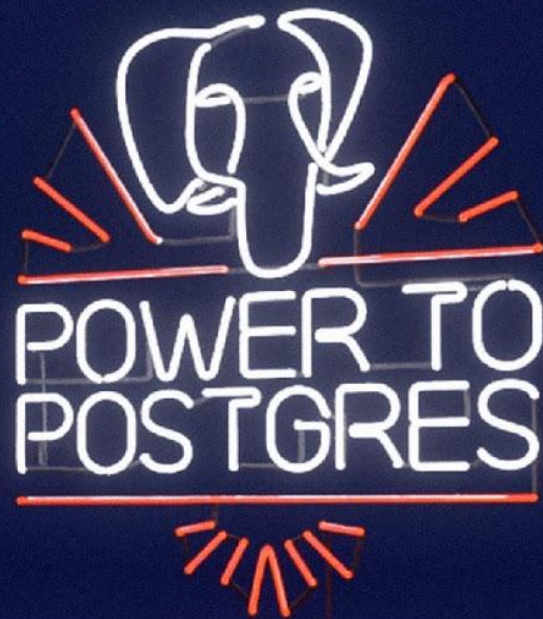


Develop Cloud-Native Apps on K8s with Postgres

Adam Wright | Sr. Product Manager

Wed Jun 30



Agenda

- Why EDB?
- PostgreSQL + K8s
- Declarative Cluster Configuration
- Deploy Anywhere
- Kubernetes Services
- Advanced database features, no DBA required
- Conclusions and recommended next steps



Why EDB?

The Largest PostgreSQL Company



5,300+

lifetime customers



Over 300

dedicated PostgreSQL
technologists



91%

customer
satisfaction rating



28%

of Fortune 500 companies
are customers



40%

of customers
deploying in cloud



2

Founders of Postgres
Community



The most PostgreSQL experts

EDB team includes:

300+ PostgreSQL technologists

26 PostgreSQL community contributors and committers

Including founders and leaders like



Michael Stonebraker

“Father of Postgres”
and EDB Advisor



Bruce Momjian

Co-founder, PostgreSQL
Development Corp and
PostgreSQL Core Team



Peter Eisentraut

PostgreSQL
Core Team member



Robert Haas

PostgreSQL Major
Contributor and Committer



Simon Riggs

PostgreSQL Major Contributor,
Founder
of 2ndQuadrant



EDB and Kubernetes



- 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)



What makes up Cloud Native PostgreSQL?

Downloadable software solution that includes [1] a Kubernetes Operator and [2] PostgreSQL/EPAS container images

Cloud Native PostgreSQL



PostgreSQL and EDB Postgres Advanced Container images

Docker container images containing the Database server with only the Postgres service exposed



Kubernetes Operator

Responsible for deploying and managing PostgreSQL and EDB Postgres Advanced containers and maintaining the desired state

What makes up Cloud Native PostgreSQL? **NEW!!**

Tools with working with Cloud Native PostgreSQL in an Operations environment [3] Kubectl plugin
[4] Cloud Native benchmarking tool

Cloud Native PostgreSQL



kubectl-cnp

Plugin for kubectl to manage a Cloud Native PostgreSQL cluster in Kubernetes



cnp-bench

Helm charts for benchmarking both storage and the database itself for pre-production readiness testing.



Cloud Native Postgres capabilities



Deploy anywhere

Lightweight, immutable
PostgreSQL containers



Automate DBA Tasks

Failover, switchover, backup,
recovery, and rolling updates



Avoid lock-in

Operator and images are portable
to any cloud

What makes EDB unique?



EDB Technology

Unique products - EDB Postgres
Advanced & Bi-Directional
Replication (BDR)

No external dependencies -
backups and HA management all
EDB developed technologies



Largest dedicated PostgreSQL company

5,000 lifetime customers

25% of Fortune 500

300+ PostgreSQL technologists

26 PostgreSQL community
contributors and committers



Kubernetes experience

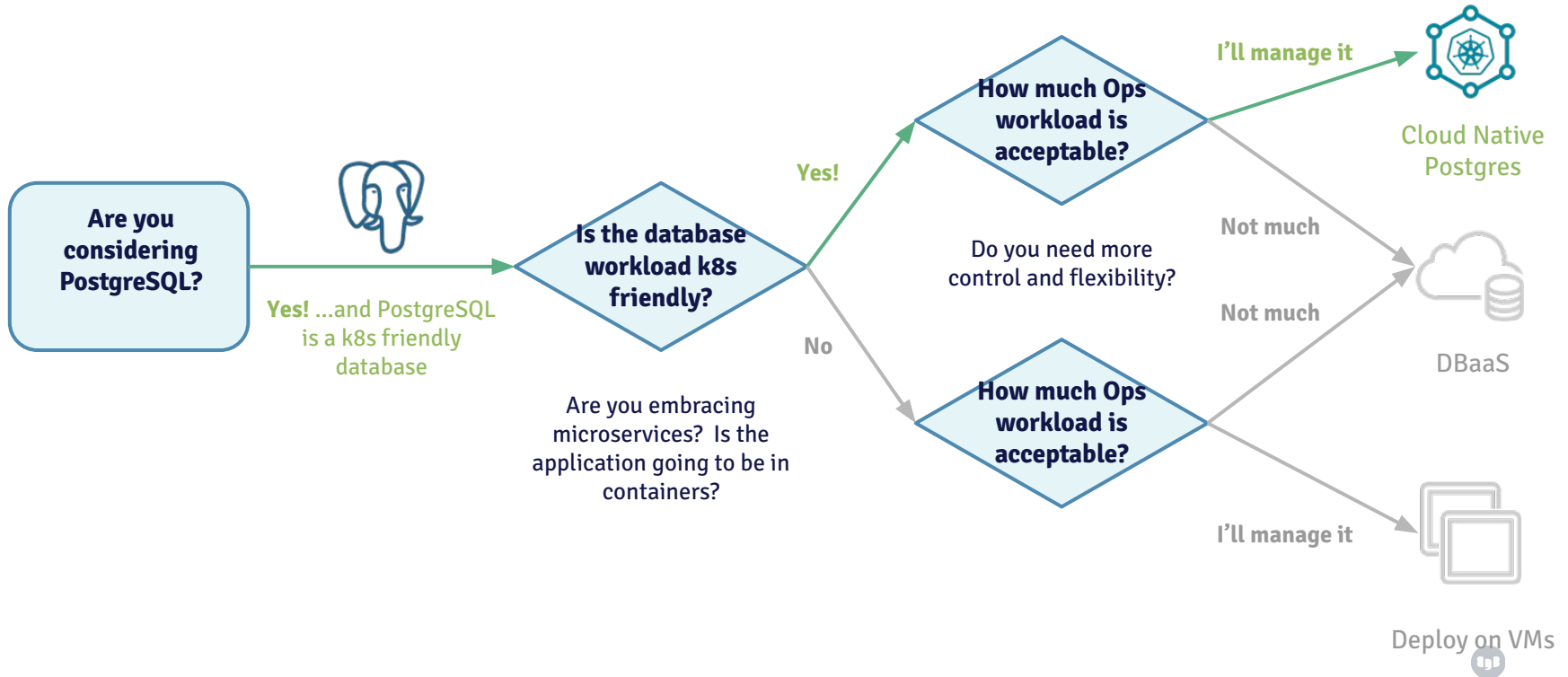
CNCF Kubernetes Certified
Services Provider (KCSP)

Red Hat Certified
Kubernetes Operators



PostgreSQL + K8s

When Cloud Native Postgres is a fit



PostgreSQL Flexibility

All Document

```
create table customers(  
  customer_document jsonb not null);
```

All Relational

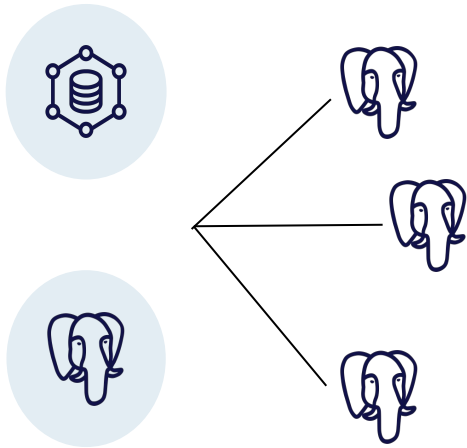
```
create table customers (  
  customer_id bigserial primary key,  
  customer_name text not null,  
  age int,  
  address text,  
  city char(50),  
  state char(2),  
  zip_code char(10),  
  CONSTRAINT fk_state  
  FOREIGN KEY(state)  
  REFERENCES state(name));
```

Mix of Document and Relational

```
create table customers(  
  customer_id bigserial primary key,  
  customer_data jsonb not null);
```



Postgres Foreign Data Wrappers (FDWs)



- Communicate with other Postgres databases
- Communicate with the data store of other microservices without pulling data into the application layer
- Query optimizations
- {join, predicate, aggregate} push-down

EDB Postgres Advanced - Database compatibility support

SQL extension support

Decode, NVL, Substr, NVL2, Date/time functions

DDL syntax support

PL/SQL support – native language

REF Cursors, Implicit and explicit cursors

Looping, variable declarations, conditional statements

Collections: Associative Arrays, Varrays, Nested tables

Pragmas

Named parameters

User Defined Exceptions

Explicit Transaction Control (within sp)

Tools

EDB*Plus – SQL*Plus look-a-like

EDB*Loader – SQL*Loader equivalent

Oracle-like Data Dictionary

ALL_, DBA_, USER_ views

Wait Events

System and session waits

Statspack-like reporting

PL/SQL supplied packages

18 DBMS

7 UTL

Data types

Blobs, Clobs, XMLTYPE, VARCHAR2, NUMBER, CHAR, Integer

Drivers

JDBC, ODBC, .NET with Oracle extensions

OCI & ProC compatible drivers

Features

Packages

Stored procedures

Functions

Triggers

Hints

Hierarchical Queries

Synonyms – Public and Private

Sequences

Rownum

Users/Roles

Dynamic SQL

Materialized Views

Partitioning

EDB Postgres Advanced Server 11:

Pragma Autonomous Transaction and DBMS_REDACT

EDB Postgres Advanced Server 12:

Interval partition; MEDIAN, LISTAGG, COMPOUND TRIGGER



Declarative Cluster Configuration

Cloud Native Postgres and BDR

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



Imperative vs Declarative

Imperative

- 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

Declarative

There's a PostgreSQL 13 cluster with 2 replicas

(At any time)



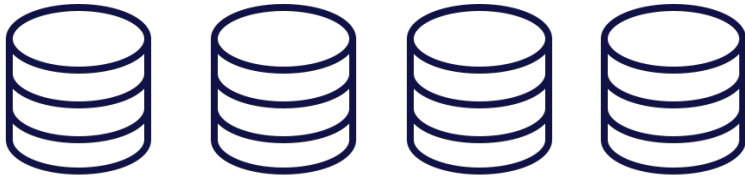
States in Kubernetes

The role of the Kubernetes controller

- Definition of the **desired state** of an object and the overall infrastructure as **configuration**
- **Reconciliation loops:**
 - Current state of the infrastructure matches the desired one
 - If not, the operator reacts to restore the desired state
- Foundation of **self-healing**



Desired state



Actual state



Convention over configuration

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: myapp-db
spec:
  instances: 3
  imageName: quay.io/enterprisedb/postgresql:13.3-4

  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
```



Deploy Anywhere

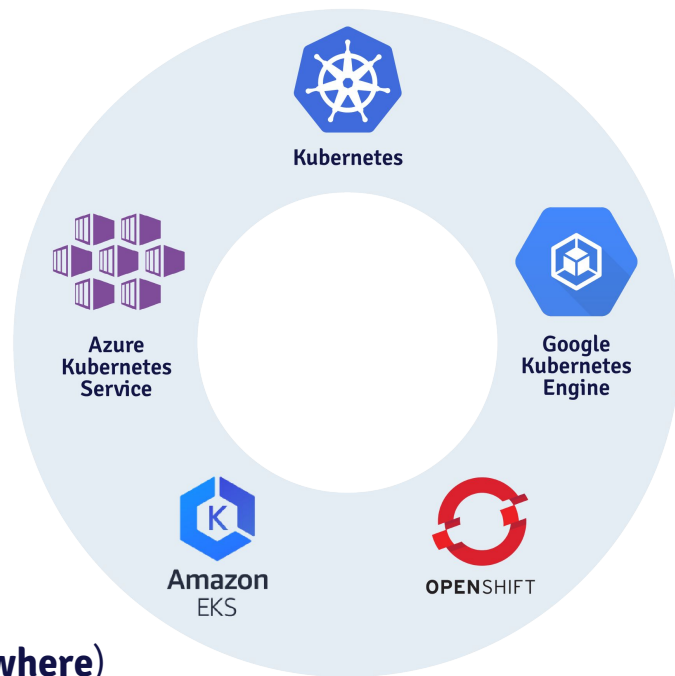


Story time

Deploy anywhere

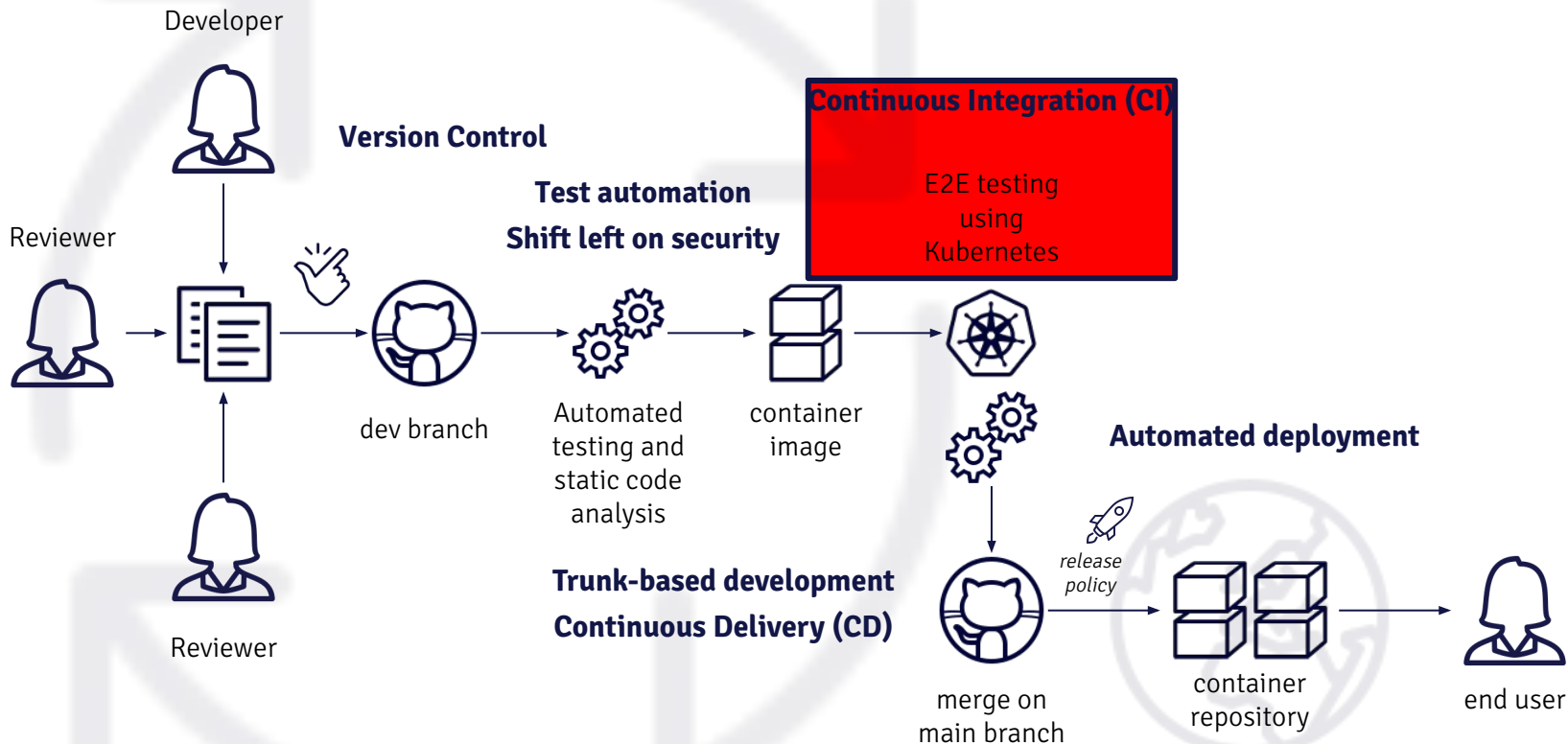
On Premises, Multi/Hybrid Cloud

- 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
- Risk mitigation for vendor lock-in of the cloud platform
 - Standard PostgreSQL backups you own (restore **anywhere**)



CNP - Deploy anywhere (including pipelines)

Real world EDB example



End-To-End tests

Part of the automated CI/CD pipelines with Github actions

- Rely on Kind (Kubernetes IN Docker) and K3d
 - Tested K8s versions: 1.16, 1.16, 1.17, 1.18, 1.20
 - Including OpenShift and, once a day, major Public Clouds
 - Tested Postgres versions: 10, 11, 12, 13
- Relevant set of tests:
 - Installation and creation of a cluster
 - Scale-up/down of a cluster
 - Failover and Switchover - including performance tests
 - Continuous backup and recovery



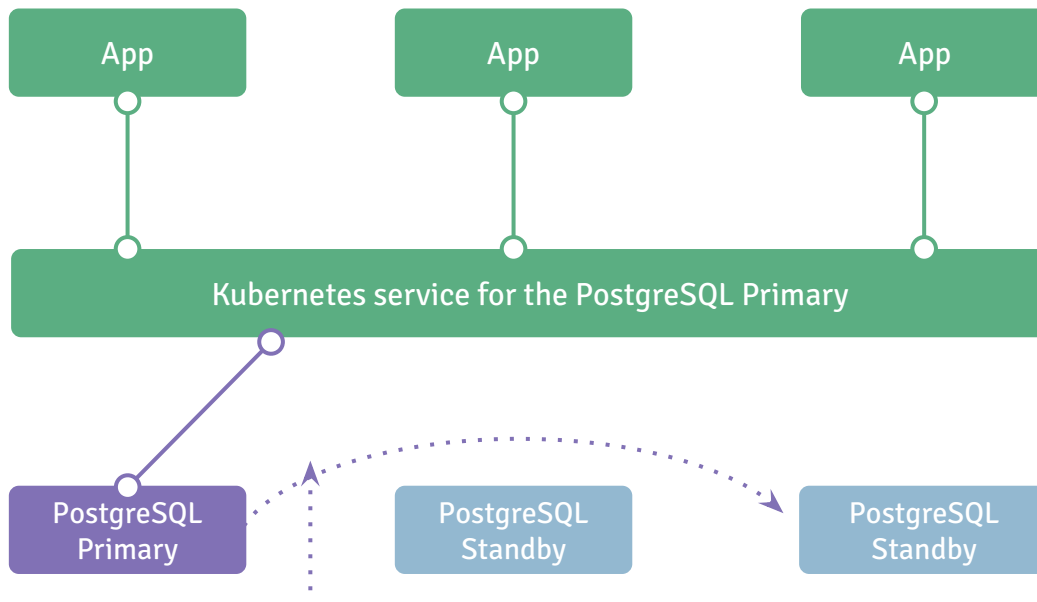
Services

PostgreSQL cluster architecture

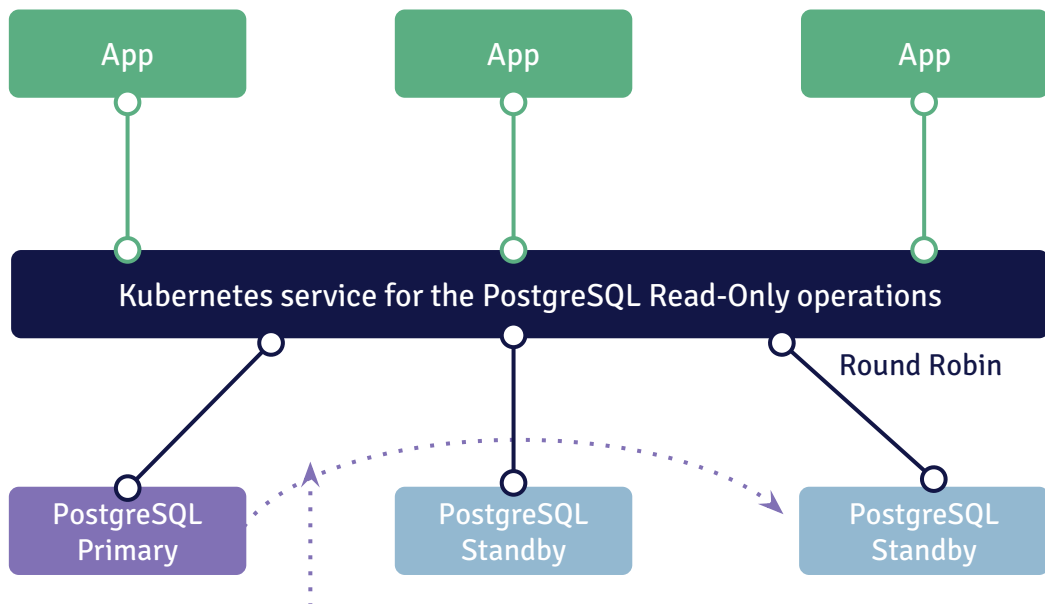
- One PostgreSQL primary
- Zero or more hot standby servers
 - PostgreSQL native streaming replication
 - Async (default) and Sync (quorum-based)
 - Required for HA
 - Transparent support for pg_rewind
- Provides apps with three K8s Services:
 - -rw suffix (read-write workloads)
 - -r suffix (read workloads)
 - -ro suffix (read-only workloads)



Read-write workloads (“-rw” service)



Read workloads (“-r” service)



Reliable DNS endpoints for your application

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

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

```
# kubectl get services -o wide
```

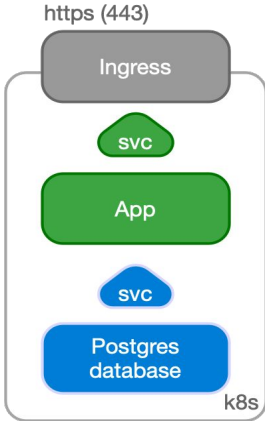
| NAME | TYPE | CLUSTER-IP | EXTERNAL-IP | PORT(S) | SELECTOR |
|-------------------|-----------|---------------|-------------|----------|--|
| pg-ha-dolores-any | ClusterIP | 10.96.184.65 | ##### | 5432/TCP | postgresql=pg-ha-dolores |
| pg-ha-dolores-r | ClusterIP | 10.96.225.137 | ##### | 5432/TCP | 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 |



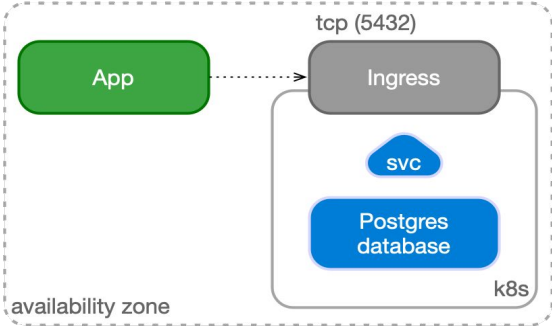
System architectures

Main classification is based on where the application reside

Use case 1: Application and Database in K8s

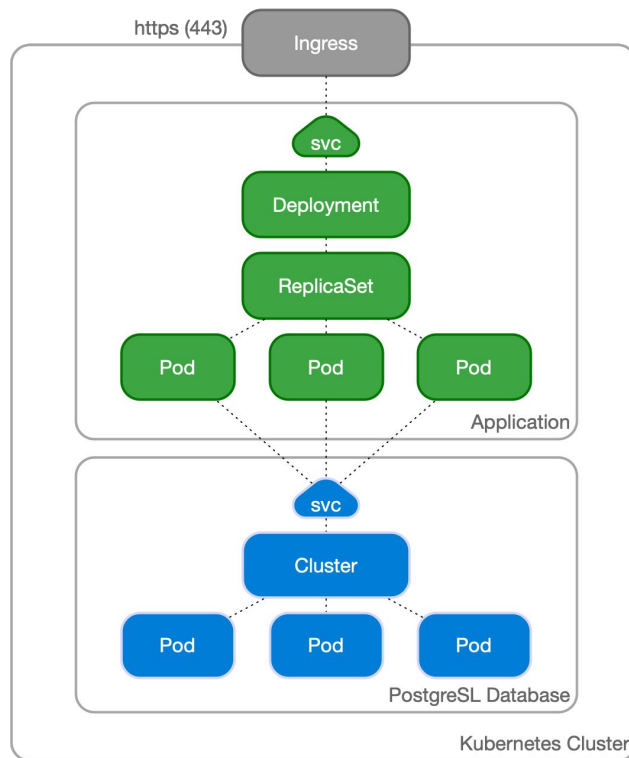


Use case 2: Application outside K8s



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



Connecting to the Service

```
1  const (  
2  host   ="pg-ha-dolores-rw"  
3  port   =5432  
4  user   ="dolores"  
5  password = "#####"  
6  )  
7  ...  
8  dbWrite := fmt.Sprintf(...)
```

```
1  const (  
2  host   ="pg-ha-dolores-ro"  
3  port   =5432  
4  user   ="dolores-ro"  
5  password = "#####"  
6  )  
7  ...  
8  dbRead  := fmt.Sprintf(...)
```



**Advanced database features, no
DBA required**

Default pg_hba.conf settings

```
# Grant local access
local all all peer
# TLS for replication
hostssl postgres streaming_replica all cert clientcert=1
hostssl replication streaming_replica all cert clientcert=1

# CUSTOM SETTINGS GO HERE

# Require md5 authentication elsewhere
host all all all md5
```

Refer to [“How to Issue a TLS Certificate for Cloud Native PostgreSQL”](#) for details



Synchronous replication

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: cluster-example
spec:
  instances: 3
  minSyncReplicas: 1
  maxSyncReplicas: 2
  storage:
    size: 1Gi
```

Refer to [“Streaming replication overview”](#) for details



PostgreSQL configuration

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: cluster-example
spec:
  instances: 3
  postgresql:
    parameters:
      work_mem: "8MB"
      # ...
  # ...
```

Refer to [“PostgreSQL configuration”](#) for details



Enabling EDB Postgres Server

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: cluster-example
spec:
  instances: 3
  imageName: quay.io/enterprisedb/edb-postgres-advanced:13.3.6-2
  licenseKey: <PASTE_YOUR_LICENSE_KEY_HERE>

# ...
```

Refer to [“Evaluation of EDB Postgres Advanced”](#) for details



Cloud Native BDR (Q4 2021)

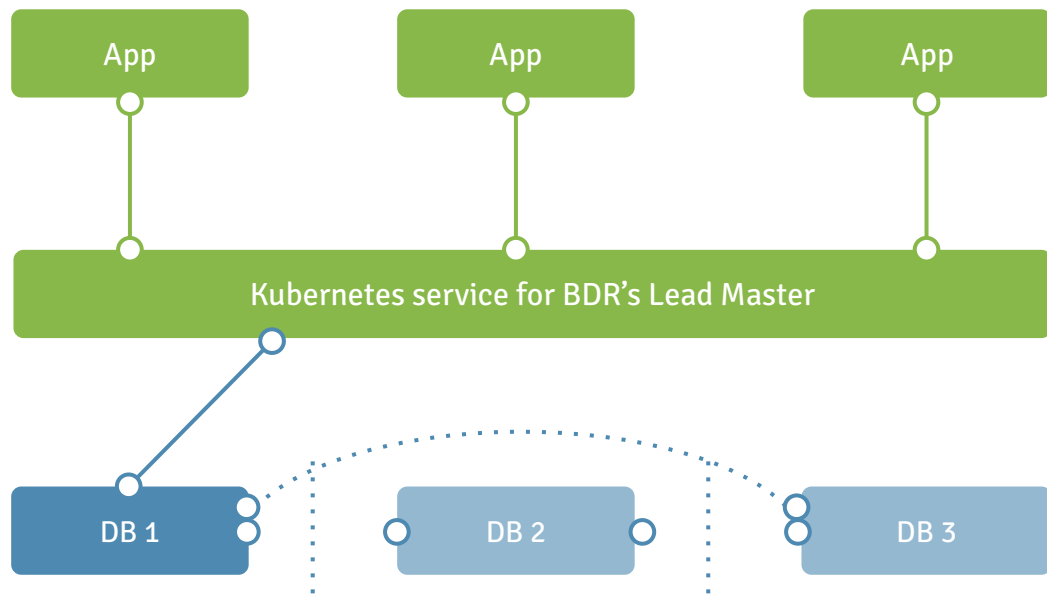
Cloud Native BDR

Kubernetes operator that manages multi-master architectures with BDR

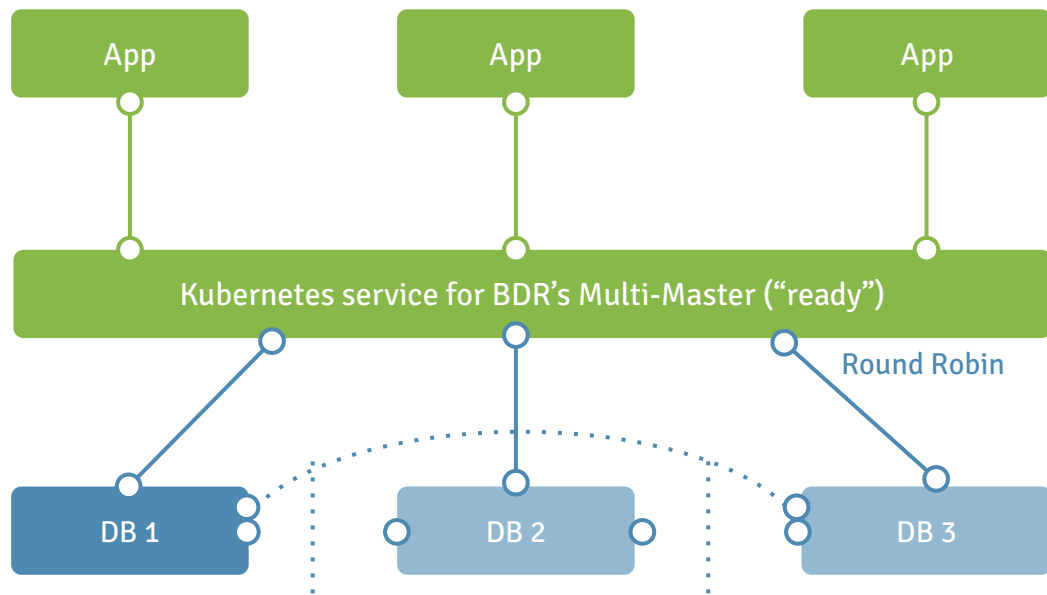
- First EDB rebranded version expected in Q4/2021
- Multi-Master technology
 - Full mesh replication solution
 - Including DDL and DML statements
 - BDR is an extension for PostgreSQL
- Based on Cloud Native PostgreSQL
 - All concepts explained earlier apply here as well
 - “BdrGroup” CRD implementing the **WriteAnywhere** architecture



Lead-master workloads (HA)



Multi-master workloads (“ready” service)



Recommended Next Steps

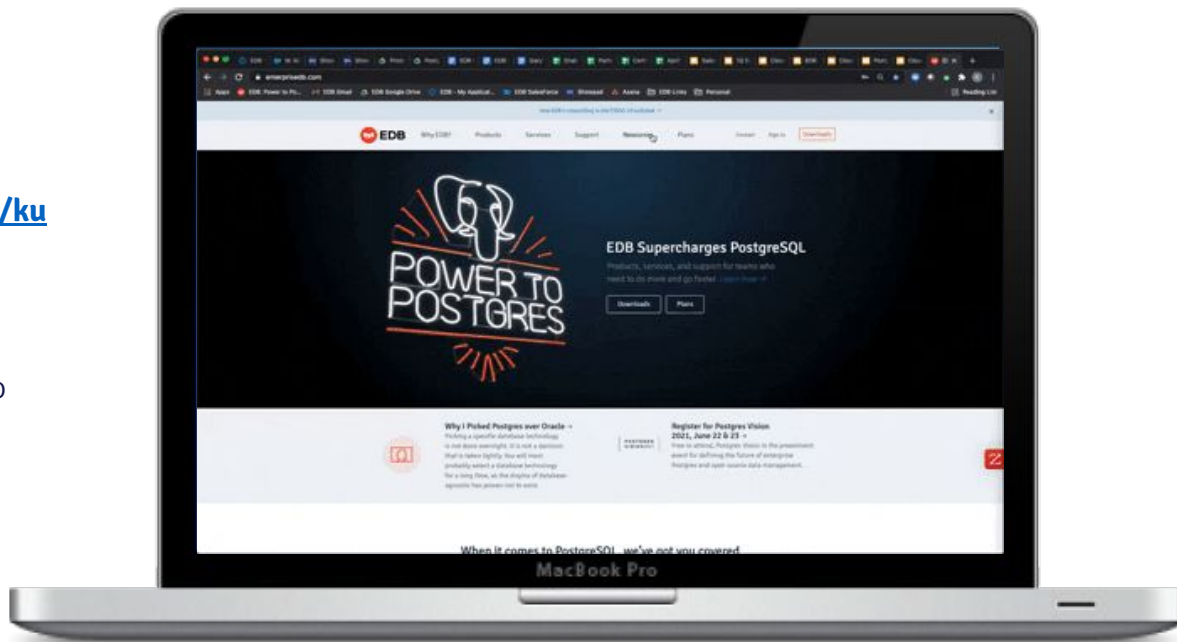
EDB docs

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!)



Evaluation/Trial

Operator with PostgreSQL

- No trial license key required
- No reconciliation attempt after 30d
 - From the creation of each Postgres cluster

Operator with EDB Postgres Advanced

- A trial license key is required
- Expiry date = 60d from license request
- No reconciliation attempt after expiry date



Recommended Next Steps

Proof of Concept

- ✓ Plan to deploy and test on YOUR K8s cluster
- ✓ How can EDB help ensure success?

