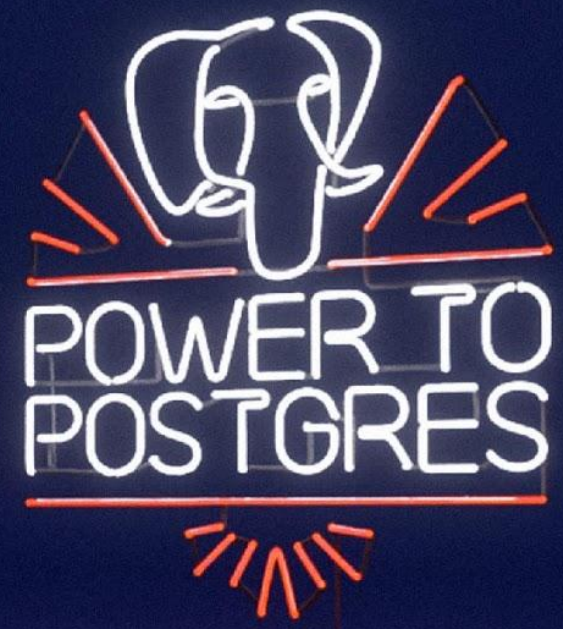
Deploying and Optimizing Clusters with Cloud Native PostgreSQL

Adam Wright | Senior Product Manager

October 2021



EDB

Agenda

1. Recap - Cloud Native PostgreSQL
2. Deploying PostgreSQL on K8s
3. Connecting to PostgreSQL
4. Insights
5. Backups
6. Recap - CNP Capabilities
7. Next steps

Quick recap

EDB's qualifications

- Kubernetes Certified Service Provider (KCSP)
  - First PostgreSQL Company to reach this status

- Silver Member of CNCF & Linux Foundation

- Red Hat Certified Kubernetes Operators
  - Cloud Native PostgreSQL (aka PostgreSQL Operator)
  - Cloud Native BDR (aka BDR Operator)

- Platinum founding sponsor of the Data on Kubernetes Community

What's a Kubernetes operator?

Extends Kubernetes controller and defines how a complex application works

- A Kubernetes operator automates actions of a human being, in a programmatic way
- **A PostgreSQL cluster is a complex application**
  - Deployment and configuration
  - Failure detection and Failover
  - Updates and switchovers
  - Backup and Recovery
- Relies on Kubernetes' native components and capabilities:
  - Self-healing, high availability, scalability, resource control, access, …
  - Declarative and fully automated

Available images

- Hosted on Quay.io
- Regularly rebuilt and scanned for vulnerabilities
  - Updating dependencies as soon as fixes are available upstream
- **Operator images:** quay.io/enterprisedb/cloud-native-postgresql
  - multi-arch support: amd64, ppc64le and s390x
- **Operand images**
  - PostgreSQL: quay.io/enterprisedb/postgresql
  - EnterpriseDB PostgreSQL Advanced Server (EPAS): quay.io/enterprisedb/edb-postgres-advanced
  - multi-arch support: amd64, ppc64le and s390x

Operand images

- PostgreSQL:
  - **Debian** and **UBI** based
  - all the **supported versions PostgreSQL** (10 .. 13.4 and 14 in preview)
  - additional libraries for CNP: **Barman Cloud, PGAudit, PostGIS**
  - **Dockerfiles**: https://github.com/EnterpriseDB/docker-postgresql
- EPAS (licensed):
  - **UBI** based
  - all the **supported versions of EPAS** (10 .. 14)
  - additional libraries: **Barman cloud, EDB Audit**

Deploying PostgreSQL on K8s

Convention over configuration

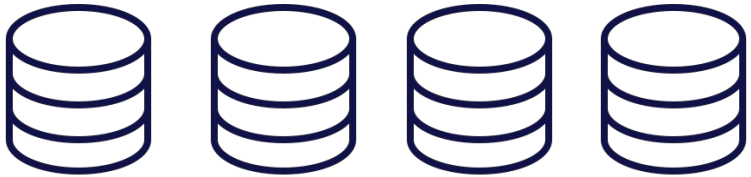
```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: myapp-db
spec:
  instances: 3
  imageName: quay.io/enterprisedb/postgresql:13.4-3

  storage:
    size: 10Gi
```


Declarative deployment

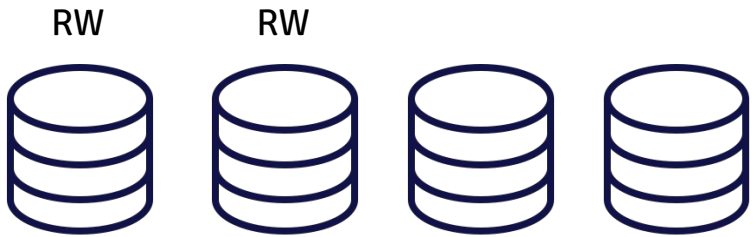
```
# Install the operator in the cluster
kubectl apply -f <OPERATOR_MANIFEST_URL>

# Deploy the cluster (declarative)
kubectl apply -f myapp-cluster.yaml
```


EDB

Desired state



Actual state

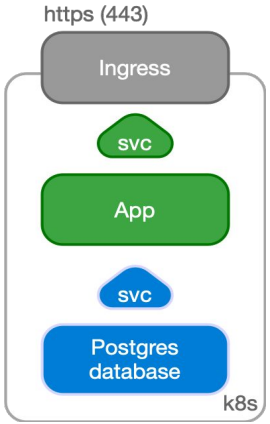


Connecting to PostgreSQL

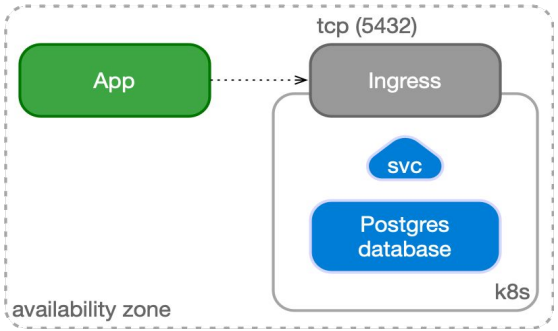
System architectures

Main classification is based on where the application reside

Use case 1: Application and Database in K8s

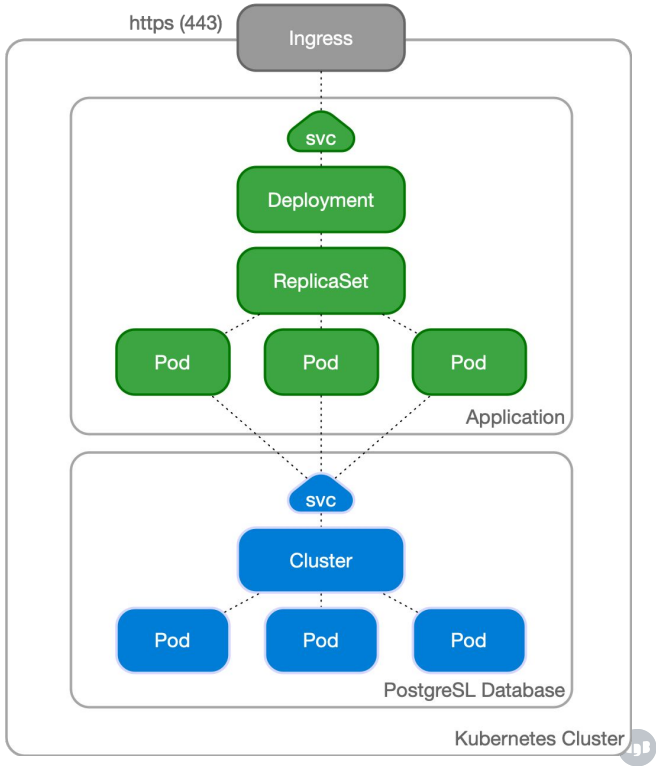


Use case 2: Application outside K8s



Application and Database in K8s

- Stateless application as a deployment
  - Rolling upgrades
  - ReplicaSet for scaling and HA
  - Custom application images (Go, Django, Java, Python, C, C++, …)
- Stateful database using our operator
  - Embeds primary/standby logic
  - Service for RW and Read operations
  - Rolling upgrades, scaling, HA, …
  - “Cluster” CRD



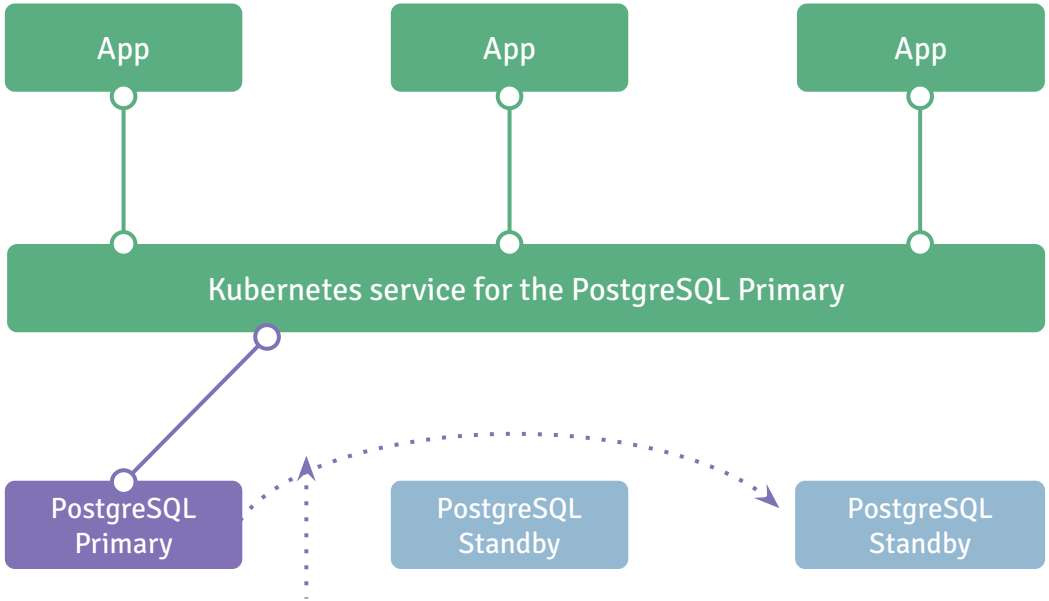
Reliable DNS endpoints
for your application

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
   name: pg-ha-dolores
spec:
   instances:  3
```

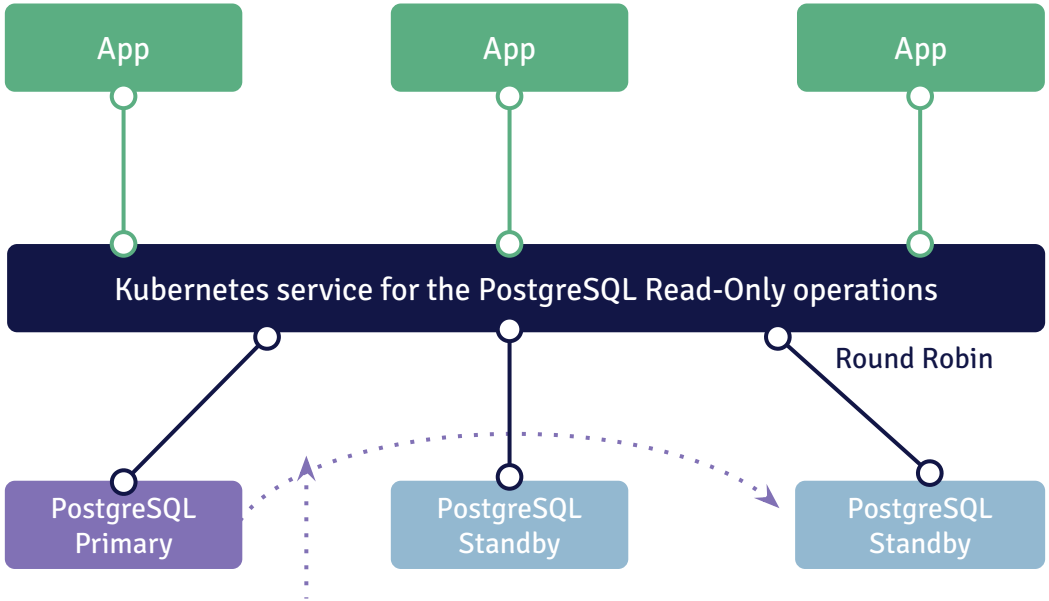
```
#  kubectl apply -f myapp-db.yaml
```

```
#  kubectl get services -o wide
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    SELECTOR
pg-ha-dolores-any   ClusterIP   10.96.184.65    ######        5432/TCP   postgresql=pg-ha-dolores
pg-ha-dolores-r     ClusterIP   10.96.225.137   ######        5432/TCP   postgresql=pg-ha-dolores
pg-ha-dolores-ro    ClusterIP   10.96.129.1     ######        5432/TCP   postgresql=pg-ha-dolores, role=replica
pg-ha-dolores-rw    ClusterIP   10.96.168.65    ######        5432/TCP   postgresql=pg-ha-dolores, role=primary
```

Read-write workloads (“-rw” service)



Read workloads (“-r” service)



Insights

Observability in Cloud Native PostgreSQL

- **Metrics:**
  - Native Prometheus exporter for the operator
  - Native Prometheus exporter for each PostgreSQL instance
- **Logs:**
  - Native support for Kubernetes events
  - Direct stdout logging in JSON format
    - Operator messages
    - Operand messages (PostgreSQL logs)

Monitoring PostgreSQL

- Each PostgreSQL instance pod locally exposes an exporter
  - Via HTTP, port 9187
- User defined queries ConfigMap or Secret
  - Syntax compatible with PostgreSQL Prometheus Exporter
- All monitoring queries are:
  - transactionally atomic (one transaction per query)
  - executed as user **postgres**
  - executed with the **pg_monitor** role
  - executed with **application_name** set to **cnp_metrics_exporter**
  - executed against the main application database

Step 1: Define the metric in a ConfigMap (or Secret)

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-monitoring
  namespace: test
data:
  custom-queries: |
    pg_replication:
      query: "SELECT CASE WHEN NOT pg_is_in_recovery()
              THEN 0
              ELSE GREATEST (0,
                EXTRACT(EPOCH FROM (now() - pg_last_xact_replay_timestamp())))
              END AS lag"
      metrics:
        - lag:
            usage: "GAUGE"
            description: "Replication lag behind primary in seconds"
```

Step 2: Add it to the PostgreSQL cluster spec

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: cluster-example
  namespace: test
spec:
  instances: 3

  storage:
    size: 1Gi

  monitoring:
    customQueriesConfigMap:
      - name: example-monitoring        # ConfigMap must be in the same namespace
        key: custom-queries
```

Grafana dashboards from CNP metrics

Logging and Auditing

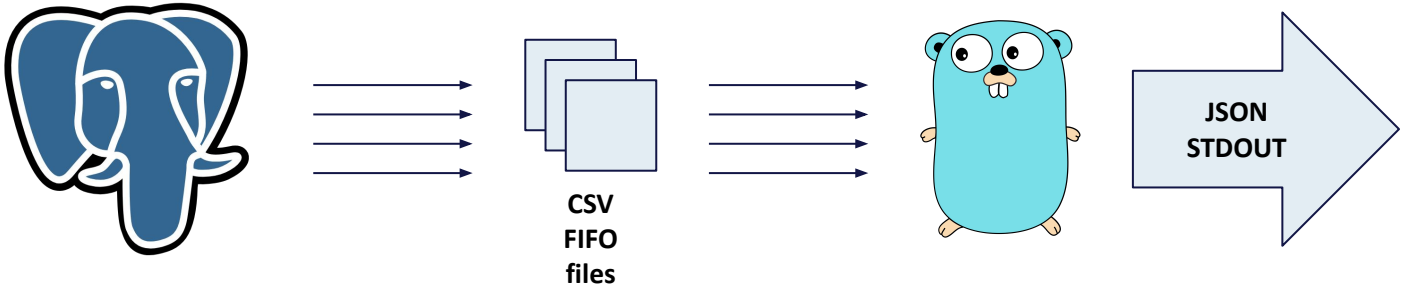
- **Logging:**
  - *logging_collector = “on”* and *"log_destination": "csvlog"* configure PostgreSQL to log in **CSV format on a given file.**
- **Auditing:**
  - **Pg Audit**: piggybacks on logging_collector logs, **CSV messages in a CSV log lines in a file.**
  - **EDB Audit**: can be configured to log to a **different file** with a specific **CSV format**.
- **We are on Kubernetes, we want JSON formatted logs on STDOUT.**

- **N.B.: Formats change with PostgreSQL versions**

Logging and Auditing

We chose to implement a custom log reader:

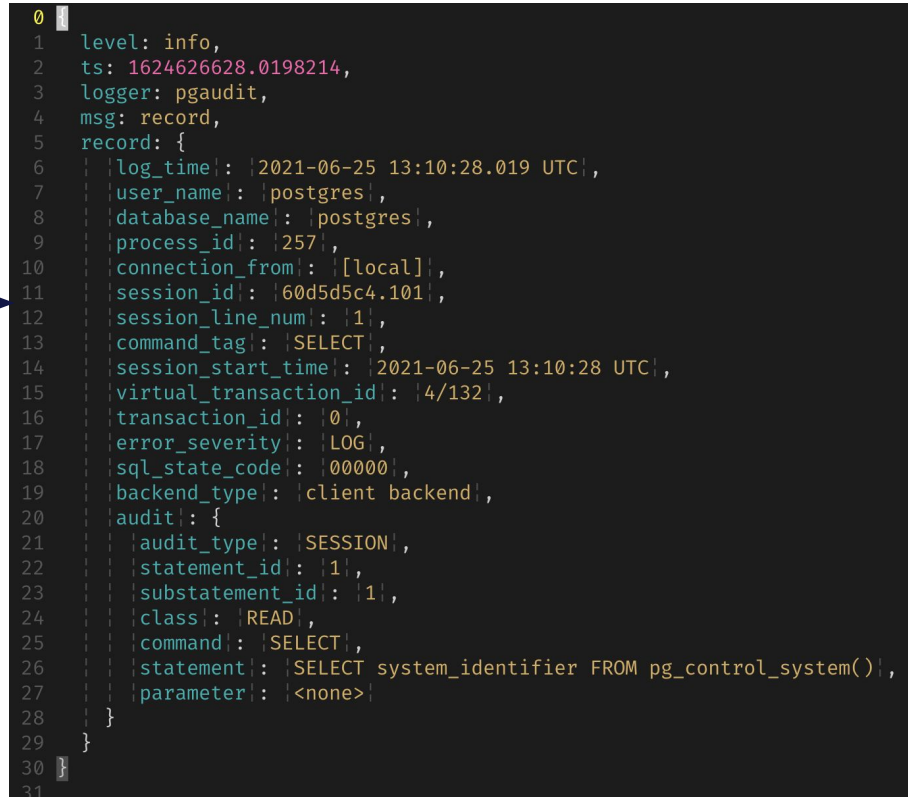
- reads all known format of CSV logs for logging_collector, PG Audit and EDB Audit
- outputs them in JSON format to STDOUT



PGAudit JSON logs

```
postgresql:
  parameters:
    "pgaudit.log": "all, -misc"
    "pgaudit.log_catalog" : "off"
    "pgaudit.log_parameter" : "on"
    "pgaudit.log_relation" : "on"
```

pgaudit extension automatically created in all databases



Backups

Continuous physical backup

- Scheduled and on-demand
- Support for object stores (S3 compatible)
  - Public clouds
    - Today: Amazon S3, Azure Blob Storage
    - Coming Soon: IBM Cloud Object Storage (COS)
  - Private clouds (e.g. MinIO)
- Rely on Barman Cloud technology
  - barman-cloud-wal-archive
  - barman-cloud-backup

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Backup
metadata:
  name: backup-example
spec:
  cluster:
    name: pg-backup
```

Recovery

- Create a new cluster from a backup
- Restore the base backup
  - Full or PITR
- Pull and replay backup WAL files
- Rely on Barman Cloud technology
  - barman-cloud-restore
  - barman-cloud-wal-restore

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: cluster-restore
spec:
  instances: 3

  storage:
    size: 5Gi

  bootstrap:
    recovery:
      backup:
        name: backup-example
```

Capabilities (Recap)

Deploy anywhere

On Premise, Multi/Hybrid Cloud

- Use the same container images from EDB
- Use the same Deployment files
- Only requires K8s 1.16 or higher
    - Tested on AWS, Google, Azure, and OpenShift

Automate DBA tasks

- Provision compute resources automatically
- Setup HA automatically with the Operator
- Update PostgreSQL minor versions in a rolling fashion
- Setup and maintain services for application connections
- Backup & recover data as needed
- Implement security best practices

Avoid lock-in

- Available on **your** platform of choice
- **You** choose between PostgreSQL or EDB Postgres Advanced
- Backup data to **your** storage bucket of choice

Next Steps

EDB docs

Don’t forget about the rich experience for Cloud Native Postgres in EDB Docs!

Visit the Cloud Native Postgres docs site:
https://www.enterprisedb.com/docs/kubernetes/cloud_native_postgresql/

- Detailed documentation
- Trial guidance
- Interactive demo (as visualized to the right!)