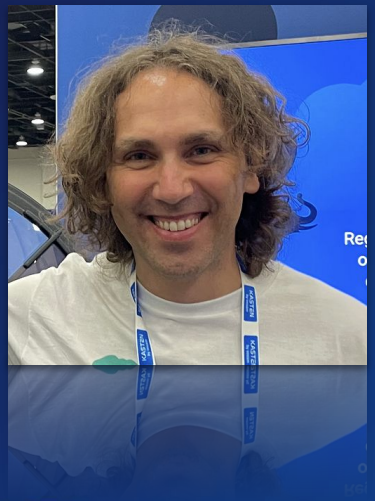
CloudNativePG & EDB Postgres for Kubernetes

Gabriele Bartolini
VP Cloud Native at EDB

June 2023

About me



- VP/CTO of Cloud Native at EDB
  - Previously at 2ndQuadrant
- PostgreSQL user since ~2000
  - Community member since 2006
  - Co-founder of PostgreSQL Europe
- DevOps evangelist
- Open source contributor
  - Barman (2011)
  - CloudNativePG (2022)

Follow me: **@_GBartolini_**

EDB

CloudNativePG Website	cloudnative-pg.io
GitHub project	github.com/cloudnative-pg/cloudnative-pg
PostgreSQL	postgresql.org
EnterpriseDB	enterprisedb.com
EDB Postgres for Kubernetes	[scan the QR code below]

@CloudNativePG

@EDBPostgres

@_GBartolini_

EDB

Introduction

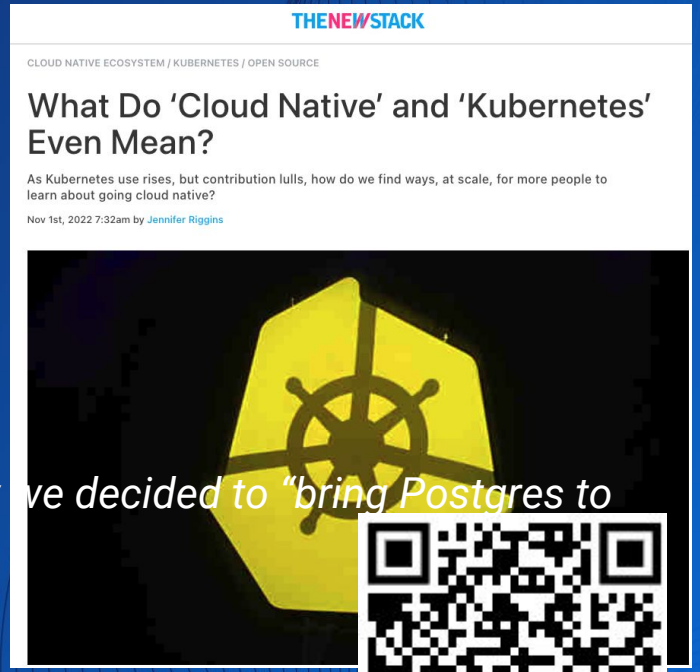
What we mean by “Cloud Native”

Cloud Native means all of the following:

- A **“generative” high-trust DevOps culture**
- **Immutable Application Containers**
- A **Container orchestrator** like Kubernetes

(Believe it or not, but the above 3 items are the reasons why ve decided to “bring Postgres to Kubernetes” - not the other way around)

For DevOps please refer to the DORA research project



#2 most read article in 2022

Container Journal

EDB

Databases like PostgreSQL are complex stateful applications.

To run in Kubernetes, they need a good operator

CloudNativePG in a nutshell

- Kubernetes operator for Postgres
- “Level 5”, Production ready
- Open source
  - Apache License 2.0
  - Vendor neutral openly governed
  - Originally created by EDB
- Extends the K8s controller
  - Status of the `Cluster`
  - “no Patroni, No statefulsets”
- Immutable application containers
- Fully declarative
  - Convention over configuration

- Automated failover
- Services for RW and RO workloads
- mTLS
- Affinity control
- Backup and recovery
- Rolling updates
- Scale up/down of read replicas
- Fencing and hibernation
- Native Prometheus exporters
- Log in JSON format to stdout
- Kubectl plug-in

… and much more

URL: **github.com/cloudnative-pg**

EDB

Support policy from the CloudNativePG community

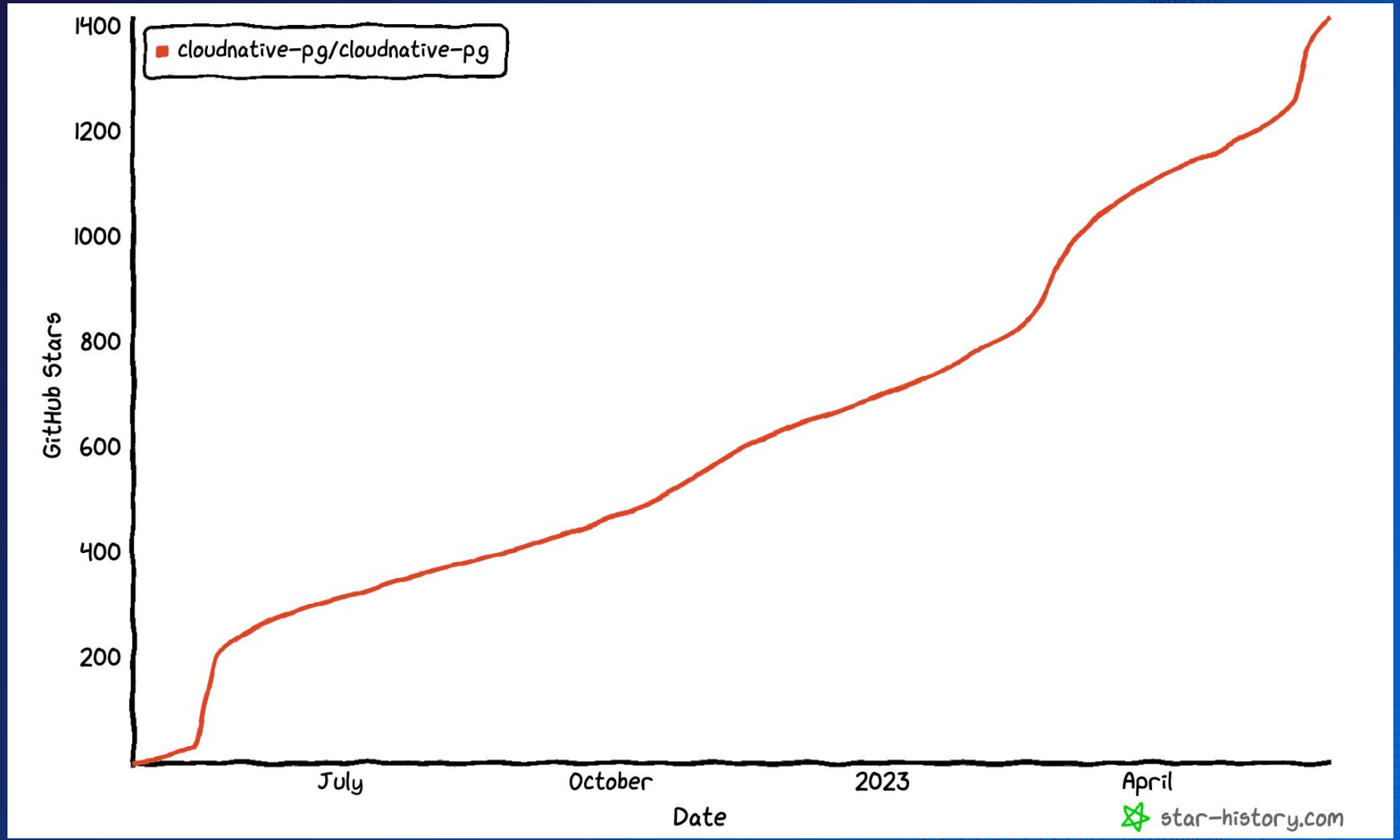
Type	Support Level	Quality and Recommended Use
Development Build	No support	Dangerous, may not be fully reliable. Useful to experiment with.
Minor Release	Support provided until 1 month after the N+2 minor release (e.g. 1.19 supported until 1 month after 1.21.0 is released)	
Patch	Same as the corresponding Minor release	Users are encouraged to adopt patch releases as soon as they are available for a given release.
Security Patch	Same as a Patch, however, it will not contain any additional code other than the security fix from the previous patch	Given the nature of security fixes, users are **strongly** encouraged to adopt security patches after release.

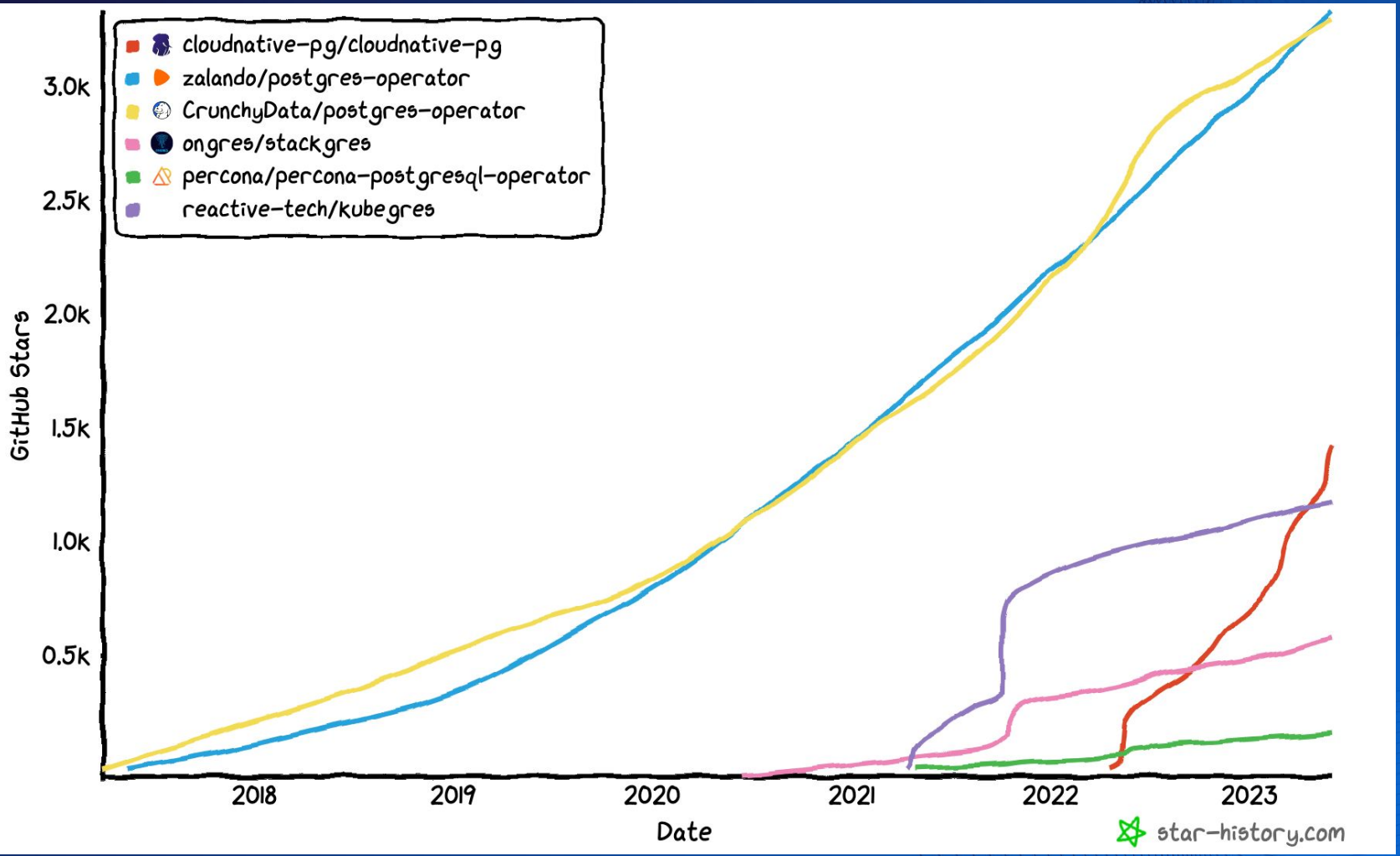
EDB

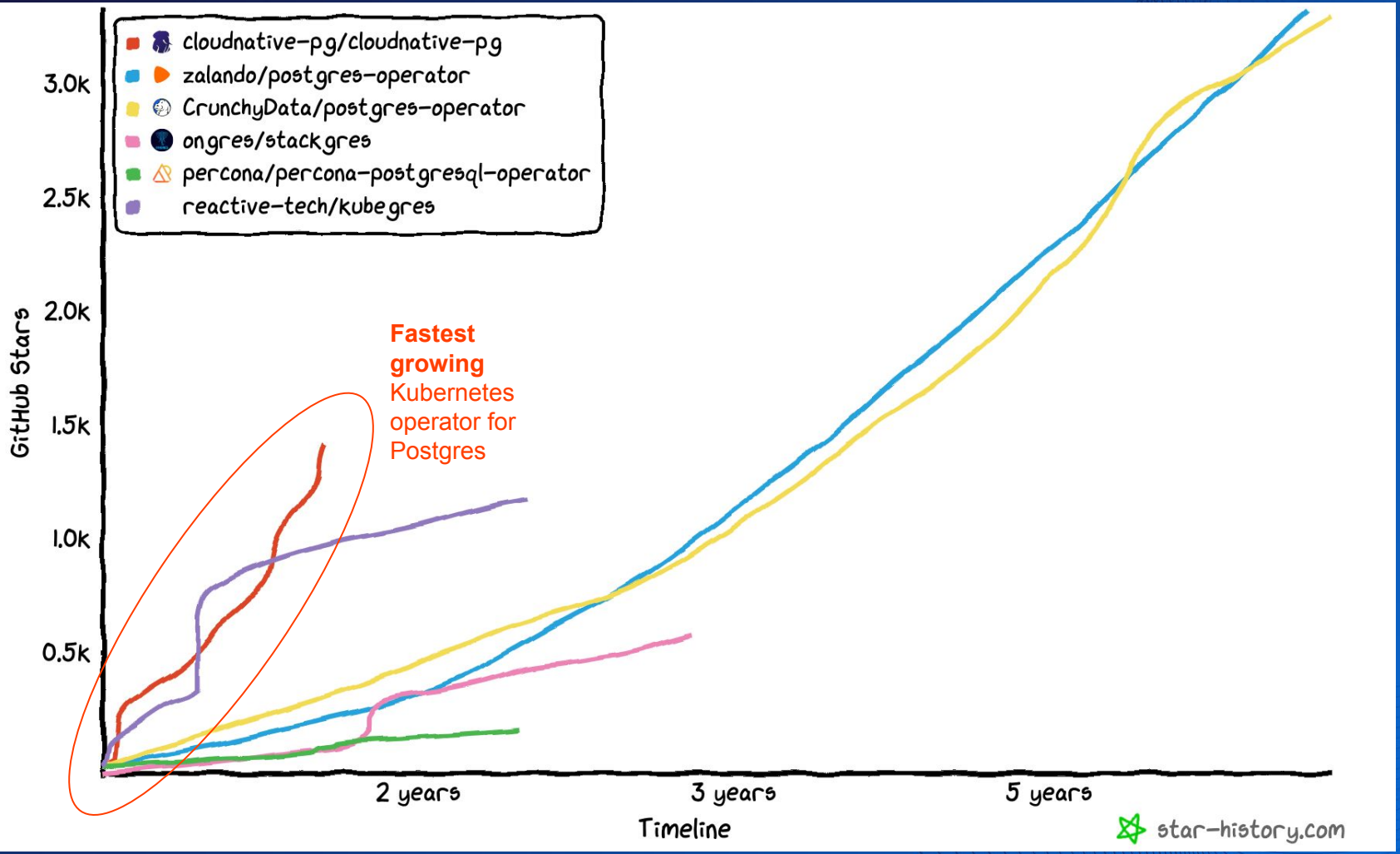
Support status of CloudNativePG releases

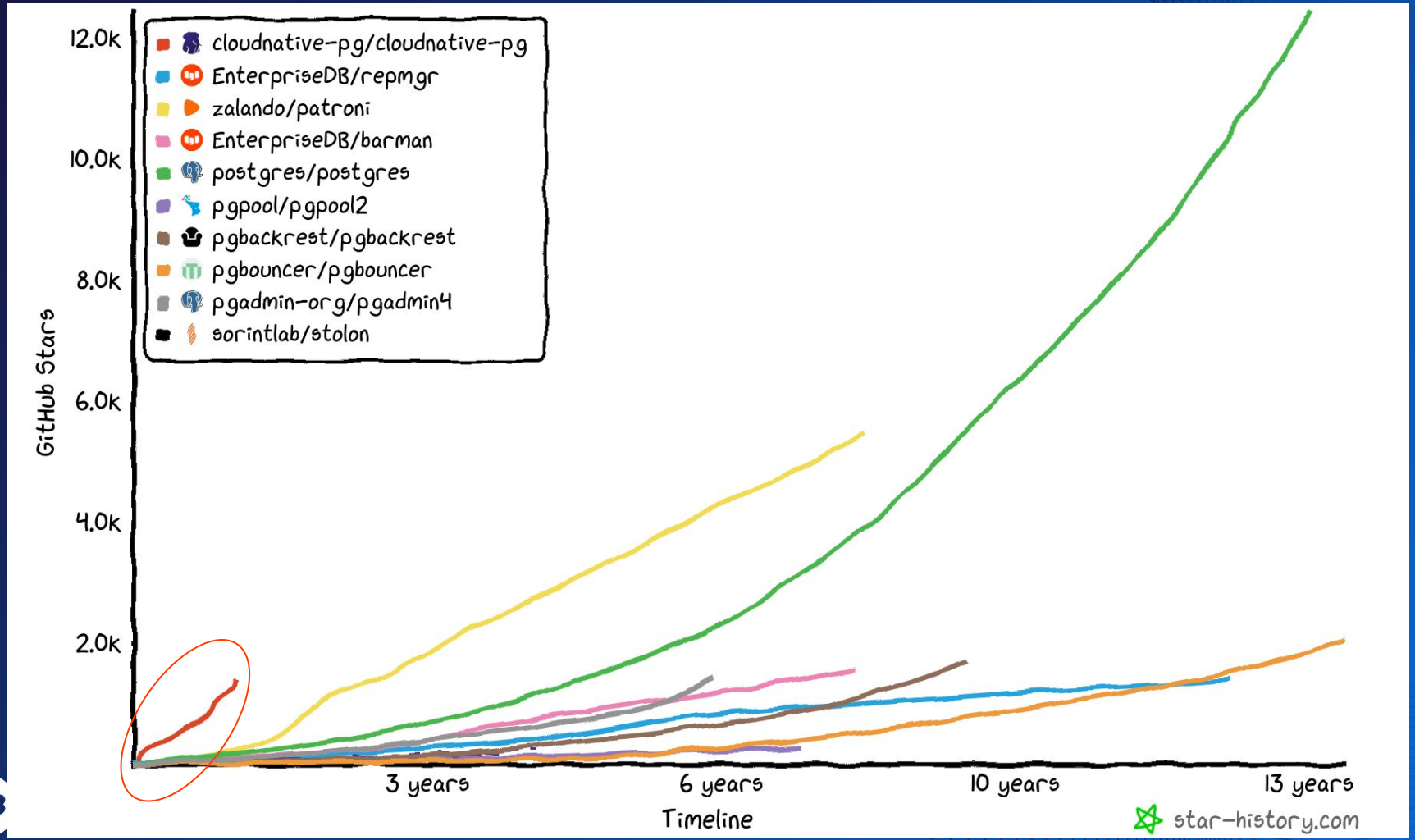
Version	Currently Supported	Release Date	End of Life	Supported Kubernetes Versions	Tested, but not supported
1.20.x	Yes	April 27, 2023	~ September 28, 2023	1.24 - 1.27	1.23
1.19.x	Yes	February 14, 2023	~ July 15, 2023	1.23 - 1.26	1.27
1.18.x	No	November 10, 2022	May 27, 2023		
main	No, development only				

EDB


cloudnative-pg/cloudnative-pg
1400
1200
1000
800
600
400
200
GitHub Stars
July
October
2023
April
Date
star-history.com


cloudnative-pg/cloudnative-pg
zalando/postgres-operator
CrunchyData/postgres-operator
ongres/stackgres
percona/percona-postgresql-operator
reactive-tech/kubegres
3.0k
2.5k
2.0k
1.5k
1.0k
0.5k
GitHub Stars
2018
2019
2020
2021
2022
2023
Date
star-history.com


cloudnative-pg/cloudnative-pg
zalando/postgres-operator
CrunchyData/postgres-operator
ongres/stackgres
percona/percona-postgresql-operator
reactive-tech/kubegres
3.0k
2.5k
2.0k
1.5k
1.0k
0.5k
GitHub Stars
Fastest growing Kubernetes operator for Postgres
2 years
3 years
5 years
Timeline
star-history.com


cloudnative-pg/cloudnative-pg
EnterpriseDB/repmgr
zalando/patroni
EnterpriseDB/barman
postgres/postgres
pgpool/pgpool2
pgbackrest/pgbackrest
pgbouncer/pgbouncer
pgadmin-org/pgadmin4
sorintlab/stolon
12.0k
10.0k
8.0k
6.0k
4.0k
2.0k
GitHub Stars
3 years
6 years
10 years
13 years
Timeline
star-history.com

cloudnative-pg / cloudnative-pg Public

Edit Pins | Unwatch 20 | Fork 106 | Starred 1.4k

Code | Issues 60 | Pull requests 29 | Discussions | Actions | Projects 1 | Wiki | Security | Insights | Settings

cloudnative-pg

Installation | OS / Arch 3 | Learn more about packages

Install from the command line

```
$ docker pull ghcr.io/cloudnative-pg/cloudnative-pg:1.20.0
```

Recent tagged image versions

latest 1.20.0 — 646,912
Published about 1 month ago · Digest

1.19.2 — 30,011
Published about 1 month ago · Digest

1.18.4 — 29,612
Published about 1 month ago · Digest

1.17.5 — 47,275
Published 3 months ago · Digest

Details

cloudnative-pg
cloudnative-pg
Apache License 2.0

Last published
1 month ago

Discussions
140

Issues
89

Total downloads
8.46M

Collaborators 5

mnencia Marco Nenciarini
gbartolini Gabriele Bartolini
sxd Jonathan Gonzalez V.
leonardoce Leonardo Cecchi

Quickstart

EDB

Installing CloudNativePG

1. Via the official Kubernetes manifest
2. Via a custom Kubernetes manifest generated by the plug-in
3. Via the official Helm Chart

```
kubectl apply -f \
  https://raw.githubusercontent.com/cloudnative-pg/cloudnative-pg/main/releases/cnpg-1.20.1.yaml
```

Deploy a 3-node HA Postgres Cluster

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

myapp-db.yaml

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

postgresql.conf

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

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

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

replication

storage

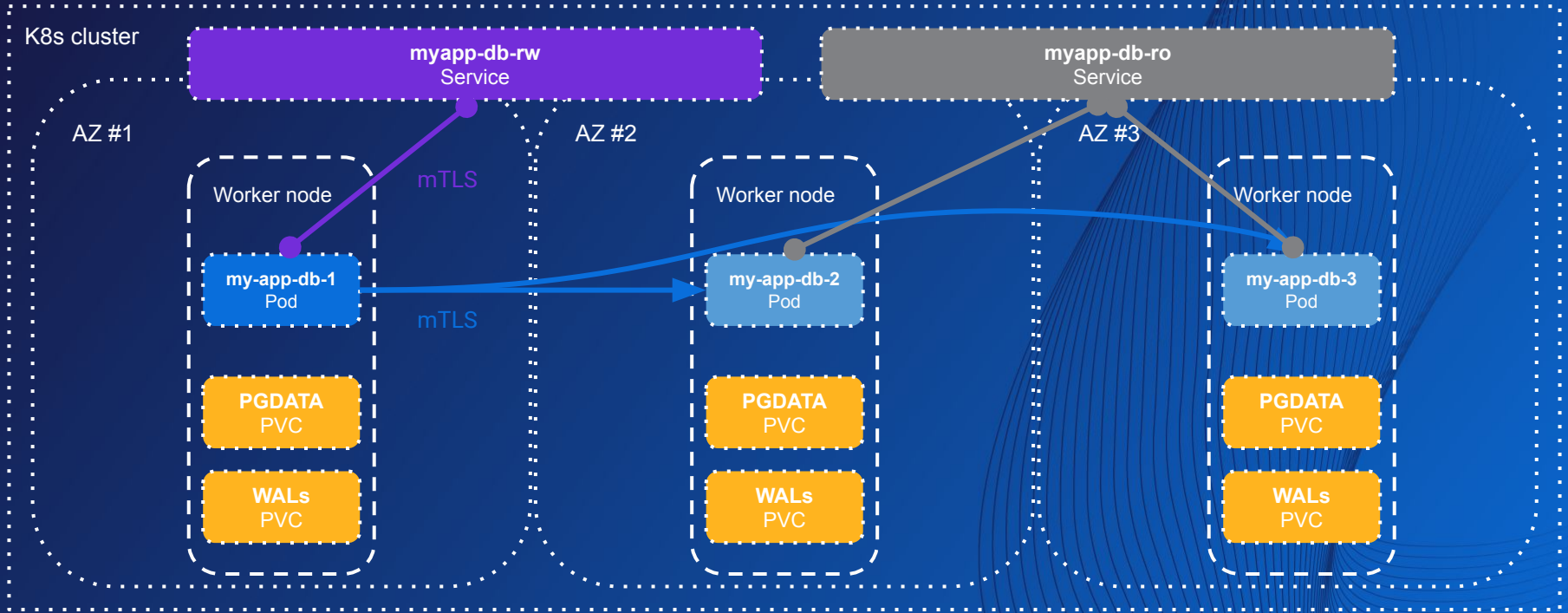
EDB

How to deploy the PostgreSQL Cluster

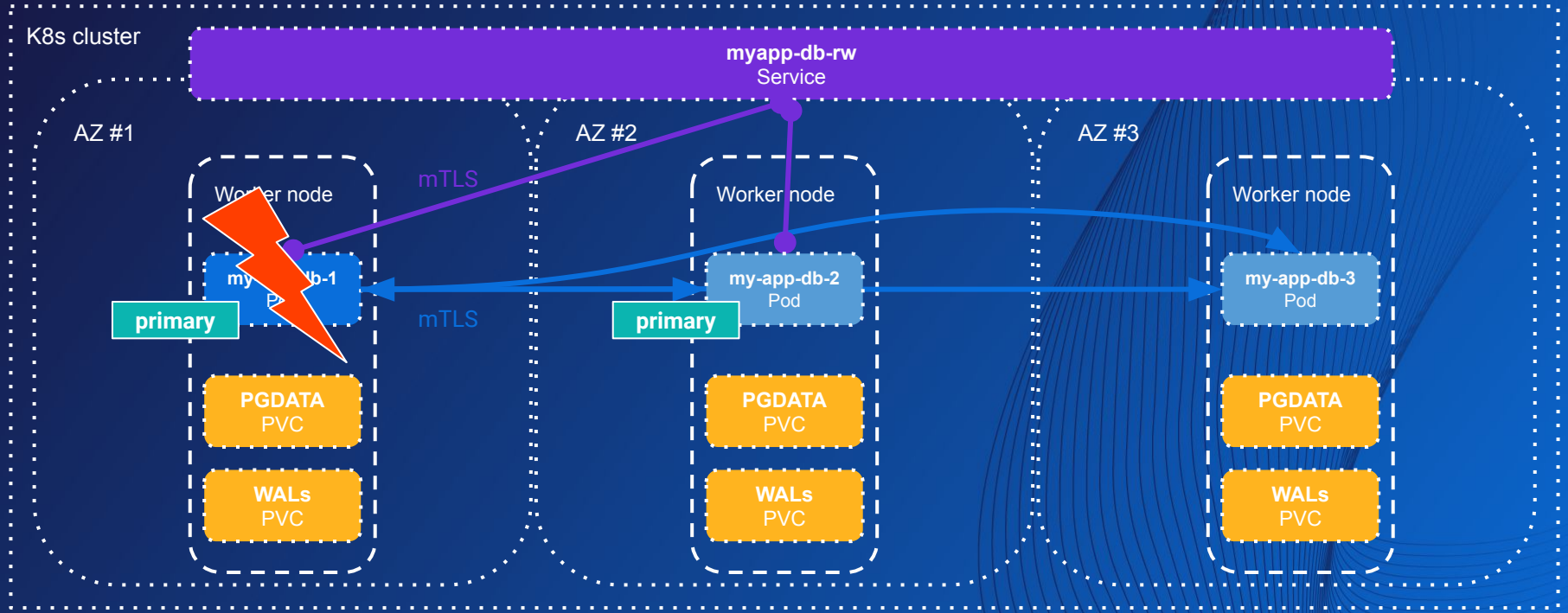
```
kubectl apply -f myapp-db.yaml
```

EDB

This is what happens under the hood



Automated failover



Main capabilities

Bootstrap

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

EDB

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

Gabriele Bartolini

August 15, 2022

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

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

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

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

Blog article

EDB

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

EDB

Backup and Recovery

- Continuous physical backup on “backup object stores”
  - Scheduled and on-demand base backups
  - Continuous WAL archiving (including parallel)
  - From primary (default) 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

EDB

Native Prometheus exporter for monitoring

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

Logging

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

The “cnpg” plugin for kubectl

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

```
muddy:cnpg gabriele.bartolini$ k cnpg status cnpg-eu-central-1
Cluster Summary
Name:               cnpg-eu-central-1
Namespace:          default
System ID:          7205268042997518356
PostgreSQL Image:   ghcr.io/cloudnative-pg/postgresql:15
Primary instance:   cnpg-eu-central-1-1
Status:             Cluster in healthy state
Instances:          3
Ready instances:    3
Current Write LSN:  3/34000000 (Timeline: 1 - WAL File: 000000010000000030000000C)

Certificates Status
Certificate Name               Expiration Date                Days Left Until Expiration
----------------               ---------------                --------------------------
cnpg-eu-central-1-ca           2023-05-29 17:54:38 +0000 UTC  89.41
cnpg-eu-central-1-replication  2023-05-29 17:54:38 +0000 UTC  89.41
cnpg-eu-central-1-server       2023-05-29 17:54:38 +0000 UTC  89.41

Continuous Backup status
First Point of Recoverability:  2023-02-28T18:01:47Z
Working WAL archiving:          OK
WALs waiting to be archived:    0
Last Archived WAL:              000000010000000030000000C   @   2023-03-01T00:30:28.211457Z
Last Failed WAL:                000000010000000000000001   @   2023-02-28T18:00:38.10286Z

Streaming Replication status
Replication Slots Enabled
Name                Sent LSN    Write LSN   Flush LSN   Replay LSN  Write Lag  Flush Lag  Replay Lag  State      Sync State  Sync Priority  Replication Slot
----                --------    ---------   ---------   ----------  ---------  ---------  ----------  -----      ----------  -------------  ----------------
cnpg-eu-central-1-2 3/34000000  3/34000000  3/34000000  3/34000000  00:00:00   00:00:00   00:00:00    streaming  quorum      1              active
cnpg-eu-central-1-3 3/34000000  3/34000000  3/34000000  3/34000000  00:00:00   00:00:00   00:00:00    streaming  quorum      1              active

Unmanaged Replication Slot Status
No unmanaged replication slots found

Instances status
Name                Database Size  Current LSN  Replication role  Status  QoS         Manager Version  Node
----                -------------  -----------  ----------------  ------  ---         ---------------  ----
cnpg-eu-central-1-1  15 GB          3/34000000   Primary           OK      Guaranteed  1.19.0           ip-192-168-23-150.eu-central-1.compute.internal
cnpg-eu-central-1-2  15 GB          3/34000000   Standby (sync)    OK      Guaranteed  1.19.0           ip-192-168-87-100.eu-central-1.compute.internal
cnpg-eu-central-1-3  15 GB          3/34000000   Standby (sync)    OK      Guaranteed  1.19.0           ip-192-168-41-249.eu-central-1.compute.internal
```

Connection pooling with PgBouncer

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

Major features added in the last year

- Fencing (1.15)
- LDAP authentication (1.15)
- Offline data import (1.16)
- Offline major upgrades for PostgreSQL (1.16)
- Separate volume for WAL files (1.17)
- Cluster-managed physical replication slots for High Availability (1.18)
- Backup from a standby (1.19)
- Delayed failover (1.19)
- Declarative role management (1.20)
- Declarative Postgres cluster hibernation (1.20)

EDB

Major features added in the last year

1.15 (April 2022)

- First open source release
- Fencing
- LDAP authentication

1.16 (July 2022)

- Offline data import
- Offline major upgrades

1.17 (September 2022)

- Separate volume for WALs

1.18 (November 2022)

- Cluster-managed physical replication slots for HA

1.19 (February 2023)

- Backup from a standby
- Delayed failover

1.20 (April 2023)

- Role management
- Postgres cluster hibernation

EDB

The road ahead

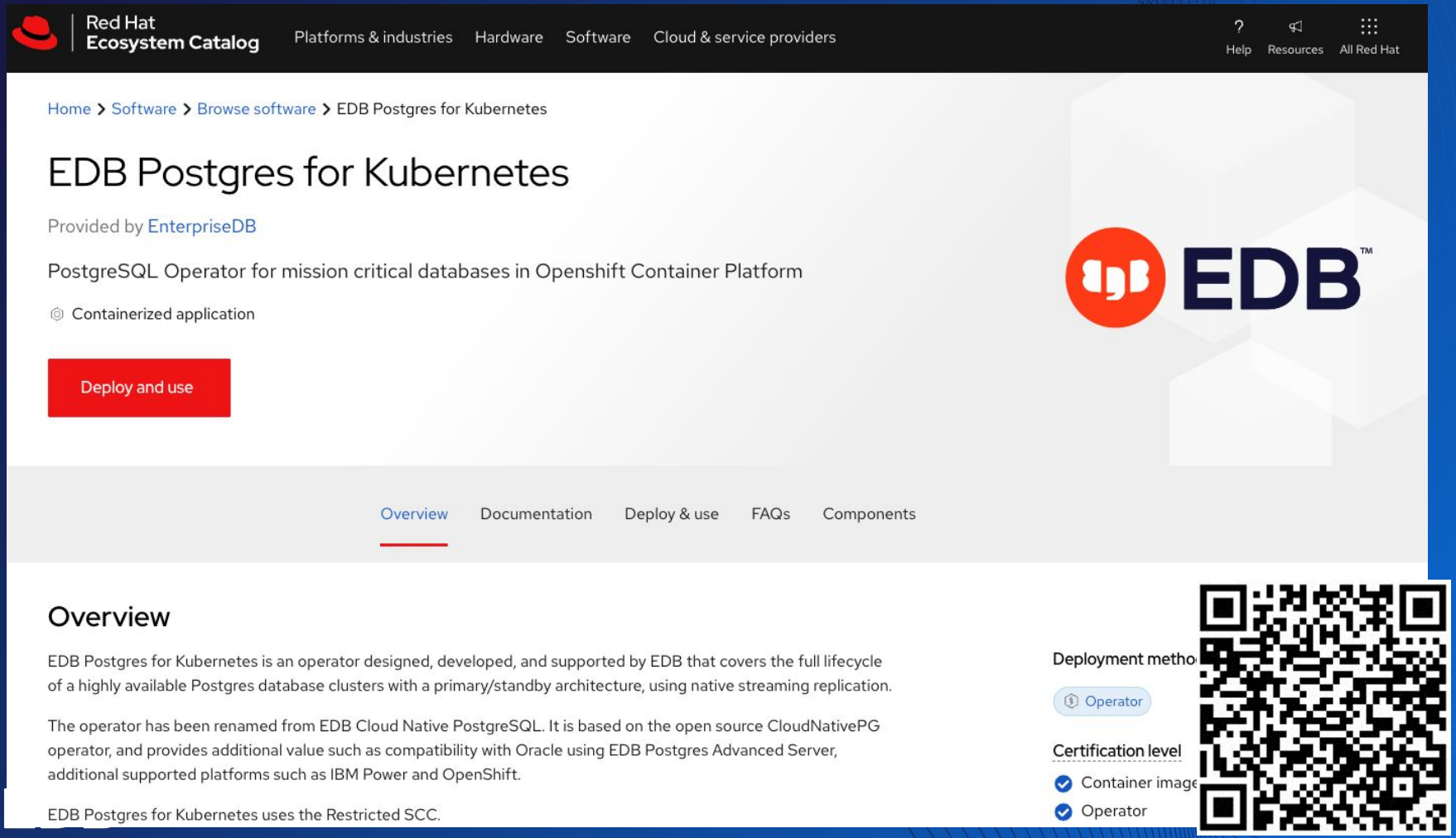
- **pg_failover_slots for CDC use cases (1.20.1!)**
- Support for Kubernetes ‘VolumeSnapshot’ API
  - Transparent incremental & differential copy for backup & recovery
  - PVC cloning
  - In-place major upgrades with pg_upgrade
  - Data mobility at storage level
- Declarative tablespaces
  - Including temporary ones
- PostgreSQL 16
- Replica cluster switchover enhancements
- Major online upgrades through logical replication

EDB

Conclusions

CloudNativePG has been originally created by

Reach out to us for support on the entire stack!


Red Hat
Ecosystem Catalog
Platforms & industries
Hardware
Software
Cloud & service providers
Help
Resources
All Red Hat
Home > Software > Browse software > EDB Postgres for Kubernetes
EDB Postgres for Kubernetes
Provided by EnterpriseDB
PostgreSQL Operator for mission critical databases in Openshift Container Platform
Containerized application
Deploy and use
EDB
Overview
Documentation
Deploy & use
FAQs
Components
Overview
EDB Postgres for Kubernetes is an operator designed, developed, and supported by EDB that covers the full lifecycle of a highly available Postgres database clusters with a primary/standby architecture, using native streaming replication.
The operator has been renamed from EDB Cloud Native PostgreSQL. It is based on the open source CloudNativePG operator, and provides additional value such as compatibility with Oracle using EDB Postgres Advanced Server, additional supported platforms such as IBM Power and OpenShift.
EDB Postgres for Kubernetes uses the Restricted SCC.
Deployment metho
Operator
Certification level
Container image
Operator

Choose EDB!

- Help us develop the open source PostgreSQL and CloudNativePG!
- **Open source mandate?** Choose **CloudNativePG** and **Community 360 plan**
  - Only latest 2 versions supported - OK to live on the edge counting on break/fix support
- **Need Oracle compatibility?** Choose **EPG4K** and the **Enterprise plan**
- For all the other cases, choose **EPG4K** and the **Standard plan**
  - OpenShift certified operator
  - Long Term Supported version (1.18 will be supported until April 2024)

EDB

Freedom

Run Postgres anywhere!
It's your choice!

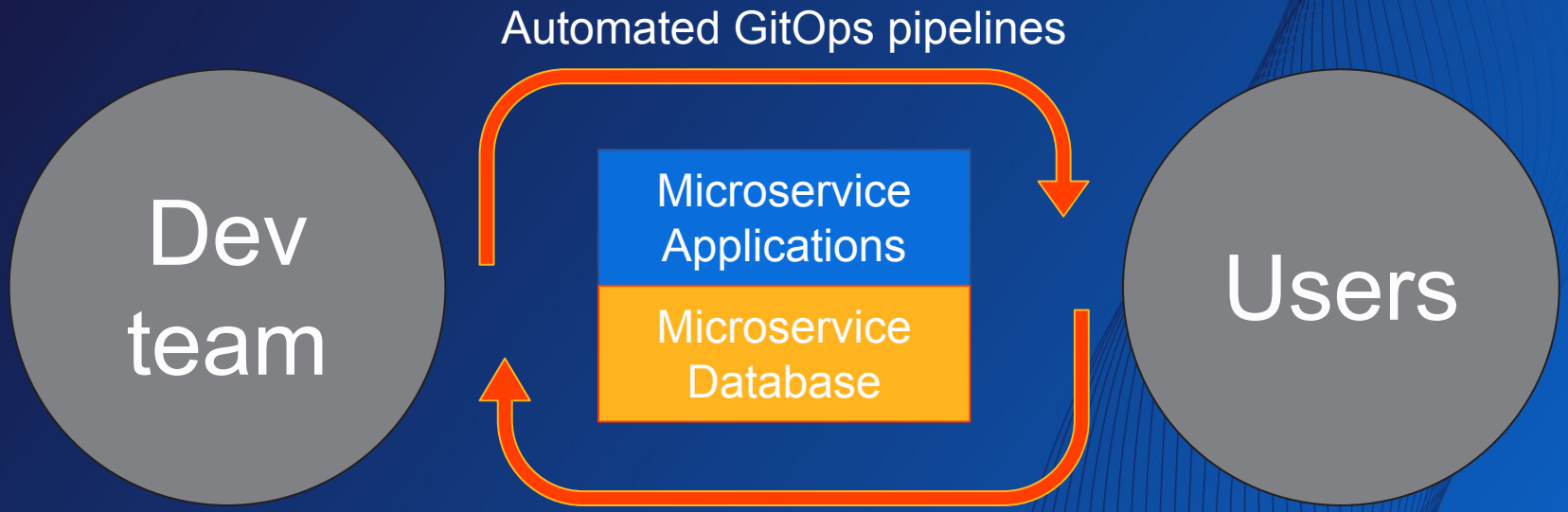
EDB™

Own your data

Retain full control of your databases and infrastructure

devops

“Generative culture”
more than tools or processes



Increased velocity

EDB

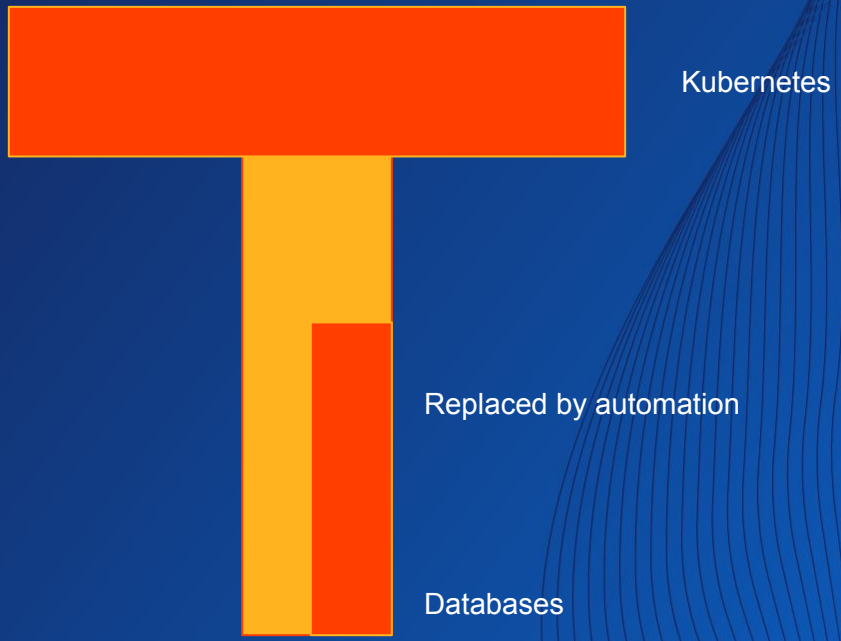
On Kubernetes or In Kubernetes?

EDB

How CloudNativePG changes the Postgres DBA role

- Most infrastructure related problems are automated
- You as a DBA are crucial in the organization
  - Leverage skills and experience from traditional environments
  - Subject Matter Expert of PostgreSQL in a multidisciplinary DevOps team
    - Stream-aligned team or Platform team (according to the “Team Topologies” book)
- Unlearn to learn
- Protect Postgres, from Day 0:
  - **Infrastructure:** help choose the right architecture and storage for Postgres!
  - **Application:** model the database with developers!
- Examples of day 2 operations:
  - Infrastructure: monitoring, alerting, backup verification
  - Application: query optimization, index optimization, data modeling

Long live the Postgres DBA



EDB