CloudNativePG: PostgreSQL on Kubernetes

Vojtěch Mareš, Prague PostgreSQL Developer Day 2025

Vojtěch Mareš

Freelance DevOps engineer, consultant, lector

X	@vojtechmares_
GitHub	@vojtechmares
Web	www.mares.cz

Agenda

- Kubernetes introduction
- Stateful appliactions on Kubernetes
- Databases onKubernetes
- PostgreSQL on Kubernetes
- Not only databases
- CloudNativePG

What is Kubernetes

- Container orchestration platform (build with Docker, run with Kubernetes across many machines)
  - Many machines (*Nodes*) join cluster and the workload is spread across all of them given some rules for allocating resources
- Running containers at scale (from a single machine to large clusters with hundreds or thousands of nodes)
- Declarative approach – you define how many instances (*Pods*) you want
- Extensible – everything is built with APIs, so it’s easy to add features – operators

Kubernetes glossary

- *Pod* – smallest deployable unit, one or more containers
- *Service* – network interface of *Pods* in Kubernetes cluster
- *Deployment* – Stateless application like web server with X amount of *Pods*
- *StatefulSet* – *Deployment*, but for stateful application
- *Persistent Volume* – Volume (network attached or local)
- *Persistent Volume Claim* – attachment of *PV* to a *Pod*

Working with Kubernetes

- Everything is in YAML file
- A lot of command line work (thanks `kubectl`)
- Operatos – programs running on Kubernetes extending it’s capabilities and handling domain-specific issues

Stateful applications on Kubernetes

- Databases (relational, NoSQL, KV,...)
- Message broker (RabbitMQ, Kafka,...)
- StatefulSet
  - No concern over who is leader
- Operators & Custom Resources (extending Kubernetes)

Databases on Kubernetes

- StatefulSet
  - Does not handle leader/primary election
  - One or more StatefulSets?
  - “Koordinator” as sidecar?
  - ”Koordinator” as stanalone application?
- Kubernetes Operator

PostgreSQL on Kubernetes

- StatefulSet
- Helm Chart (Bitnami?)
- Zalando Operator (Patroni 😮💨)
- CunchyData (license 👎)
- CloudNativePG ❤️

It’s not only about database

- Connection pooler
  - Pgpool
  - pgBouncer
  - HAProxy

CloudNativePG

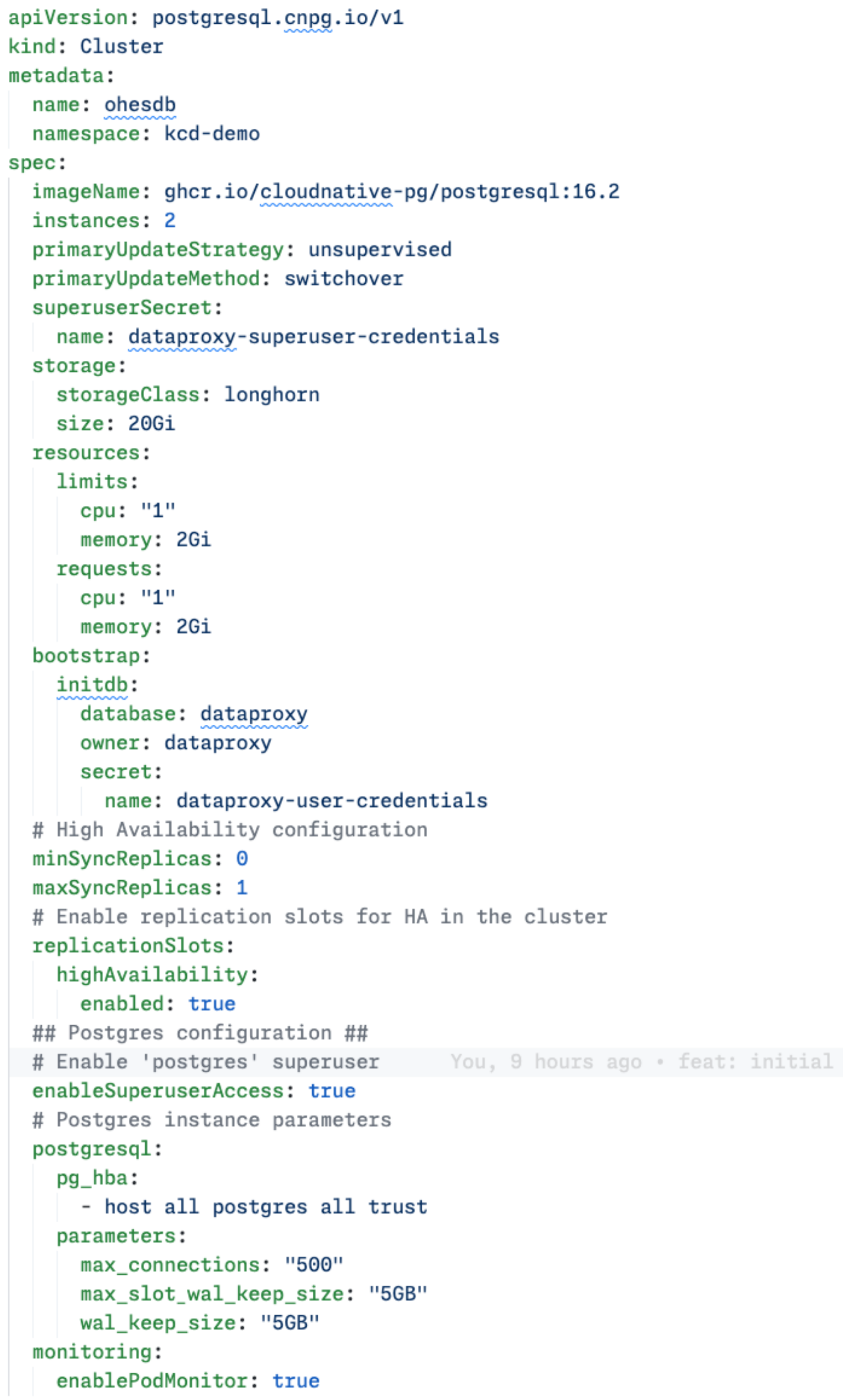
cloudnative-pg.io

What is CNPG?

- Open-source from EnterpriseDB
- Accepted as Cloud Native Compute Foundation Sandbox project (currently onboarding)
- Kubernetes Operator
  - Level 5 (autopilot)
- 5.2k ⭐

Postgres cluster

- Manifest (CRD)
- Supports native Kubernetes Secrets
- PostgreSQL parameters
- 3 Kubernetes Services
    - Read-Write
    - Read-Only
    - Replicas



```yaml
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: ohesdb
  namespace: kcd-demo
spec:
  imageName: ghcr.io/cloudnative-pg/postgresql:16.2
  instances: 2
  primaryUpdateStrategy: unsupervised
  primaryUpdateMethod: switchover
  superuserSecret:
    name: dataproxy-superuser-credentials
  storage:
    storageClass: longhorn
    size: 20Gi
  resources:
    limits:
      cpu: "1"
      memory: 2Gi
    requests:
      cpu: "1"
      memory: 2Gi
  bootstrap:
    initdb:
      database: dataproxy
      owner: dataproxy
      secret:
        name: dataproxy-user-credentials
  # High Availability configuration
  minSyncReplicas: 0
  maxSyncReplicas: 1
  # Enable replication slots for HA in the cluster
  replicationSlots:
    highAvailability:
      enabled: true
  ## Postgres configuration ##
  # Enable 'postgres' superuser      You, 9 hours ago • feat: initial
  enableSuperuserAccess: true
  # Postgres instance parameters
  postgresql:
    pg_hba:
      - host all postgres all trust
    parameters:
      max_connections: "500"
      max_slot_wal_keep_size: "5GB"
      wal_keep_size: "5GB"
  monitoring:
    enablePodMonitor: true
```

Initialized database

- initdb
- Database
  - Owner
  - User (Secret)
- Or…
  - Restore database from a backup
  - Create database/cluster from existing database cluster

Backups and WAL archiving

- Backups goes to object storage (S3/GCS/ASB)
- Backup of Kubernetes volumes (Velero)
- WAL archiving → Point In Time Recovery
- Automatic backups (ScheduledBackup)
- On-demand backup (via kubectl cli)

Restore cluster from backup

- Restore cluster from backup
- Could easily be done for feature branches with Git

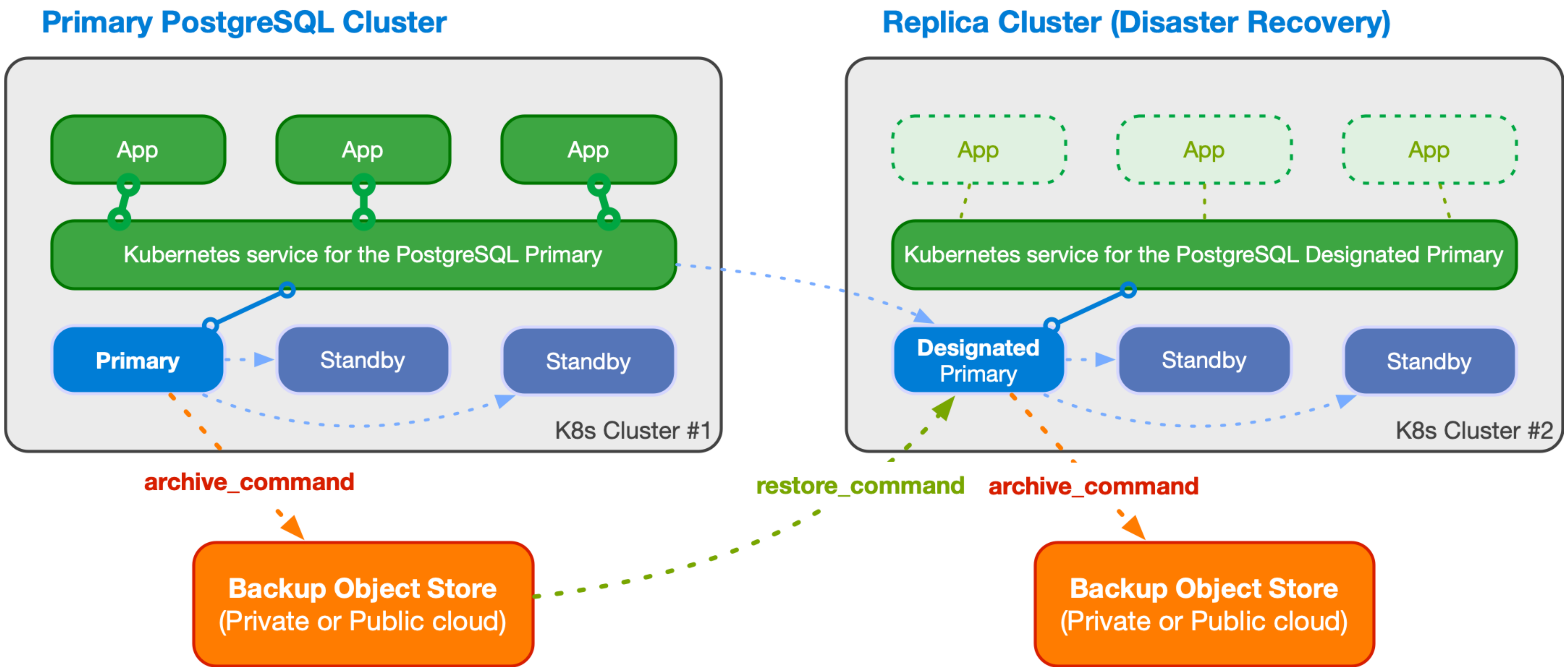
Replication and replicas

- Postgres native replication
  - Streaming replication (sync/async)
- 2 kinds of replicas
  - Synchronyous (read-only traffic or failover target)
  - Asynchronyous (Write performance)
- Replication slots
  - Dedicated replication connections between instances
- Support for Kubernetes affinity (spread *Pods* across machines)

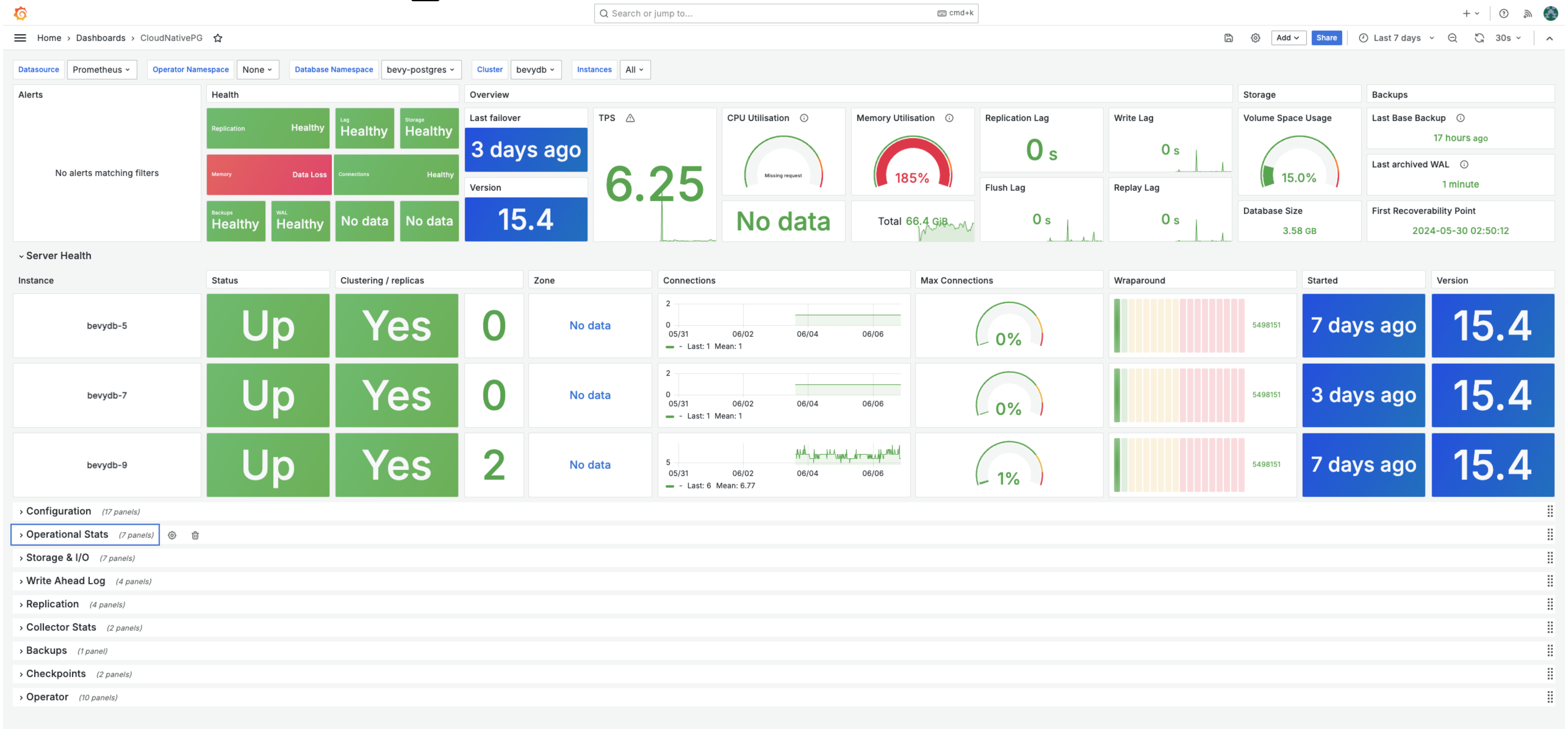
pgBouncer

```yaml
apiVersion: postgresql.cnpg.io/v1
kind: Pooler
metadata:
  name: pgbouncer
  namespace: bevy-postgres
  annotations:
    argocd.argoproj.io/sync-wave: "20"
spec:
  cluster:
    name: bevydb
  instances: 3
  type: rw
  pgbouncer:
    poolMode: session
    parameters:
      # 3 replicas with 100 connections = 300 connections total
      # postgres has max of 500 connections
      max_client_conn: "100"
      default_pool_size: "10"
      ignore_startup_parameters: "search_path"
  deploymentStrategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
  monitoring:
    enablePodMonitor: true
  # PodTemplateSpec
  template:
    metadata:
      labels:
        app.kubernetes.io/name: pooler
    spec:
      containers: [] # suppress error
      affinity:
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
            - weight: 100
              podAffinityTerm:
                labelSelector:
                  matchExpressions:
                    - key: app.kubernetes.io/name
                      operator: In
                      values:
                        - pooler
                topologyKey: kubernetes.io/hostname # node hostname
```

Replica cluster


Primary PostgreSQL Cluster
Replica Cluster (Disaster Recovery)
App
App
App
Kubernetes service for the PostgreSQL Primary
Primary
Standby
Standby
K8s Cluster #1
App
App
App
Kubernetes service for the PostgreSQL Designated Primary
Designated Primary
Standby
Standby
K8s Cluster #2
archive_command
restore_command
archive_command
Backup Object Store (Private or Public cloud)
Backup Object Store (Private or Public cloud)

Monitoring



Search or jump to... cmd+k

Home › Dashboards › CloudNativePG

Add · Share · Last 7 days · 30s

Datasource: Prometheus · Operator Namespace: None · Database Namespace: bevy-postgres · Cluster: bevydb · Instances: All

Alerts

No alerts matching filters

Health

Replication	Healthy
Lag	Healthy
Storage	Healthy
Memory	Data Loss
Connections	Healthy
Backups	Healthy
WAL	Healthy
	No data
	No data

Overview

Last failover	3 days ago
Version	15.4
TPS	6.25
CPU Utilisation	Missing request / No data
Memory Utilisation	185% / Total 66.4 GiB
Replication Lag	0 s
Write Lag	0 s
Flush Lag	0 s
Replay Lag	0 s

Storage

Volume Space Usage	15.0%
Database Size	3.58 GB

Backups

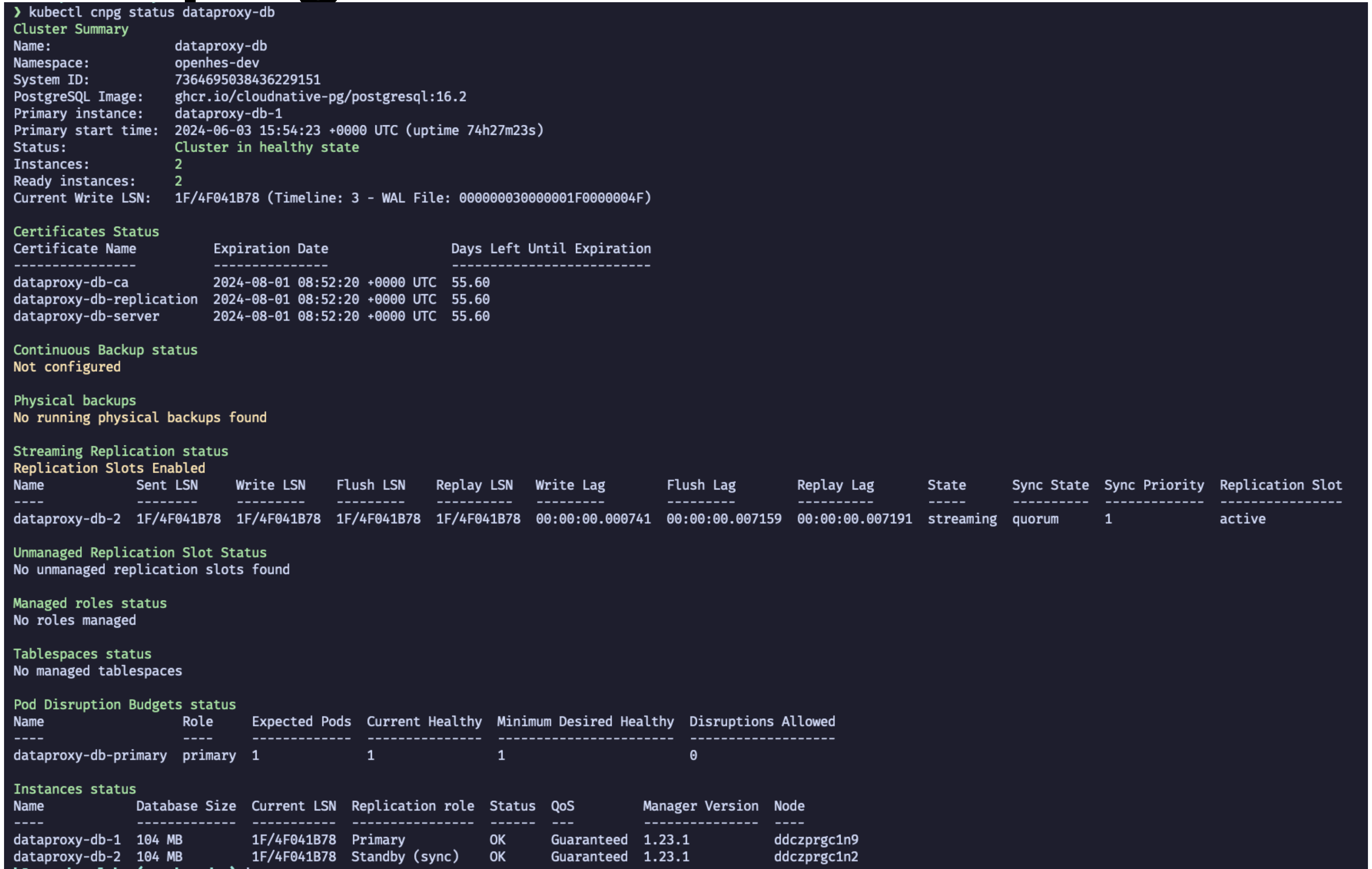
Last Base Backup	17 hours ago
Last archived WAL	1 minute
First Recoverability Point	2024-05-30 02:50:12

Server Health

Instance	Status	Clustering / replicas		Zone	Connections	Max Connections	Wraparound	Started	Version
bevydb-5	Up	Yes	0	No data	Last: 1 Mean: 1	0%	5498151	7 days ago	15.4
bevydb-7	Up	Yes	0	No data	Last: 1 Mean: 1	0%	5498151	3 days ago	15.4
bevydb-9	Up	Yes	2	No data	Last: 6 Mean: 6.77	1%	5498151	7 days ago	15.4

- › Configuration (17 panels)
- › Operational Stats (7 panels)
- › Storage & I/O (7 panels)
- › Write Ahead Log (4 panels)
- › Replication (4 panels)
- › Collector Stats (2 panels)
- › Backups (1 panel)
- › Checkpoints (2 panels)
- › Operator (10 panels)

kubectl plugin



```
❯ kubectl cnpg status dataproxy-db
Cluster Summary
Name:                dataproxy-db
Namespace:           openhes-dev
System ID:           7364695038436229151
PostgreSQL Image:    ghcr.io/cloudnative-pg/postgresql:16.2
Primary instance:    dataproxy-db-1
Primary start time:  2024-06-03 15:54:23 +0000 UTC (uptime 74h27m23s)
Status:              Cluster in healthy state
Instances:           2
Ready instances:     2
Current Write LSN:   1F/4F041B78 (Timeline: 3 - WAL File: 000000030000001F0000004F)

Certificates Status
Certificate Name         Expiration Date                Days Left Until Expiration
----------------         ---------------                --------------------------
dataproxy-db-ca          2024-08-01 08:52:20 +0000 UTC  55.60
dataproxy-db-replication 2024-08-01 08:52:20 +0000 UTC  55.60
dataproxy-db-server      2024-08-01 08:52:20 +0000 UTC  55.60

Continuous Backup status
Not configured

Physical backups
No running physical backups found

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
----            --------     ---------    ---------    ----------   ---------        ---------        ----------       -----      ----------  -------------  ----------------
dataproxy-db-2  1F/4F041B78  1F/4F041B78  1F/4F041B78  1F/4F041B78  00:00:00.000741  00:00:00.007159  00:00:00.007191  streaming  quorum      1              active

Unmanaged Replication Slot Status
No unmanaged replication slots found

Managed roles status
No roles managed

Tablespaces status
No managed tablespaces

Pod Disruption Budgets status
Name                  Role     Expected Pods  Current Healthy  Minimum Desired Healthy  Disruptions Allowed
----                  ----     -------------  ---------------  -----------------------  -------------------
dataproxy-db-primary  primary  1              1                1                        0

Instances status
Name            Database Size  Current LSN  Replication role  Status  QoS         Manager Version  Node
----            -------------  -----------  ----------------  ------  ---         ---------------  ----
dataproxy-db-1  104 MB         1F/4F041B78  Primary           OK      Guaranteed  1.23.1           ddczprgc1n9
dataproxy-db-2  104 MB         1F/4F041B78  Standby (sync)    OK      Guaranteed  1.23.1           ddczprgc1n2
```

Questions? 🙋‍♀️

Follow me

X @vojtechmares_ | GitHub @vojtechmares | www.mares.cz