How to run PostgreSQL on Kubernetes

EDB

Sergio Romera | Senior Sales Engineer
February 2023

About me

- Sergio Romera (🇪🇸 🇫🇷 🇬🇧)

- Based in France, (Île-de-France near to Paris)
- Database fanatic since 1997
- Developer, DBA, Architect, Sales Engineer
- Companies: BNPParibas, Oracle, Quest Software
- Senior Sales Engineer at EDB

Why did PostgreSQL win?

It does everything…

- Migration
- New App Development
- Replatforming to Cloud and Containers
- System of Record
- System of Analysis
- System of Engagement

It works everywhere…

- Public Cloud - IaaS
- Public Cloud - DBaaS
- Private Cloud
- Virtual Machines
- Containers

EDB™

A kubernetes operator for Postgres

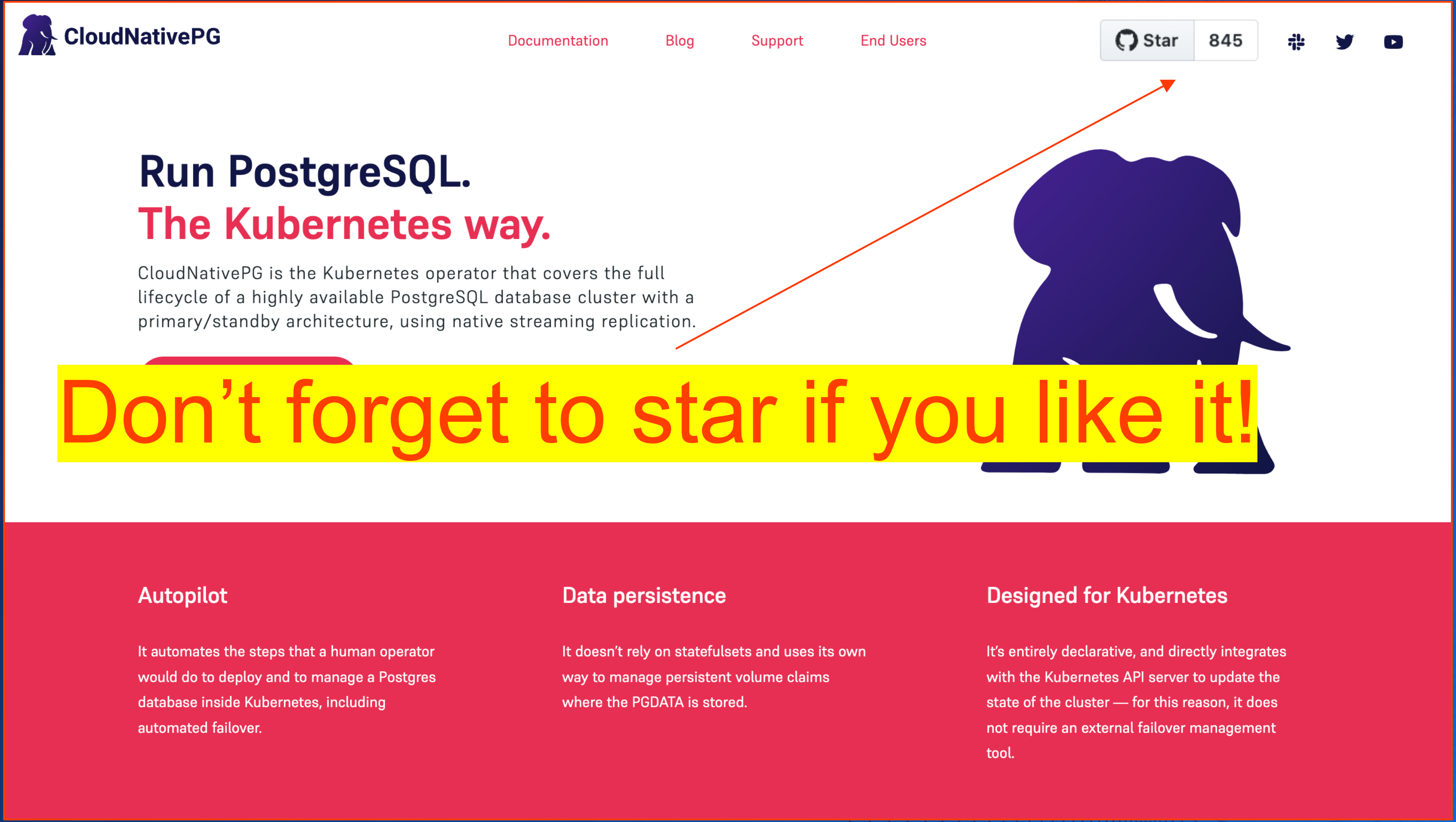
- Operators are software extensions to Kubernetes that make use of custom resources to manage applications and their components.
- Operators follow Kubernetes principles, notably the control loop.
- (Kubernetes definition link)

- Our PostgreSQL operator must simulate the work of a DBA

CloudNativePG

- Kubernetes operator for PostgreSQL
- "Level 5", Production ready
- Day 1 & 2 operations of a PostgreSQL database
  - In traditional environments usually reserved to humans
- Open source
  - Originally created and developed by EDB
  - Vendor neutral/openly governed community
  - Apache 2.0 license
  - Submitted to the CNCF Sandbox
- Fully declarative

EDB™



CloudNativePG

Documentation Blog Support End Users

Star 845

Run PostgreSQL.
The Kubernetes way.

CloudNativePG is the Kubernetes operator that covers the full lifecycle of a highly available PostgreSQL database cluster with a primary/standby architecture, using native streaming replication.

Don’t forget to star if you like it!

Autopilot

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

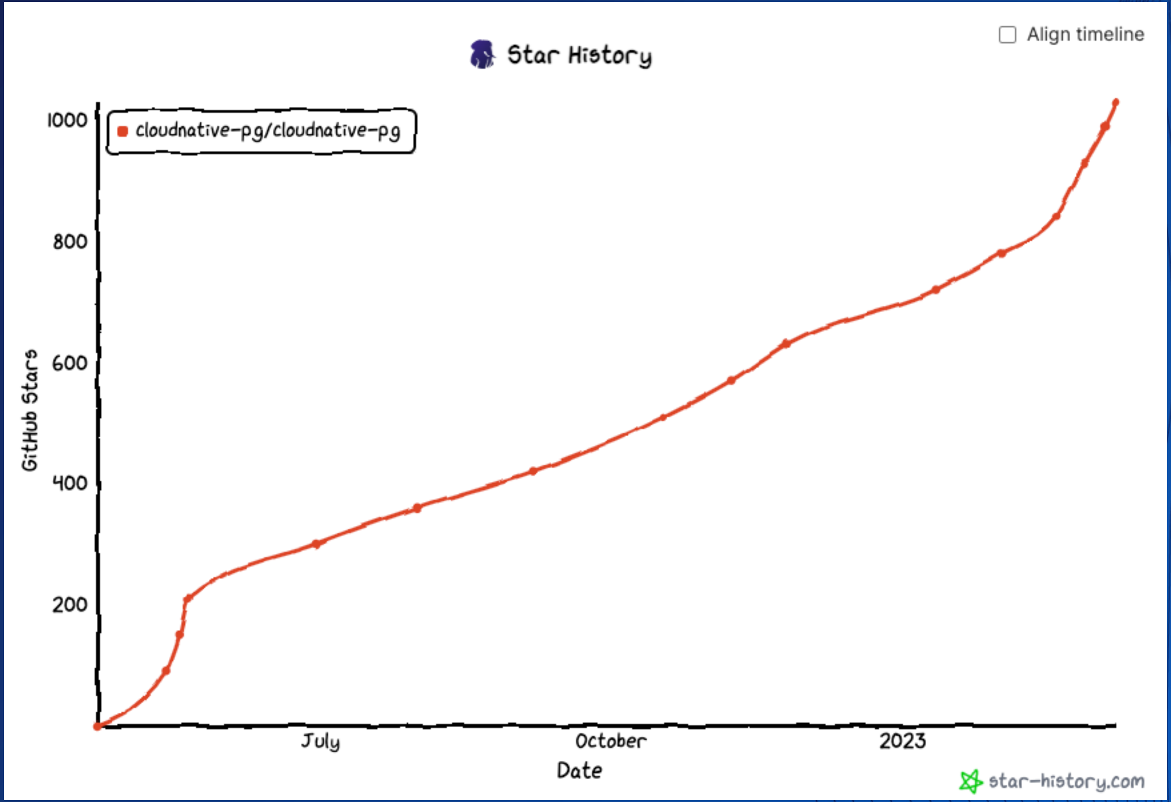
Data persistence

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

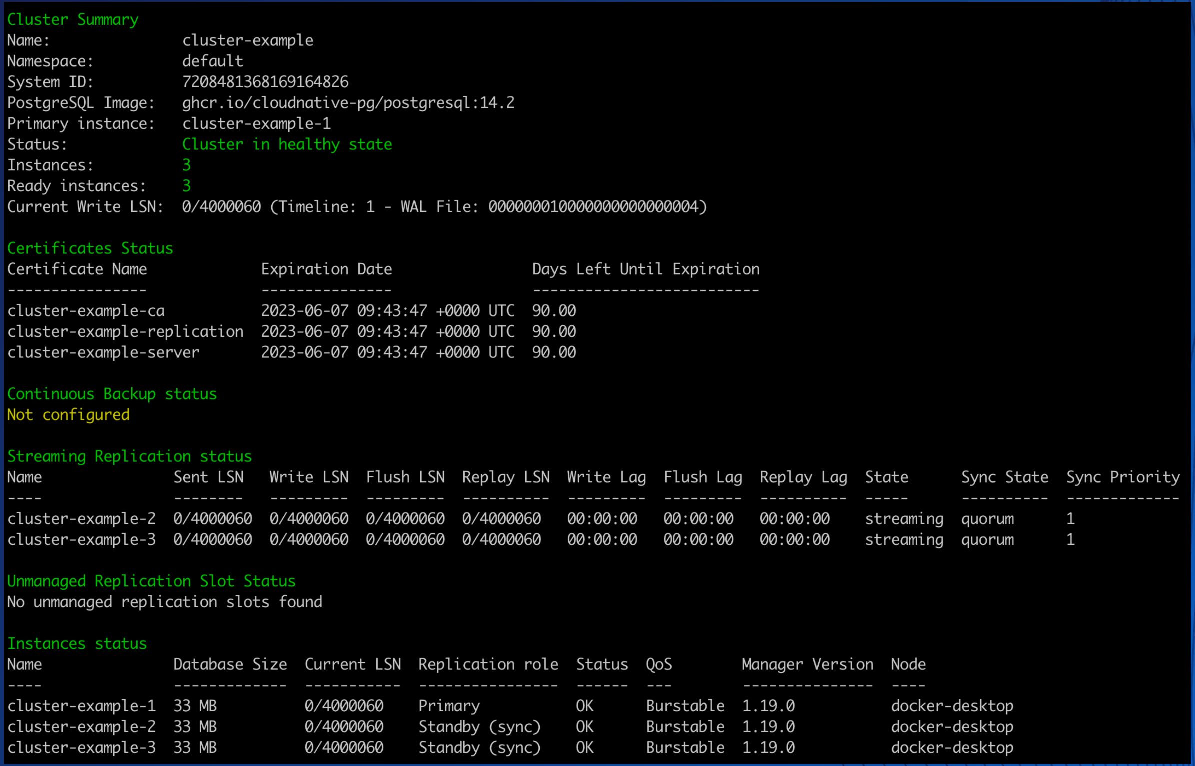
Designed for Kubernetes

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.

EDB


Align timeline
Star History
● cloudnative-pg/cloudnative-pg
GitHub Stars
1000
800
600
400
200
July
October
2023
Date
star-history.com

Command line interface



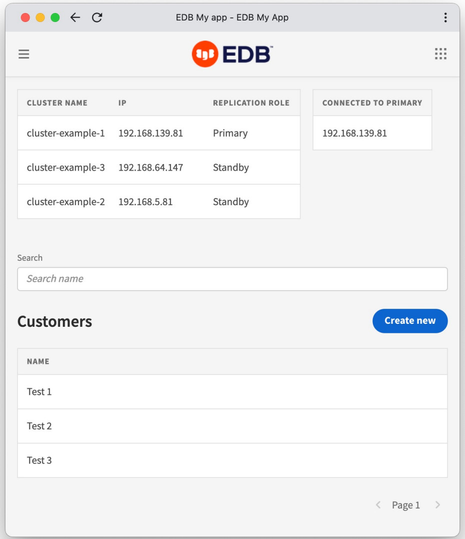
```
Cluster Summary
Name:               cluster-example
Namespace:          default
System ID:          7208481368169164826
PostgreSQL Image:   ghcr.io/cloudnative-pg/postgresql:14.2
Primary instance:   cluster-example-1
Status:             Cluster in healthy state
Instances:          3
Ready instances:    3
Current Write LSN:  0/4000060 (Timeline: 1 - WAL File: 000000010000000000000004)

Certificates Status
Certificate Name             Expiration Date                Days Left Until Expiration
----------------             ---------------                --------------------------
cluster-example-ca           2023-06-07 09:43:47 +0000 UTC  90.00
cluster-example-replication  2023-06-07 09:43:47 +0000 UTC  90.00
cluster-example-server       2023-06-07 09:43:47 +0000 UTC  90.00

Continuous Backup status
Not configured

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-2  0/4000060  0/4000060  0/4000060  0/4000060   00:00:00   00:00:00   00:00:00    streaming  quorum      1
cluster-example-3  0/4000060  0/4000060  0/4000060  0/4000060   00:00:00   00:00:00   00:00:00    streaming  quorum      1

Unmanaged Replication Slot Status
No unmanaged replication slots found

Instances status
Name               Database Size  Current LSN  Replication role  Status  QoS        Manager Version  Node
----               -------------  -----------  ----------------  ------  ---        ---------------  ----
cluster-example-1  33 MB          0/4000060    Primary           OK      Burstable  1.19.0           docker-desktop
cluster-example-2  33 MB          0/4000060    Standby (sync)    OK      Burstable  1.19.0           docker-desktop
cluster-example-3  33 MB          0/4000060    Standby (sync)    OK      Burstable  1.19.0           docker-desktop
```

EDB

Demo

Demo Architecture

kubernetes



CLUSTER NAME	IP	REPLICATION ROLE
cluster-example-1	192.168.139.81	Primary
cluster-example-3	192.168.64.147	Standby
cluster-example-2	192.168.5.81	Standby

CONNECTED TO PRIMARY
192.168.139.81

Search

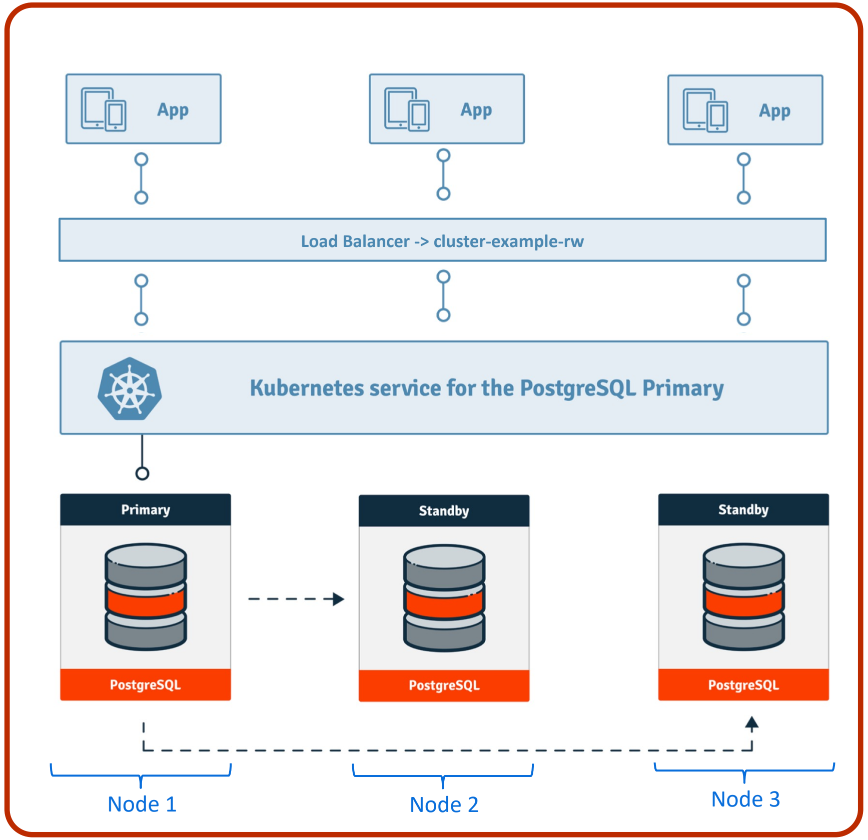
Search name

Customers

Create new

NAME
Test 1
Test 2
Test 3

Page 1


App
App
App
Load Balancer -> cluster-example-rw
Kubernetes service for the PostgreSQL Primary
Primary
PostgreSQL
Standby
PostgreSQL
Standby
PostgreSQL
Node 1
Node 2
Node 3

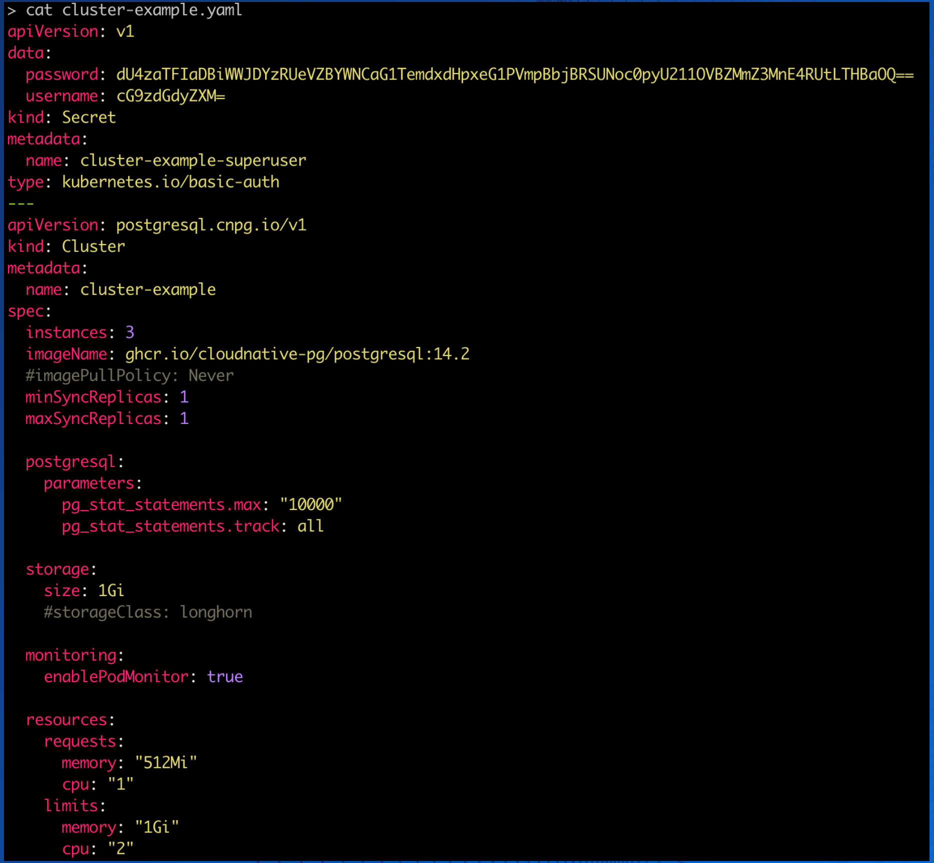
EDB

Features demo

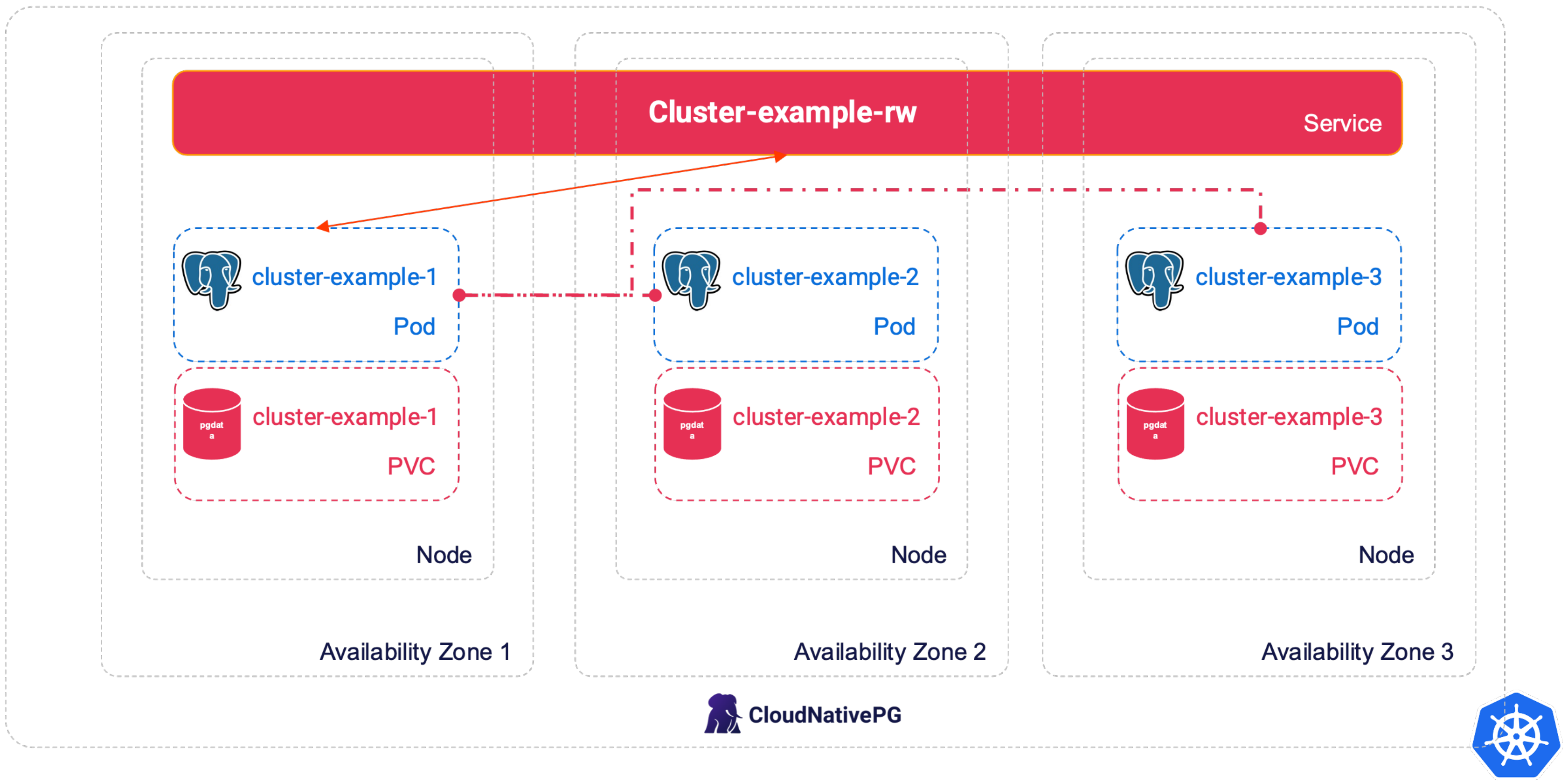
- Kubernetes plugin install
- CloudNativePG operator install
- Postgres cluster install
- Insert data in the cluster
- Switchover (promote)
- Failover
- Backup
- Recovery
- Rolling updates (minor and major)
- Last CloudNativePG tested version is 1.19.0

cluster-example.yaml

- Cluster name: cluster-example
- 3 Instances
  - 1 Primary
  - 2 Standby’s
- PostgreSQL 14.2
- Min 1 sync replica
- Activate pg_stat_statement extensión
- 1GB disk
- Activate monitoring metrics
- CPU
  - Request: 1
  - Limit: 2



```
> cat cluster-example.yaml
apiVersion: v1
data:
  password: dU4zaTFIaDBiWWJDYzRUeVZBYWNCaG1TemdxdHpxeG1PVmpBbjBRSUNoc0pyU211OVBZMmZ3MnE4RUtLTHBaOQ==
  username: cG9zdGdyZXM=
kind: Secret
metadata:
  name: cluster-example-superuser
type: kubernetes.io/basic-auth
---
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: cluster-example
spec:
  instances: 3
  imageName: ghcr.io/cloudnative-pg/postgresql:14.2
  #imagePullPolicy: Never
  minSyncReplicas: 1
  maxSyncReplicas: 1

  postgresql:
    parameters:
      pg_stat_statements.max: "10000"
      pg_stat_statements.track: all

  storage:
    size: 1Gi
    #storageClass: longhorn

  monitoring:
    enablePodMonitor: true

  resources:
    requests:
      memory: "512Mi"
      cpu: "1"
    limits:
      memory: "1Gi"
      cpu: "2"
```


Cluster-example-rw
Service
cluster-example-1
Pod
cluster-example-2
Pod
cluster-example-3
Pod
pgdata
cluster-example-1
PVC
pgdata
cluster-example-2
PVC
pgdata
cluster-example-3
PVC
Node
Node
Node
Availability Zone 1
Availability Zone 2
Availability Zone 3
CloudNativePG

EDB

This demo is in

https://github.com/sergioenterprisedb/kubecon2022-demo

DEMO In yellow

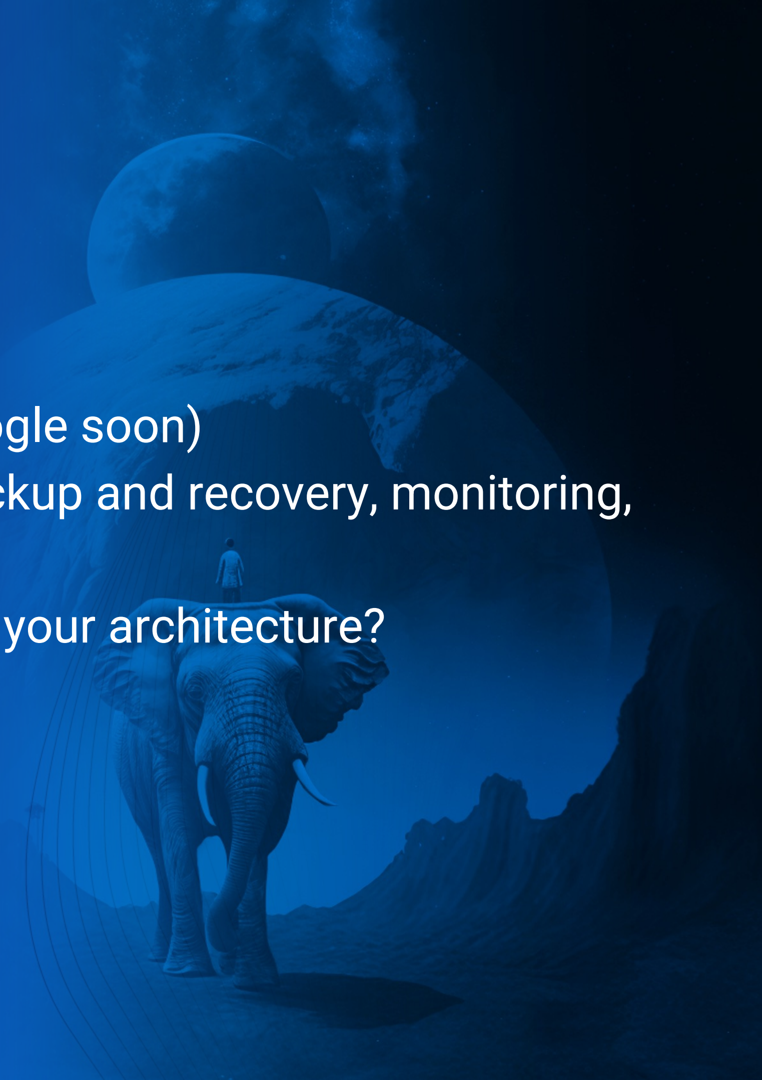
Key capabilities

- Direct integration with Kubernetes API server for High Availability, without requiring an external tool
- Failover of the primary instance by promoting the most aligned replica
- Automated recreation of a replica
- Planned switchover of the primary instance by promoting a selected replica
- Scale up/down capabilities
- Definition of the *read-write* service, to connect your applications to the only primary server of the cluster
- Definition of the *read-only* service, to connect your applications to any of the instances for reading workloads
- Declarative management of PostgreSQL configuration, including certain popular Postgres extensions through the cluster spec: pg_audit, auto_explain, and pg_stat_statements
- Support for Local Persistent Volumes with PVC templates
- Reuse of Persistent Volumes storage in Pods
- Separate volume for WAL files
- Rolling updates for PostgreSQL minor versions
- In-place or rolling updates for operator upgrades
- TLS connections and client certificate authentication
- Support for custom TLS certificates (including integration with cert-manager)
- Continuous backup to an object store (AWS S3 and S3-compatible, Azure Blob Storage, and Google Cloud Storage)
- Backup retention policies (based on recovery window)
- Full recovery and Point-In-Time recovery from an existing backup in an object store
- Offline import of existing PostgreSQL databases, including major upgrades of PostgreSQL
- Parallel WAL archiving and restore to allow the database to keep up with WAL generation on high write systems
- Support tagging backup files uploaded to an object store to enable optional retention management at the object store layer Replica clusters for
- PostgreSQL deployments across multiple Kubernetes clusters, enabling private, public, hybrid, and multi-cloud architectures
- Support for Synchronous Replicas
- Support for HA physical replication slots at cluster level
- Connection pooling with PgBouncer
- Support for node affinity via nodeSelector
- Native customizable exporter of user defined metrics for Prometheus through the metrics port (9187)
- Standard output logging of PostgreSQL error messages in JSON format
- Automatically set readOnlyRootFilesystem security context for pods
- cnpg plugin for kubectl
- Fencing of an entire PostgreSQL cluster, or a subset of the instances
- Simple bind and search+bind LDAP client authentication
- Multi-arch format container images
- Postgres cluster hibernation

EDB™

Contact EDB if you need:

- Support for PostgreSQL Opensource
- Oracle migrations to PostgreSQL
- Managed Postgres on Azure or AWS (and Google soon)
- Enterprise tools for Postgres (HA, failover, backup and recovery, monitoring, trainings, …)
- Do you need a workshop to better understand your architecture?



EDB™

Sergio Romera
EDB – Senior Sales Engineer

Oracle Cloud Infrastructure Architect
Oracle Autonomous Database Cloud Specialist
Database Administrator

AWS Cloud Practitioner

Microsoft Azure Fundamentals
Microsoft Data Azure Fundamentals

EDB Certified Associate - Postgres Advanced Server 12
EDB Certified Professional – Postgres Advanced Server 13

Thank you