tembo

# Kubernetes Killed the High Availability Star

How to stop worrying and embrace Postgres in the cloud

**Postgres Conf**
**Seattle 2024**

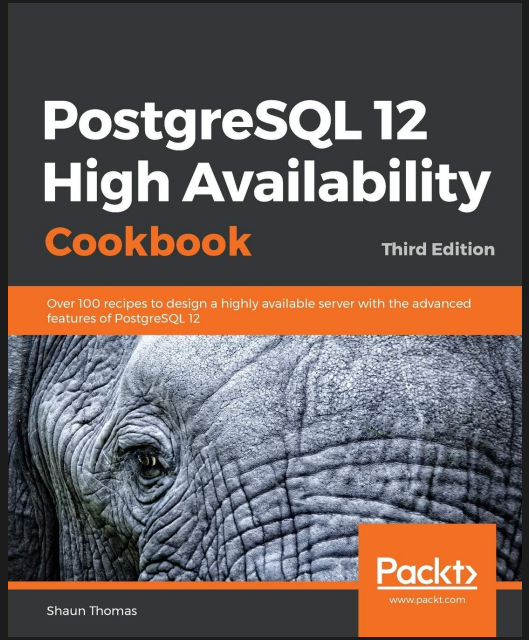
# Who are Tembo?

- Cloud Services - cloud.tembo.io
- Trunk (Extensions) - pgt.dev
- pg_vectorize
- pgmq
- pg_tier
- pg_timeseries
- And more!

# Who am I?

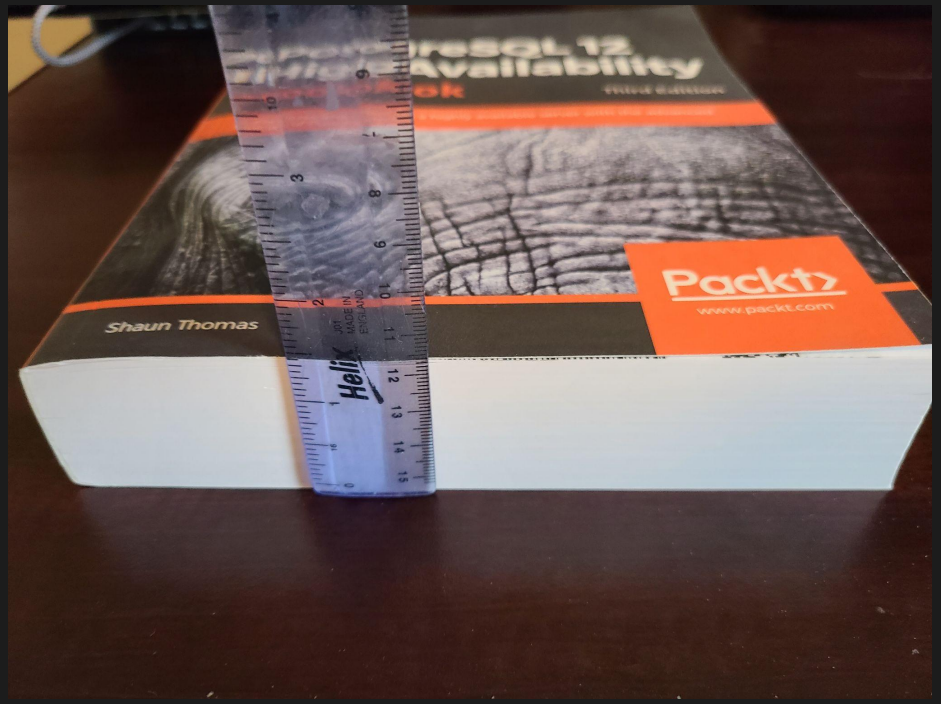
- Author
- Speaker
- Blogger
- Mentor
- Dev
- **High Availability Star**

shaun@tembo.io



# Postgres HA is Hard

# How Hard is Postgres High Availability?



# But why is Postgres High Availability Hard?

# The Difficulty of Postgres HA

**Postgres is a bag of tools**

~~Some~~ All assembly required

Used by every Postgres HA stack

- pg_basebackup - build replicas
- pg_rewind - “Fix” old Primary nodes
- pg_ctl - start / stop Postgres

Then add orchestration

# The Amount of HA Concepts

**High Availability is a content-rich topic**

A plethora of theoretical frameworks

Minimum familiarity with:

- Quorum
- Sync / Async Replication
- Split Brain
- Network Partition
- Fencing
- CAP / PACELC

# The Ecosystem of HA Tools

## HA Management

- repmgr
- pg_auto_failover
- Patroni
- Stolon
- EDB Failover Manager (EFM)
- EDB Postgres Distributed (PGD)
- Bucardo

## Pooling Proxies

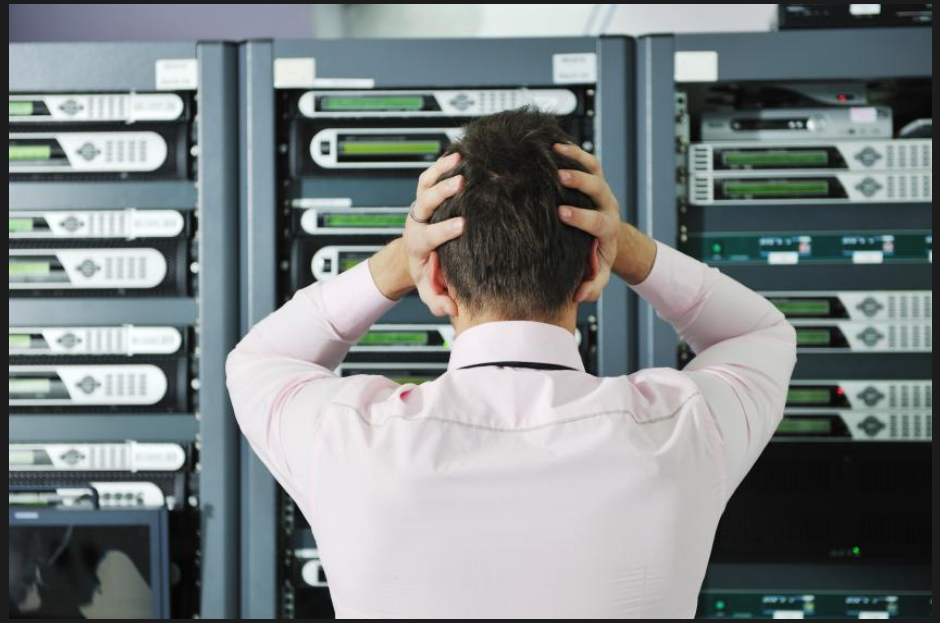
- PGBouncer
- PgPool-II
- PgCat
- Odyssey
- Supavisor

# And you still have to put it together




tembo

# All the Steps



- Configure each server
- Build each replica
- Activate HA stack
- Design access topography
- Account for edge cases
- Cross fingers

# Enough Complaining

# What’s the Solution?

# Everyone says “use Patroni!”

# Create Three VMs

**Ok, we need somewhere to run Patroni...**

Create some servers!

- AWS / GCS / Azure / OCI / etc.
- Terraform
- Docker compose
- Buy three physical systems

# Install All Software

## Necessary steps:

- Set up repos
- Install software
- Enable required services
- Repeat for all 3 systems

## Everything involved:

- Postgres
- Patroni
- HAProxy
- etcd
- pgBackrest / Barman

# Configure etcd

## etcd.conf

```
name: pgha1
data-dir: /db/etcd
initial-advertise-peer-urls:
http://pgha1:2380
listen-peer-urls: http://0.0.0.0:2380
listen-client-urls:
http://0.0.0.0:2379
advertise-client-urls:
http://pgha1:2379
initial-cluster:
"pgha1=http://pgha1:2380,pgha2=http:/
/pgha2:2380,pgha3=http://
pgha3:2380"
```

Don’t forget:

- Do this for each node
- Won’t work until etcd is running
- This is the bootstrap phase

# Configure Patroni

## cluster-name.yml

```
scope: stampede
name: pgha1
restapi:
  listen: pgha1:8008
  connect_address: pgha1:8008
etcd:
  host: pgha1:2379

postgresql:
  listen: pgha1:5432
  connect_address: pgha1:5432
  data_dir: /db/pgdata

bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 1048576
```

## cluster-name.yml (continued)

```
  postgresql:
    use_pg_rewind: true
    use_slots: true
    parameters:
      wal_level: logical
      wal_log_hints: "on"

initdb:
- encoding: UTF8
- data-checksums

pg_hba:
- host replication rep_user 10.0.30.1/24 md5
- host all 10.0.30.1/24 md5

users:
  admin:
    password: adminpass
  options:
  - createrole
  - createdb
```

# Configure HAProxy

## haproxy.cfg

```
global
    maxconn 100

defaults
    log     global
    mode    tcp
    retries 2
    timeout client 30m
    timeout connect 4s
    timeout server 30m
    timeout check 5s

listen postgres
    bind *:5000
    option httpchk
    default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
    server postgres_pg1 pgha1:5432 check port 8008
    server postgres_pg2 pgha2:5432 check port 8008
    server postgres_pg3 pgha3:5432 check port 8008
```

# Don’t forget to do that 3 times

# But That's Not All!


Still need:

- PgBouncer
- Backup (Barman, pgBackrest)
- Monitoring (Prometheus)
- Log capture (Elasticsearch?)

# Also, set everything up perfectly

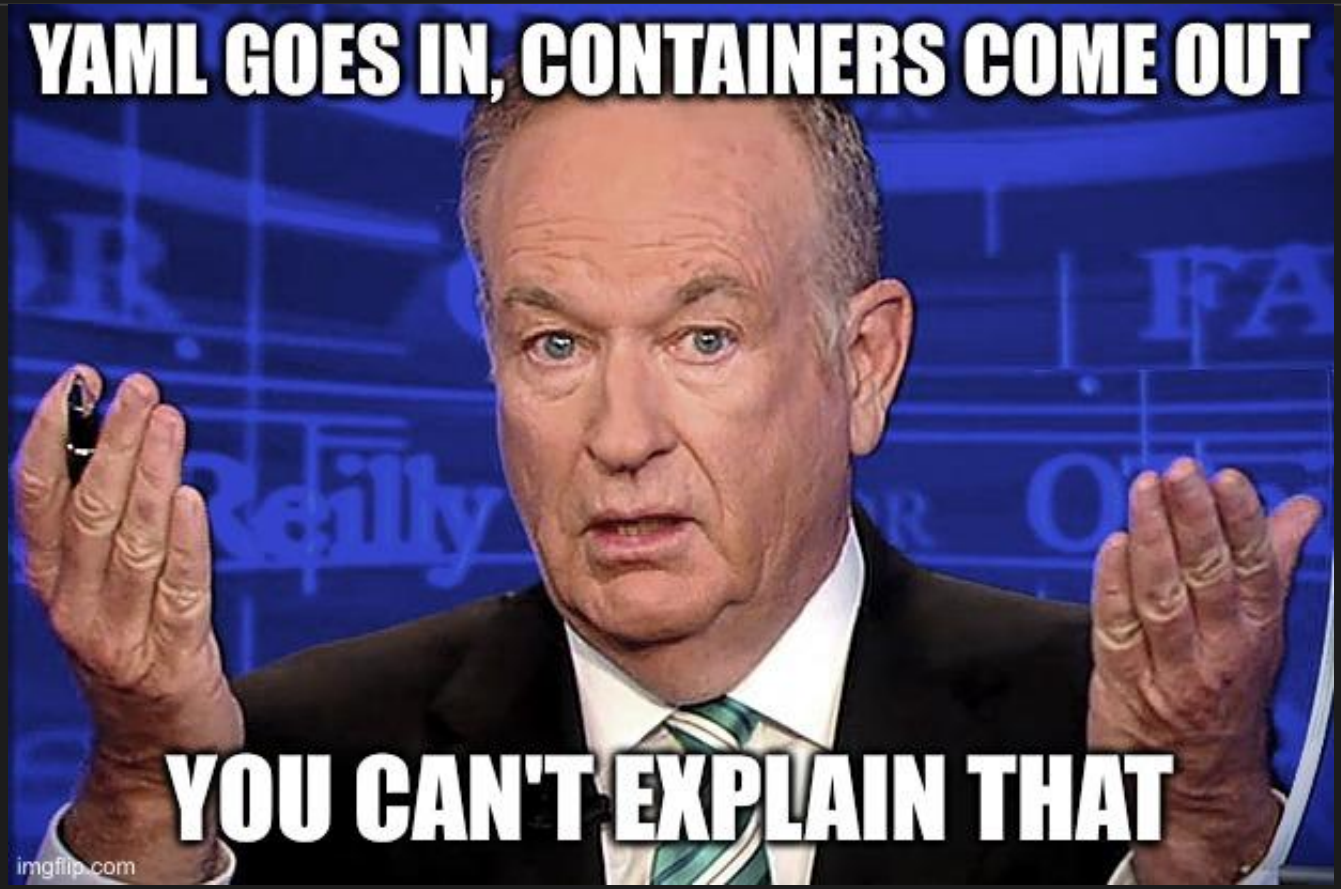
# Just Kidding!

# This works better

Kubernetes

CloudNativePG

# What is Kubernetes?



# Virtually Yours

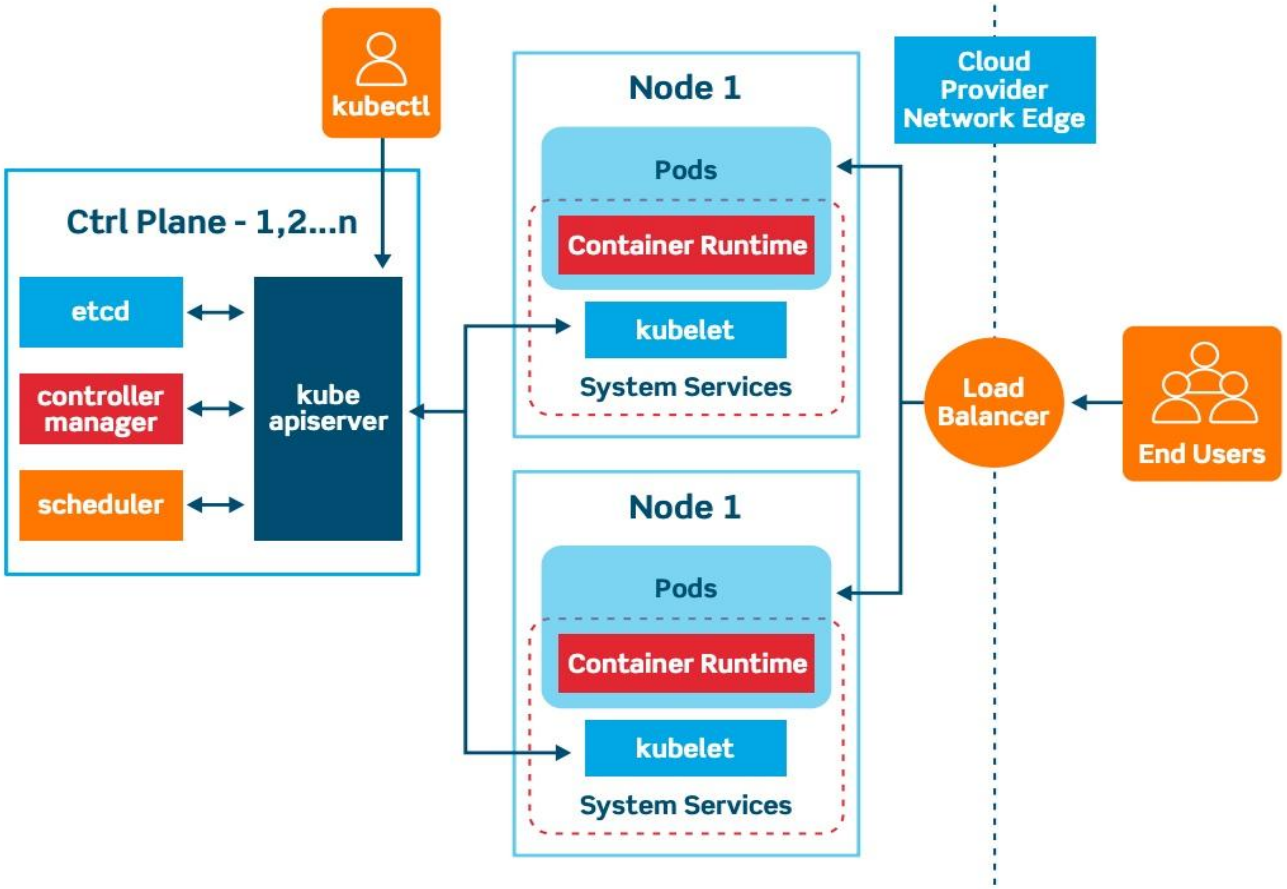
**Infrastructure as Code**

Docker Compose, but more

Intent-based (declarative) deployment

- Describe required resources
- Define limits
- Choose software
- Configure intended result

Kubernetes continuously provisions


kubectl
Ctrl Plane - 1,2...n
etcd
controller manager
scheduler
kube apiserver
Node 1
Pods
Container Runtime
kubelet
System Services
Node 1
Pods
Container Runtime
kubelet
System Services
Cloud Provider Network Edge
Load Balancer
End Users

# Four Important Parts

**Control Plane**
- Maintains state
- Coordinates everything

**Worker Nodes**
- Host workloads
- Provide resources

**Storage**
- Files, objects, etc.
- Various volume types

**Compute**
- CPUs, GPUs, etc.
- Considered transitory

# What Kubernetes Does

Do, or do not:

- Give you what you asked for
- Keep everything operational

Like the force, it’s a medium for action

# HA Without HA

One node cluster!

- Restarts faster than failover
- Quorum is built in
- Connections go here
- ***Storage is all that matters***

# What about Two Nodes?

Kubernetes can't do this by itself

- Which node is the primary?
- How do failovers work?
- What about making replicas?
- And proper backups?
- Where should connections go?

# What is CloudNativePG?

# DBA in a Box

**What do Postgres DBAs do?**

What does Kubernetes need?

CloudNativePG is a Kubernetes Operator

- Need extra nodes? It makes more
- Backups and restores? Done
- Connection pool? Ready to go
- Node routing? Easy
- Monitoring and logging? Obviously
- Fencing? Of course!

# How Easy it Can Be

## A sample cluster definition

```
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: cluster-example
spec:
  instances: 3

  storage:
    size: 1Gi
```

## What it does:

- Creates a 3-node cluster
- Each node has 1GB of storage
- Automatic failover
- Read-write endpoint to primary
- Read-only endpoint to replicas
- Read endpoint to all nodes

# What’s the Point?

# Nodes are ephemeral

# Declarative Focuses on Results

# Only Storage Matters

# Other Operators

# Other Popular Operators

## Zalando Postgres Operator

github.com/zalando/postgres-operator

## Crunchy Postgres Operator

github.com/CrunchyData/postgres-operator

## Percona Postgres Operator

github.com/percona/percona-postgresql-operator

## KubeDB

kubedb.com

## StackGres

stackgres.io

# Thanks!

shaun@tembo.io
@BonesMoses
/in/bonesmoses

**tembo.io**

**Want to experiment?**
**Use the Tembo free trial!**

- Two weeks to test
- $300 USD credit
- Reverts to Hobby tier instance after trial ends

learn more at

tembo.io

**Easily deploy one of our Postgres stacks**

- AI / RAG
- Geospatial
- Analytics
- Timeseries

tembo

Other ways to extend / focus slides

- To get equivalent of an operator, need to configure:
  - Postgres + replication
  - HA system
  - PgBouncer
  - Barman + HAProxy (round robin)
  - If Patroni, also consensus layer (etcd)
- Must be familiar with all systems
  - Better be good at reading docs
  - Many edge cases
  - Build it yourself, even though it’s all 3rd party tools

Other ways to extend / focus slides

- Question Slides
  - Have you ever used a SAN or other external storage?
    - So you agree, ephemeral is best!
  - Etc.
- Series of slides *showing* how difficult it is to build HA cluster
  - But wait, there’s more!
-