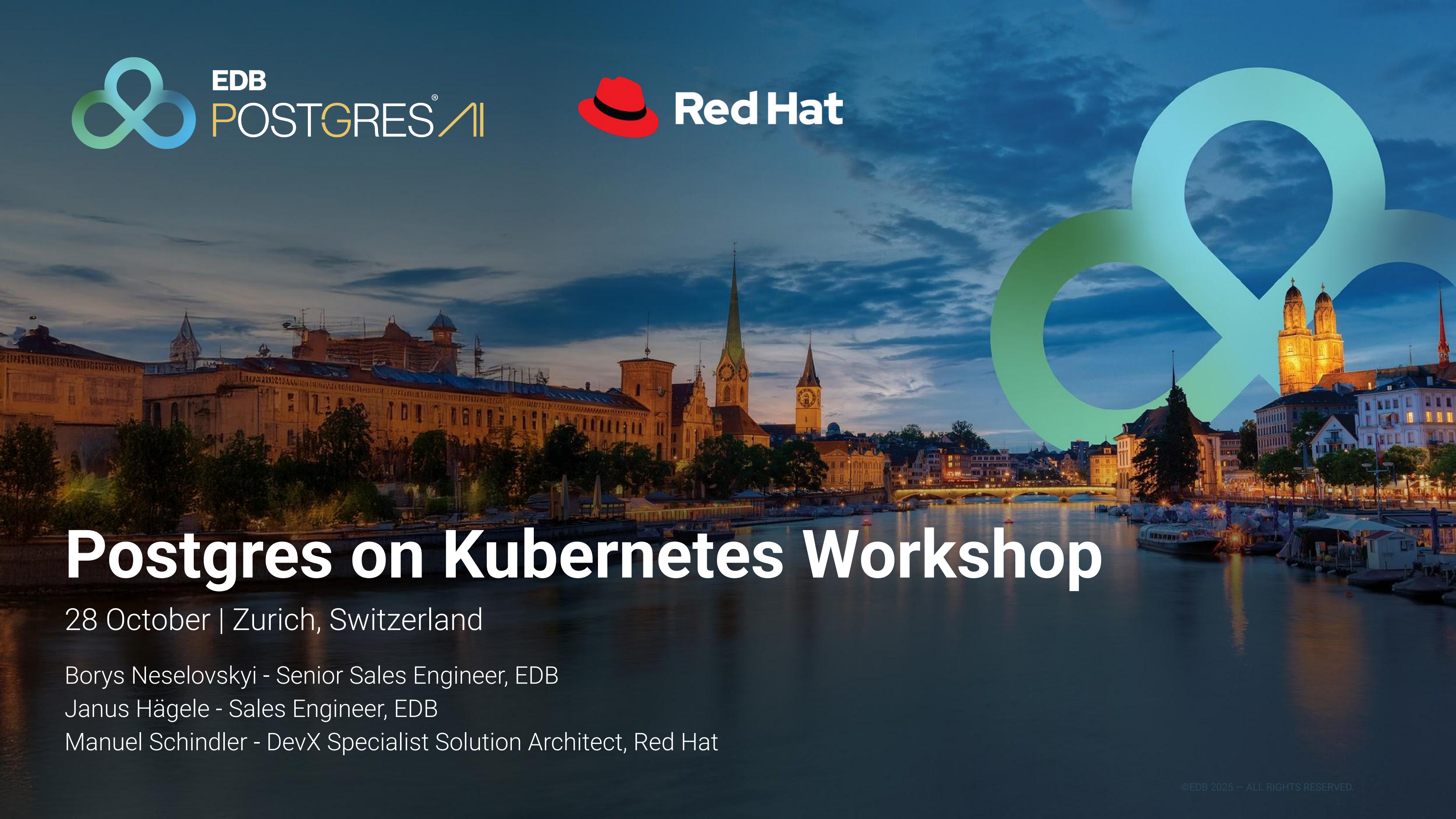


Postgres on Kubernetes Workshop

28 October | Zurich, Switzerland

Borys Neselovskyi - Senior Sales Engineer, EDB
Janus Hägele - Sales Engineer, EDB
Manuel Schindler - DevX Specialist Solution Architect, Red Hat

Agenda

Start	End	Session
13:00	13:30	Registration & Welcome
13:30	13:45	Red Hat OpenShift & EDB Partnership (Red Hat - Manuel Schindler)
13:45	14:00	Introduction to Postgres marketplace and EDB (EDB - Cyrille Sauvain)
14:00	14:30	CNPG Operator Reference Architecture and Functionalities (EDB - Borys Neselovskyi)
14:30	16:30	Interactive session & demo (EDB - Borys Neselovskyi & Janus Hägele)
16:30	17:00	What more?
17:00	18:00	Drinks and pizza

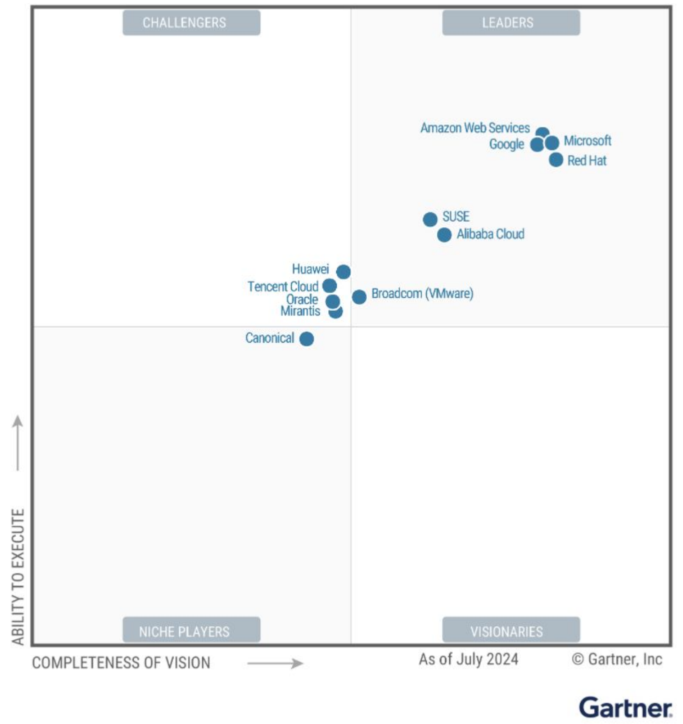
Red Hat Openshift and EDB Partnership

Red Hat OpenShift with EDB

Manuel Schindler
DevX Specialist Solution Architect

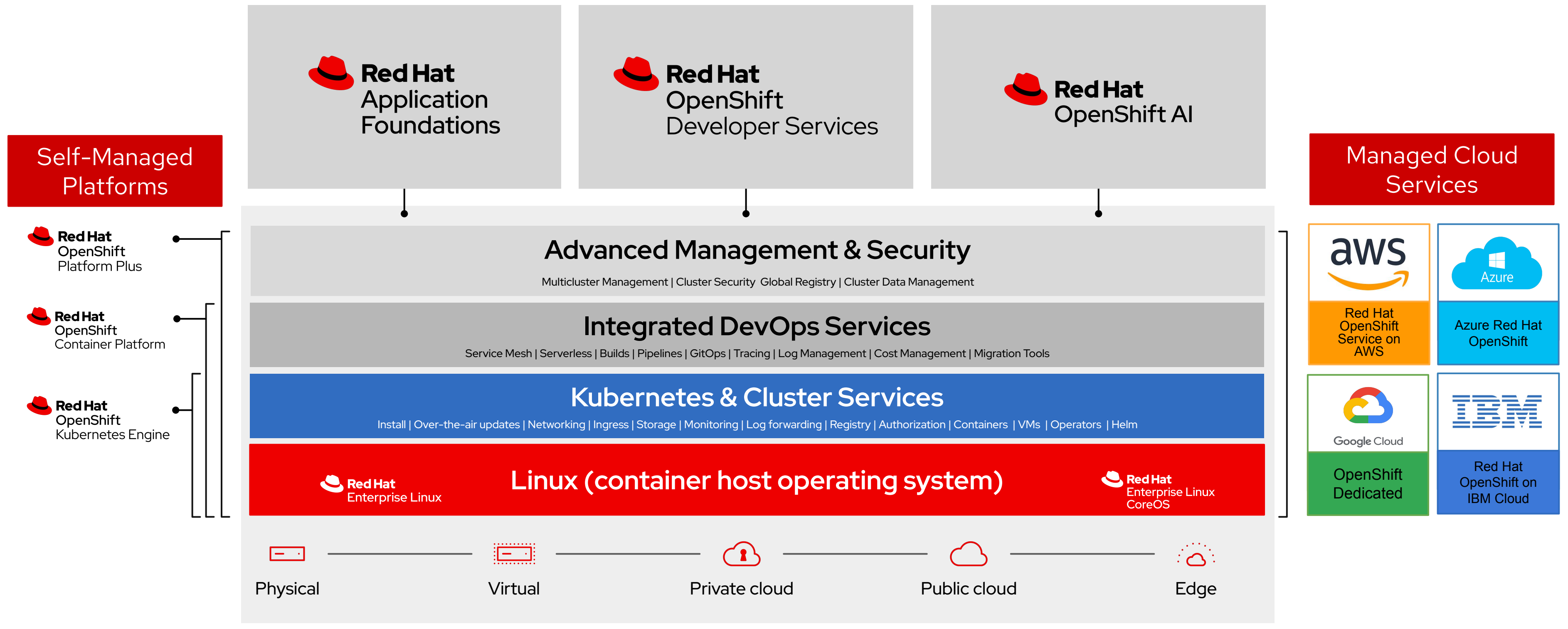
Red Hat is a Leader in the 2024 Gartner® Magic Quadrant™: Container Management

Figure 1: Magic Quadrant for Container Management



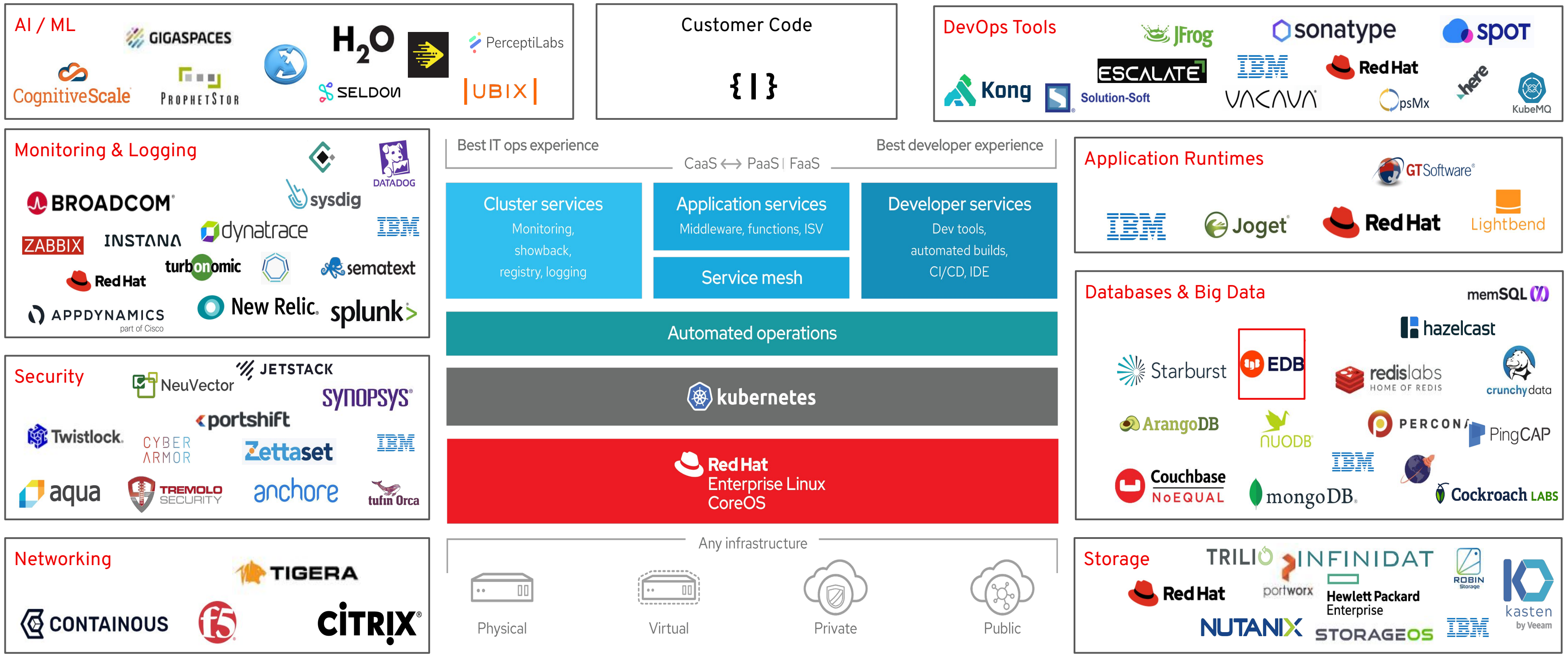
Source: Gartner, "Magic Quadrant for Container Management," September 2024.

Hybrid Cloud Application Platform

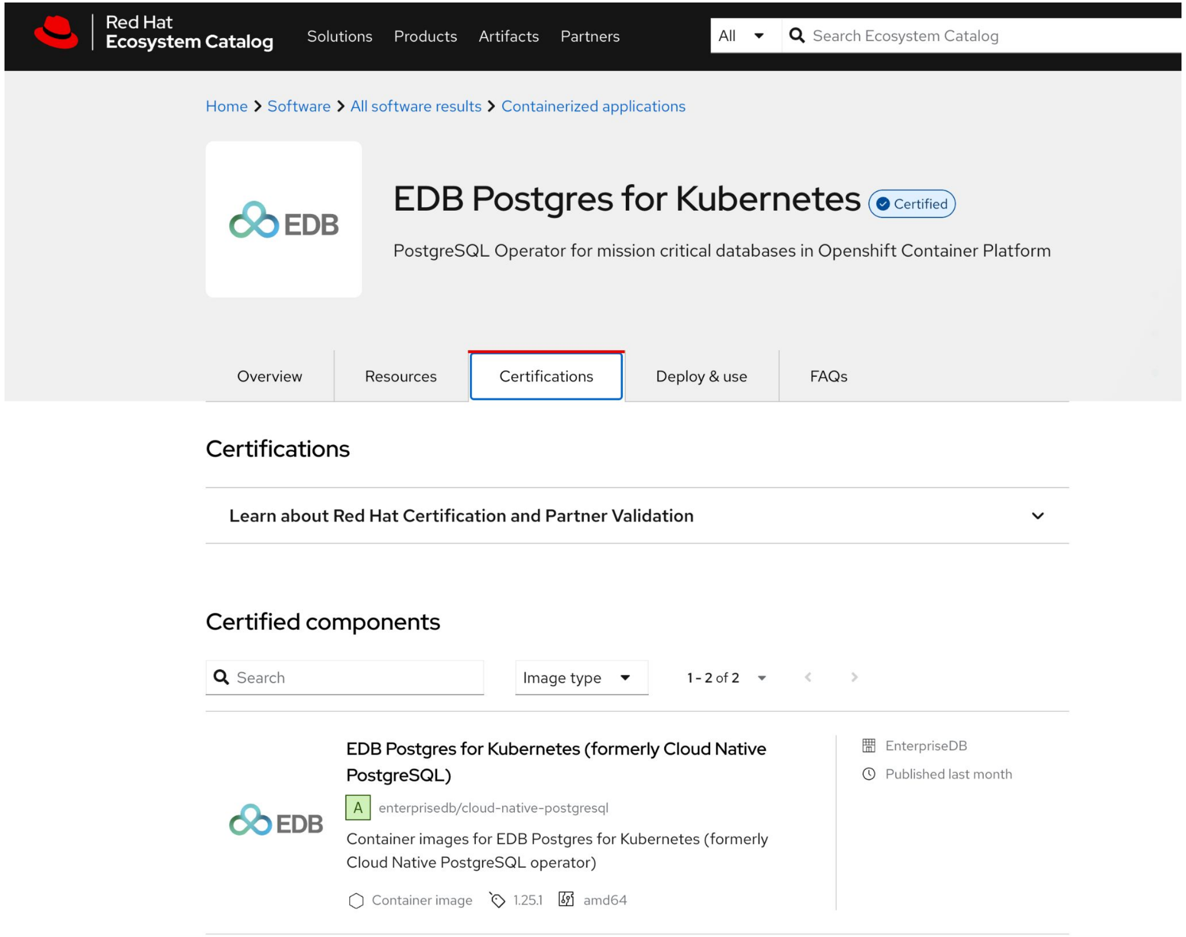

Red Hat Application Foundations
Red Hat OpenShift Developer Services
Red Hat OpenShift AI
Self-Managed Platforms
Red Hat OpenShift Platform Plus
Red Hat OpenShift Container Platform
Red Hat OpenShift Kubernetes Engine
Advanced Management & Security
Multicluster Management | Cluster Security Global Registry | Cluster Data Management
Integrated DevOps Services
Service Mesh | Serverless | Builds | Pipelines | GitOps | Tracing | Log Management | Cost Management | Migration Tools
Kubernetes & Cluster Services
Install | Over-the-air updates | Networking | Ingress | Storage | Monitoring | Log forwarding | Registry | Authorization | Containers | VMs | Operators | Helm
Red Hat Enterprise Linux
Linux (container host operating system)
Red Hat Enterprise Linux CoreOS
Physical
Virtual
Private cloud
Public cloud
Edge
Managed Cloud Services
aws
Red Hat OpenShift Service on AWS
Azure
Azure Red Hat OpenShift
Google Cloud
OpenShift Dedicated
IBM
Red Hat OpenShift on IBM Cloud

Red Hat

Red Hat open hybrid cloud platform with ISV ecosystem



Why Red Hat OpenShift for EDB: operator certification



Home > Software > All software results > Containerized applications

EDB Postgres for Kubernetes

PostgreSQL Operator for mission critical databases in Openshift Container Platform

Certifications

Learn about Red Hat Certification and Partner Validation

Certified components

EDB Postgres for Kubernetes (formerly Cloud Native PostgreSQL)

A enterprisedb/cloud-native-postgresql

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

Container image 1.25.1 amd64

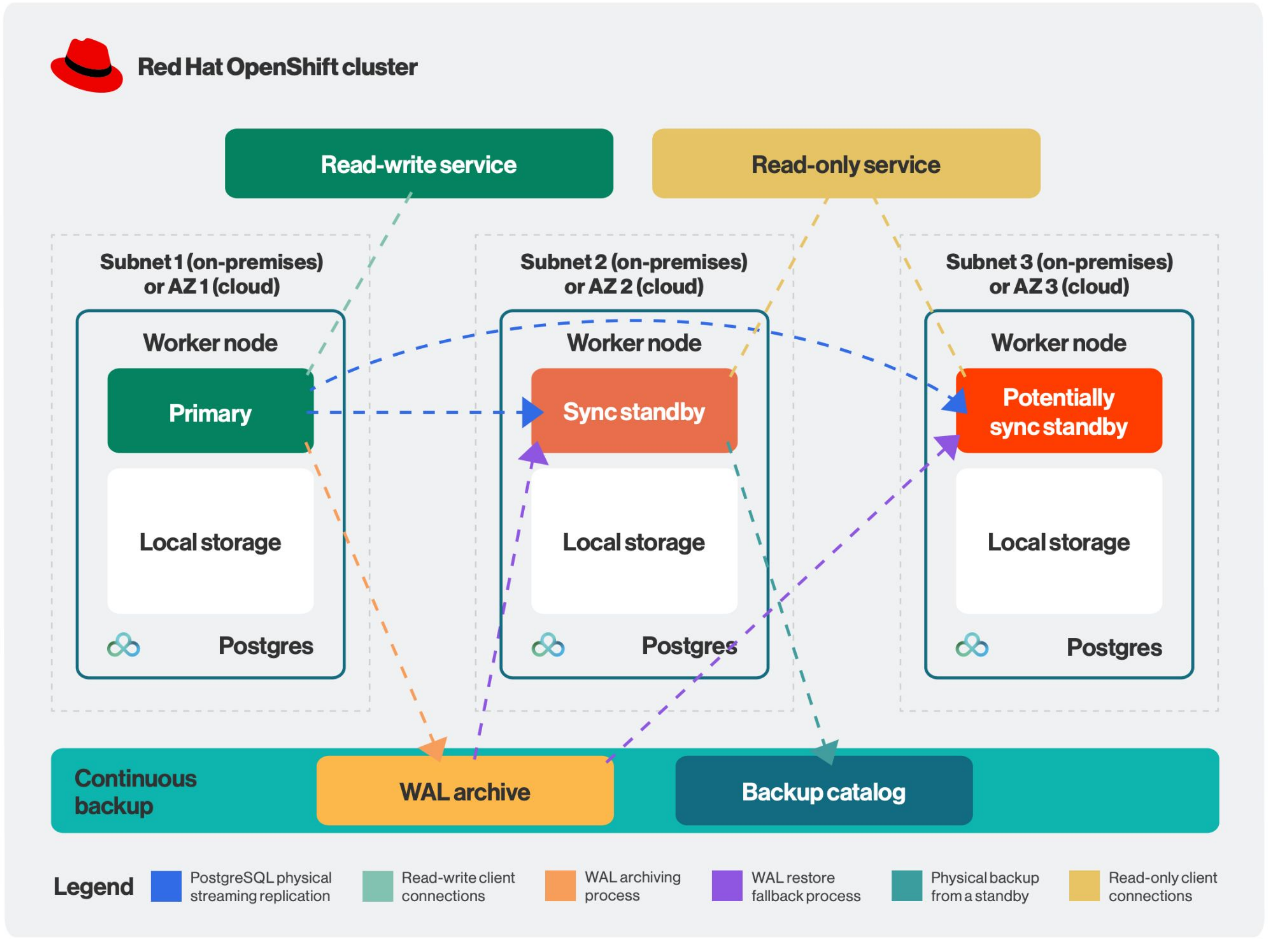
EnterpriseDB

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


Red Hat OpenShift cluster
Read-write service
Read-only service
Subnet 1 (on-premises) or AZ 1 (cloud)
Subnet 2 (on-premises) or AZ 2 (cloud)
Subnet 3 (on-premises) or AZ 3 (cloud)
Worker node
Primary
Sync standby
Potentially sync standby
Local storage
Postgres
Continuous backup
WAL archive
Backup catalog
Legend
PostgreSQL physical streaming replication
Read-write client connections
WAL archiving process
WAL restore fallback process
Physical backup from a standby
Read-only client connections

Red Hat

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

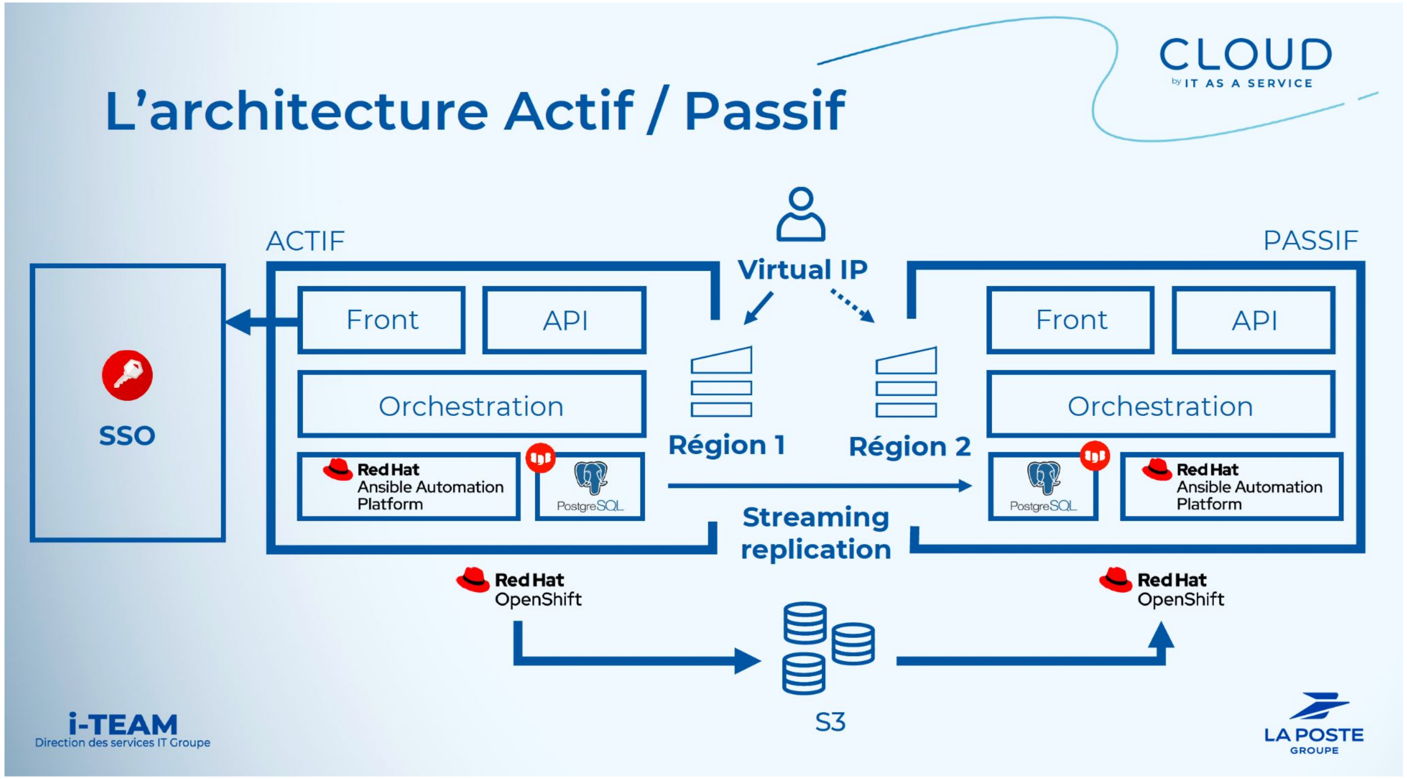
Solution

- 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.

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

Red Hat

Introduction to CloudNativePG and EDB

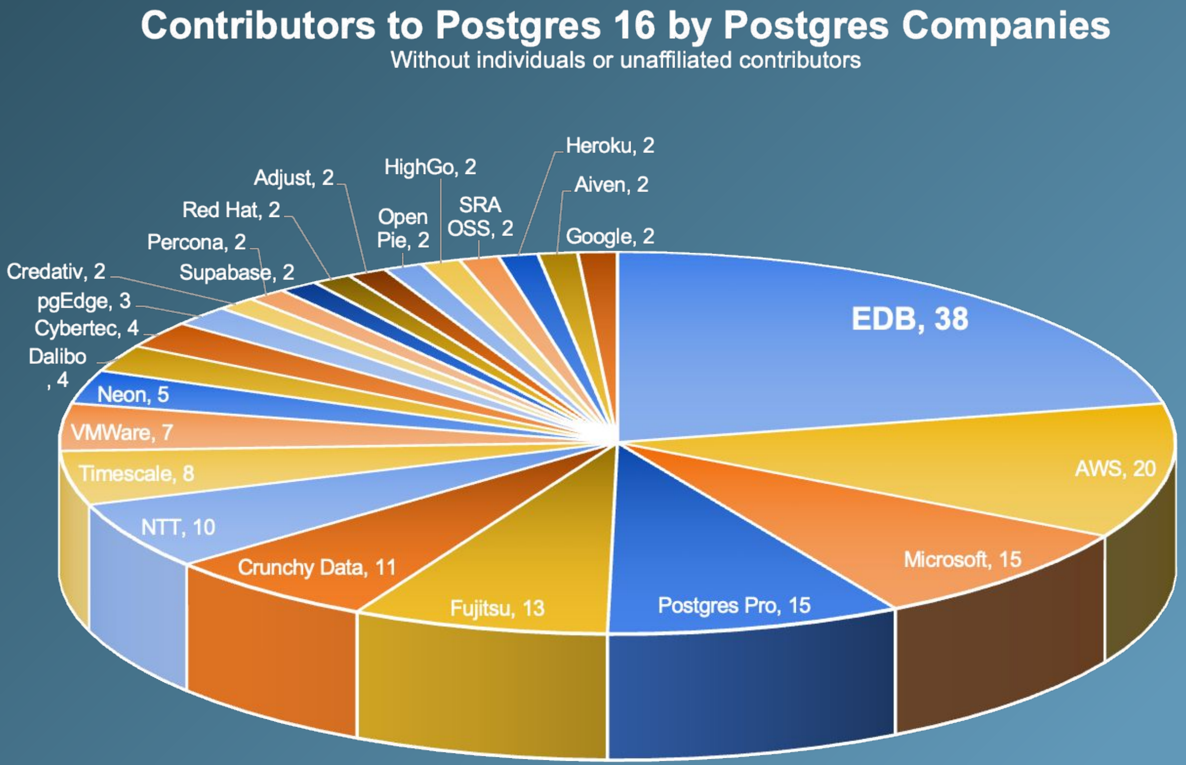
EDB
Postgres® for the AI Generation

20+ years of Postgres innovation & adoption

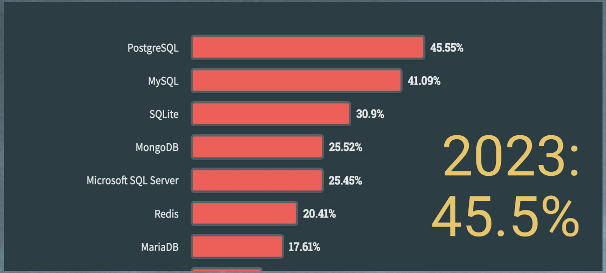
- Number one contributor to Postgres, fastest-growing and most loved Database in the world
  - 2 Core Team members, 7 Committers, 9 Major Contributors, 10 Contributors, #1 site for desktop downloads
- Over 700 employees in more than 30 countries
- EDB Postgres AI
  - The industry's first platform that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

Large Developer Community

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

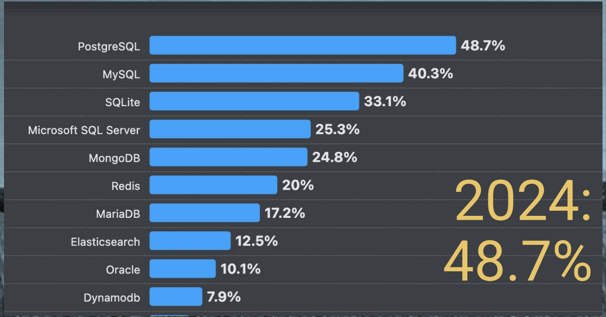


Postgres has won the database race



PostgreSQL	45.55%
MySQL	41.09%
SQLite	30.9%
MongoDB	25.52%
Microsoft SQL Server	25.45%
Redis	20.41%
MariaDB	17.61%

2023: 45.5%



PostgreSQL	48.7%
MySQL	40.3%
SQLite	33.1%
Microsoft SQL Server	25.3%
MongoDB	24.8%
Redis	20%
MariaDB	17.2%
Elasticsearch	12.5%
Oracle	10.1%
Dynamodb	7.9%

2024: 48.7%

Stack Overflow Survey 2023/2024

LEADING ENTERPRISES TRUST EDB

BANKING FINANCIAL

ABN·AMRO
BBVA
AON
Accertify
AN AMERICAN EXPRESS COMPANY
AMERICAN EXPRESS
mastercard
London Stock Exchange
Santander
ANZ
IAG Insurance Australia Group
RBC
AAA

TECHNOLOGY

SONY
tomtom
Braintree
A PayPal Service
ENTRUST
RSA
hp
Alibaba.com
SAMSUNG
Postmates
IBM
NOKIA
DELL EMC

TELCO

AT&T
kt
OPTUS
T Systems
VONAGE
Nokia Siemens Networks
Telefónica
vodafone
Telstra
verizon

OUR PLATFORM

EDB POSTGRES® AI

DATABASE
- ENTERPRISE POSTGRES
- ORACLE COMPATIBILITY
- COMMUNITY POSTGRESQL
- MULTI-MODEL EXTENSIONS
- MGMT. & OBSERVABILITY
- KUBERNETES OPERATORS
- SUPPLY CHAIN SECURITY
- MIGRATION TOOLS
- HIGH AVAILABILITY

HYBRID MANAGEMENT
- HYBRID OBSERVABILITY
- HYBRID DBAAS
- HYBRID MULTI-MODAL DATA
- PERFORMANCE MGMT
- MIGRATION MGMT

AI FACTORY
- VECTOR ENGINE
- AI PIPELINE
- GENAI BUILDER
- AGENT STUDIO
- MODEL SERVING

ANALYTICS ACCELERATOR
- ANALYTICS ENGINE
- LAKEHOUSE CONNECTOR
- GREENPLUM WORKLOADS

DEPLOY ANYWHERE	ENGINEERED SYSTEM	HYBRID SOFTWARE	MANAGED PLATFORM
SOVEREIGN ASSURANCE	MANAGED SERVICES	PILOT TO PRODUCTION SLAS	OUTCOME EXECUTION

DELIVERED WITH WORLD-CLASS STRATEGIC PARTNERS:

aws Google Cloud Azure IBM Red Hat NVIDIA NUTANIX SUPERMICRO Prolifics

CNPG Operator: Reference Architecture and functionalities

EDB
Postgres® for the AI Generation

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

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

CNPG 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

EDB CloudNativePG

Autopilot

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

EDB CloudNativePG

Security

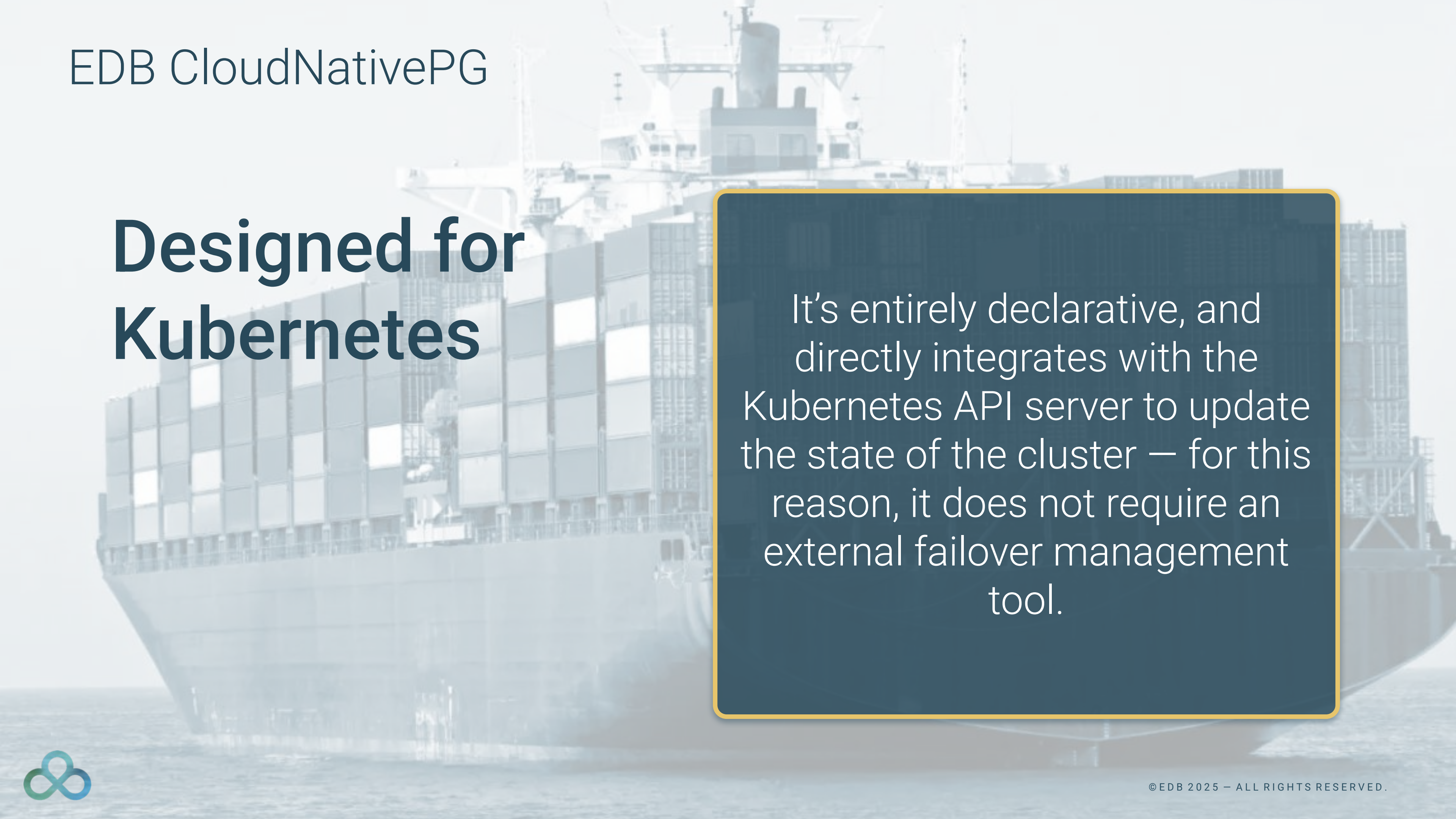
CloudNativePG is secured by default.

EDB CloudNativePG

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

Data persistence

EDB CloudNativePG

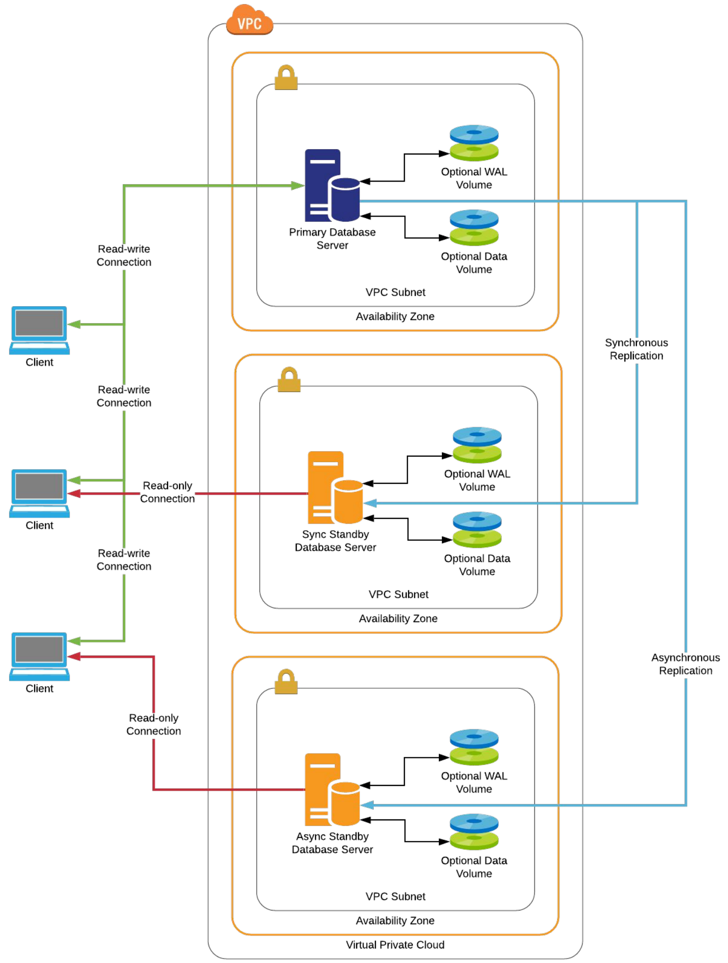


Designed for Kubernetes

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.

Imperative vs Declarative

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



Convention over configuration

Declarative - simple to install, simple to maintain

There's a PostgreSQL 17 cluster with 2 replicas:

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: myapp-db
spec:
  instances: 3
  imageName: quay.io/enterprisedb/postgresql:17

  storage:
    size: 10Gi
```

Features

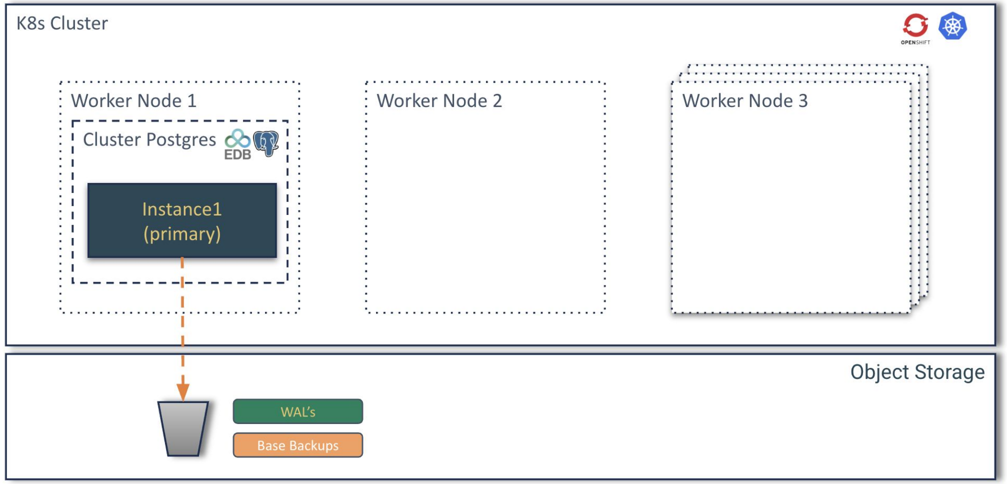
Deployment	Administration	Backup & Recovery	Monitoring	Security	High Availability
Kubernetes operator	Single node	Backup	Prometheus	TDE	Switchover
Kubernetes plugin	Cluster (Multi node)	Recovery	Grafana dashboards	Certificates	Failover
EDB Postgres (EPAS)	PostgreSQL configuration	PITR	Postgres Enterprise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

Use Cases

Use case 1 architecture

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

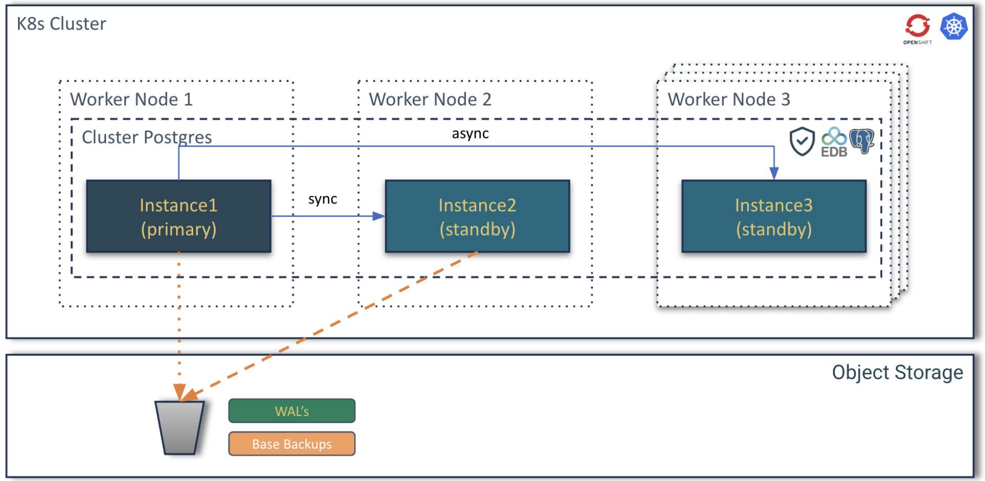
- Development and testing environments
- Small applications with low traffic
- Non-critical data analysis
- Applications with high tolerance for downtime
- Cost-sensitive projects



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.

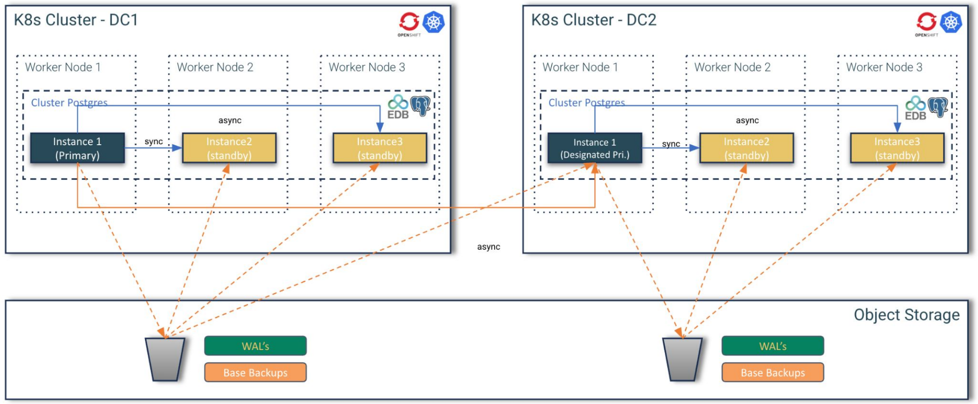
- Business critical Applications
- Applications with stringent SLAs
- Real-time systems
- Improving user experience
- Minimizing planned downtime



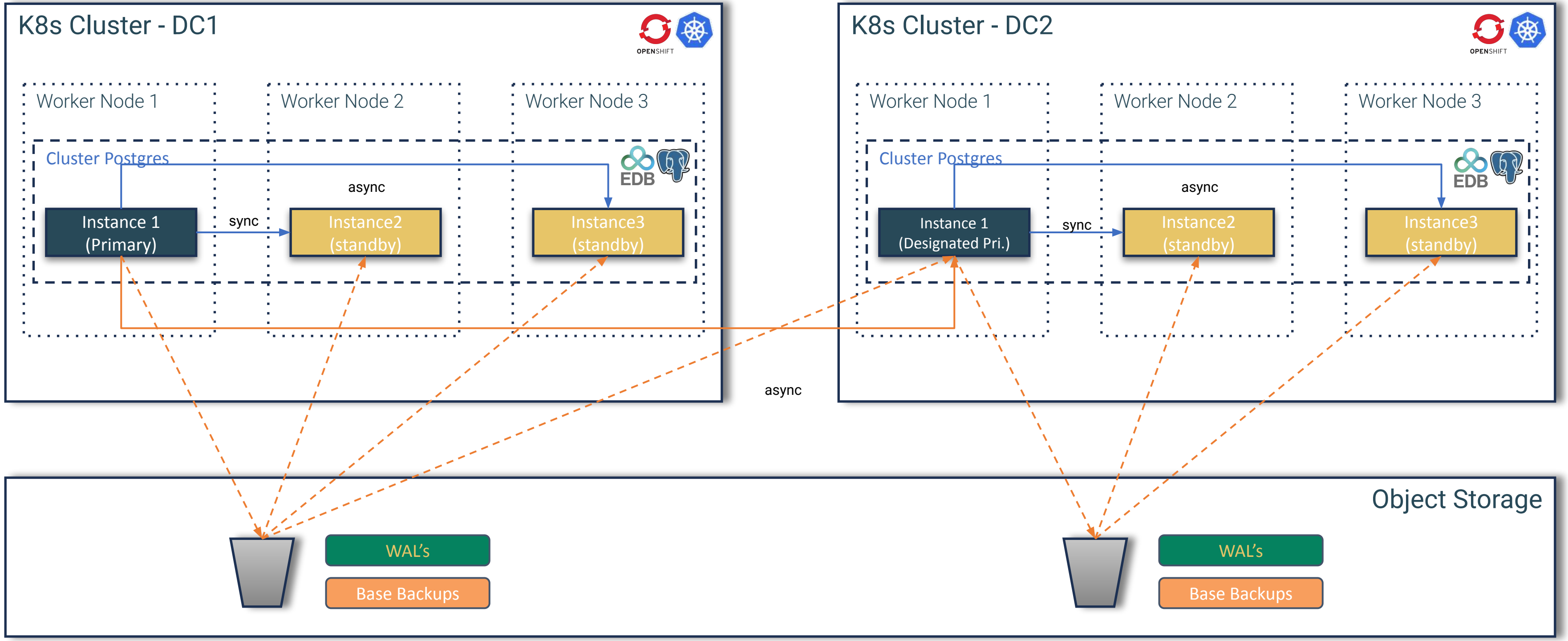
Use case 3 architecture

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://tinyurl.com/3j7cbjh3

Links:

Openshift Console:
https://console-openshift-console.apps.cluster-m6pll.m6pll.sandbox3121.opentlc.com

Users:

name:	user2..user40
Password:	edb-workshop

Devspaces url:
https://devspaces.apps.cluster-m6pll.m6pll.sandbox3121.opentlc.com/

Short url to Devspaces:
https://tinyurl.com/yy9dswmk

Minio:

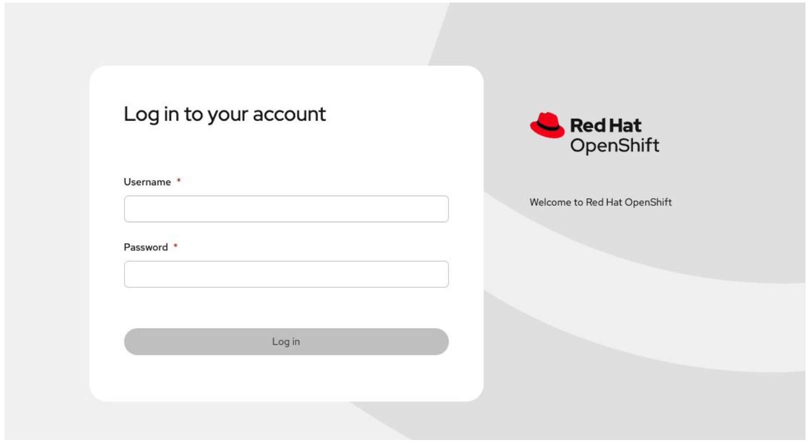
UI:	https://minio-ui-default.apps.cluster-m6pll.m6pll.sandbox3121.opentlc.com
API:	https://minio-api-default.apps.cluster-m6pll.m6pll.sandbox3121.opentlc.com

User:	minio
Password:	edb-workshop

Call to action: Open the Openshift Console

Open the following URL in your browser:

https://tinyurl.com/yy9dswmk


Log in to your account

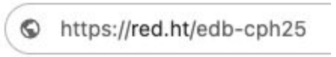
Username *

Password *

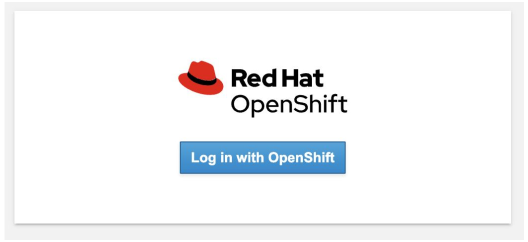
Username and password provided to you

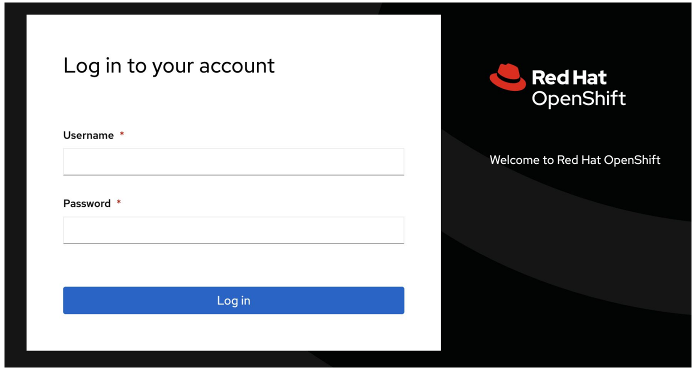
Call to action: Open the DevSpace

Open the following URL in your browser:


https://red.ht/edb-cph25

https://tinyurl.com/yy9dswmk




Log in to your account

Username *

Password *

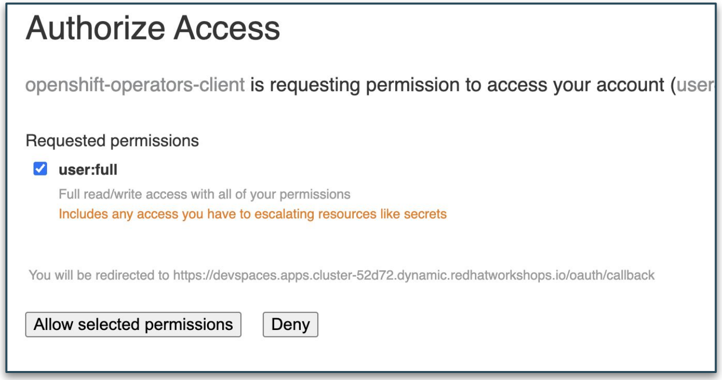
Log in

Red Hat OpenShift

Welcome to Red Hat OpenShift

Username and password provided to you

Call to action: Open the DevSpace



Authorize Access

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

Requested permissions

✓ **user:full**

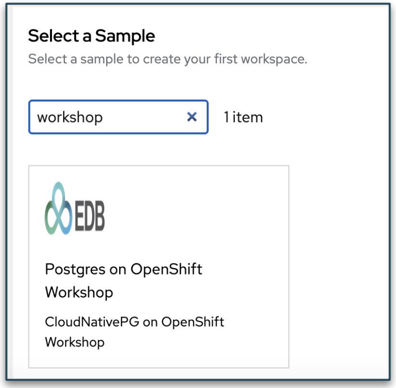
Full read/write access with all of your permissions

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.

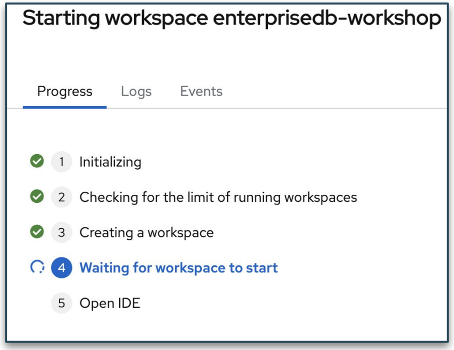
workshop   1 item

Postgres on OpenShift Workshop

CloudNativePG on OpenShift Workshop

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

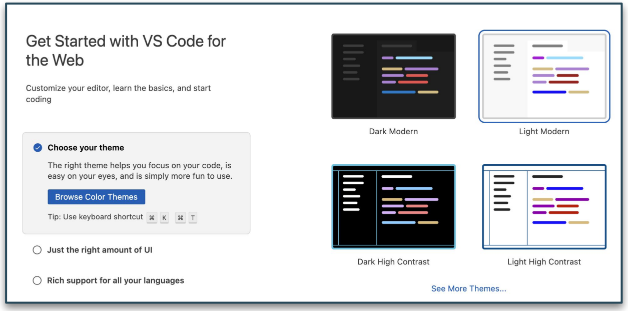
Call to action: Open the DevSpace


Starting workspace enterprisedb-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.

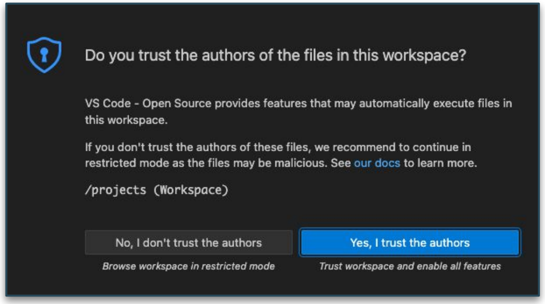
Browse Color Themes

Tip: Use keyboard shortcut ⌘ K ⌘ T

Just the right amount of UI

Rich support for all your languages

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.

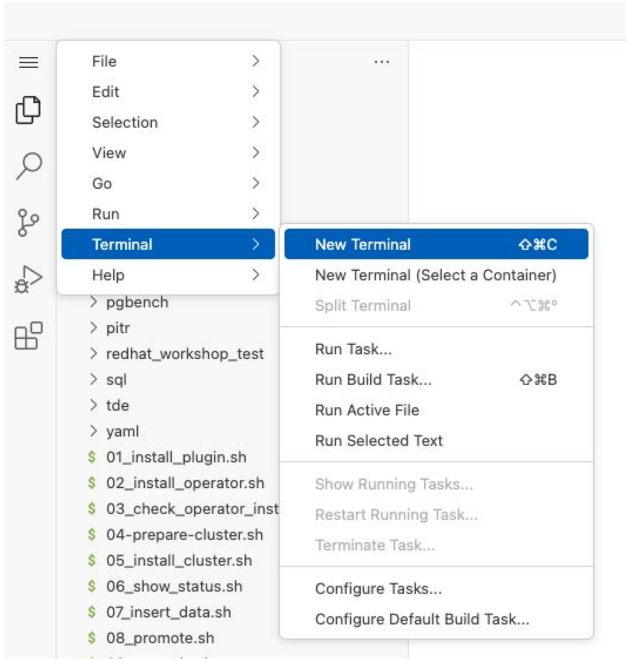
/projects (Workspace)

No, I don't trust the authors — *Browse workspace in restricted mode*

Yes, I trust the authors — *Trust workspace and enable all features*

And trust the authors

Call to action: Open the new terminal window



Open two terminal windows

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

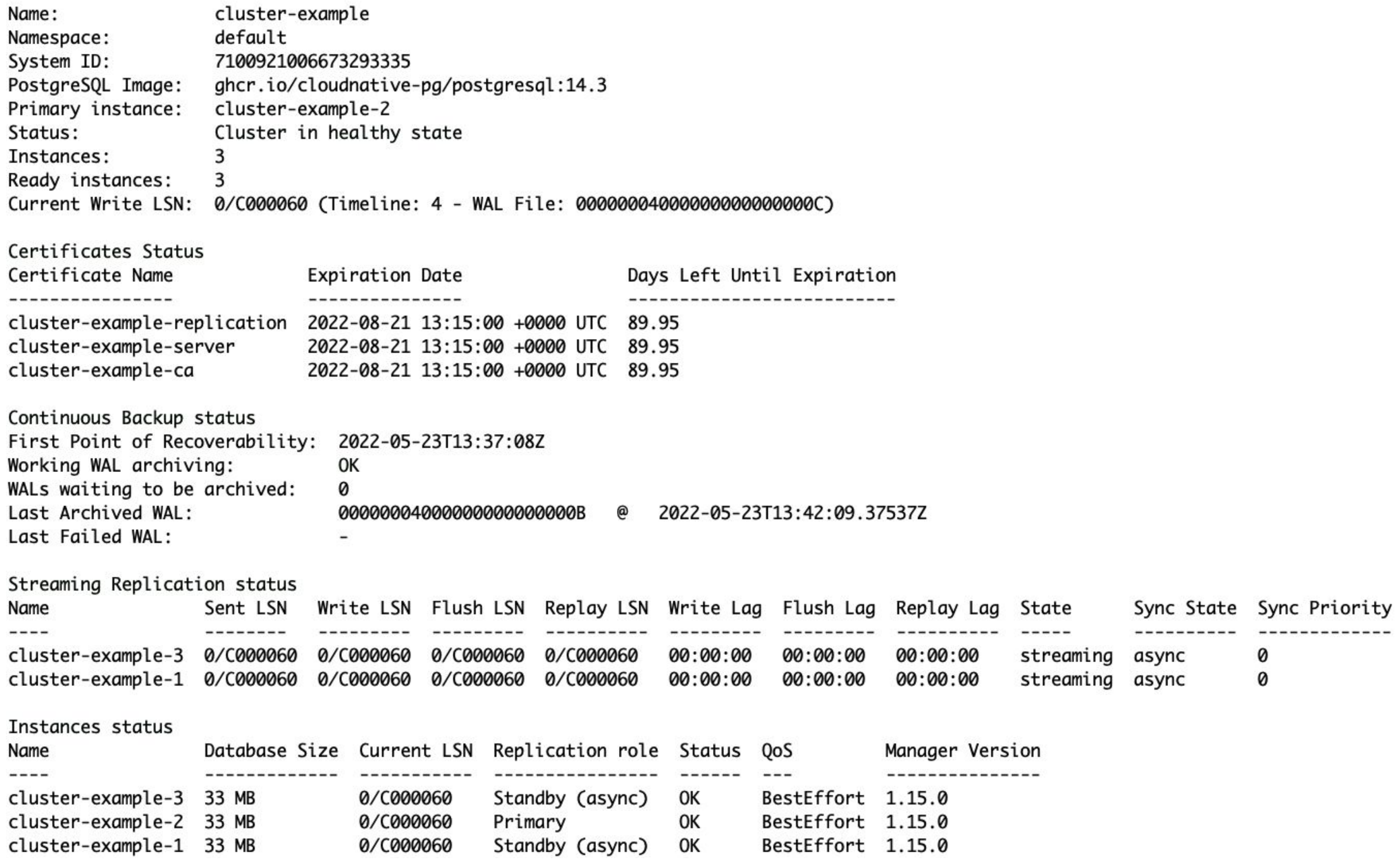
https://github.com/sergioenterprisedb/edb-postgres-for-kubernetes-in-openshift

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



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

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

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

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

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

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 to action: Check CNPG plugin

- Call the help for the CNPG Plugin, run:

```
kubectl-cnp help
```

Use case
Operator installation

Operator Installation demonstration

NOT NEEDED DURING WORKSHOP
For illustrative purposes.

- Install the operator
- 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

Install the CNPG Operator and check the it in the web terminal

NOT NEEDED DURING WORKSHOP
For illustrative purposes.

- In the web terminal install the operator:

```
./02_install_operator.sh
```

-> will require admin privs on Openshift

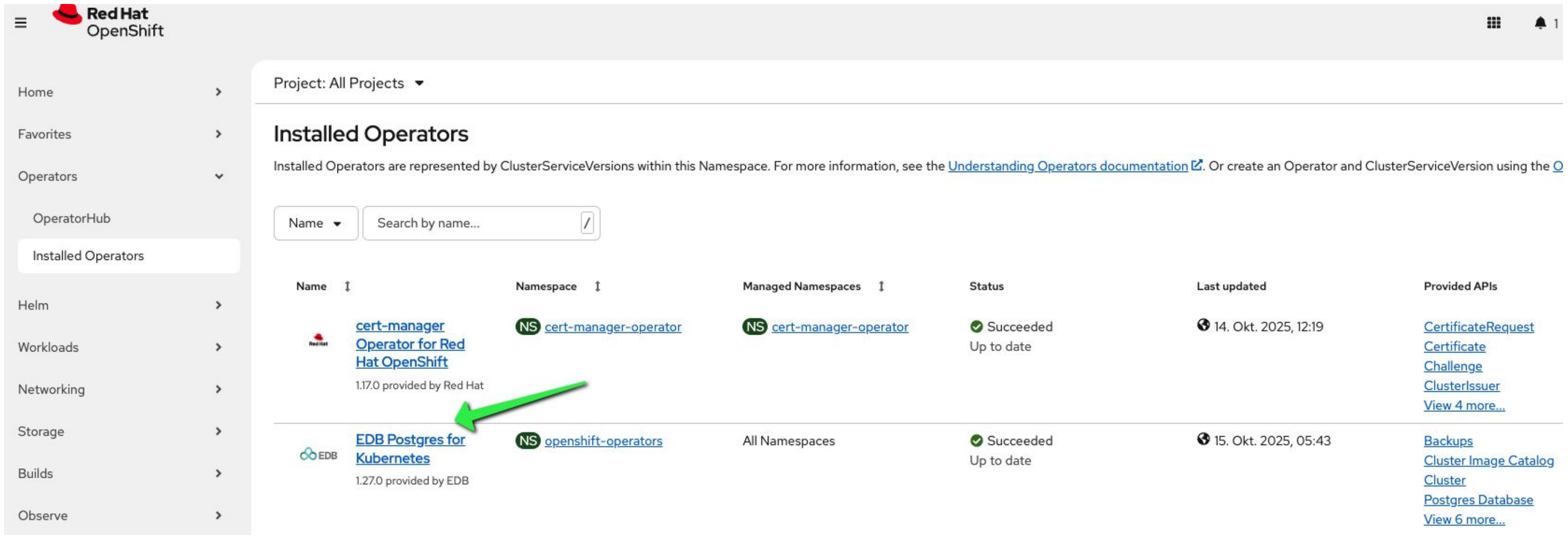
Call to action: Check the it in the web terminal

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

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

Call to action: Check the installed CNPG Operator in the Openshift console

- In the OpenShift console navigate to:
  - -> Operators
  - -> Installed Operators
  - -> Klick on the Operator installed:

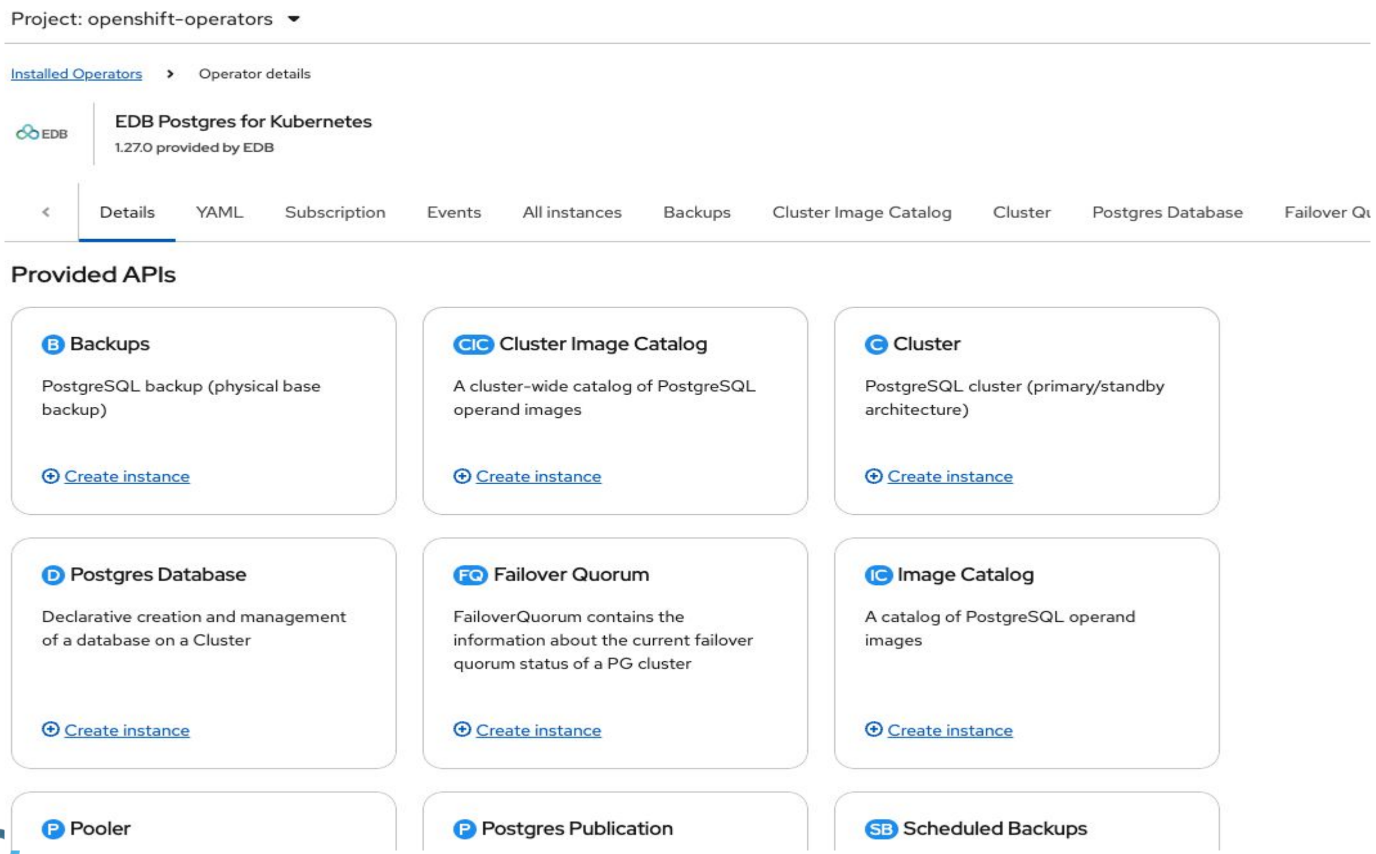


Installed Operators

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 O

Name	Namespace	Managed Namespaces	Status	Last updated	Provided APIs
cert-manager Operator for Red Hat OpenShift 1.17.0 provided by Red Hat	NS cert-manager-operator	NS cert-manager-operator	Succeeded Up to date	14. Okt. 2025, 12:19	CertificateRequest Certificate Challenge ClusterIssuer View 4 more...
EDB Postgres for Kubernetes 1.27.0 provided by EDB	NS openshift-operators	All Namespaces	Succeeded Up to date	15. Okt. 2025, 05:43	Backups Cluster Image Catalog Cluster Postgres Database View 6 more...

Call to action: Discover the features of the Operator in the OpenShift environment



Project: openshift-operators

Installed Operators > Operator details

EDB Postgres for Kubernetes
1.27.0 provided by EDB

Details | YAML | Subscription | Events | All instances | Backups | Cluster Image Catalog | Cluster | Postgres Database | Failover Qu

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
Postgres Database Declarative creation and management of a database on a Cluster Create instance	**Failover Quorum** FailoverQuorum contains the information about the current failover quorum status of a PG cluster Create instance	**Image Catalog** A catalog of PostgreSQL operand images Create instance
Pooler	**Postgres Publication**	**Scheduled Backups**

Use case
Create the postgres cluster

Synchronizing the state of a Postgres database

- Being a DBMS, PostgreSQL is a stateful workload in Kubernetes
- Stateless workloads achieve HA and DR mainly through traffic redirection
- Stateful workloads require the state to be replicated in multiple locations:
  - **Storage-level** replication
  - **Application-level** replication (in our case, application = Postgres)
- Postgres has a very robust and powerful native replication system
  - We've built it
  - Founded on the Write Ahead Log
  - Read-only standby servers
  - Supports also synchronous replication controlled at the transaction level
- **We recommend application-level** over storage-level replication for Postgres

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

Storage management

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

Call to action: 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
  ```

Call to action: Create table test with 1000 rows

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

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

- Check data in the database, use the kubectl plugin to connect to the database:

```
echo "select count(*) from test;" | kubectl-cnp psql cluster-user<X>
```

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

Call to action: 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

Call to action: 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
    ```

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

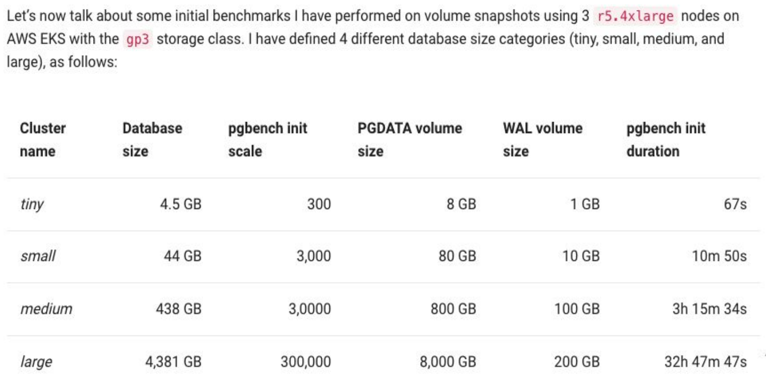
Backup and Recovery - Part 2

- 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

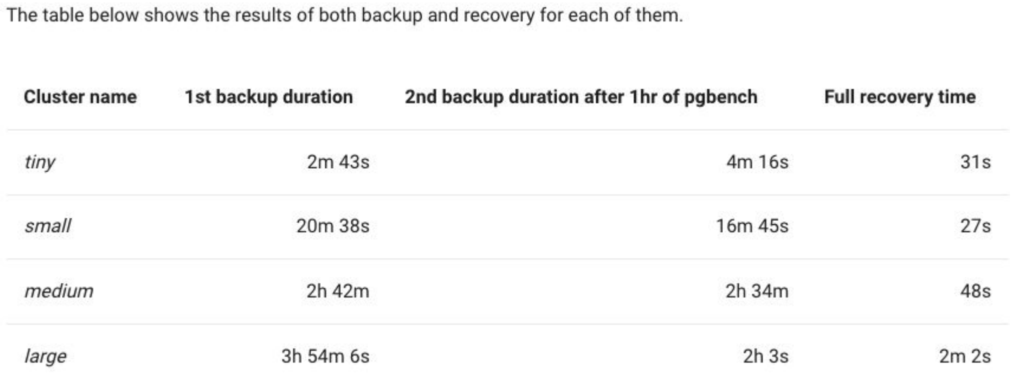
- Transparent support for:
  - Incremental backup and recovery at block level
  - Differential backup and recovery at block level
  - Based on copy on write
- Leverage the storage class to manage the snapshots, including:
  - Data mobility across network (availability zones, Kubernetes clusters, regions)
  - 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
tiny	4.5 GB	300	8 GB	1 GB	67s
small	44 GB	3,000	80 GB	10 GB	10m 50s
medium	438 GB	3,0000	800 GB	100 GB	3h 15m 34s
large	4,381 GB	300,000	8,000 GB	200 GB	32h 47m 47s



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

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

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

Call to action: 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:

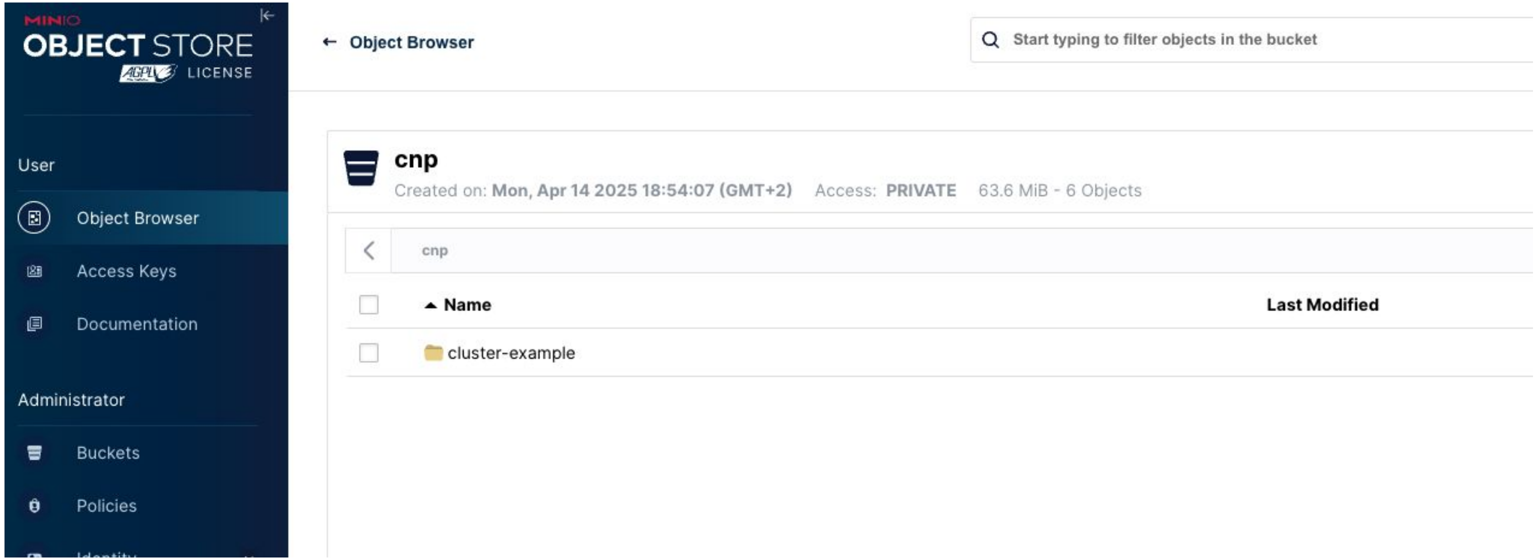
      ```
      ./10_backup_cluster.sh
      ```

    - Check the backup status:

      ```
      ./11_backup_describe.sh
      ```

Call to action: Check Backup in MinIO UI

- Obtain the MinIO URL from the Slide **#56** and open the URL:
- Connect as user **minio** with the password: **edb-workshop**
- The page will appear:



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

cnp

Name	Last Modified
cluster-example	

Call to action: 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  "select count(*) from test;"
```

  - Delete the cluster-restore-user<x> to avoid resource problems during the workshop:

```
oc delete cluster cluster-restore-user<X>
```

Use case: Failover

Call to action: 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
    ```

Use case
Scale-out and scale-down

Scale up and down of replicas

- The operator allows you to scale up and down the number of instances in a PostgreSQL cluster.
- New replicas are started up from the primary server and participate in the cluster's HA infrastructure.
- The CRD declares a "scale" subresource that allows you to use the kubectl scale command.

Call to action: 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

Call to action: 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
    ```

Use Case
Fencing

Fencing

- **Fencing** is the process of protecting the data in one, more, or even all instances of a PostgreSQL cluster when they appear to be malfunctioning.
- When an instance is fenced, the PostgreSQL server process is guaranteed to be shut down, while the pod is kept running.
- This ensures that, until the fence is lifted, data on the pod isn't modified by PostgreSQL and that you can investigate file system for debugging and troubleshooting purposes.

Call to action: 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
```

Call to action: 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
    ```

Use case
Hibernation

Hibernation

- CloudNativePG supports **hibernation** of a running PostgreSQL cluster in a declarative manner
  - through the cnpg.io/hibernation annotation
- Hibernation enables saving CPU power by removing the database pods while keeping the database PVCs
- This feature simulates scaling to 0 instances.

Call to action: 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
```

Call to action: Start the postgres cluster

- In the terminal 1:
  - Run the script:

```
./33_hibernation_off.sh
```

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

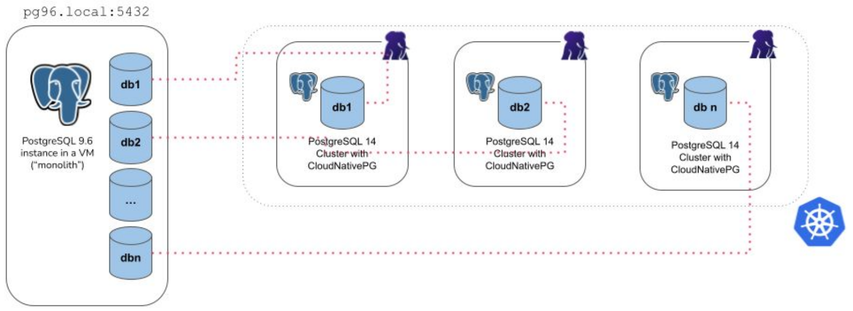
```
./06_show_status.sh
```

Use case
Database Migration / Major Version Upgrade

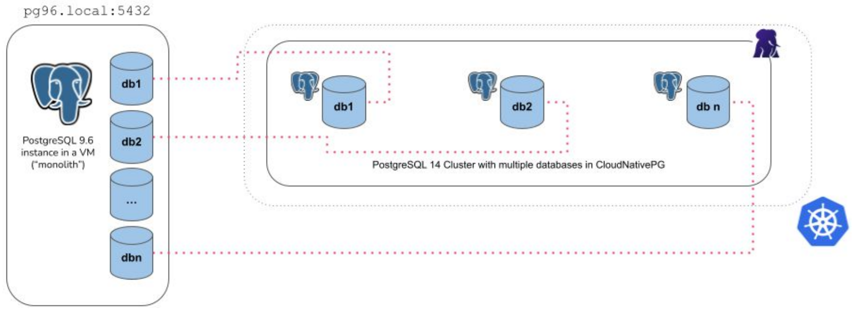
Database Migration

- In this step we will migrate the app database from our existing cluster to the new cluster
- We will create the yaml file with the setting "**import**" in the bootstrap section
- The operator uses internally postgres tools pg_dump and pg_restore
- This method can be used to **migrate** another database or to run **out-of-the-place upgrade**
- Possible settings:

Microservices:


pg96.local:5432
db1
db2
...
dbn
PostgreSQL 9.6 instance in a VM ("monolith")
db1
PostgreSQL 14 Cluster with CloudNativePG
db2
PostgreSQL 14 Cluster with CloudNativePG
db n
PostgreSQL 14 Cluster with CloudNativePG

Monolith:


pg96.local:5432
db1
db2
...
dbn
PostgreSQL 9.6 instance in a VM ("monolith")
db1
db2
db n
PostgreSQL 14 Cluster with multiple databases in CloudNativePG

Call to action: Run migration or out of the place upgrade

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

    ```
    ./20_upgrade_major_version.sh
    ```

  - Check the cluster creation process:

    ```
    kubectl get pods -w
    ```

  - Check the table test in the cluster-user<X>-17, run the command:
    - Connect to the cluster:

      ```
      oc cnp psql cluster-user<X>-17
      ```

    - Connect to the database app:

      ```
      \c app
      ```

    - Run sql commands:

      ```
      select version();
      select count(*) from test;
      ```

What more?

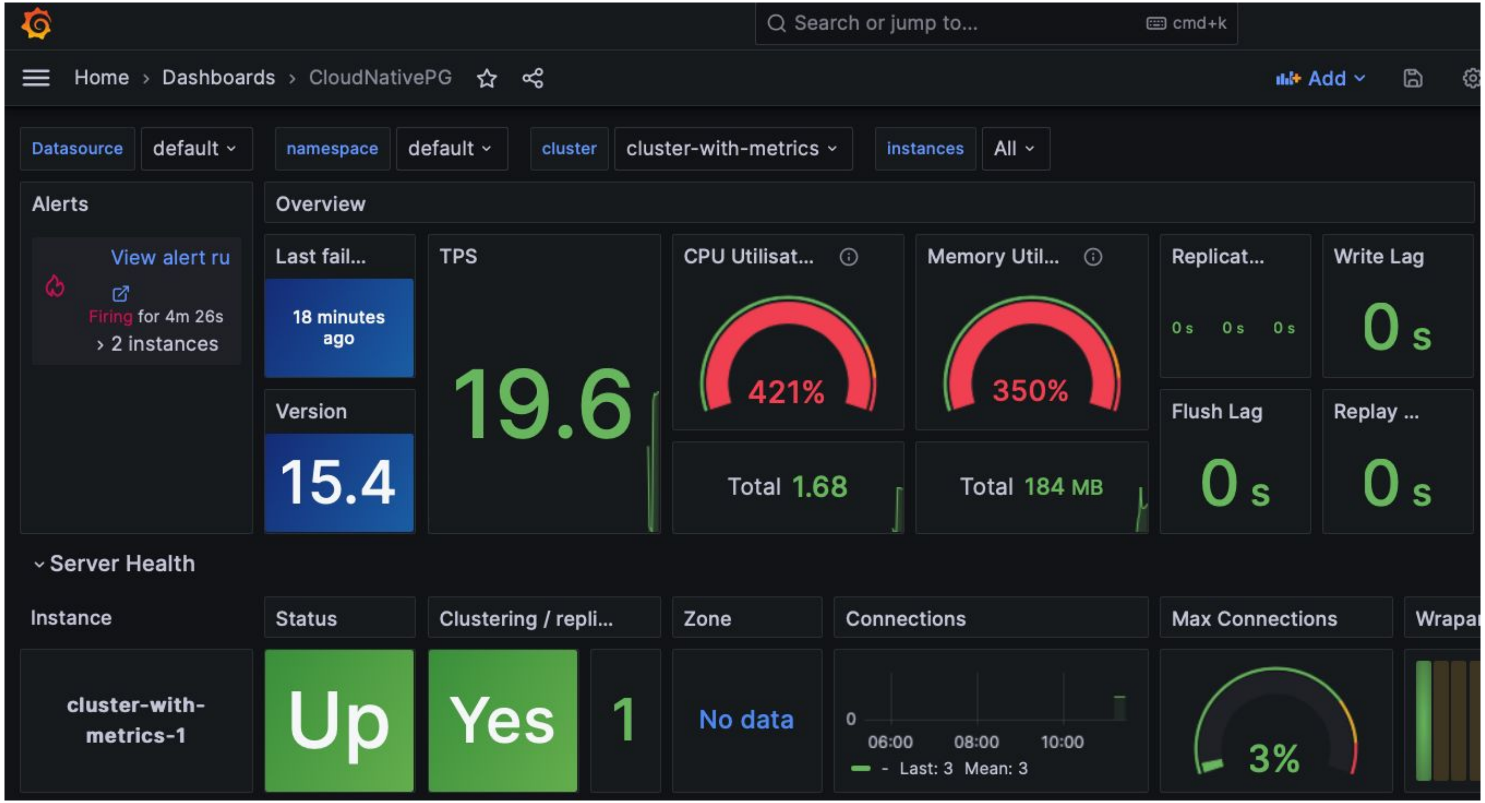
(some additional features from EDB)

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



Specific to EDB Operator

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

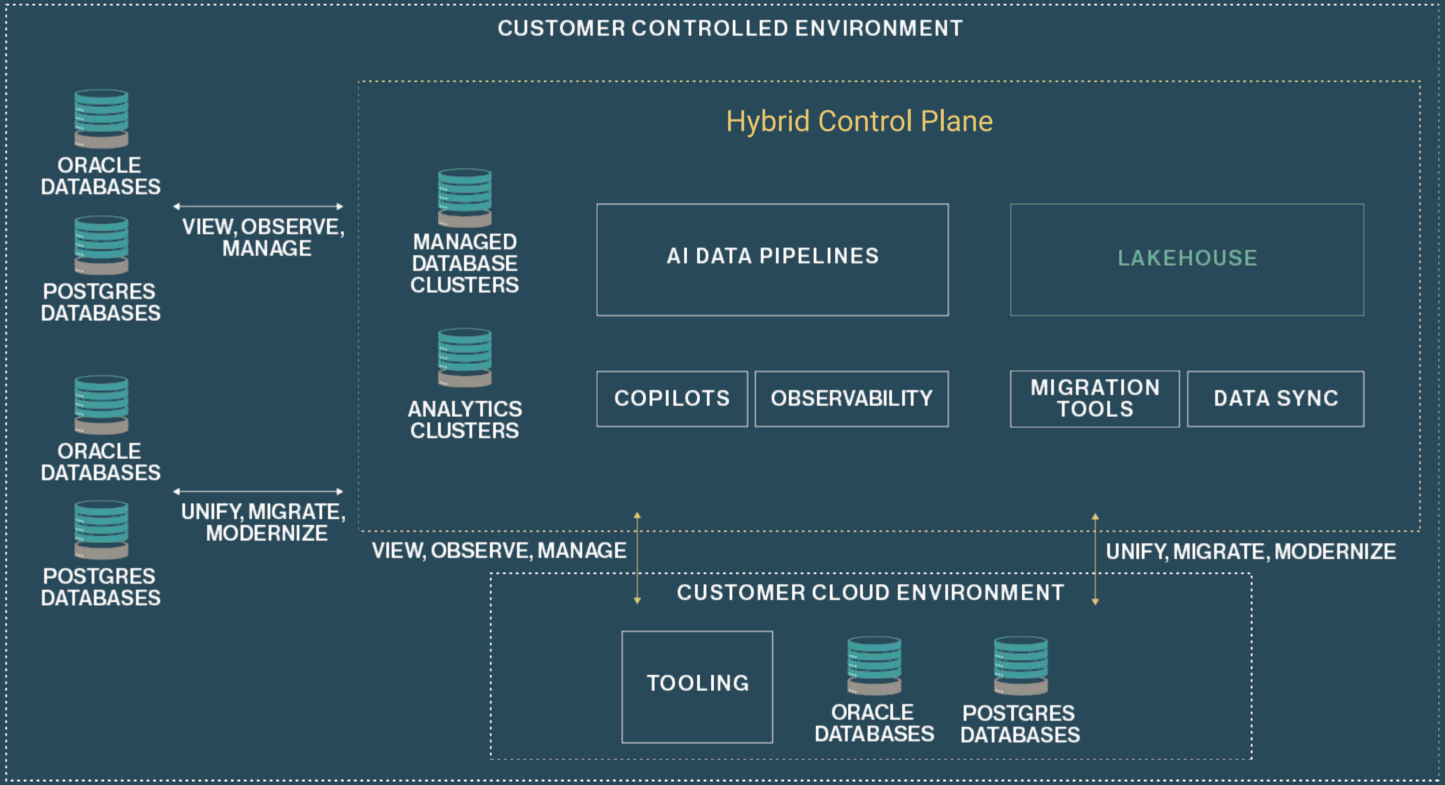
Data redaction
Protect sensitive information for GDPR, PCI and HIPAA compliance

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



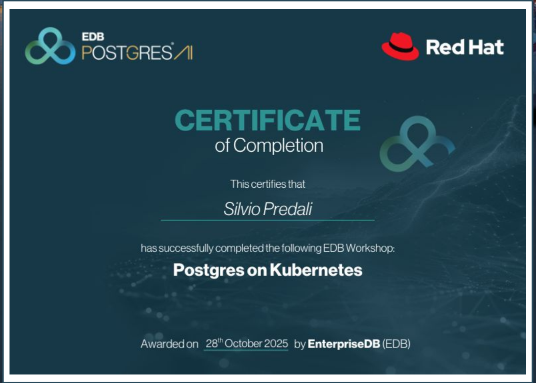
Hybrid Control Plane

LIVE DEMO

Thank you for participating in the

Postgres on Kubernetes Workshop

Your certificate will be emailed to you after the workshop!



EDB POSTGRES AI

Red Hat

CERTIFICATE
of Completion

This certifies that

Silvio Predali

has successfully completed the following EDB Workshop:

Postgres on Kubernetes

Awarded on 28th October 2025 by **EnterpriseDB** (EDB)