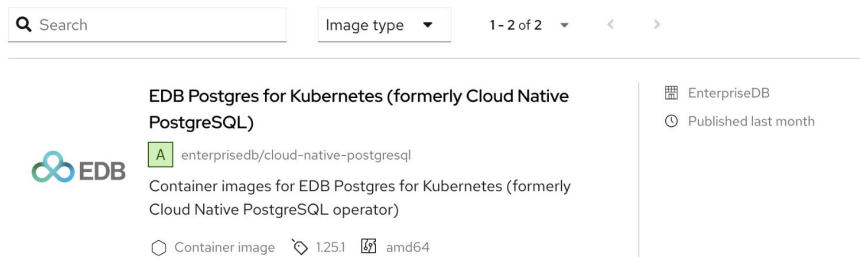
PostgreSQL Operator for mission critical databases in Openshift Container Platform

Overview Resources **Certifications** Deploy & use FAQs


Certifications


Learn about Red Hat Certification and Partner Validation

Certified components







Search Image type 1 - 2 of 2 < >


 **EDB Postgres for Kubernetes (formerly Cloud Native PostgreSQL)**

 enterisedb/cloud-native-postgresql

Container images for EDB Postgres for Kubernetes (formerly Cloud Native PostgreSQL operator)

 Container image  1.25.1  amd64

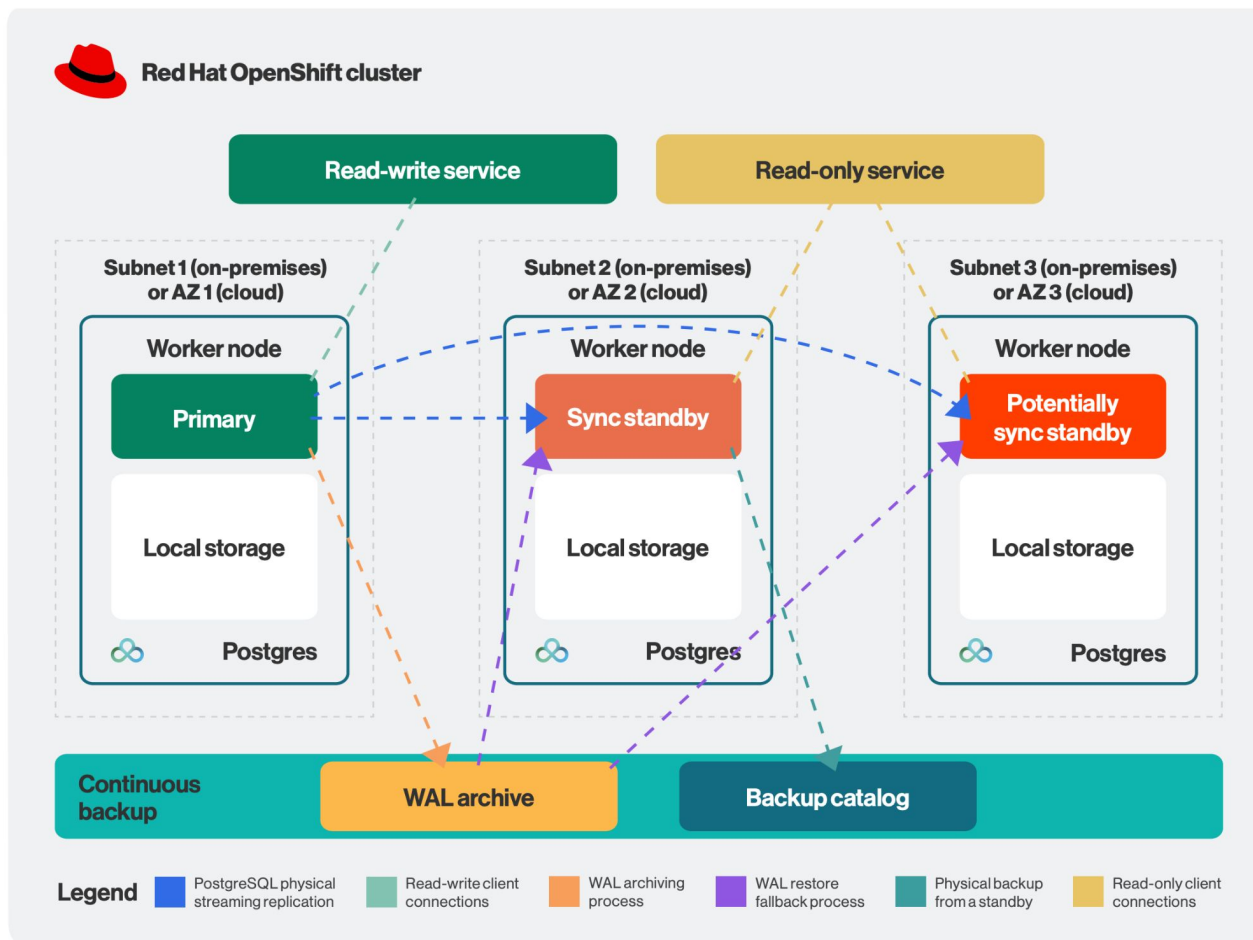
 EnterpriseDB

 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.



- 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

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



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.



- 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

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.

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.



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



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

Summary

Use Case

On prem DBaaS (Production)

Workload

Transactional

Application Name

All VMWare PostgreSQL databases

EDB Tools of Interest

EDB Postgres Advanced Server with Oracle and TDE (optional)

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

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.



- Red Hat OpenShift
- CloudNativePG
- PostgreSQL



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

Summary

Use Case

OnPrem DBaaS (in production)

Workload

Transactional

Application Name

Multiple applications

EDB Tools of Interest

PostgreSQL, CloudNativePG

Problem

- Fast database deployment
- Provide containerized Postgres DBaaS

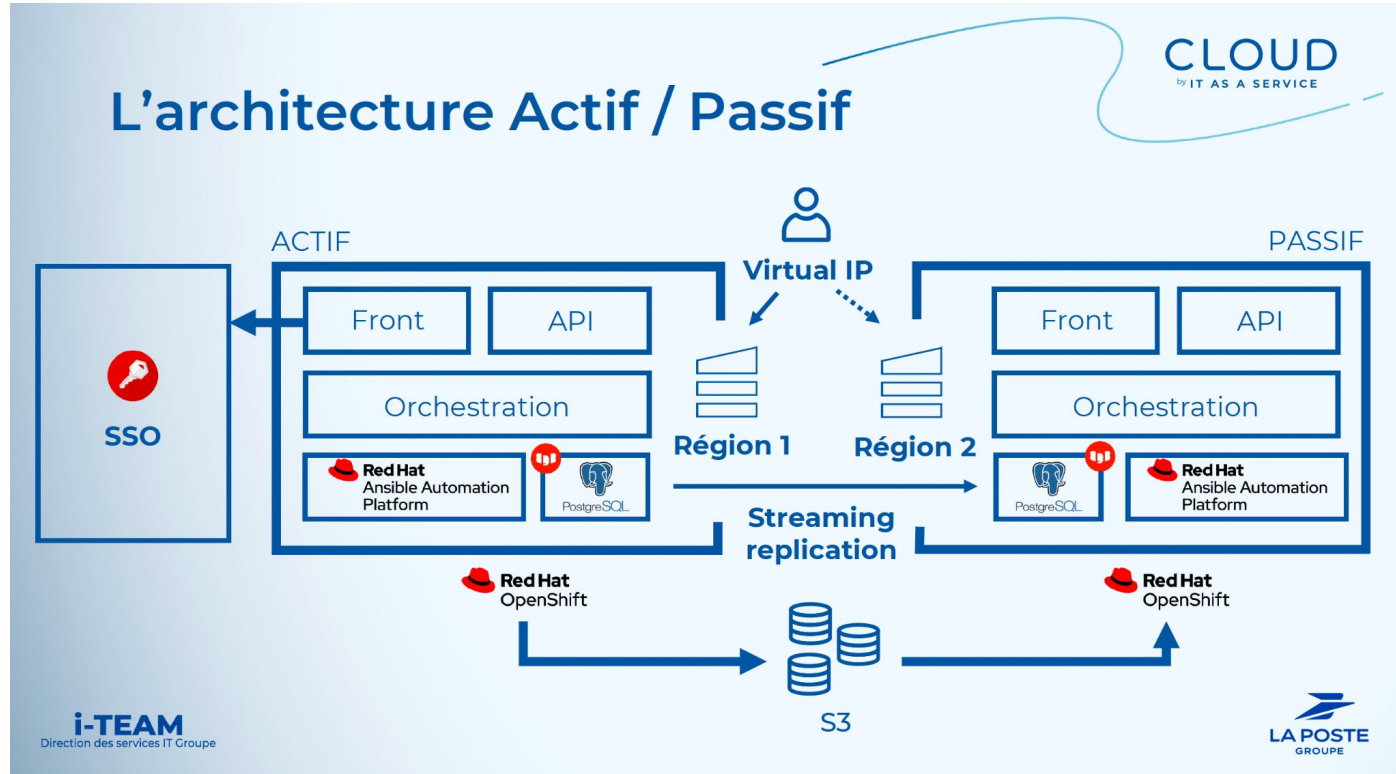
Solution

- 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

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



youtube.com/user/RedHatVideos



facebook.com/redhatinc



twitter.com/RedHat



Introduction to Postgres and EDB



EDB

Postgres® for the AI Generation

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



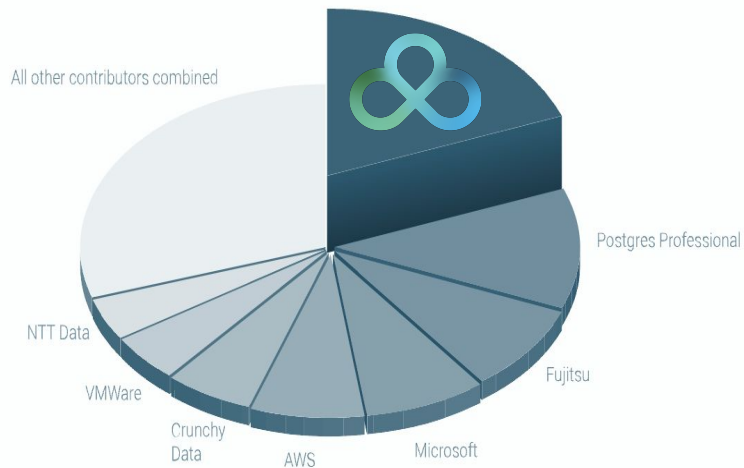
20+ years of innovation

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

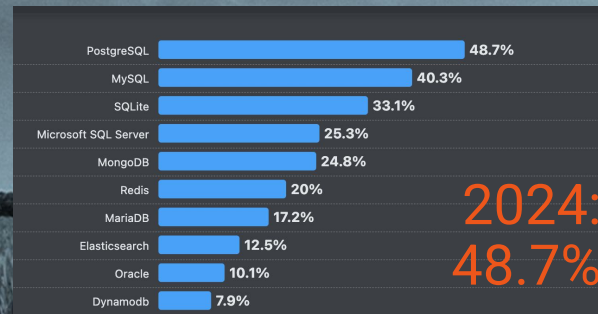
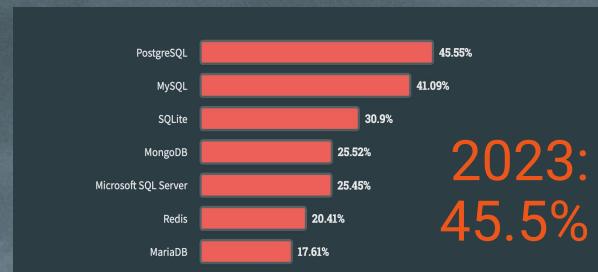
#1 contributor to Postgres

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

30% + of Postgres Code Contributed in 2024



Postgres has won the database race

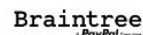


Stack Overflow Survey 2023/2024

BANKING FINANCIAL



TECHNOLOGY



TELCO



EDB POSTGRES AI PLATFORM

UNIFIED WORKLOAD MANAGEMENT

TRANSACTIONAL

ANALYTICAL

ARTIFICIAL INTELLIGENCE

SINGLE PANE OF GLASS ADMINISTRATION

HYBRID
CONTROL PLANE

DATABASE AUTOMATION
AND MANAGEMENT

INTELLIGENT
OBSERVABILITY

ENTERPRISE
SECURITY

MULTI-MODEL DATA MANAGEMENT

RELATIONAL

JSON

TIME SERIES

VECTOR

COLUMNAR

HYBRID AND MULTI-CLOUD DEPLOYMENT

SOFTWARE
DEPLOYMENT

CLOUD
SERVICE

HARDWARE
INTEGRATED SOLUTION

PLATFORM TOOLS AND SERVICES

MIGRATION
PORTAL

CONTINUOUS HIGH
AVAILABILITY

BACKUP AND
RECOVERY

EXTENSIBILITY

CSP INTEGRATIONS

DEVOPS TOOLING

KUBERNETES TOOLING

GENAI & LLM INTEGRATIONS

LAKEHOUSE INTEGRATIONS



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



 **Red Hat**
OpenShift



CloudNativePG



EDB
Postgres® AI



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 man in a tactical vest with the word "SECURITY" printed on the back, holding a walkie-talkie to his mouth. The image is faded and serves as a background for the slide.

SECURITY

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

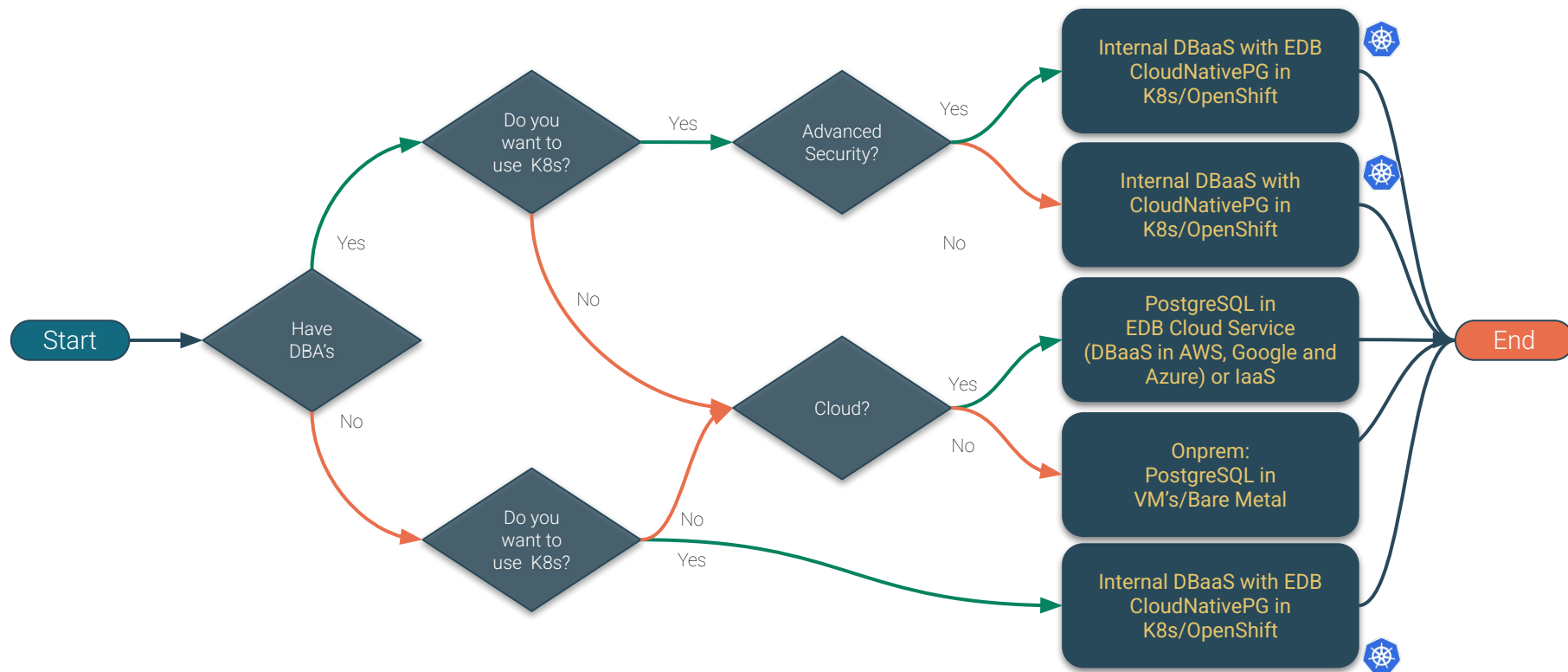
- 05 |
- Automated backups
 - Multi-region failover

Security & access control

- 06 |
- RBAC
 - Secret management



Decision-making for choosing the deployment platform

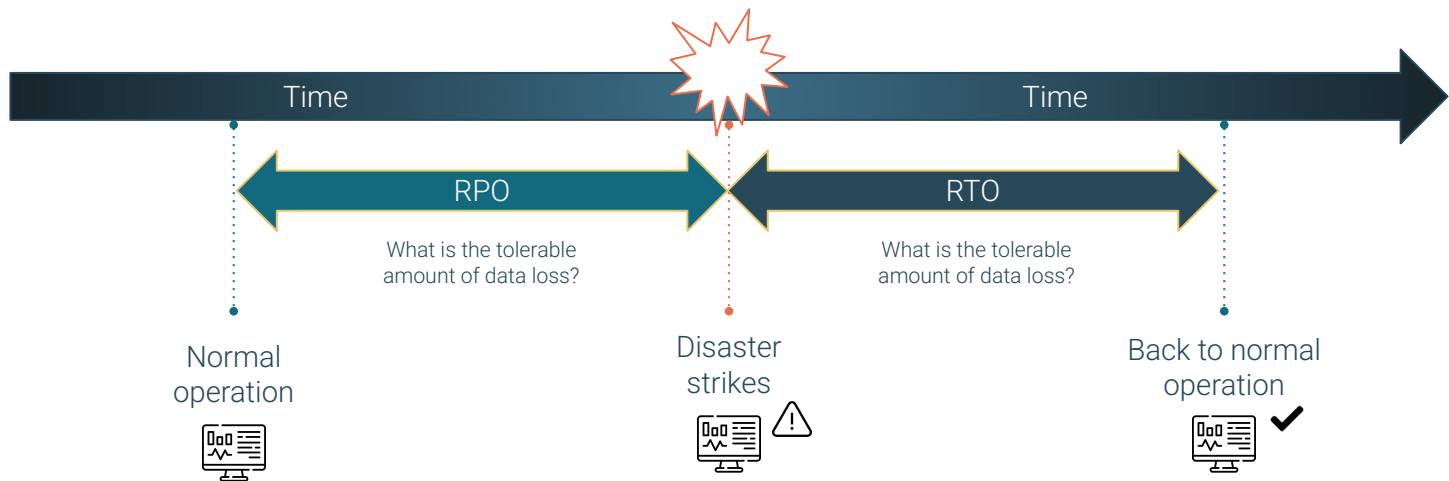


Architectures



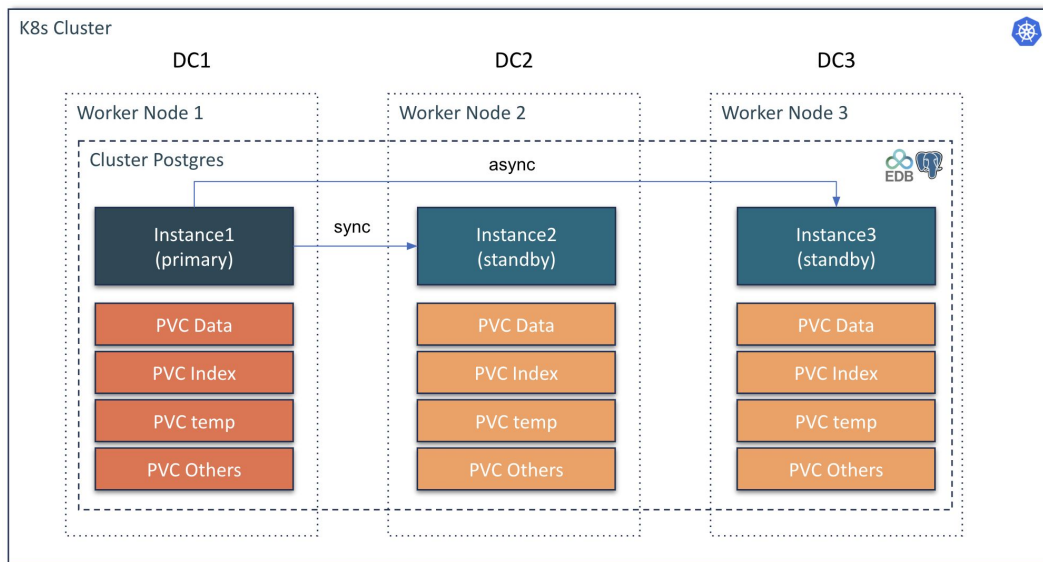
Concepts

- 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.



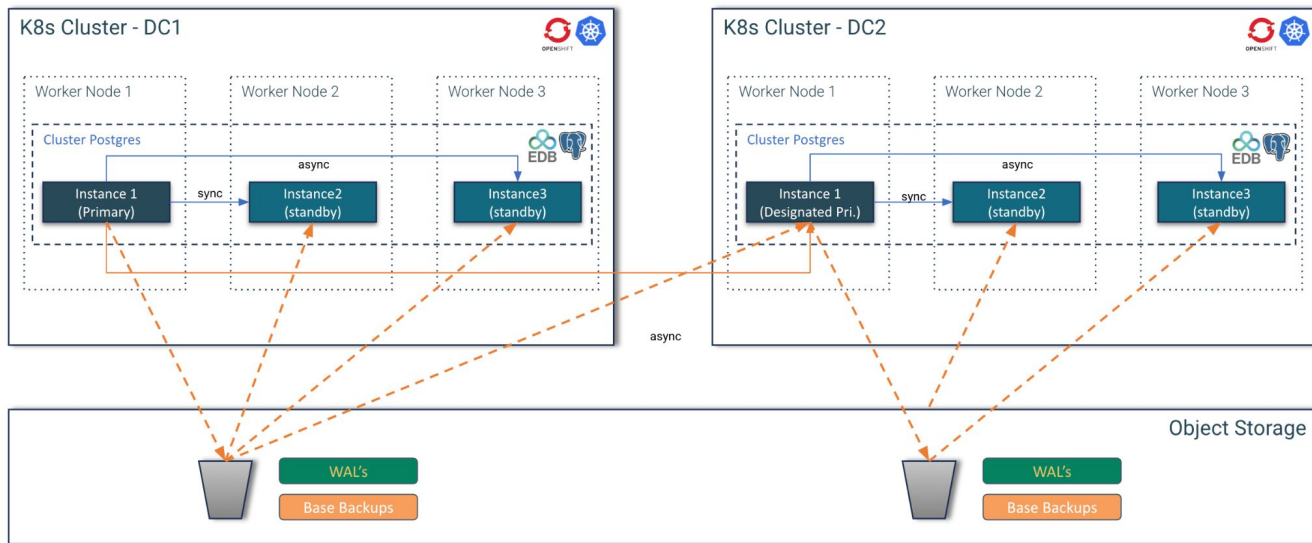
Red Hat Recommendation

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.

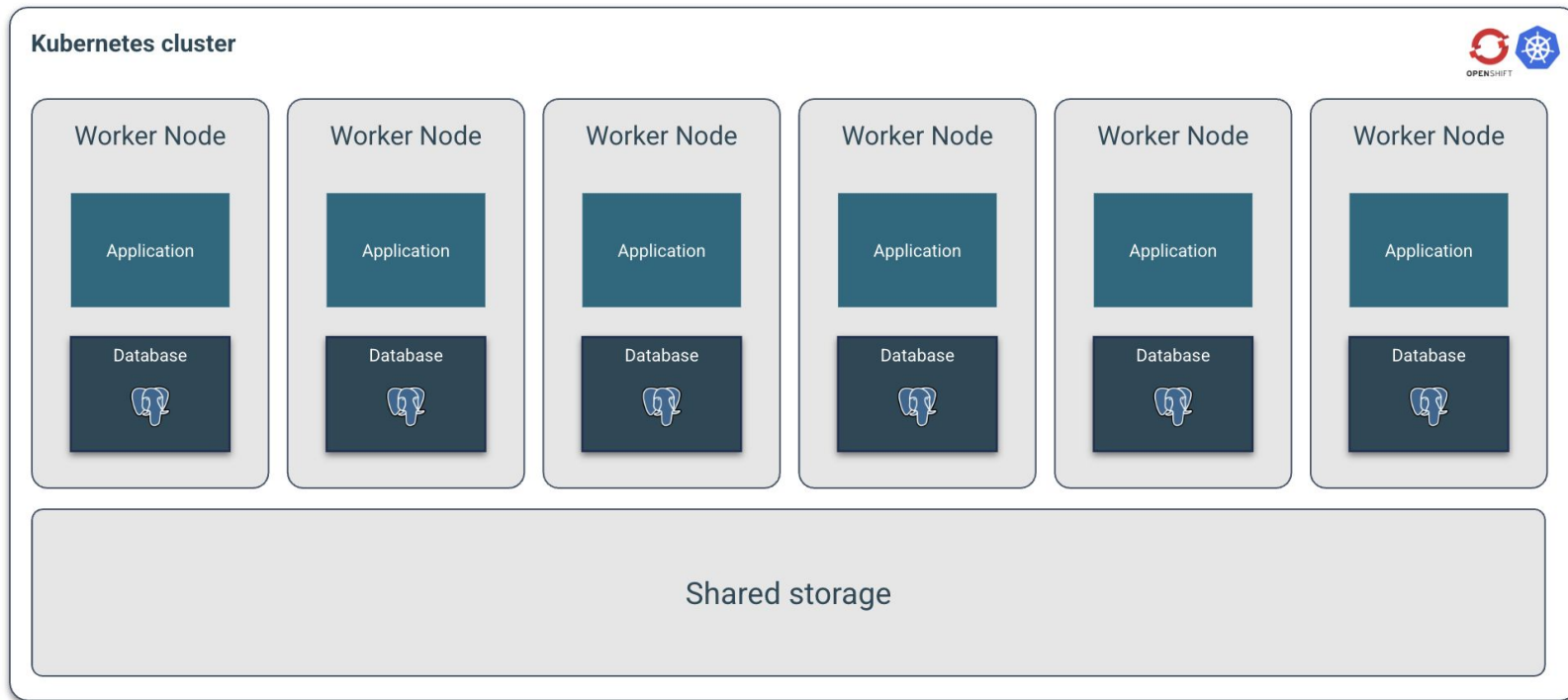


Two separate single data center Kubernetes clusters

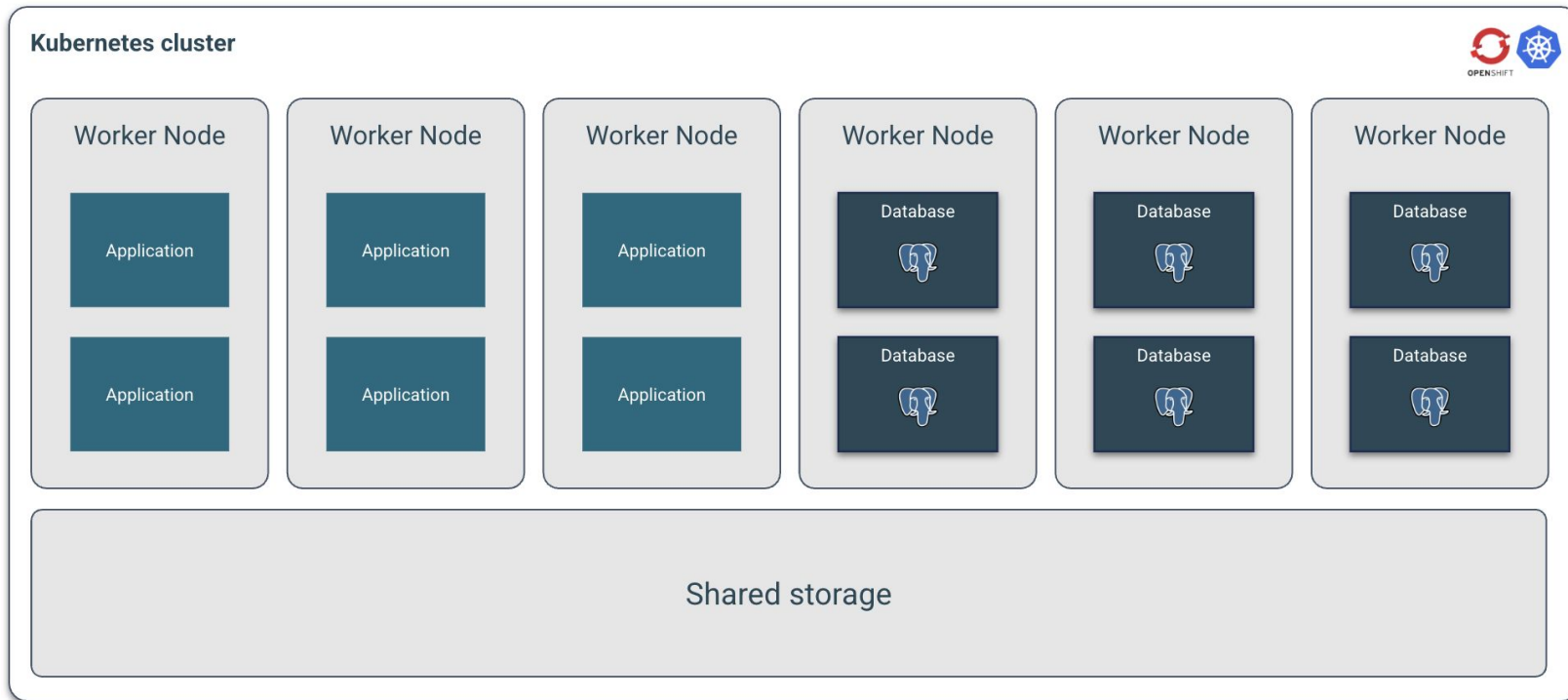
In case you cannot go beyond two data centers and you end up with two separate Kubernetes clusters, don't despair.



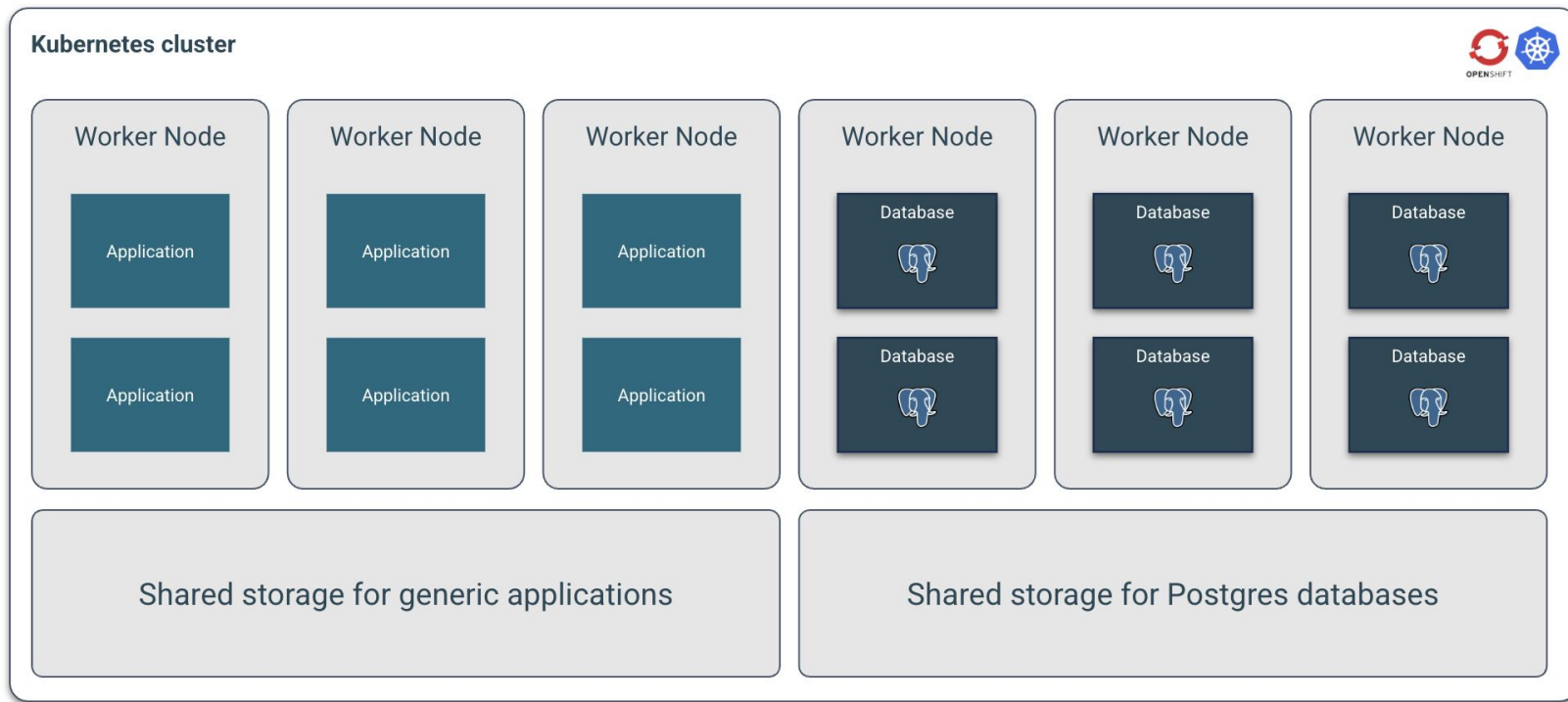
Shared workload, shared storage



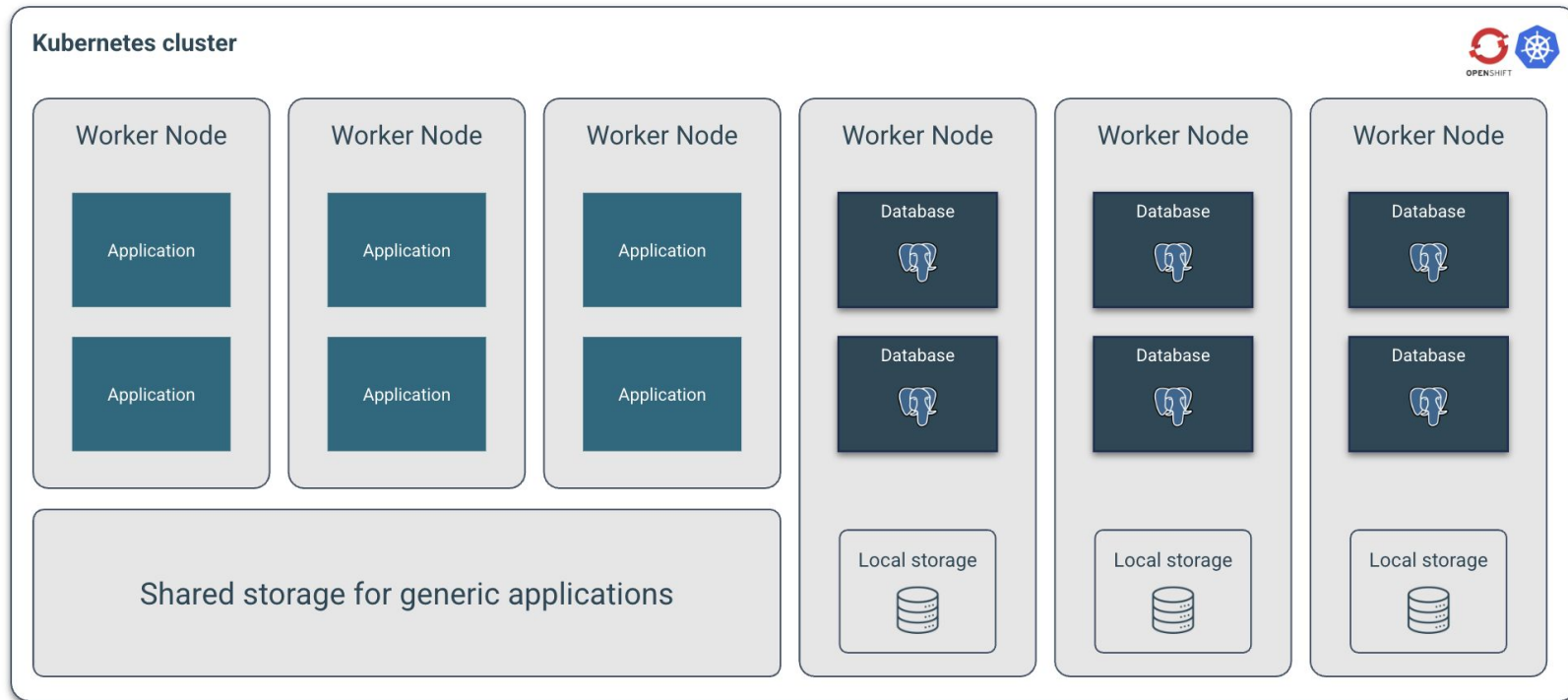
Shared workload, shared storage



Shared workload, shared storage

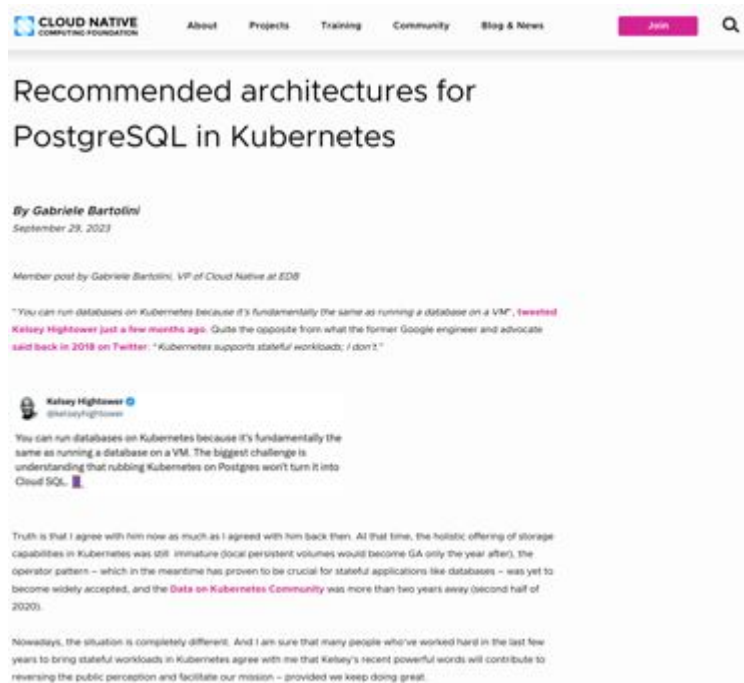


Shared workloads, local storage



Recommended architectures

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



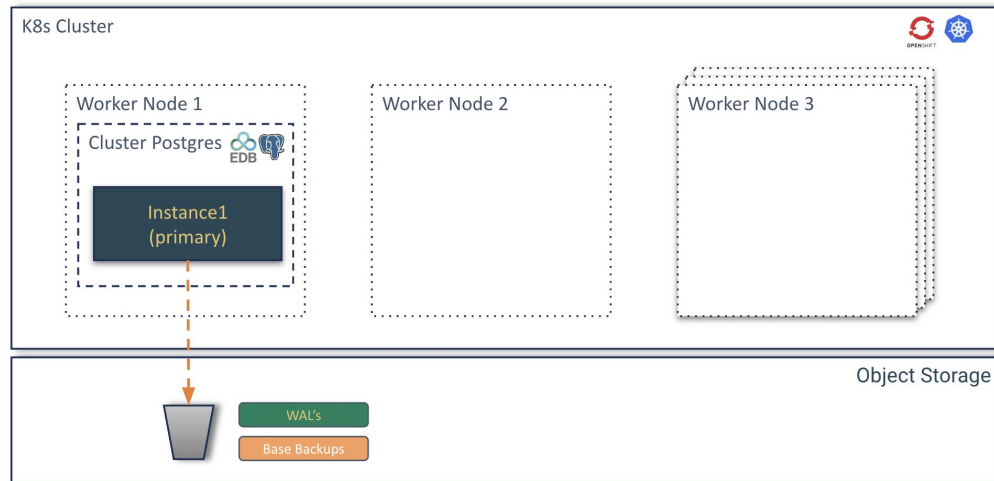
Use Cases



Use case 1 architecture

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

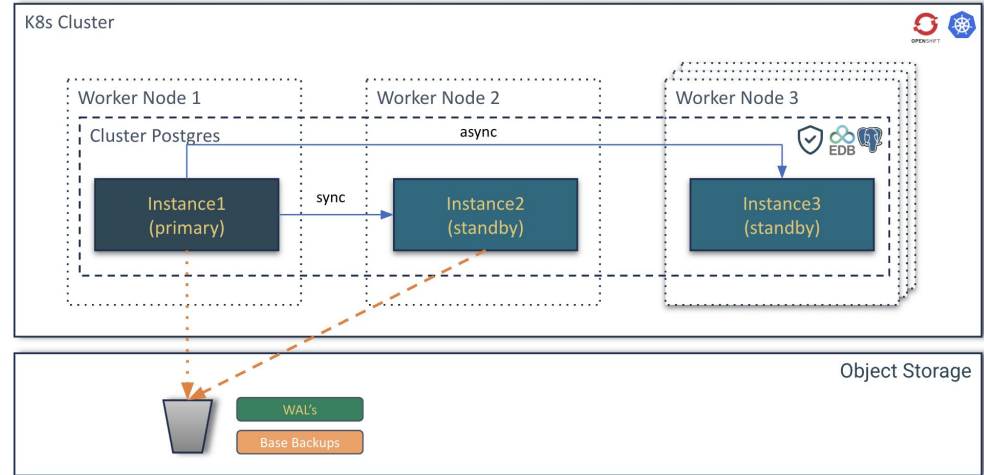
- Development and testing environments
- Small applications with low traffic
- Non-critical data analysis
- Applications with high tolerance for downtime
- Cost-sensitive projects



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.

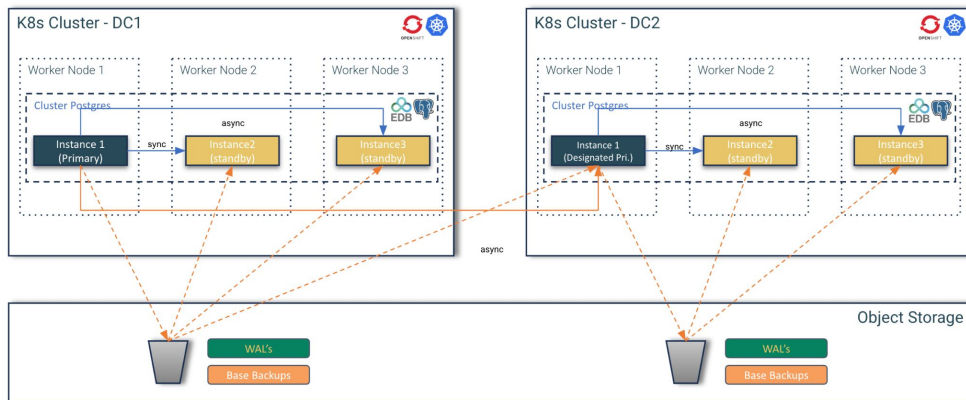
- Business critical Applications
- Applications with stringent SLAs
- Real-time systems
- Improving user experience
- Minimizing planned downtime



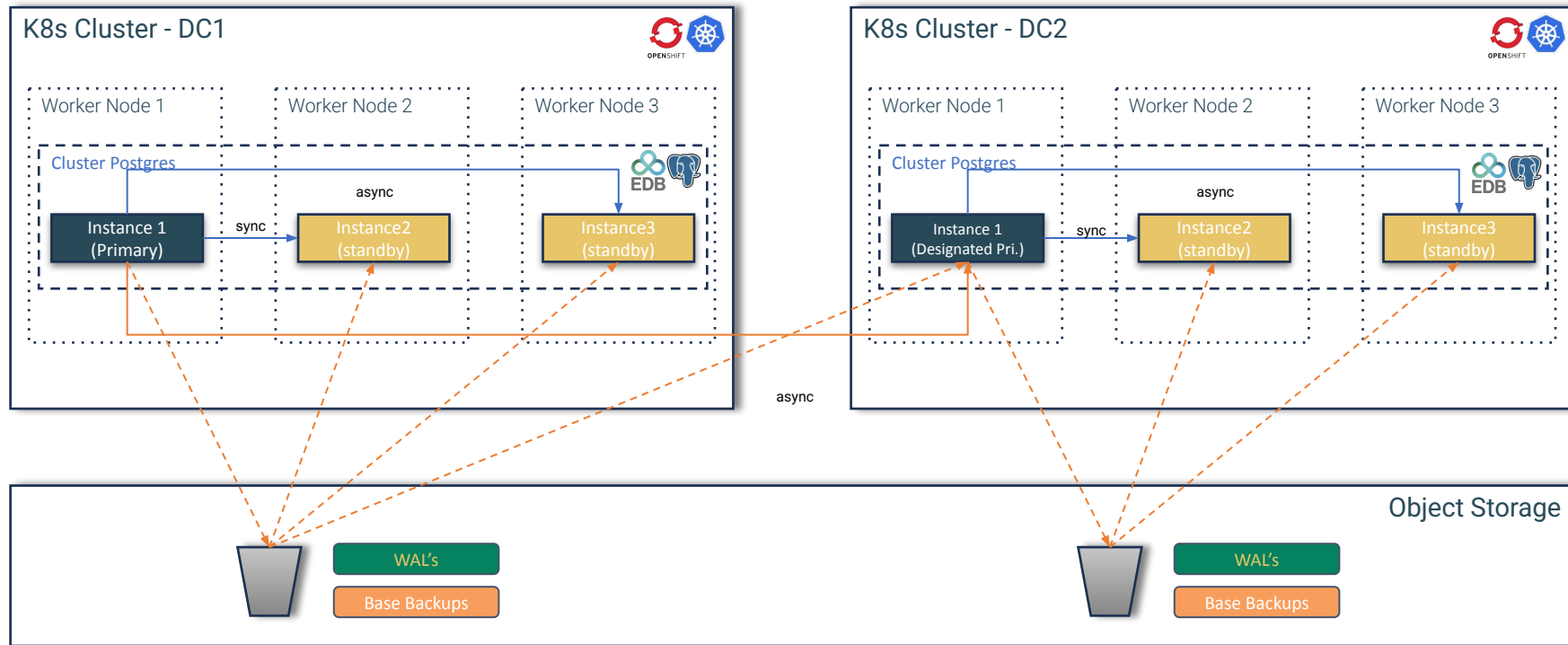
Use case 3 architecture

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-bdb5x.bdb5x.sandbox789.opentlc.com>

Users:

name: user4..user40
password: AqJDpZKUYiQp0eI8

Devspaces url:

<https://devspaces.apps.cluster-bdb5x.bdb5x.sandbox789.opentlc.com/>

URL for devfile (use to create devspace):

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

Minio:

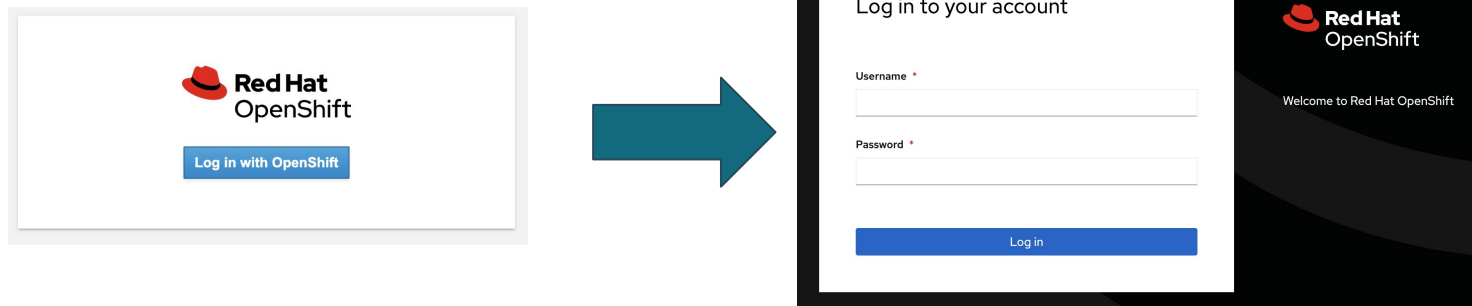
UI: <https://minio-ui-minio.apps.cluster-bdb5x.bdb5x.sandbox789.opentlc.com>
API: <https://minio-api-minio.apps.cluster-bdb5x.bdb5x.sandbox789.opentlc.com>

User: minio
Password: edb-workshop



Open the following URL in your browser:

<https://devspaces.apps.cluster-bdb5x.bdb5x.sandbox789.opentlc.com/>



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



1 item



Postgres on OpenShift
Workshop

CloudNativePG on OpenShift
Workshop

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

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"

Import from Git

Git repo URL *

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



Create & Open

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

In the Import from Git section insert:

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

And select Create & Open

Starting workspace enterisedb-workshop

Progress Logs Events

- ✓ 1 Initializing
- ✓ 2 Checking for the limit of running workspaces
- ✓ 3 Creating a workspace
- 🔄 4 **Waiting for workspace to start**
- 5 Open IDE

Your workshop is loading ...

Get Started with VS Code for the Web

Customize your editor, learn the basics, and start coding

Choose your theme

The right theme helps you focus on your code, is easy on your eyes, and is simply more fun to use.

[Browse Color Themes](#)

Tip: Use keyboard shortcut `Ctrl+K Ctrl+T`

☐ Just the right amount of UI

☐ Rich support for all your languages

[See More Themes...](#)

Dark Modern

Light Modern

Dark High Contrast

Light High Contrast

Select your theme

Do you trust the authors of the files in this workspace?

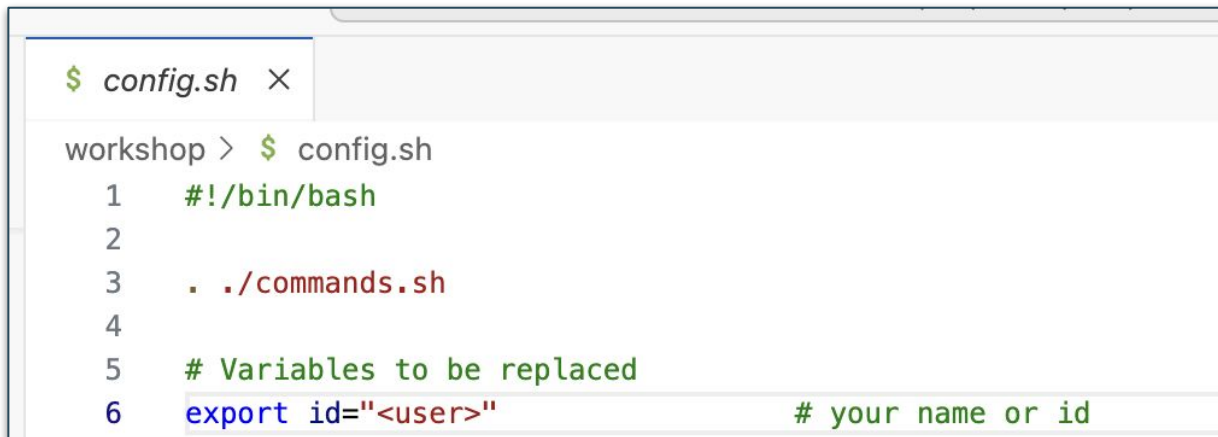
VS Code - Open Source provides features that may automatically execute files in this workspace.

If you don't trust the authors of these files, we recommend to continue in restricted mode as the files may be malicious. See [our docs](#) to learn more.

/projects (Workspace)

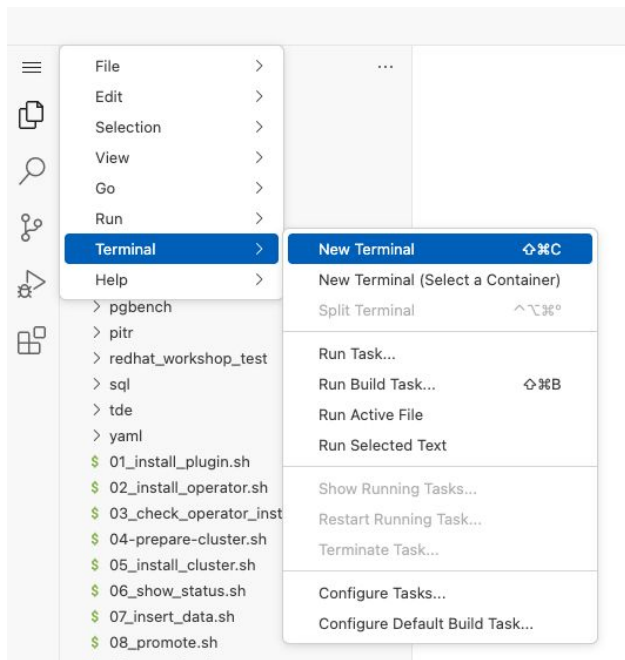
Browse workspace in restricted mode *Trust workspace and enable all features*

And trust the authors

A terminal window with a tab labeled '\$ config.sh'. The prompt is 'workshop >'. The command '\$ config.sh' has been executed, displaying the following script content:

```
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

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

Administration


Backup and
Recovery

High Availability

Monitoring

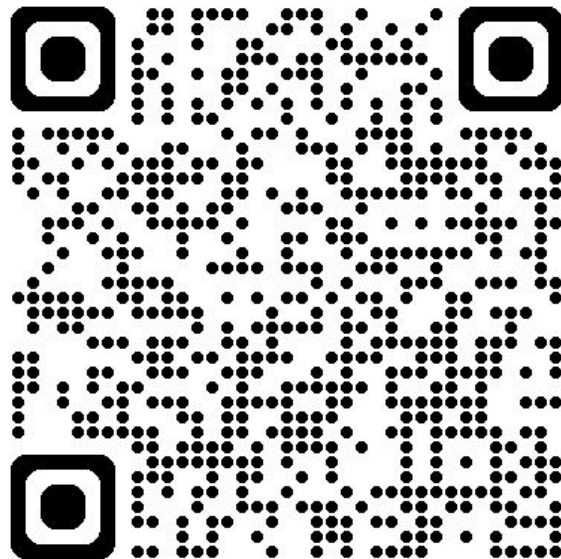
Last CloudNativePG tested version is 1.25



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: 00000004000000000000000C)

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



Operator Installation demonstration

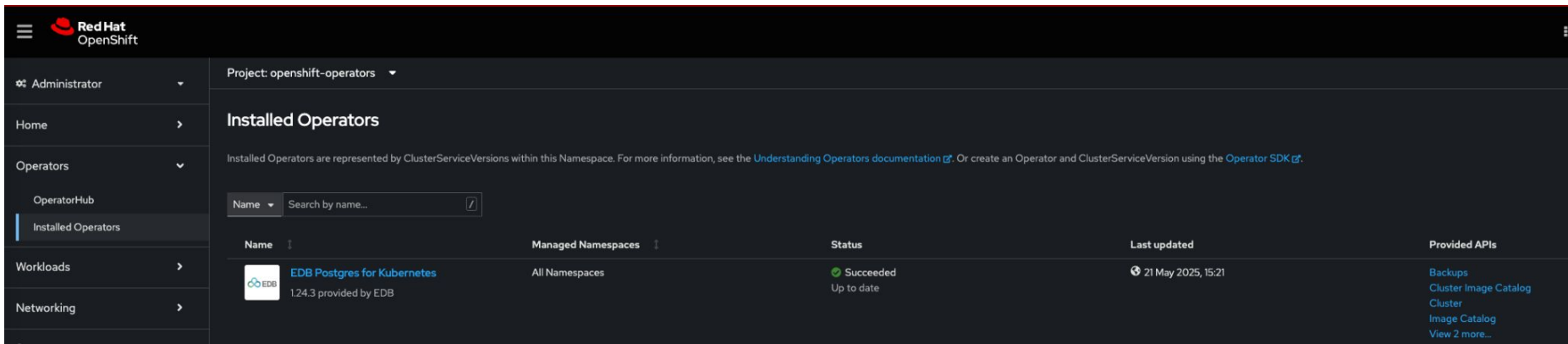
- Check the installed CNP Operator in the console
- Discover the features of the Operator in the OpenShift environment
- Check the installed CNP Operator in the web terminal




NOT NEEDED DURING WORKSHOP
For illustrative purposes.

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:

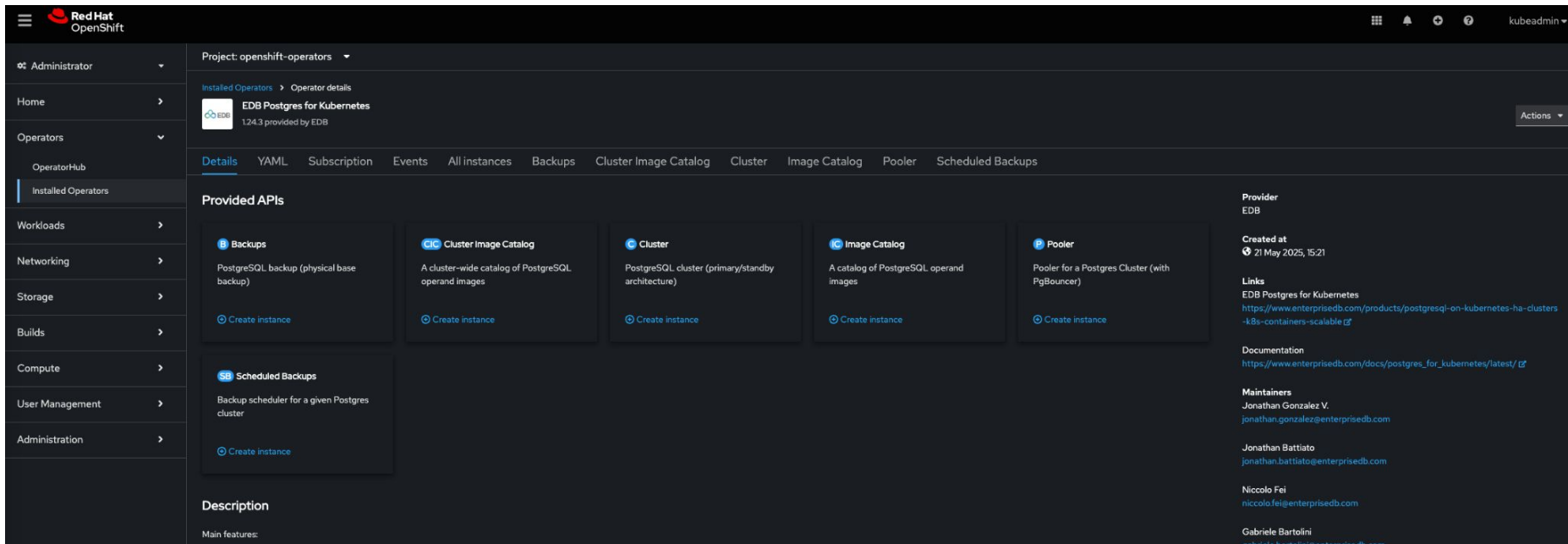


The screenshot shows the Red Hat OpenShift console interface. The left sidebar contains navigation links: Administrator, Home, Operators (selected), OperatorHub, Installed Operators (active), Workloads, and Networking. The main content area is titled 'Installed Operators' and shows a table of installed operators. The table has columns for Name, Managed Namespaces, Status, Last updated, and Provided APIs. One operator is listed: 'EDB Postgres for Kubernetes' (version 1.24.3 provided by EDB), managed in 'All Namespaces', with a status of 'Succeeded' and 'Up to date', last updated on '21 May 2025, 15:21'. The provided APIs include Backups, Cluster Image Catalog, Cluster, Image Catalog, and a link to 'View 2 more...'. Above the table, there is a search bar and a note about ClusterServiceVersions.

Name	Managed Namespaces	Status	Last updated	Provided APIs
 EDB Postgres for Kubernetes 1.24.3 provided by EDB	All Namespaces	✓ Succeeded Up to date	🕒 21 May 2025, 15:21	Backups Cluster Image Catalog Cluster Image Catalog View 2 more...



Discover the features of the Operator in the OpenShift environment



The screenshot displays the OpenShift OperatorHub interface. On the left is a navigation sidebar with links to Administrator, Home, Operators, OperatorHub, Installed Operators, Workloads, Networking, Storage, Builds, Compute, User Management, and Administration. The main content area shows the details for the 'EDB Postgres for Kubernetes' operator, version 1.24.3, provided by EDB. A top navigation bar includes tabs for Details, YAML, Subscription, Events, All instances, Backups, Cluster Image Catalog, Cluster, Image Catalog, Pooler, and Scheduled Backups. The 'Details' tab is active, showing 'Provided APIs' with cards for Backups, Cluster Image Catalog, Cluster, Image Catalog, Pooler, and Scheduled Backups. Each card includes a brief description and a 'Create instance' button. On the right side of the details view, there is a sidebar with information about the provider (EDB), creation date (21 May 2025, 15:21), links to documentation and maintainers, and contact information for Jonathan Battiato, Niccolò Fei, and Gabriele Bartolini.

OperatorHub

Project: openshift-operators

Installed Operators > Operator details

EDB Postgres for Kubernetes
1.24.3 provided by EDB

Actions

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

Provided APIs

API	Description	Create instance
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/

Maintainers
Jonathan Gonzalez V.
jonathan.gonzalez@enterprisedb.com

Jonathan Battiato
jonathan.battiato@enterprisedb.com

Niccolò Fei
niccolo.fei@enterprisedb.com

Gabriele Bartolini
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.

Use case

Create the postgres cluster



Bootstrap - different 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.sh`

- 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	31s
<i>small</i>	20m 38s	16m 45s	27s
<i>medium</i>	2h 42m	2h 34m	48s
<i>large</i>	3h 54m 6s	2h 3s	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 image consists of two screenshots from the OpenShift console. The left screenshot shows the 'Operators' section under 'Installed Operators'. A red arrow points to the 'Operators' link in the left sidebar, and another red arrow points to the 'Backups' tab in the top navigation bar. The 'Backups' tab is selected, showing a list of backup instances. The right screenshot shows the 'Backups' section for the 'EDB Postgres for Kubernetes' operator. A search bar is visible, and a table lists the backup instances. The table has columns for Name, Kind, Namespace, and Status.

Name	Kind	Namespace	Status
backup-test	Backup	NS edb-emea-user1	Phase: co
cluster-user1-backup-test	Backup	NS edb-emea-user1	Phase: co



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:

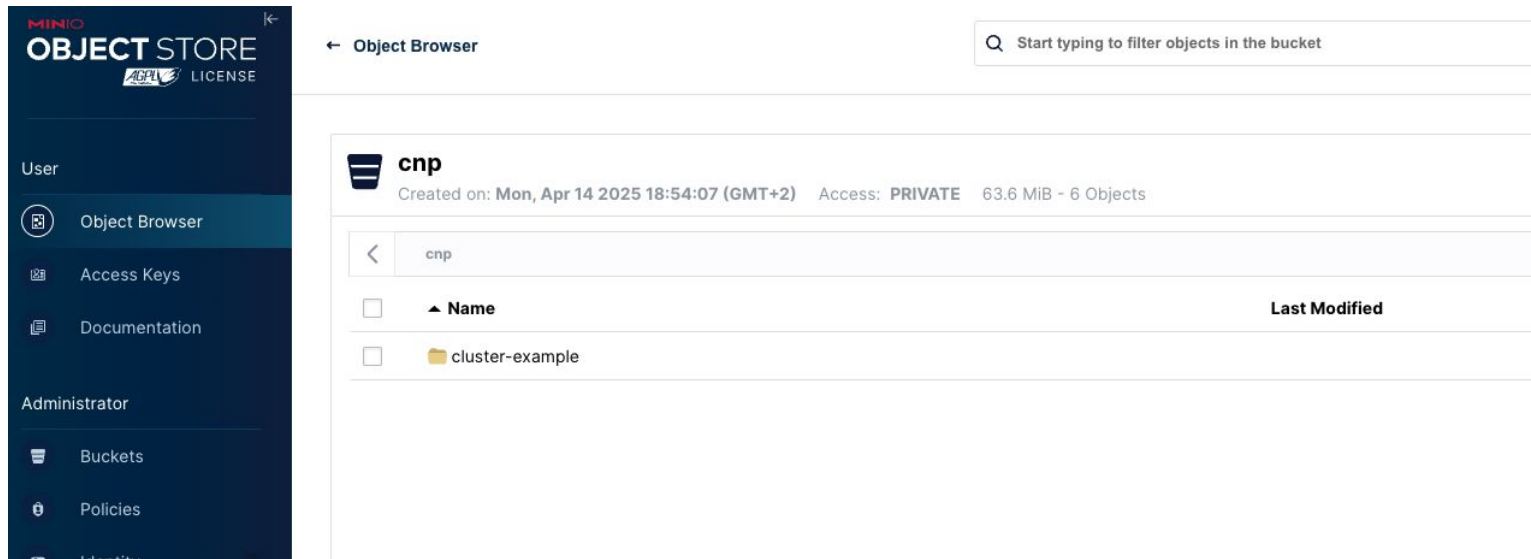
The screenshot shows the OpenShift console interface. On the left sidebar, the 'Networking' menu is expanded, and 'Routes' is selected. Three green circles with arrows indicate the navigation path: 1 points to 'Networking', 2 points to 'Routes', and 3 points to the 'minio-ui' route entry. The route entry is highlighted with a green box and shows the URL 'https://minio-ui-user1.apps.cluster-4cfvh.dynamic.redhatworkshops.io'.

Route	Namespace	Status	URL	Service
minio-api	user20	Accepted	https://minio-api-user20.apps.cluster-4cfvh.dynamic.redhatworkshops.io	minio-service
minio-api	admin	Accepted	https://minio-api-admin.apps.cluster-4cfvh.dynamic.redhatworkshops.io	minio-service
minio-ui	user1	Accepted	https://minio-ui-user1.apps.cluster-4cfvh.dynamic.redhatworkshops.io	minio-service



Check Backup in MinIO UI

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



The screenshot displays the MinIO Object Store interface. On the left is a dark blue sidebar with the 'MINIO OBJECT STORE' logo and 'AGPL LICENSE' text. The sidebar contains sections for 'User' (with 'Object Browser' selected), 'Access Keys', 'Documentation', and 'Administrator' (with 'Buckets' and 'Policies' listed). The main content area is titled 'Object Browser' and features a search bar with the placeholder text 'Start typing to filter objects in the bucket'. Below the search bar, the bucket 'cnp' is shown with details: 'Created on: Mon, Apr 14 2025 18:54:07 (GMT+2)', 'Access: PRIVATE', and '63.6 MiB - 6 Objects'. A table lists the objects in the bucket, with columns for a checkbox, 'Name', and 'Last Modified'. One object is listed: 'cluster-example' with a folder icon.

<input type="checkbox"/>	Name	Last Modified
<input type="checkbox"/>	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 - 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

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)

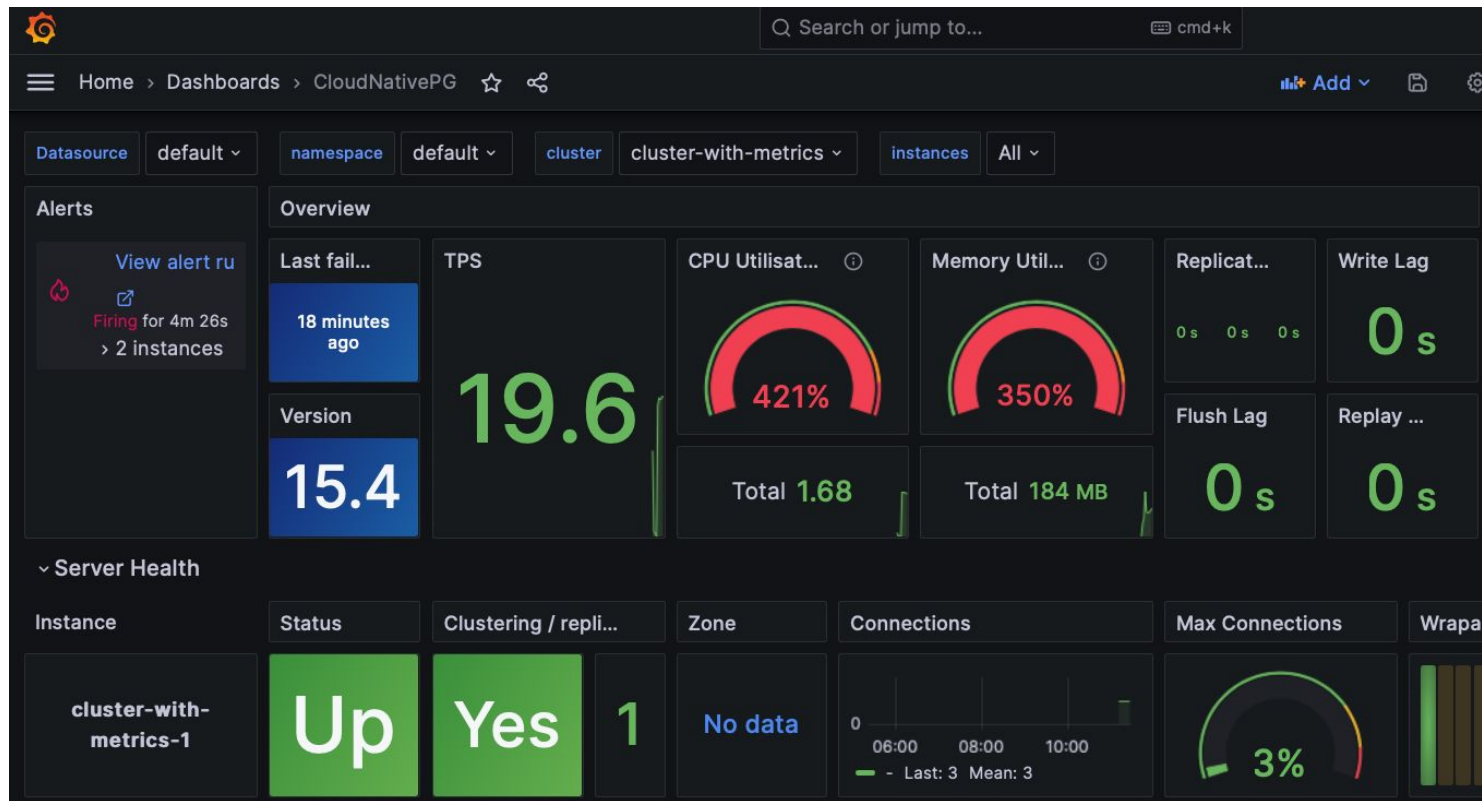


What we didn't show you today

- 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

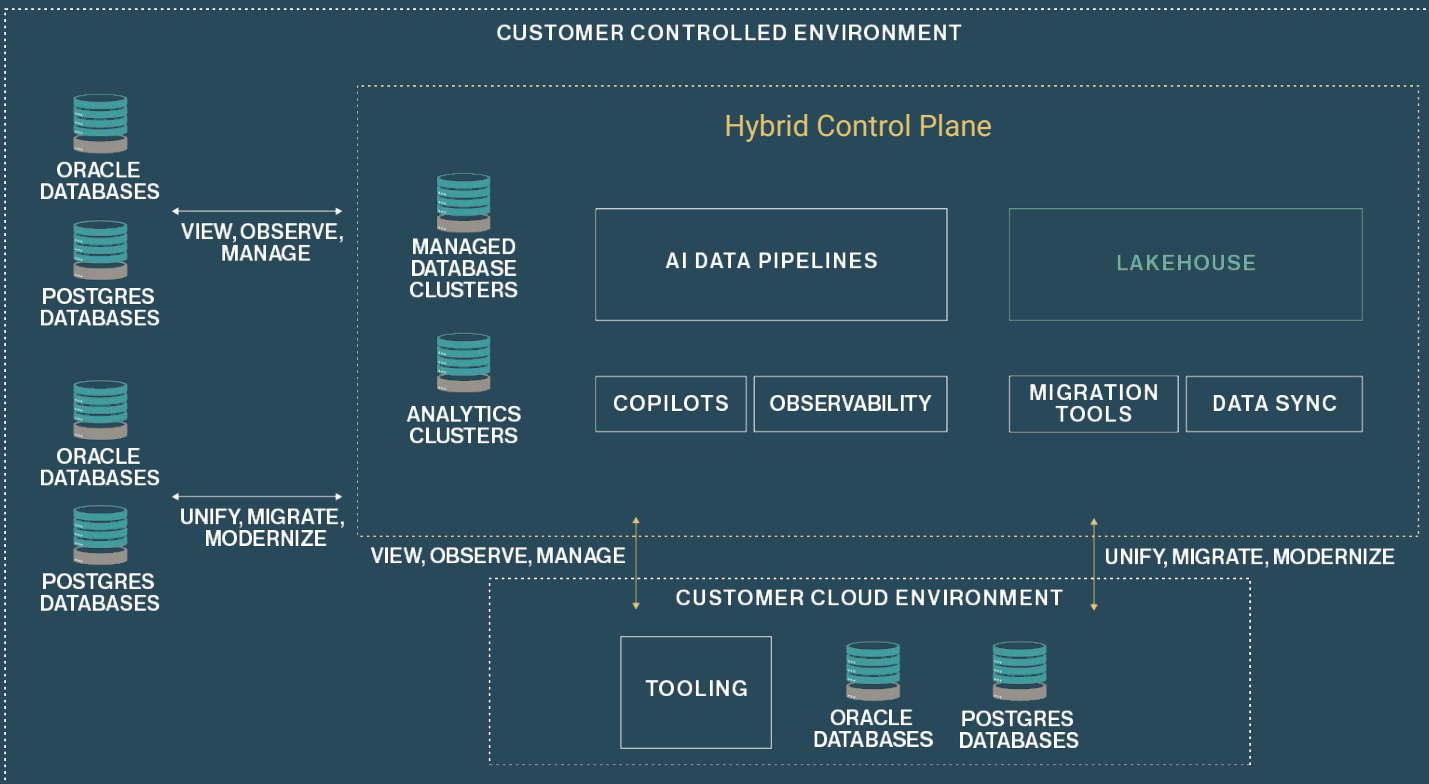


Transparent Data Encryption (EDB-only features)

- 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 Key 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

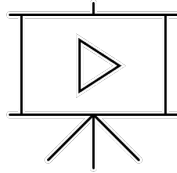


Hybrid Control Plane at a glance



Hybrid Control Plane

LIVE DEMO





Thank you for participating in the
Postgres on Kubernetes Workshop

Please pick up your certificate :-)

