CYBERTEC
POSTGRESQL SERVICES & SUPPORT

PostgreSQL on Kubernetes, across data centers

How the CYBERTEC PG Operator keeps a database alive when a whole site fails

Hans-Jürgen Schönig — CYBERTEC
www.cybertec-postgresql.com

What we will cover

From one cluster to several data centers, and how to start

01 **What it is**
An operator, and who does what.

02 **Why one data center is not enough**
A cluster is one failure domain.

03 **Images**
What actually runs in the pod.

04 **Multiple data centers**
Failover, and how a site rejoins.

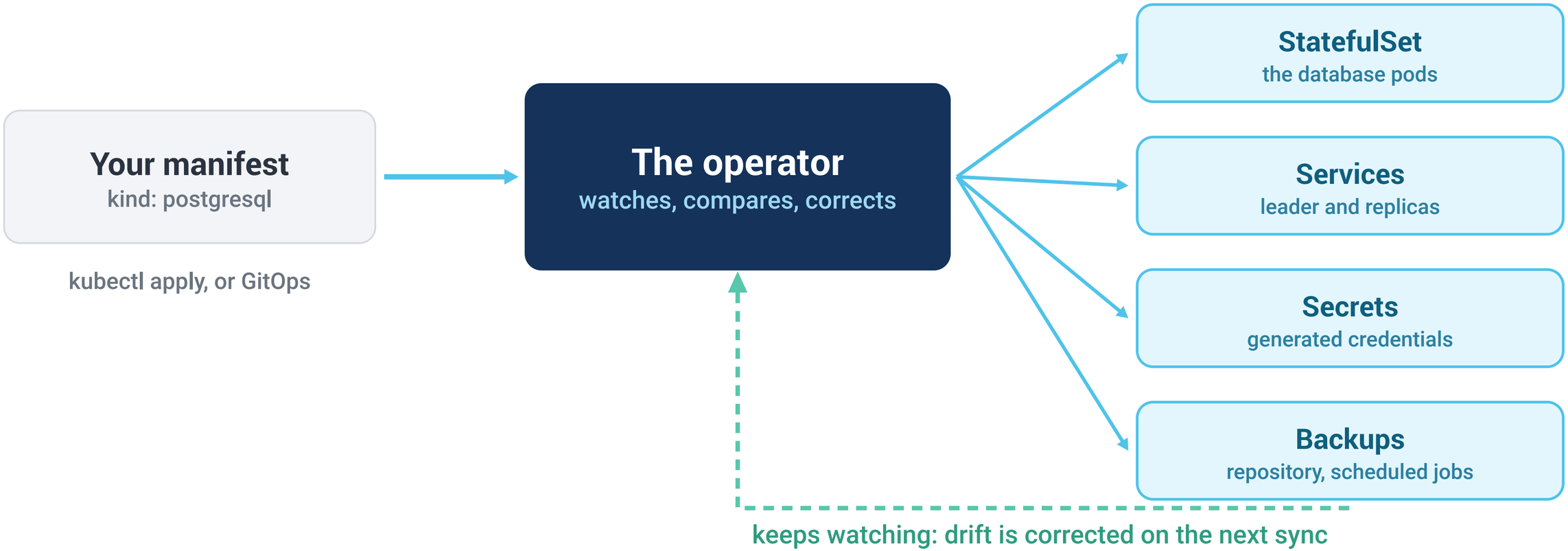
05 **From zero to running**
What you actually type.

What it is

A Kubernetes operator for PostgreSQL

An operator is a loop

You declare the cluster you want. It makes it so, and keeps it so



The CYBERTEC PG Operator

CPO: open source, built for production PostgreSQL

Declarative

Describe a cluster in a custom resource. Provisioning needs no direct Kubernetes API access.

Highly available

Patroni handles leader election and automatic failover.

Backup and recovery

Point-in-time recovery, to a volume, S3, GCS or Azure.

Updates and upgrades

Rolling updates, minor upgrades and in-place major upgrades. PostgreSQL 13 to 18.

Beyond one cluster

Connection pooling, standby clusters, and multisite with automatic failover.

Open source

MIT licensed. Kubernetes and OpenShift, in the cloud or on your own hardware.

Who does what

Kubernetes and Patroni protect different things

Kubernetes

keeps the pod alive

Recreates a crashed or evicted database pod.

Patroni

decides who is primary

Holds the leader lease, streams WAL, promotes a replica when the leader is lost.

The operator

shapes the cluster

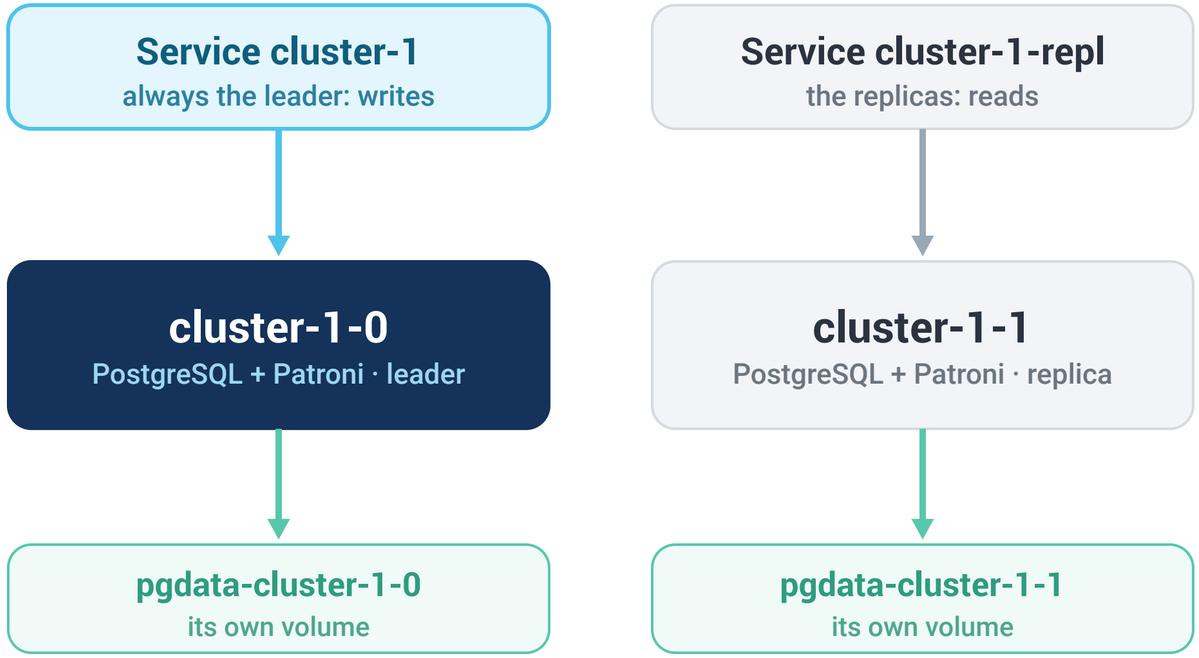
Turns your manifest into pods, services and secrets, and reconciles every change.

Patroni runs inside the database pods.

So failover does not depend on the operator being up: an operator restart never touches the data path.

What it creates for one cluster

Everything below comes from a single manifest



Secrets

Generated credentials for the superuser, replication and application users.

A disruption budget

Kubernetes may not drain the pod that holds the leader role.

Backups, if configured

A backup repository pod and scheduled backup jobs.

Why one data center is not enough

A cluster is still one failure domain

Failure domains

Each layer fails on its own. Something has to cover each one

A pod or process crashes
Kubernetes restarts it

A node dies, the leader with it
Patroni promotes a replica on another node

The whole data center is gone
Power, network, the cluster's own control plane

Everything in one cluster shares the same fate, whatever the replica count.

The usual answers, and where they stop

Three ways to survive a site, with different trade-offs

Backups, restore elsewhere

You rebuild

- Always works.
- The recovery time is the time it takes to restore.

A standby cluster

You promote by hand

- A second cluster follows the first.
- A primary cannot be demoted: the old one is rebuilt as a standby.

Multisite

Automatic

- Automatic failover across sites.
- The old primary steps down and rejoins.
- Needs a third place for the quorum.

You do the work

The rest of this talk is the third column.

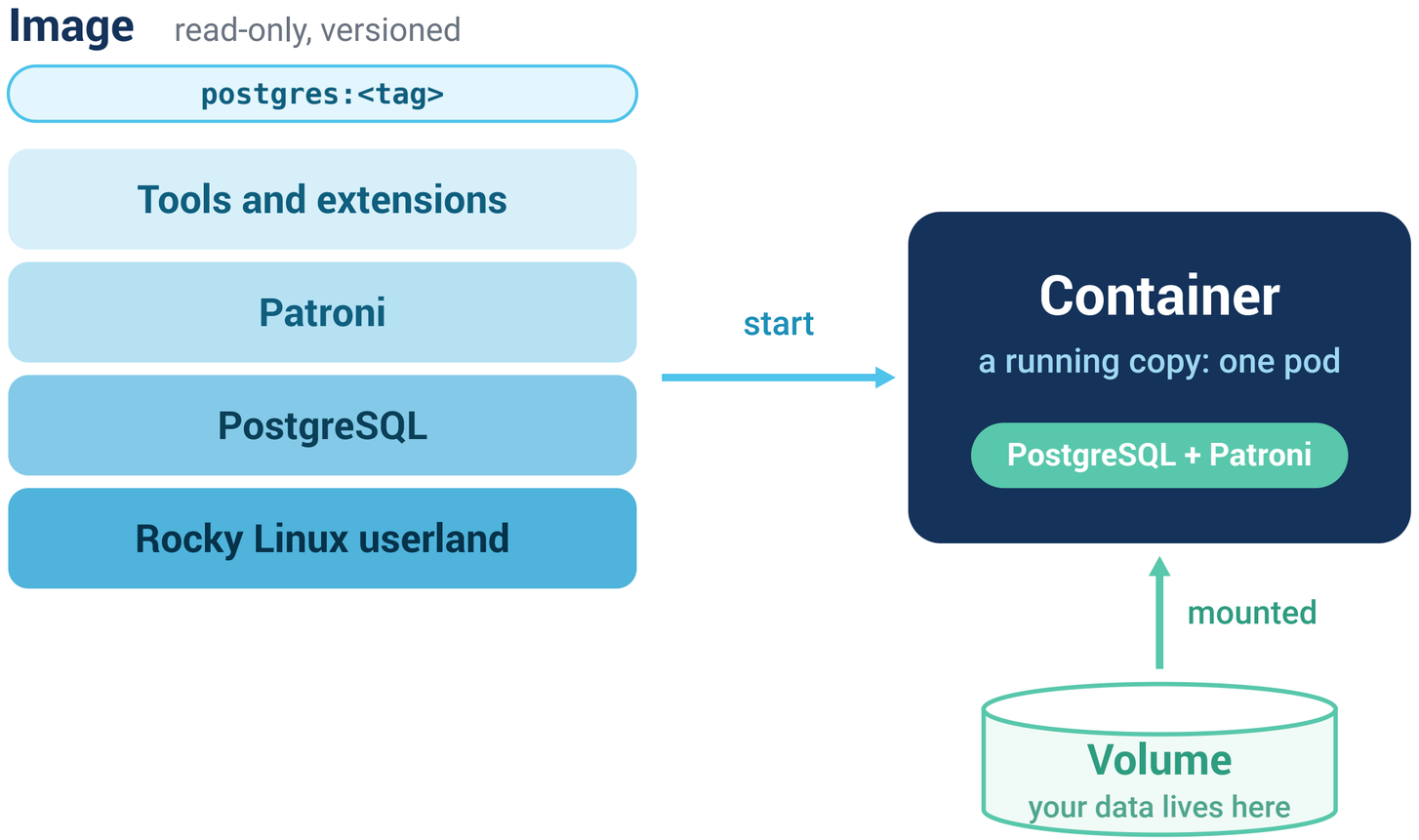
The system does the work

Images

What actually runs in the pod

What an image is

A sealed, versioned bundle. The data stays outside



Data stays outside

Throw the container away and the volume is untouched.

Same bits everywhere

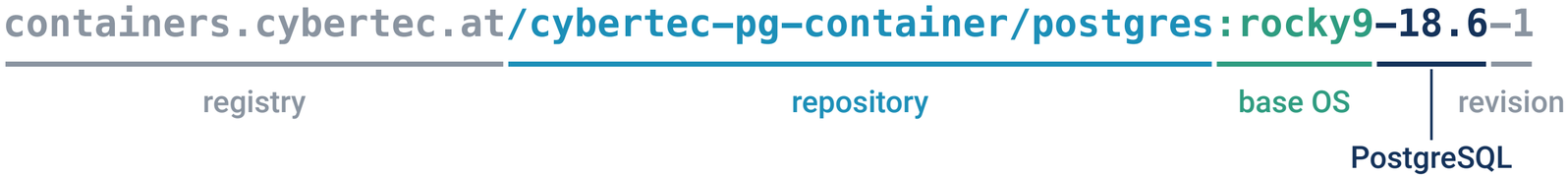
Development, test and production start from the same image.

A version you can name

The tag says which PostgreSQL you are running.

The image a cluster runs

The tag is the version



A cluster names its image in the manifest

`dockerImage`

PostgreSQL and Patroni: the database pods

`containers.cybertec.at/cybertec-pg-container/postgres:rocky9-18.6-1`

Updating is changing one line

update

```
# A minor version update is a one-line change to the manifest.
spec:
  dockerImage: 'containers.cybertec.at/cybertec-pg-container/postgres:rocky9-18.6-1'

# apply it
kubectl apply -n cpo -f postgres.yaml

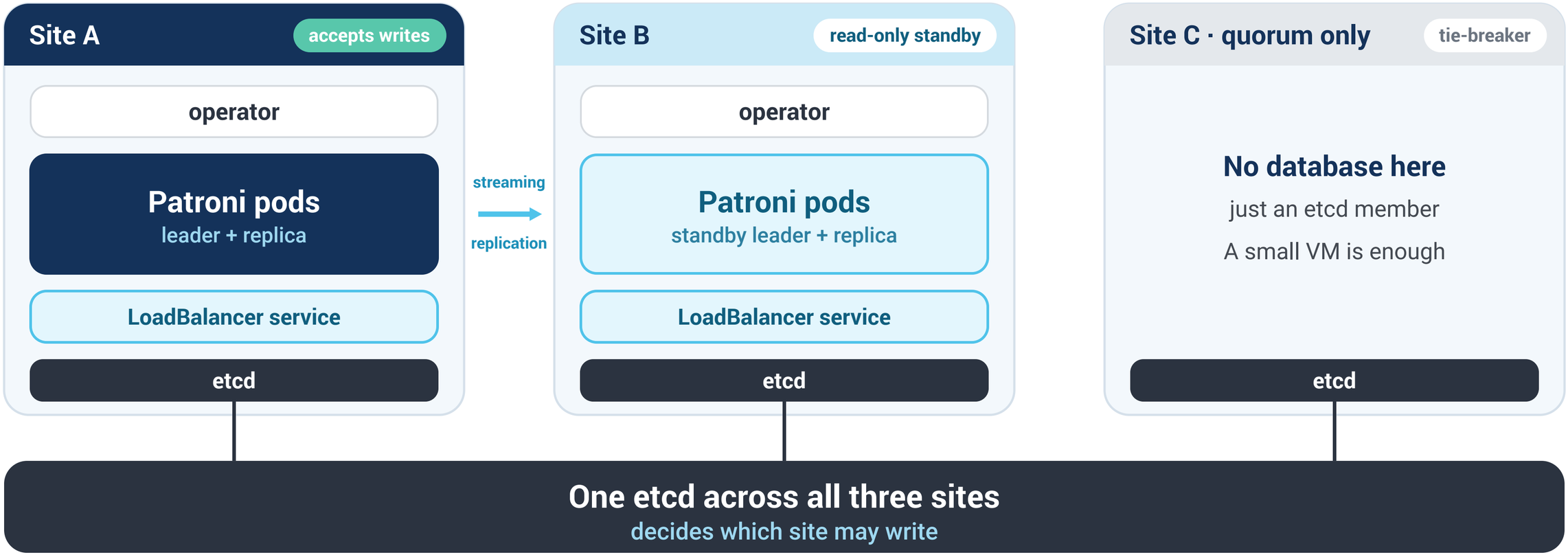
# what happens, according to the operator docs:
#   the pods restart onto the new image, on the same data directory
#   on a multi-instance cluster the update is rolling
```

Multiple data centers

Two consensus layers, one logical cluster

The architecture

Independent Kubernetes clusters, one agreed leader



Why a third place

Two data centers cannot tell a dead neighbour from a cut cable

Everything is fine



3 of 3 agree

One site is lost



2 of 3: still a majority, decide

Only two sites, link cut



1 of 2 each: no majority, nobody may write

An odd number of voters, at least three.

Any one site may hold only one of them, so a whole site can be lost without losing the majority.

That agreement is what stops two sites from both accepting writes: a split brain.

What multisite needs

A short list, and none of it is exotic

Two or more clusters

Kubernetes 1.25 or later, or OpenShift 4.12 or later.

Reachable LoadBalancers

Services with external IPs that the other clusters can reach.

An etcd in three places

API v3, one voter per site. The quorum site can be a small VM: 2 vCPU, 2 GB, 20 GB.

Object storage

S3-compatible storage for the backups.

Who may write

etcd

```
# The whole multisite state is a few keys. Read them from any database pod.
/multisite/cpo/multisite-cluster/leader
s1-multisite-cluster

/multisite/cpo/multisite-cluster/members/s1-multisite-cluster
{"host":"192.168.49.102","port":5432}

/multisite/cpo/multisite-cluster/members/s2-multisite-cluster
{"host":"192.168.50.102","port":5432}

# /leader names the site that may accept writes.
# Each site's leader advertises its externally visible address under /members.
```

Failover, step by step

A whole site disappears

1

Site A is lost

Its pods stop renewing the leader lease in etcd.

2

The lease lapses

etcd still has a majority: sites B and C.

3

Site B takes the lead

Its standby leader acquires the multisite lease.

4

Site B promotes

Its Patroni cluster becomes the primary.

5

Clients move

They connect to site B's service.

about 90 seconds

for an automatic cross-site failover, in the validated test in the CPO docs

Measure it in your own environment: network, storage and load move it.

What you see during a failover

CYBERTEC
POSTGRESQL SERVICES & SUPPORT

kubectl describe postgresql

```
# the site that now holds the lead
Status:
  Multisite:                  Leader
  Postgres Cluster Status:  Running
Events:
  Normal  Promote  13s   patroni  Acquired multisite leader status

# a site that lost the lead
Status:
  Multisite:                  Standby
Events:
  Normal  Demote   62s   patroni  Lost leader lock to s1-multisite-cluster
```

Rejoin: the old site comes back

Nobody has to decide anything by hand

01 **The site returns**
Its pods start and ask etcd who leads.

02 **It finds a new leader**
Site B holds the lease, so A does not take over.

03 **It steps down**
Patroni demotes the old primary to a standby.

04 **It follows**
Replication catches it up from site B.

Compare: a plain standby

A primary cannot be demoted to a standby. After a failover you create a new cluster as a standby and let it copy.

Multisite does the demotion for you. How much has to be copied again depends on how far the old primary got ahead of the new one.

Failing back is a switchover you choose

switchover

```
# Move the lead to site s1 on purpose: maintenance, or failing back.
curl --data-raw '{"switchover_to": "s1"}' \
     -H "Content-type: application/json" \
     http://postgres-operator.default.svc.cluster.local:8080/clusters/cpo/multisite-cluster/multisite/

# The call returns at once. The switchover itself takes a moment to coordinate.
# Watch the status and the events:
kubectl describe postgresql multisite-cluster

# Automatic failback took about 20 seconds in the validated test in the CPO docs.
```

Measure it

A failover you have not tested is a hope

Break it on purpose

Kill the leader, or simulate a site outage, and measure the real recovery time and data loss.

Know your lag

Replication between the sites decides what a failover can lose. Measure the lag you really have.

Restore, do not just back up

Test point-in-time restores, not just the creation of backups.

Plan the client path

Decide how clients find the active site. That part belongs to your platform.

From zero to running

What you actually type

Install the operator

install

```
# the tutorials repository holds the example manifests and the Helm chart
git clone https://github.com/cybertec-postgresql/CYBERTEC-operator-tutorials.git
cd CYBERTEC-operator-tutorials

# a namespace, then the operator
kubectl create namespace cpo
helm install -n cpo cpo setup/helm/operator/.

# is it up?
kubectl get pods -n cpo
```

Your first cluster

CYBERTEC
POSTGRESQL SERVICES & SUPPORT

postgres.yaml

```
apiVersion: cpo.opensource.cybertec.at/v1
kind: postgresql
metadata:
  name: cluster-1
spec:
  dockerImage: 'containers.cybertec.at/cybertec-pg-container/postgres:rocky9-18.6-1'
  numberOfInstances: 1
  postgresql:
    version: '18'
  resources:
    requests: { cpu: 500m, memory: 500Mi }
    limits:   { cpu: 500m, memory: 500Mi }
  volume:
    size: 5Gi

kubectl apply -n cpo -f postgres.yaml
```

Highly available: change one number

ha

```
# one number turns a single node into a highly available cluster
spec:
  numberOfInstances: 2          # was 1

kubectl apply -n cpo -f postgres.yaml
kubectl get pods -n cpo

NAME          READY   STATUS    RESTARTS   AGE
cluster-1-0   1/1     Running   0          3d
cluster-1-1   1/1     Running   0          31s

# the new pod is cloned from the leader and starts streaming
```

A multisite cluster from scratch

The same manifest on every site

01 **Make an etcd for three places**
One voter per site, plus a small quorum site.

02 **Tell each operator**
The etcd address, and a unique site name.

03 **Apply the same manifest twice**
Same name, same namespace, on every site.

04 **Watch it settle**
The first site to boot leads and bootstraps.

The other sites

They find the leader in etcd, copy a base backup from it, and start replicating.

Read the docs first

The manifest alone is not enough. The tutorial itself warns about it.

The multisite configuration

multisite

```
# operator configuration, once per Kubernetes cluster
multisite:
  etcd_host: global-etcd-etcd-helm.default.svc.cluster.local
  site: s1                              # s2 on the other cluster

# the cluster manifest: identical on both sites
spec:
  numberOfInstances: 1
  patroni:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    multisite:
      enable: true
```

Day two is an edit

Change the manifest, let the operator reconcile

Resize

Edit the resources: a rolling update, replicas first, then a leader switchover.

Scale

Change numberOfInstances. A new pod is cloned from the leader.

Update

Change the image tag. Pods restart onto it, on the same data.

Back up and restore

A backup block, and a restore block for point-in-time recovery.

Pool connections

pgBouncer, enabled from the same manifest.

Clone

A new cluster from a backup, described in the manifest.

When to use it, and when not

Said before you find out

It fits

- Cross-site disaster recovery is a hard requirement.
- You run your own Kubernetes or OpenShift, on premises or in the cloud.
- You want the database declared in Git, like everything else.

It does not

- One site is enough and a restore is an acceptable recovery time.
- You cannot run a third place for the quorum.
- You expect it to replace testing: restores and failovers still need drills.

Open source, MIT licensed.
24×7 commercial support for it is available from CYBERTEC.

CYBERTEC
POSTGRESQL SERVICES & SUPPORT

Try out

Source	**github.com/cybertec-postgresql/CYBERTEC-pg-operator**
Docs	**cybertec-postgresql.github.io/CYBERTEC-pg-operator**
Tutorials	**github.com/cybertec-postgresql/CYBERTEC-operator-tutorials**

Built and supported by CYBERTEC · www.cybertec-postgresql.com