



Fuel the DevOps Movement

Innovate Faster with EDB Postgres
for Kubernetes

14 July 2022



EDBTM

OUR STORY

Ridha Sulaiman

Country Sales Lead, Indonesia

EDB

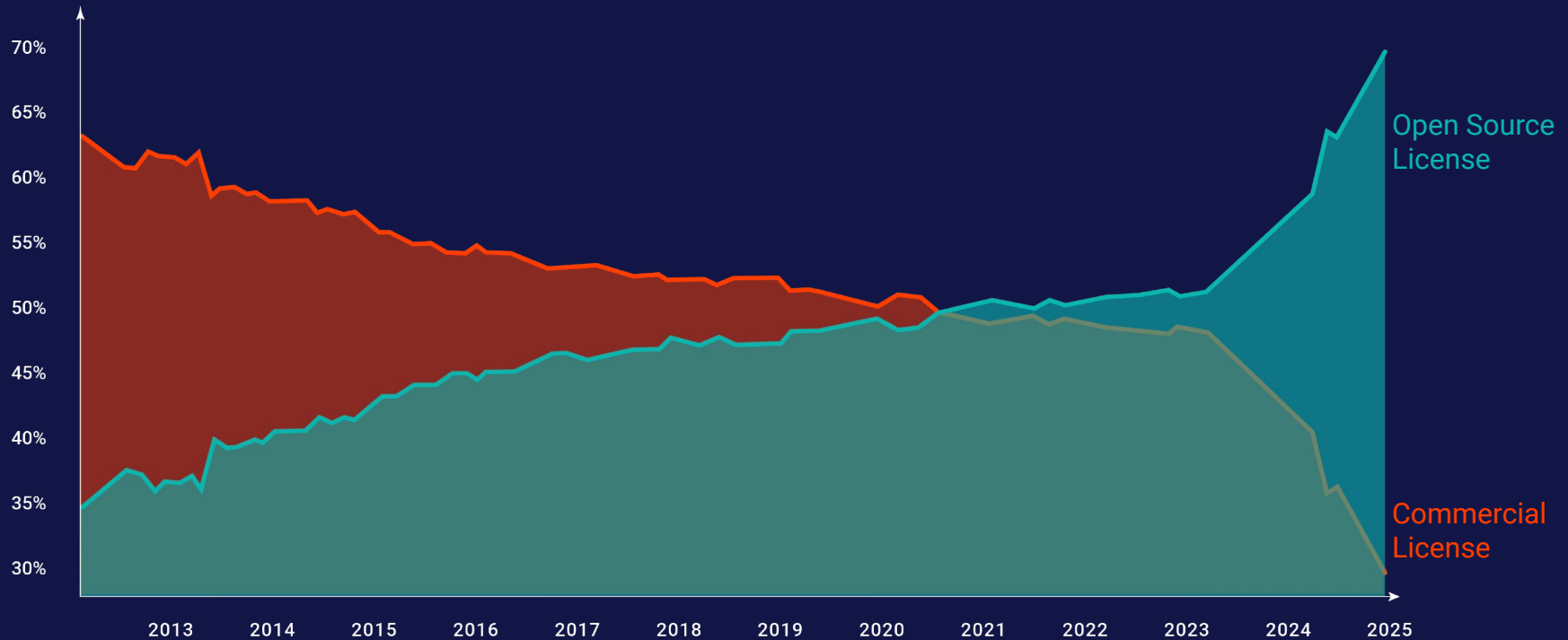
A NEW WORLD HAS EMERGED

Companies must shift to **next-gen** operating models to compete and **stay relevant**

- Data is the currency
- Data ownership is an economic advantage
- Databases are a strategic asset

A TIPPING POINT

Open source database licenses now outpace legacy



source: db-engines, IDC

POSTGRES IS THE DRIVER OF THIS TIPPING POINT

The Standard

Postgres is the
database of choice

- #1 Fastest growing DBMS
- #1 Largest developer community
- #1 Most loved database by developers



EDB™

Fuel the DevOps
Movement and Innovate
Faster with EDB Postgres
for Kubernetes

Habiburrokhman A. Sjarbini - Abip
July 2022

About Me



IT Consultant since 2007

- Oracle DBA Consultant for various SI companies, 2007-2010
- Oracle Indonesia ACS 2010-2013
- Oracle Singapore ACS Engineered Systems 2014-2017
- Self-Employed Oracle Engineered Systems Specialist 2017-2019
- Red Hat Indonesia, Global Professional Services - 2019-2022. Platform Consultant for RH OCP 3 and 4, RH OCS/ODF, RH Ansible Automation Platform, RH Gluster Storage, RHEL High Availability
- EDB, Sales Engineer

<https://www.linkedin.com/in/abip/>

Running Postgres in Kubernetes with EDB Postgres for Kubernetes

- Why Postgres
- Why EDB
- Evolution of database deployment
- EDB Postgres for Kubernetes
- Success Stories
- EDB Postgres for Kubernetes in Action

Why Postgres

“

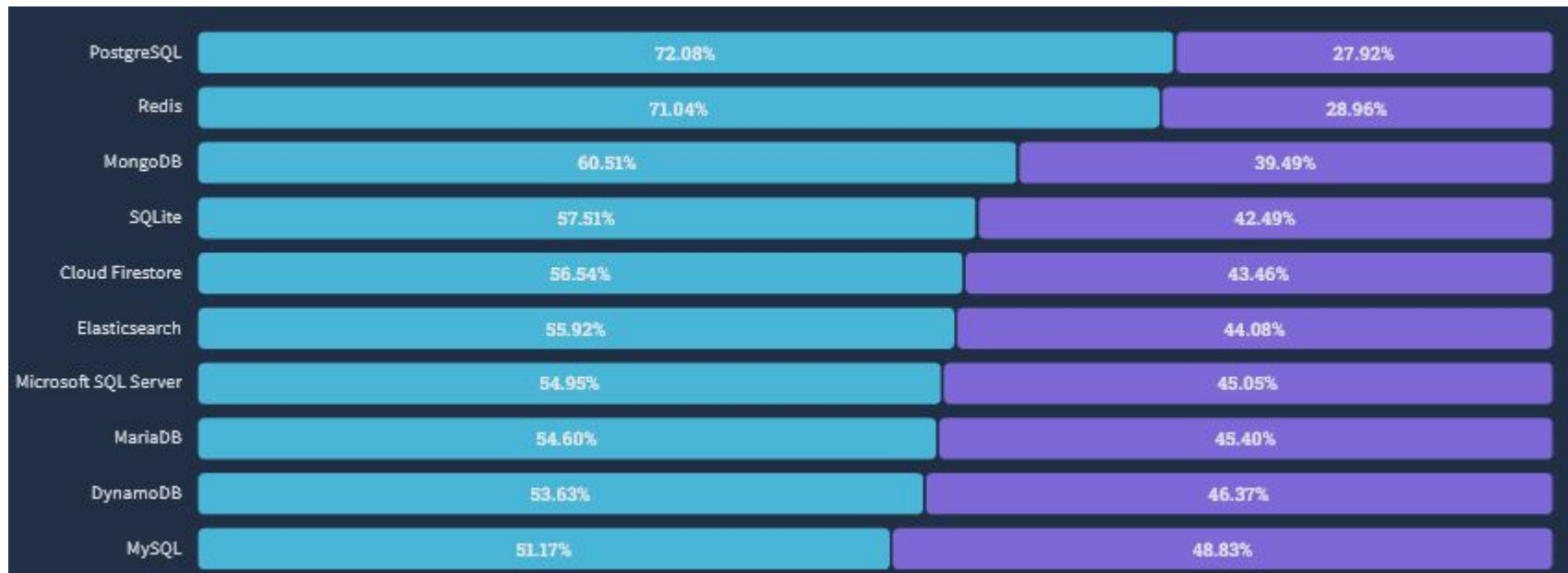
**“Postgres is
the most transformative tech
since Linux”**

Marc Linster - EDB CTO



Postgres won

If you bet... you bet on Postgres



https://survey.stackoverflow.co/2022/?utm_source=results#most-loved-dreaded-and-wanted-database-love-dread





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



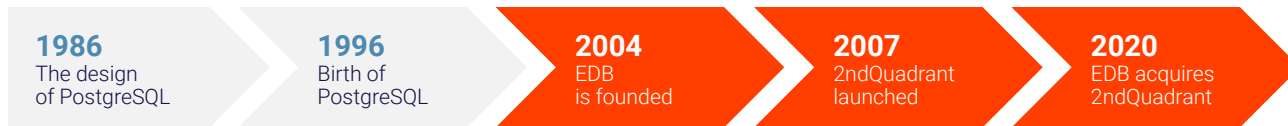
Containers

and doesn't
lock you in

Why EDB



We're the PostgreSQL experts



Key PostgreSQL Contributions

EDB

- Heap Only Tuples (HOT)
- Materialized Views
- Parallel Query
- JIT Compilation
- Serializable Parallel Query

2ndQuadrant

- Hot Standby
- Logical Replication
- Transaction Control in Procedures
- Generated Columns

No company
has contributed
more to
PostgreSQL



We have 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

Evolution of Database Deployment

Monolithic to Agile

Cloud changes much more than just your deployment method



Enter Containers & Kubernetes

Cloud Native technologies that enable microservices architectures and continuous delivery

Containers

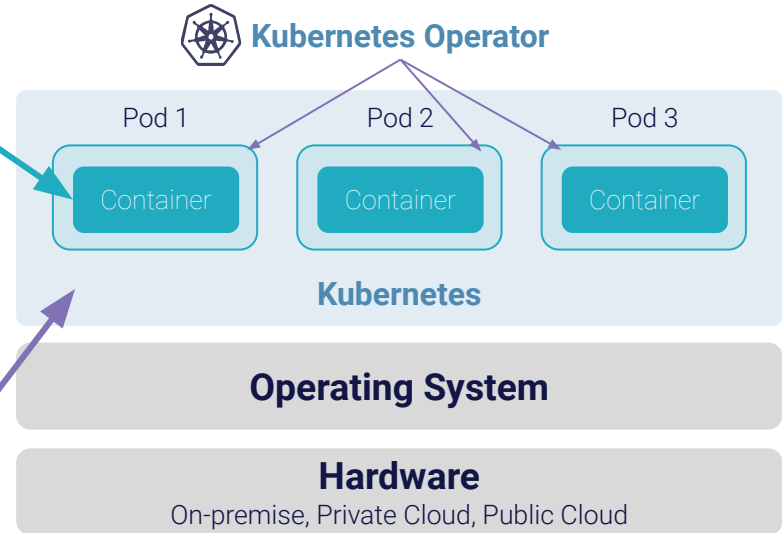
Packages the code for a software product along with the dependencies it needs at run time (i.e. when it is executed.).

Containers are managed in Pods.

Kubernetes

A container management system used for orchestrating software management and utilizing infrastructure more efficiently.

It makes deploying and managing containers at scale a reality.



Why Kubernetes?

Kubernetes includes many system services needed for managing software

Services, Load Balancing,
and Networking



Health
checking



Storage
management



Automated
Scheduling



Scalability:
scale-up/down



Rolling
Deployments



Built on Cloud Native Principles



Future proof
with Kubernetes



No additional tools required



Fully declarative deployment



Introducing EDB Postgres for Kubernetes



Kubernetes Certified Service Provider



Platinum Sponsor



“

“EDB Postgres for Kubernetes is an operator designed by EnterpriseDB to manage PostgreSQL workloads on any supported Kubernetes cluster running in private, public, hybrid, or multi-cloud environments.

EDB Postgres for Kubernetes adheres to DevOps principles and concepts such as **declarative configuration** and **immutable infrastructure**.”

https://www.enterprisedb.com/docs/postgres_for_kubernetes/latest/



EDB Postgres for Kubernetes



EDB Technology

Stack of enterprise grade products designed, developed, and supported by EDB

Exploiting Postgres workloads in Kubernetes



Kubernetes Experience

EDB Postgres for Kubernetes operator - based on the open source CloudNativePG operator

Adds EDB Postgres Advanced Server workloads with a primary/standby architecture



Red Hat Certified

Red Hat Certified Kubernetes Operator

Available on Red Hat OpenShift



Key capabilities

Relying exclusively on the Kubernetes API



Deploy anywhere

Lightweight, immutable
Postgres containers



Automate DBA Tasks

Failover, switchover, backup,
recovery, and rolling updates



Avoid lock-in

Operator and images are
portable to any cloud



Core benefits



Security

Secure data in-flight, automate patch management, implement least privilege model



Flexibility

Choose Postgres or use EDB Postgres Advanced Server for additional enterprise-class features



Support

24/7 global support to guide you on best practices and resolve issues



Design

Accelerate time to market with a solution built for microservice architectures



Business benefits

Business benefits of using EDB Postgres for Kubernetes



Go to market sooner

Enable a microservices approach at the data tier on any platform



Keep staff focused on business objectives

Minimize time spent on routine database management tasks

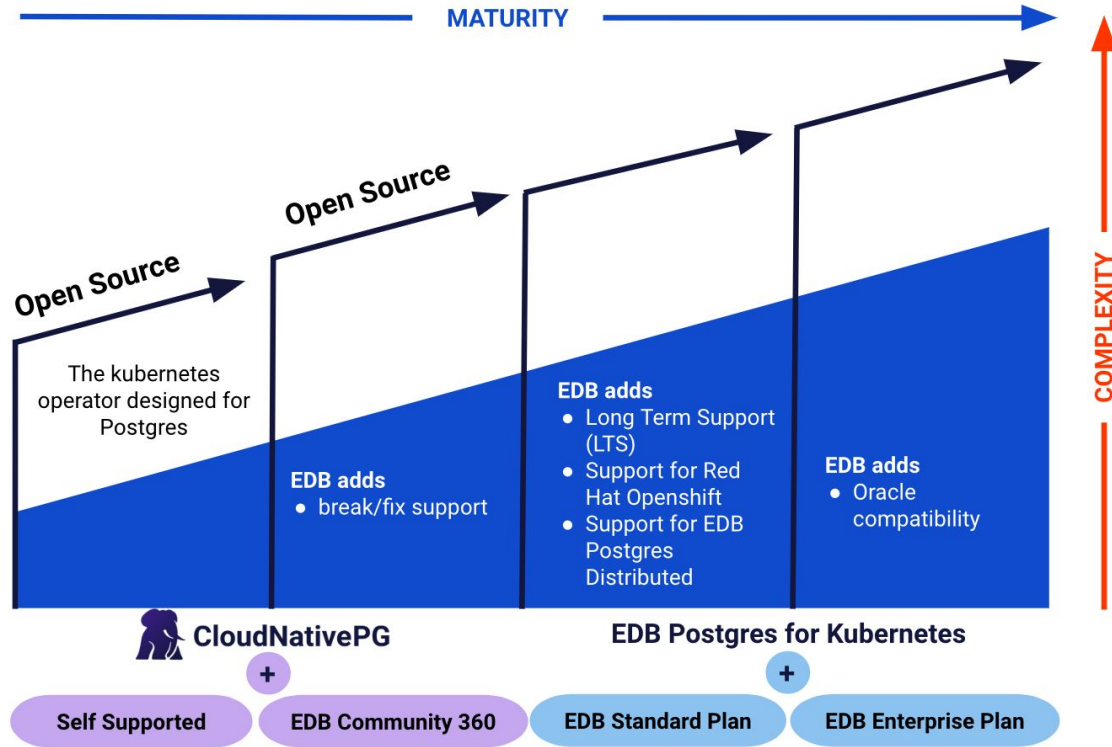


Keep the business running

Add resiliency for uptime: self-healing, less error-prone, auto-scalable

CloudNativePG and EDB Postgres for Kubernetes

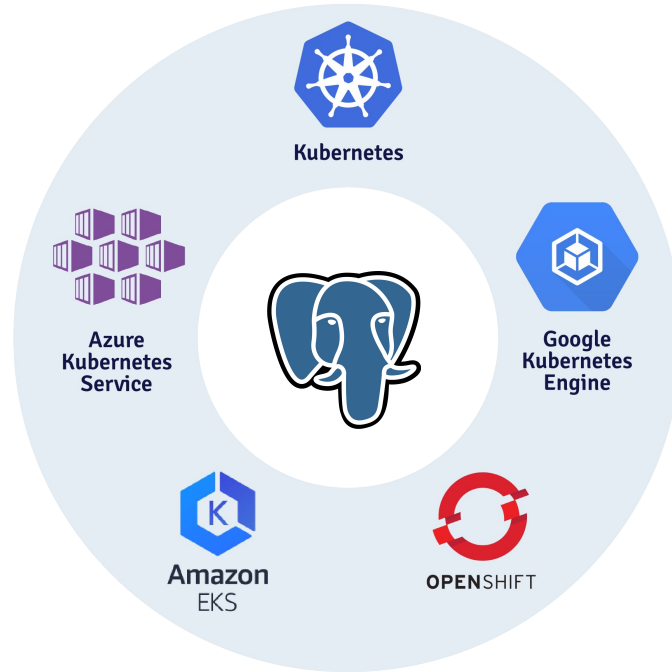
Modernize your infrastructure and power your performance with Cloud Native and EDB



LANDSCAPE OF POSTGRES FOR KUBERNETES JOURNEY



Deploy anywhere



EDB Postgres for Kubernetes Use Cases

Modernize your infrastructure and power your performance with Cloud Native and EDB



New application
development



Migrate applications
to the cloud



Embrace multi- and
hybrid-cloud strategies



Application development

Benefits of using EDB Postgres for Kubernetes



Microservices architecture

Make Postgres an integral part of your microservices architecture.

There is no longer a need to rely on a database system somewhere in your organization.



Increase velocity and customer experience

Allows you to integrate data management in an agile strategy that help you deliver features, functionality, and fixes at the touch of a button.



Infrastructure abstraction

Adding Postgres to your Cloud Native approach, meant a lot of tweaking and tuning both Postgres and Kubernetes.

This is fixed out of the box for you with the most advanced Postgres operator available today.



Kubernetes administration

Benefits of using EDB Postgres for Kubernetes



Declarative configuration

The complexity of replica databases, automated failover, and self-healing, unique to stateful database workloads on Kubernetes, is reduced to a set of parameters that let you control your environment.



Immutable application containers

No worrying about upgrades and patches; containers are not updated, they are simply replaced.

Fully integrate Postgres deployments as you would deploy any other application in Kubernetes.



Security concepts

Critical applications require guarantees around security (shift-left on security). Running Postgres on Kubernetes is no exception. You get a full implementation of Kubernetes specific security policies combined with industry leading support.



Project direction

Benefits of using EDB Postgres for Kubernetes



Standardized way of running Postgres

Simplify using Postgres in your architecture to such an extent that the integration of this database management system no longer requires highly specialized staff.



Deliver applications at high velocity

Enable Postgres to be part of the DevOps and Agile technological foundations and add its unparalleled technological versatility to any of your projects.



Increase speed to benefit and pace of innovation

In bringing the two very powerful communities of Postgres and Kubernetes together, you gain an unmatched advantage for your project. The level of innovation in both technologies drive exceptional speed to your project's development.

Success Stories

EDB Postgres for Kubernetes success story

A Government Institution



New single sign-on application (KeyCloak) for citizens to access government services developed and deployed with Kubernetes.

Additionally, create a new self-service environment to build new applications.



Initiatives

- Develop a new SSO application that is scalable, secure, and performant
- Create self-service database environment for developers



EDB's value

- EDB Postgres for Kubernetes empowers developers with development speed and flexibility
- RDBA for staff augmentation and expertise



Business benefits

- Improve user experience for the citizens accessing government websites and services
- Deliver the apps in a cost-effective manner



Solution details

- Application running in production on EDB Postgres for Kubernetes deployed on RedHat OpenShift
- Applications include queue management systems and a BPM engine for workflows like customer feedback, COVID travel plan callback, and more



EDB Postgres for Kubernetes in Action

Day 0 Operations

Objective - Plan Kubernetes Infrastructure

- First impressions last
 - K8s infrastructure often planned for stateless-only workloads
 - Common choice: database outside Kubernetes - DBaaS
- You can run databases inside Kubernetes
 - Fully leverage devops
 - Shared/Shared nothing architectures
 - Storage sector in K8s is growing fast
- Choose your storage wisely
 - Like you are used to in VMs and bare metal

Shared nothing architecture

Node

Availability zone 1

Node

Availability zone 2

Node

Availability zone 3



Kubernetes cluster



Disclaimer: this is a simplified view

Installing EDB Postgres for Kubernetes Operator

```
kubectl apply -f \  
https://get.enterprisedb.io/cnp/postgresql-operator-1.15.1.yaml
```

Declarative configuration via YAML manifest

Lab Env Info

Cluster Name & Namespace Name - Primary Cluster Cluster Name & Namespace Name - Replica Cluster	epkclu1 on epkclu1 epkclu2 on epkclu2
OS	Ubuntu 18.04.5 LTS
IP Address	192.168.24.40
Host Name	k8s
# of Node	1 - Untainted Control Plane
k8s Version	1.20.0
helm Version	3.7.1

```
root@k8s:~# kubectl get nodes -owide
NAME      STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
k8s      Ready    control-plane,master   72d   v1.20.0   192.168.24.40   <none>        Ubuntu 18.04.5 LTS   4.15.0-188-generic   docker://20.10.7
```



StorageClasses - NFS Helper - For Demo Env Only!

```
root@k8s:~$ kubectl get sc
NAME                                PROVISIONER                RECLAIMPOLICY   VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
helper-nfs-k8s-es                  helper.ocpdev/nfs         Delete          Immediate            false                  20h
helper-nfs-storage (default)       helper.ocpdev/nfs         Delete          Immediate            false                  72d
local-storage                      kubernetes.io/no-provisioner Delete          WaitForFirstConsumer false                  9d
localblock-sc                      kubernetes.io/no-provisioner Delete          WaitForFirstConsumer false                  2d8h
```

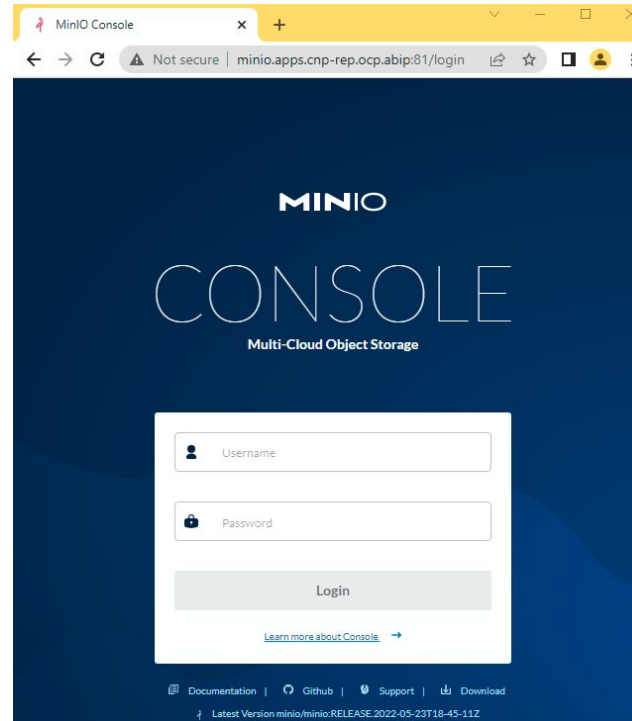


Object Storage - MinIO

S3 Compatible Object Storage - Expose MinIO



```
root@k8s:~$ kubectl get ingress/minio-ingress -n velero -ojson | jq .spec
{
  "ingressClassName": "nginx",
  "rules": [
    {
      "host": "minio.apps.cnp-rep.ocp.abip",
      "http": {
        "paths": [
          {
            "backend": {
              "service": {
                "name": "minio",
                "port": {
                  "number": 9001
                }
              }
            },
            "path": "/",
            "pathType": "Prefix"
          }
        ]
      }
    }
  ]
}
```



Day 1 Operations

Objective - 2-Node Postgres cluster

- Install the 14.1 version of PostgreSQL
- Create a new PostgreSQL 14 Cluster
- One primary and one standby server
- mTLS authentication with replicas
- 1 Gi of storage
- A way to access the primary via network
- A user for the application
- A database for the application

EDB Postgres for Kubernetes

Create Simple Cluster - Apply Cluster Resource Definition

```
root@k8s:cluster-ocpsno$ cat epkclul.yaml
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: epkclul
  namespace: epkclul
spec:
  backup:
    barmanObjectStore:
      destinationPath: 's3://epkclul/'
      endpointURL: 'http://minio.velero.svc:9000'
      s3Credentials:
        accessKeyId:
          key: MINIO_ACCESS_KEY
          name: minio-creds
        inheritFromIAMRole: false
        secretAccessKey:
          key: MINIO_SECRET_KEY
          name: minio-creds
  imageName: 'quay.io/enterprisedb/postgresql:14.1'
  monitoring:
    disableDefaultQueries: false
    enablePodMonitor: true
  bootstrap:
    initdb:
      database: app
      encoding: UTF8
      localeCTYPE: C
      localeCollate: C
      owner: app
  storage:
    resizeInUseVolumes: true
    size: 1Gi
    storageClass: helper-nfs-storage
  instances: 2
```

```
$ kubectl apply -f epkclul.yaml
cluster.postgresql.k8s.enterprisedb.io/epkclul created
```



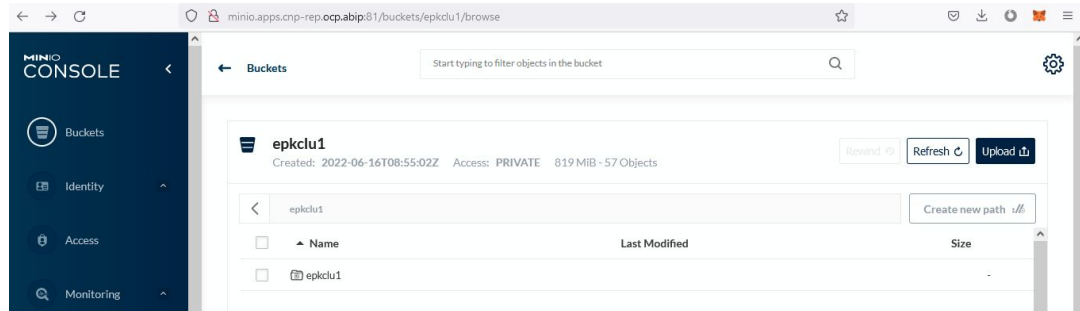
EDB Postgres for Kubernetes

Create Simple Cluster - Review Cluster Resource

```
root@k8s:cluster-ocpsno$ kubectl get pods,pvc,svc
NAME          READY   STATUS    RESTARTS   AGE
pod/epkclu1-1 1/1     Running   0           11m
pod/epkclu1-2 1/1     Running   0           14m

NAME                                STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
persistentvolumeclaim/epkclu1-1     Bound    pvc-ae792105-199c-46af-9dec-4af44d210875 1Gi        RWO             helper-nfs-storage 9d
persistentvolumeclaim/epkclu1-2     Bound    pvc-5f8c18d2-9e6c-4616-a552-ba0cd0fad2cb 1Gi        RWO             helper-nfs-storage 9d

NAME                                TYPE          CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/epkclu1-any                  ClusterIP     10.103.7.84     <none>        5432/TCP         9d
service/epkclu1-r                    ClusterIP     10.96.119.80   <none>        5432/TCP         9d
service/epkclu1-ro                   ClusterIP     10.108.130.68  <none>        5432/TCP         9d
service/epkclu1-ro-nodeport          NodePort     10.107.85.158  <none>        5432:32227/TCP  33h
service/epkclu1-rw                   ClusterIP     10.107.240.65  <none>        5432/TCP         9d
service/epkclu1-rw-nodeport          NodePort     10.110.146.212 <none>        5432:30264/TCP  33h
```



EDB Postgres for Kubernetes

Check Cluster Status using CNP Plugin

```
root@k8s:cluster-ocpsno$ kubectl cnp status epkclul
Cluster Summary
Name: epkclul
Namespace: epkclul
System ID: 7110015503906877463
PostgreSQL Image: quay.io/enterprisedb/postgresql:14.1
Primary instance: epkclul-1
Status: Cluster in healthy state
Instances: 2
Ready instances: 2
Current Write LSN: 0/30002228 (Timeline: 5 - WAL File: 000000050000000000000030)

Certificates Status
Certificate Name      Expiration Date      Days Left Until Expiration
-----
epkclul-ca           2022-09-15 01:26:22 +0000 UTC  80.53
epkclul-replication 2022-09-15 01:26:22 +0000 UTC  80.53
epkclul-server       2022-09-15 01:26:22 +0000 UTC  80.53

Continuous Backup status
First Point of Recoverability: 2022-06-17T01:36:43Z
Working WAL archiving: OK
WALs waiting to be archived: 0
Last Archived WAL: 000000040000000000000030.partial @ 2022-06-26T12:39:39.305565Z
Last Failed WAL: -

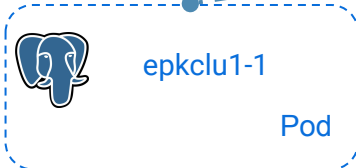
Streaming Replication status
Name      Sent LSN      Write LSN      Flush LSN      Replay LSN      Write Lag      Flush Lag      Replay Lag      State      Sync State      Sync Priority
-----
epkclul-2 0/30002228    0/30002228    0/30002228    0/30002228    00:00:00      00:00:00      00:00:00      streaming  async           0

Instances status
Name      Database Size  Current LSN      Replication role  Status  QoS      Manager Version
-----
epkclul-1 187 MB         0/30002228      Primary           OK      BestEffort 1.15.1
epkclul-2 187 MB         0/30002228      Standby (async)  OK      BestEffort 1.15.1
```





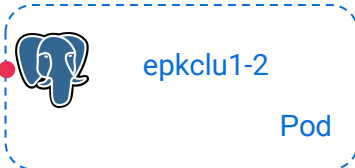
(m)TLS



Node

Availability zone 1

mTLS



Node

Availability zone 2

Node

Availability zone 3



Kubernetes cluster



There's more

- A service to access read-only replicas (epkclu1-ro)
- A service to access any instance for reads (epkclu1-r)
- Many other Kubernetes objects are created:
 - Secrets
 - ConfigMaps
 - Roles
 - RoleBindings
 - ServiceAccounts
 - ...
- Convention over configuration

PostgreSQL Configuration

- Most GUCs are configurable
 - `.postgresql.parameters` section
 - Some cannot be changed (e.g. `log_destination`)
 - Some have defaults
- Host-Based Authentication can be configured
 - `.postgresql.pg_hba` section
 - By default:
 - Requires TLS authentication for streaming replicas
 - Fallback sets sha-256/md5 authentication
- EDB Postgres for Kubernetes supports changes of configuration
 - Reload
 - Rolling updates if restart is required
 - Update of standby sensitive parameters

Day 2 Operations

EDB Postgres for Kubernetes In Action

- Switchover and Failover
- Horizontal Scaling
- Vertical Scaling

Desired state



Actual state



EDB Postgres for Kubernetes

Perform Switchover to Promote Standby Instance

```
$ kubectl cnp promote epkclul epkclul-2
```

Node epkclul-2 in cluster epkclul will be promoted

```
root@k8s:cluster-ocpsno$ kubectl get pods -L role
NAME      READY   STATUS    RESTARTS   AGE   ROLE
epkclul-1 1/1     Running   1          20m   replica
epkclul-2 1/1     Running   0          5m21s primary
```

```
root@k8s:cluster-ocpsno$ kubectl cnp status epkclul
Cluster Summary
Name:          epkclul
Namespace:    epkclul
System ID:    7110015503906877463
PostgreSQL Image: quay.io/enterprisedb/postgresql:14.1
Primary instance: epkclul-2
Status:       Cluster in healthy state
Instances:    2
Ready instances: 2
Current Write LSN: 0/32001D30 (Timeline: 6 - WAL File: 0000000600000000000000032)

Certificates Status
Certificate Name      Expiration Date          Days Left Until Expiration
-----
epkclul-ca           2022-09-15 01:26:22 +0000 UTC  80.53
epkclul-replication 2022-09-15 01:26:22 +0000 UTC  80.53
epkclul-server       2022-09-15 01:26:22 +0000 UTC  80.53

Continuous Backup status
First Point of Recoverability: 2022-06-17T01:36:43Z
Working WAL archiving:        OK
WALs waiting to be archived: 0
Last Archived WAL:           0000000500000000000000032.partial @ 2022-06-26T12:45:07.934303Z
Last Failed WAL:             -

Streaming Replication status
Name      Sent LSN      Write LSN      Flush LSN      Replay LSN      Write Lag      Flush Lag      Replay Lag      State      Sync State      Sync Priority
-----
epkclul-1 0/32001D30    0/32001D30    0/32001D30    0/32001D30    00:00:00      00:00:00      00:00:00      streaming async          0

Instances status
Name      Database Size  Current LSN      Replication role  Status  QoS      Manager Version
-----
epkclul-1 187 MB         0/32001D30      Standby (async)   OK      BestEffort 1.15.1
epkclul-2 187 MB         0/32001D30      Primary           OK      BestEffort 1.15.1
```



EDB Postgres for Kubernetes

Simulate a Failure - Delete Primary Pods

```
$ kubectl delete pods epkclul-2  
pod "epkclul-2" deleted
```

```
root@k8s:cluster-ocpsno$ kubectl cnv status epkclul  
Cluster Summary  
Name: epkclul  
Namespace: epkclul  
System ID: 7110015503906877463  
PostgreSQL Image: quay.io/enterprisedb/postgresql:14.1  
Primary instance: epkclul-1  
Status: Cluster in healthy state  
Instances: 2  
Ready instances: 2  
Current Write LSN: 0/33001050 (Timeline: 7 - WAL File: 0000000700000000000000033)  
  
Certificates Status  
Certificate Name Expiration Date Days Left Until Expiration  
-----  
epkclul-replication 2022-09-15 01:26:22 +0000 UTC 80.53  
epkclul-server 2022-09-15 01:26:22 +0000 UTC 80.53  
epkclul-ca 2022-09-15 01:26:22 +0000 UTC 80.53  
  
Continuous Backup status  
First Point of Recoverability: 2022-06-17T01:36:43Z  
Working WAL archiving: OK  
WALs waiting to be archived: 0  
Last Archived WAL: 0000000600000000000000033.partial @ 2022-06-26T12:47:38.710414Z  
Last Failed WAL: -  
  
Streaming Replication status  
Name Sent LSN Write LSN Flush LSN Replay LSN Write Lag Flush Lag Replay Lag State Sync State Sync Priority  
-----  
epkclul-2 0/33001050 0/33001050 0/33001050 0/33001050 00:00:00 00:00:00 00:00:00 streaming async 0  
  
Instances status  
Name Database Size Current LSN Replication role Status QoS Manager Version  
-----  
epkclul-1 187 MB 0/33001050 Primary OK BestEffort 1.15.1  
epkclul-2 187 MB 0/33001050 Standby (async) OK BestEffort 1.15.1
```



EDB Postgres for Kubernetes

Horizontal Scaling - Scale-Out the Cluster - from 2 to 3 Instances

```
$ kubectl scale cluster epkclul --replicas=3  
cluster.postgresql.k8s.enterprisedb.io/epkclul scaled
```

```
root@k8s:cluster-ocpsno$ kubectl cnp status epkclul  
Cluster Summary  
Name: epkclul  
Namespace: epkclul  
System ID: 7110015503906877463  
PostgreSQL Image: quay.io/enterprisedb/postgresql:14.1  
Primary instance: epkclul-1  
Status: Cluster in healthy state  
Instances: 3  
Ready instances: 3  
Current Write LSN: 0/49000060 (Timeline: 7 - WAL File: 0000000700000000000000049)  
  
Certificates Status  
-----  
Certificate Name      Expiration Date      Days Left Until Expiration  
-----  
epkclul-ca            2022-09-15 01:26:22 +0000 UTC  80.52  
epkclul-replication  2022-09-15 01:26:22 +0000 UTC  80.52  
epkclul-server        2022-09-15 01:26:22 +0000 UTC  80.52  
  
Continuous Backup status  
First Point of Recoverability: 2022-06-17T01:36:43Z  
Working WAL archiving: OK  
WALs waiting to be archived: 0  
Last Archived WAL: 0000000700000000000000048.00000028.backup @ 2022-06-26T13:02:12.800909Z  
Last Failed WAL: -  
  
Streaming Replication status  
-----  
Name      Sent LSN      Write LSN      Flush LSN      Replay LSN      Write Lag      Flush Lag      Replay Lag      State      Sync State      Sync Priority  
-----  
epkclul-2 0/49000060    0/49000060    0/49000060    0/49000060    00:00:00      00:00:00      00:00:00      streaming  async           0  
epkclul-6 0/49000060    0/49000060    0/49000060    0/49000060    00:00:00      00:00:00      00:00:00      streaming  async           0  
  
Instances status  
-----  
Name      Database Size  Current LSN      Replication role  Status  QoS      Manager Version  
-----  
epkclul-1 187 MB         0/49000060      Primary           OK      BestEffort 1.15.1  
epkclul-2 187 MB         0/49000060      Standby (async)  OK      BestEffort 1.15.1  
epkclul-6 187 MB         0/49000060      Standby (async)  OK      BestEffort 1.15.1
```



EDB Postgres for Kubernetes

Horizontal Scaling - Scale-In the Cluster - from 3 to 2 Instances

```
$ kubectl scale cluster epkclul --replicas=2  
cluster.postgresql.k8s.enterprisedb.io/epkclul scaled
```

```
root@k8s:cluster-ocpsno$ kubectl cnv status epkclul  
Cluster Summary  
Name: epkclul  
Namespace: epkclul  
System ID: 7110015503906877463  
PostgreSQL Image: quay.io/enterprisedb/postgresql:14.1  
Primary instance: epkclul-1  
Status: Cluster in healthy state  
Instances: 2  
Ready instances: 2  
Current Write LSN: 0/49000060 (Timeline: 7 - WAL File: 0000000700000000000000049)  
  
Certificates Status  
Certificate Name Expiration Date Days Left Until Expiration  
-----  
epkclul-ca 2022-09-15 01:26:22 +0000 UTC 80.51  
epkclul-replication 2022-09-15 01:26:22 +0000 UTC 80.51  
epkclul-server 2022-09-15 01:26:22 +0000 UTC 80.51  
  
Continuous Backup status  
First Point of Recoverability: 2022-06-17T01:36:43Z  
Working WAL archiving: OK  
WALs waiting to be archived: 0  
Last Archived WAL: 000000070000000000000048.00000028.backup @ 2022-06-26T13:02:12.800909Z  
Last Failed WAL: -  
  
Streaming Replication status  
Name Sent LSN Write LSN Flush LSN Replay LSN Write Lag Flush Lag Replay Lag State Sync State Sync Priority  
-----  
epkclul-2 0/49000060 0/49000060 0/49000060 0/49000060 00:00:00 00:00:00 00:00:00 streaming async 0  
  
Instances status  
Name Database Size Current LSN Replication role Status QoS Manager Version  
-----  
epkclul-1 187 MB 0/49000060 Primary OK BestEffort 1.15.1  
epkclul-2 187 MB 0/49000060 Standby (async) OK BestEffort 1.15.1
```



EDB Postgres for Kubernetes

Vertical Scaling - Update Cluster Resource Requests and Limits

```
$ kubectl edit cluster epkclul
cluster.postgresql.k8s.enterprisedb
.io/epkclul edited
```

```
$ kubectl get cluster epkclul
-ojson | jq .spec.resources
{
  "limits": {
    "cpu": "1",
    "memory": "1Gi"
  },
  "requests": {
    "cpu": "1",
    "memory": "1Gi"
  }
}
```

```
root@k8s:cluster-ocpsno$ kubectl cnp status epkclul
Cluster Summary
Name: epkclul
Namespace: epkclul
System ID: 7110015503906877463
PostgreSQL Image: quay.io/enterprisedb/postgresql:14.2
Primary instance: epkclul-1
Status: Cluster in healthy state
Instances: 2
Ready instances: 2
Current Write LSN: 0/4E001450 (Timeline: 9 - WAL File: 000000090000000000000004E)

Certificates Status
Certificate Name      Expiration Date      Days Left Until Expiration
-----
epkclul-ca           2022-09-15 01:26:22 +0000 UTC  80.49
epkclul-replication 2022-09-15 01:26:22 +0000 UTC  80.49
epkclul-server       2022-09-15 01:26:22 +0000 UTC  80.49

Continuous Backup status
First Point of Recoverability: 2022-06-17T01:36:43Z
Working WAL archiving: OK
WALs waiting to be archived: 0
Last Archived WAL: 00000008000000000000000004E.partial @ 2022-06-26T13:37:02.499853Z
Last Failed WAL: -

Streaming Replication status
Name      Sent LSN      Write LSN      Flush LSN      Replay LSN      Write Lag      Flush Lag      Replay Lag      State      Sync State      Sync Priority
-----
epkclul-2 0/4E001450    0/4E001450    0/4E001450    0/4E001450    00:00:00      00:00:00      00:00:00      streaming  async          0

Instances status
Name      Database Size  Current LSN      Replication role  Status  QoS      Manager Version
-----
epkclul-1 187 MB         0/4E001450      Primary           OK      Guaranteed  1.15.1
epkclul-2 187 MB         0/4E001450      Standby (async)   OK      Guaranteed  1.15.1
```



EDB Postgres for Kubernetes In Action

- Backup and Recovery
- Disaster Recovery Plan

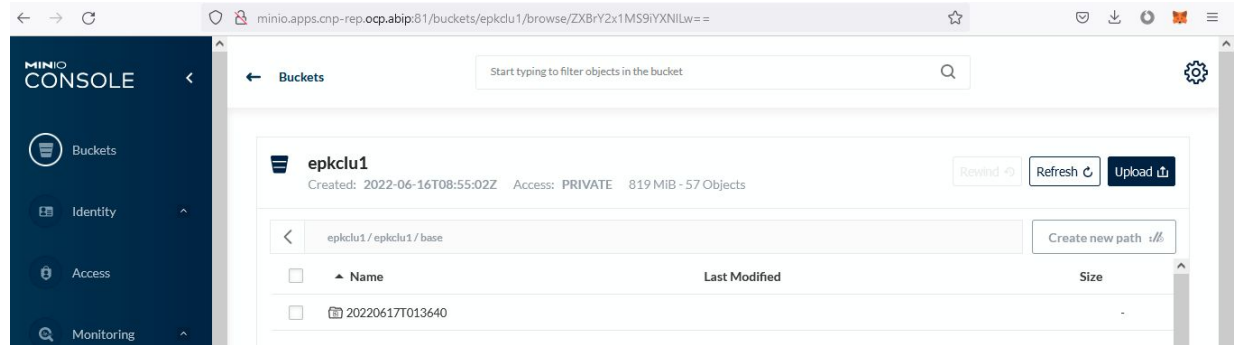
EDB Postgres for Kubernetes

Take On-Demand Backups and Verify Backup Files

```
$ cat backup-epkclul.yaml
apiVersion: postgresql.k8s.enterpris
edb.io/v1
kind: Backup
metadata:
  name: epkclul-backup1
spec:
  cluster:
    name: epkclul
```

```
$ kubectl apply -f backup-epkclul.yaml
backup.postgresql.k8s.enterprisedb.io/epkclul-backup1
created
```

```
root@k8s:cluster-ocpsno$ kubectl get backup
NAME                AGE    CLUSTER  PHASE    ERROR
epkclul-backup1    9d    epkclul  completed
```



EDB Postgres for Kubernetes

Schedule the Backup

Field name	Mandatory?	Allowed values	Allowed special characters
Seconds	Yes	0-59	* / , -
Minutes	Yes	0-59	* / , -
Hours	Yes	0-23	* / , -
Day of month	Yes	1-31	* / , - ?
Month	Yes	1-12 or JAN-DEC	* / , -
Day of week	Yes	0-6 or SUN-SAT	* / , - ?

```
root@k8s:cluster-ocpsno$ kubectl get backup
NAME                                AGE    CLUSTER  PHASE    ERROR
backup-epkclul-scheduled-1656266400 3h46m  epkclul  completed
epkclul-backup1                       10d    epkclul  completed
```

```
$ cat
backup-epkclul-scheduled.yaml
apiVersion:
postgresql.k8s.enterprisedb.io/v1
kind: ScheduledBackup
metadata:
  name: backup-epkclul-scheduled
spec:
  schedule: "0 0 0 * * *"
  cluster:
    name: epkclul
```

```
$ kubectl apply -f
backup-epkclul-scheduled.yaml
scheduledbackup.postgresql.k8s.ente
rprisedb.io/backup-epkclul-schedule
d created
```



EDB Postgres for Kubernetes

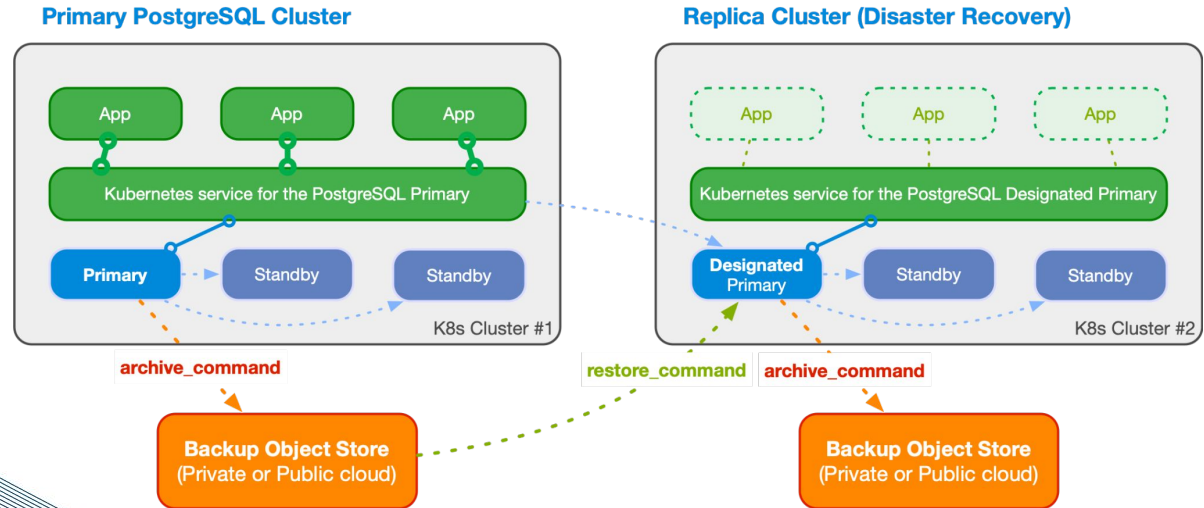
Backup Retention Policy

```
$ kubectl get cluster epkclul1 -ojson | jq .spec.backup
{
  "barmanObjectStore": {
    "destinationPath": "s3://epkclul1/",
    "endpointURL": "http://minio.velero.svc:9000",
    "s3Credentials": {
      "accessKeyId": {
        "key": "MINIO_ACCESS_KEY",
        "name": "minio-creds"
      },
      "inheritFromIAMRole": false,
      "secretAccessKey": {
        "key": "MINIO_SECRET_KEY",
        "name": "minio-creds"
      }
    }
  },
  "retentionPolicy": "30d"
}
```



EDB Postgres for Kubernetes

Physical Replica Clusters



EDB Postgres for Kubernetes

Primary/Replica Cluster - Review Source/Primary Cluster

```
root@k8s:cluster-ocpsno$ kubectl cnv status epkclul
Cluster Summary
Name: epkclul
Namespace: epkclul
System ID: 7110015503906877463
PostgreSQL Image: quay.io/enterprisedb/postgresql:14.2
Primary instance: epkclul-1
Status: Cluster in healthy state
Instances: 2
Ready instances: 2
Current Write LSN: 0/4E001450 (Timeline: 9 - WAL File: 000000090000000000000004E)

Certificates Status
Certificate Name      Expiration Date      Days Left Until Expiration
-----
epkclul-ca           2022-09-15 01:26:22 +0000 UTC  80.49
epkclul-replication 2022-09-15 01:26:22 +0000 UTC  80.49
epkclul-server       2022-09-15 01:26:22 +0000 UTC  80.49

Continuous Backup status
First Point of Recoverability: 2022-06-17T01:36:43Z
Working WAL archiving: OK
WALs waiting to be archived: 0
Last Archived WAL: 000000080000000000000004E.partial @ 2022-06-26T13:37:02.499853Z
Last Failed WAL: -

Streaming Replication status
Name      Sent LSN      Write LSN      Flush LSN      Replay LSN      Write Lag      Flush Lag      Replay Lag      State      Sync State      Sync Priority
-----
epkclul-2 0/4E001450    0/4E001450    0/4E001450    0/4E001450    00:00:00      00:00:00      00:00:00      streaming  async          0

Instances status
Name      Database Size  Current LSN      Replication role  Status  QoS      Manager Version
-----
epkclul-1 187 MB         0/4E001450      Primary           OK      Guaranteed  1.15.1
epkclul-2 187 MB         0/4E001450      Standby (async)  OK      Guaranteed  1.15.1
```



EDB Postgres for Kubernetes

Create Replica Cluster

```
$ kubectl get cluster epkclu2 -n epkclu2
-ojson | jq .spec.bootstrap
{
  "pg_basebackup": {
    "source": "epkclu1"
  }
}
```

```
$ kubectl get cluster epkclu2 -n epkclu2
-ojson \
| jq
.spec.backup.barmanObjectStore.destinationPath
"s3://epkclu2/"
```

```
$ kubectl get cluster epkclu2 -n epkclu2
-ojson | jq .spec.replica
{
  "enabled": true,
  "source": "epkclu1"
}
```

```
$ kubectl get cluster epkclu2 -n epkclu2
-ojson | jq .spec.externalClusters
[
  {
    "connectionParameters": {
      "dbname": "postgres",
      "host": "epkclu1-rw.epkclu1.svc",
      "sslmode": "verify-full",
      "user": "streaming_replica"
    },
    "name": "epkclu1",
    "sslCert": {
      "key": "tls.crt",
      "name": "epkclu1-replication"
    },
    "sslKey": {
      "key": "tls.key",
      "name": "epkclu1-replication"
    },
    "sslRootCert": {
      "key": "ca.crt",
      "name": "epkclu1-ca"
    }
  }
]
```



EDB Postgres for Kubernetes

Review Replica Cluster using CNP Plugin

```
root@k8s:cluster-ocpsno$ kubectl cnp status epkclu2
Cluster Summary
Name:          epkclu2
Namespace:    epkclu2
System ID:    7117028855415050261
PostgreSQL Image: quay.io/enterprisedb/postgresql:14.2
Primary instance: epkclu2-1
Status:       Cluster in healthy state
Instances:    2
Ready instances: 2

Certificates Status
Certificate Name      Expiration Date          Days Left Until Expiration
-----
epkclu2-ca           2022-10-10 22:52:14 +0000 UTC  89.99
epkclu2-replication 2022-10-10 22:52:15 +0000 UTC  89.99
epkclu2-server      2022-10-10 22:52:14 +0000 UTC  89.99

Continuous Backup status
First Point of Recoverability: Not Available
Working WAL archiving:       Starting Up
WALs waiting to be archived: 0
Last Archived WAL:          -
Last Failed WAL:            -

Streaming Replication status
Name      Sent LSN      Write LSN      Flush LSN      Replay LSN      Write Lag      Flush Lag      Replay Lag      State      Sync State      Sync Priority
-----
epkclu2-2 0/16000060   0/16000060   0/16000060   0/16000060   00:00:00     00:00:00     00:00:00     streaming  async           0

Instances status
Name      Database Size      Current LSN      Replication role      Status      QoS           Manager Version
-----
epkclu2-1 33 MB              0/16000060     Designated primary    OK          BestEffort    1.15.1
epkclu2-2 33 MB              0/16000060     Standby (async)       OK          BestEffort    1.15.1
```

```
root@k8s:~$ kubectl get cluster epkclu2 -ojson \
> | jq .spec.replica
{
  "enabled": true,
  "source": "epkclu1"
}
```



EDB Postgres for Kubernetes

Primary/Replica Cluster - Insert Data at Primary Cluster then Verify at Replica Cluster

```
$ kubectl get svc epkclu1-rw-nodeport -ojson | jq .spec.ports[0].nodePort  
30264
```

```
$ kubectl get svc -n epkclu2 epkclu2-ro-nodeport -ojson | jq .spec.ports[0].nodePort  
31311
```

```
[root@dns ~]# psql -h 192.168.24.40 -p 30264 -U app app -c "insert into apjse values (105,'DBaaS')"  
INSERT 0 1  
[root@dns ~]# psql -h 192.168.24.40 -p 31311 -U app app -c "select * from apjse where id=105"  
 id | name  
-----+-----  
 105 | DBaaS  
(1 row)
```



EDB Postgres for Kubernetes In Action

- External Client Access
- Monitoring
- Logging

EDB Postgres for Kubernetes

External Client Access - Port Forwarding

```
$ kubectl port-forward svc/epkclu1-rw 5555:5432 --address 192.168.24.40
Forwarding from 192.168.24.40:5555 -> 5432
```

```
$ psql -h 192.168.24.40 -p 5555 -U app app
Password for user app:
psql (14.1.0, server 14.1)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-ECDSA-AES256-GCM-SHA384, bits: 256,
compression: off)
Type "help" for help.
```



EDB Postgres for Kubernetes

External Client Access - NodePort

```
$ kubectl expose svc epkclul-rw --type=NodePort --name=epkclul-rw-nodeport  
--generator="service/v2"  
service/epkclul-rw-nodeport exposed
```

```
$ kubectl get svc | grep nodeport  
epkclul-rw-nodeport
```

```
NodePort 10.110.146.212 <none> 5432:30264/TCP 36s
```

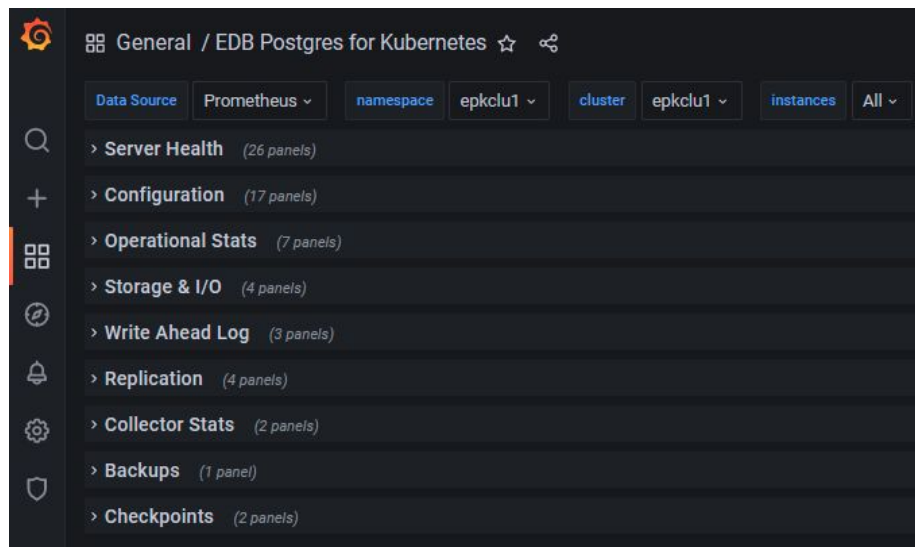
The screenshot shows the EDB Admin console interface. On the left, the 'Create - Server' dialog is open, with the 'Connection' tab selected. The fields are filled with: Host name/address: 192.168.24.40, Port: 30264, Maintenance database: app, Username: app, Password: [masked], Save password?: checked, Role: [empty], and Service: [empty]. At the bottom of the dialog are 'Cancel', 'Reset', and 'Save' buttons. On the right, the 'Admin' browser shows a tree view of servers. The 'epkclul-rw' server is expanded, showing its configuration, including a 'Databases (2)' section with 'app' and 'postgres' databases, and a 'Schemas (1)' section with 'postgres' schema. Other servers listed include PostgreSQL 12, Postgres Enterprise Manager, ascend, biganimal, and various other instances.



EDB Postgres for Kubernetes

Monitoring - Enable PodMonitor - Import CNP Grafana Dashboard

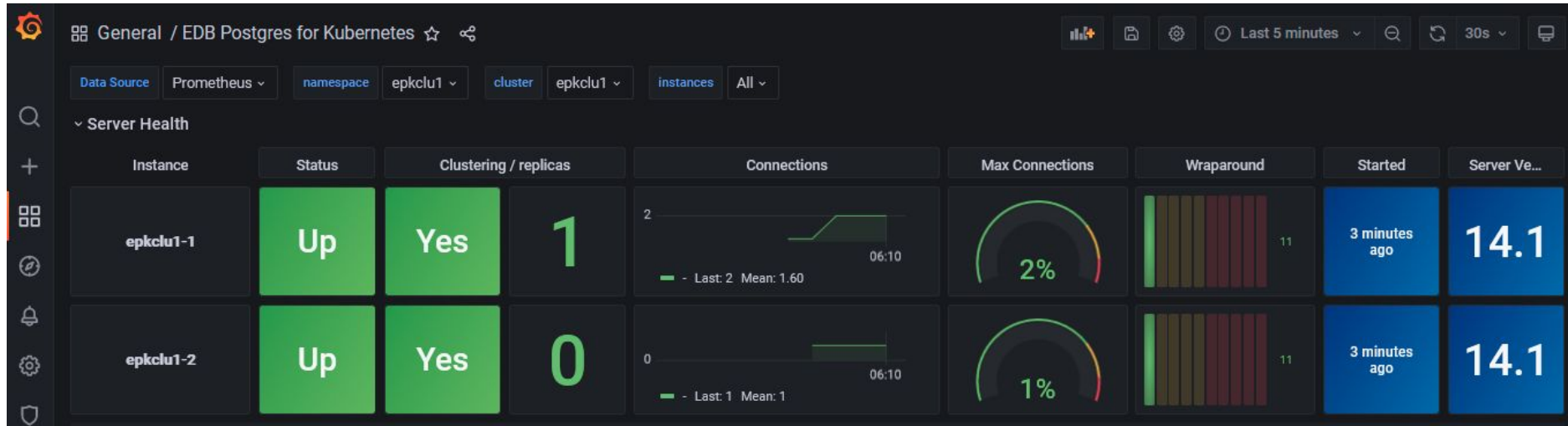
```
$ kubectl get cluster epkclu1 -ojson | jq .spec.monitoring.enablePodMonitor  
true
```



EDB Postgres for Kubernetes

Monitoring - Enable PodMonitor - Import CNP Grafana Dashboard

```
$ kubectl get cluster epkclu1 -ojson | jq .spec.monitoring.enablePodMonitor  
true
```



Dashboard:
<https://github.com/EnterpriseDB/cnp-sandbox/blob/main/charts/cnp-sandbox/dashboard.json>



EDB Postgres for Kubernetes

Logging - Check PostgreSQL Log

```
$ kubectl logs -n epkclul epkclul-1 | jq -r 'select(.logger=="postgres") |
[ (.ts|strflocaltime("%Y-%m-%dT%H:%M:%S %Z")), .record.message ] | @csv' | head
-10
"2022-06-26T13:36:43 UTC",
"2022-06-26T13:36:43 UTC",
"2022-06-26T13:36:43 UTC",
"2022-06-26T13:36:43 UTC",
"2022-06-26T13:36:43 UTC","ending log output to stderr"
"2022-06-26T13:36:43 UTC","starting PostgreSQL 14.2 on x86_64-pc-linux-gnu,
compiled by gcc (GCC) 8.5.0 20210514 (Red Hat 8.5.0-4), 64-bit"
"2022-06-26T13:36:43 UTC","listening on IPv4 address ""0.0.0.0"", port 5432"
"2022-06-26T13:36:43 UTC","listening on IPv6 address "":::"", port 5432"
"2022-06-26T13:36:43 UTC","listening on Unix socket
"/controller/run/.s.PGSQL.5432""
"2022-06-26T13:36:43 UTC","database system was shut down in recovery at
2022-06-26 13:36:05 UTC"
```



EDB Postgres for Kubernetes

Logging - Get FATAL errors from a specific PostgreSQL pod

```
$ kubectl logs -n epkclul1 epkclul1-1 | jq -r '.record | select(.error_severity == "FATAL")'
{
  "log_time": "2022-06-26 13:36:53.082 UTC",
  "process_id": "138",
  "session_id": "62b860f1.8a",
  "session_line_num": "3",
  "session_start_time": "2022-06-26 13:36:49 UTC",
  "transaction_id": "0",
  "error_severity": "FATAL",
  "sql_state_code": "08006",
  "message": "could not send end-of-streaming message to primary: SSL connection has
been closed unexpectedly\nno COPY in progress",
  "backend_type": "walreceiver",
  "query_id": "0"
}
```



EDB Postgres for Kubernetes

Logging - Query PostgreSQL Logs

The screenshot displays the Elastic Search web interface. At the top, the Elastic logo and a search bar are visible. Below the search bar, there are navigation options like 'Discover' and 'Options'. The main search area shows a query: `kubernetes.namespace_name: epkclu1` and a filter: `log:"logger":"postgres"`. The results are displayed in a table with columns for 'Time' and 'Document'. The 'Document' column contains log entries from PostgreSQL, including information about the logging pod, session details, and transaction status.

Time	Document
> Jun 26, 2022 @ 09:20:46.540	<pre>kubernetes.namespace_name: epkclu1 log: {"level":"info","ts":1656210046.5402734,"logger":"postgres","msg":"record","logging_pod":"epkclu1-2","record":{"log_time":"2022-06-26 02:20:46.539 UTC","process_id":"31","session_id":"62b7c277.1f","session_line_num":"9","session_start_time":"2022-06-26 02:20:39 UTC","virtual_transaction_id":"1/0","transaction_id":"0","error_severity":"LOG","sql_state_code":"00000","message":"redo starts</pre>
> Jun 26, 2022 @ 09:20:46.362	<pre>kubernetes.namespace_name: epkclu1 log: {"level":"info","ts":1656210046.3616192,"logger":"postgres","msg":"record","logging_pod":"epkclu1-2","record":{"log_time":"2022-06-26 02:20:46.359 UTC","process_id":"147","session_id":"62b7c27e.93","session_line_num":"1","session_start_time":"2022-06-26 02:20:46 UTC","transaction_id":"0","error_severity":"LOG","sql_state_code":"00000","message":"started streaming WAL from primary at</pre>
> Jun 26, 2022 @ 09:20:46.343	<pre>kubernetes.namespace_name: epkclu1 log: {"level":"info","ts":1656210046.3431962,"logger":"postgres","msg":"record","logging_pod":"epkclu1-2","record":{"log_time":"2022-06-26 02:20:46.338 UTC","process_id":"31","session_id":"62b7c277.1f","session_line_num":"7","session_start_time":"2022-06-26 02:20:39 UTC","virtual_transaction_id":"1/0","transaction_id":"0","error_severity":"LOG","sql_state_code":"00000","message":"consistent</pre>
> Jun 26, 2022 @ 09:20:46.343	<pre>kubernetes.namespace_name: epkclu1</pre>



EDB Postgres for Kubernetes In Action

- Rolling Updates

EDB Postgres for Kubernetes

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 Postgres for Kubernetes

Rolling Updates - Minor Upgrade - 14.1 to 14.2

```
$ kubectl edit cluster epkclul  
cluster.postgresql.k8s.enterprisedb.io/epkclul edited
```

```
$ kubectl get cluster epkclul -ojson | jq .spec.imageName  
"quay.io/enterprisedb/postgresql:14.2"
```


```
root@k8s:cluster-ocpsno$ kubectl cnp status epkclul  
Cluster Summary  
Name: epkclul  
Namespace: epkclul  
System ID: 7110015503906877463  
PostgreSQL Image: quay.io/enterprisedb/postgresql:14.2  
Primary instance: epkclul-2  
Status: Cluster in healthy state  
Instances: 2  
Ready instances: 2  
Current Write LSN: 0/4C001288 (Timeline: 8 - WAL File: 000000080000000000000004C)  
  
Certificates Status  
-----  
Certificate Name      Expiration Date      Days Left Until Expiration  
-----  
epkclul-replication  2022-09-15 01:26:22 +0000 UTC  80.50  
epkclul-server       2022-09-15 01:26:22 +0000 UTC  80.50  
epkclul-ca           2022-09-15 01:26:22 +0000 UTC  80.50  
  
Continuous Backup status  
First Point of Recoverability: 2022-06-17T01:36:43Z  
Working WAL archiving: OK  
WALs waiting to be archived: 0  
Last Archived WAL: 000000070000000000000004C.partial @ 2022-06-26T13:29:22.643921Z  
Last Failed WAL: -  
  
Streaming Replication status  
-----  
Name      Sent LSN      Write LSN      Flush LSN      Replay LSN      Write Lag      Flush Lag      Replay Lag      State      Sync State      Sync Priority  
-----  
epkclul-1 0/4C001288  0/4C001288  0/4C001288  0/4C001288  00:00:00  00:00:00  00:00:00  streaming  async          0  
  
Instances status  
-----  
Name      Database Size      Current LSN      Replication role      Status      QoS      Manager Version  
-----  
epkclul-1 187 MB              0/4C001288      Standby (async)      OK          BestEffort  1.15.1  
epkclul-2 187 MB              0/4C001288      Primary              OK          BestEffort  1.15.1
```



Get Your Evaluation Key

Evaluation Key *for* Oracle-Compatible PostgreSQL

<https://cloud-native.enterprisedb.com/trial/>



Cloud Native PostgreSQL
Request a free trial
license

Cloud Native PostgreSQL

Cluster namespace

Cluster name

I agree to the terms and conditions

Coupon

Submit

Q & A

Call to action

Submit an e-survey form and stand a chance to win our Lucky Draw Prize
– a **Samsung Galaxy Fit 2**

