PASS24
Data Community Summit

Postgres High Availability Configurations and Best Practices

Nick Ivanov

Solutions Architect

EDB

Nick Ivanov

Solutions Architect

EDB

nick.ivanov@enterprisedb.com

https://www.linkedin.com/in/nick-ivanov-toronto/

Before joining EnterpriseDB in 2022, Nick had been working at IBM Canada for more than 10 years as a database and cloud application architect. He has experience with database design, performance tuning, HA&DR implementation, migration on multiple database platforms, primarily PostgreSQL and Db2.

He's based in Toronto, Canada.

Outline

- What is High Availability (HA)
- Way to achieve HA with PostgreSQL
- HA automation – cluster managers
- Typical deployment scenarios
- Tips

https://github.com/nick-ivanov-edb/presentations

Terminology

- Cluster isn't a cluster isn't a cluster
- Traditionally — group of DBs managed by one Postgres process
- Also, group of servers replicating data, often controlled by a cluster manager
- We'll call the traditional cluster "an instance"

https://github.com/nick-ivanov-edb/presentations

What is High Availability

“It is a feature of a system that reflects its abiility to maintain a level of operational performance in adverse conditions above a certain threshold”

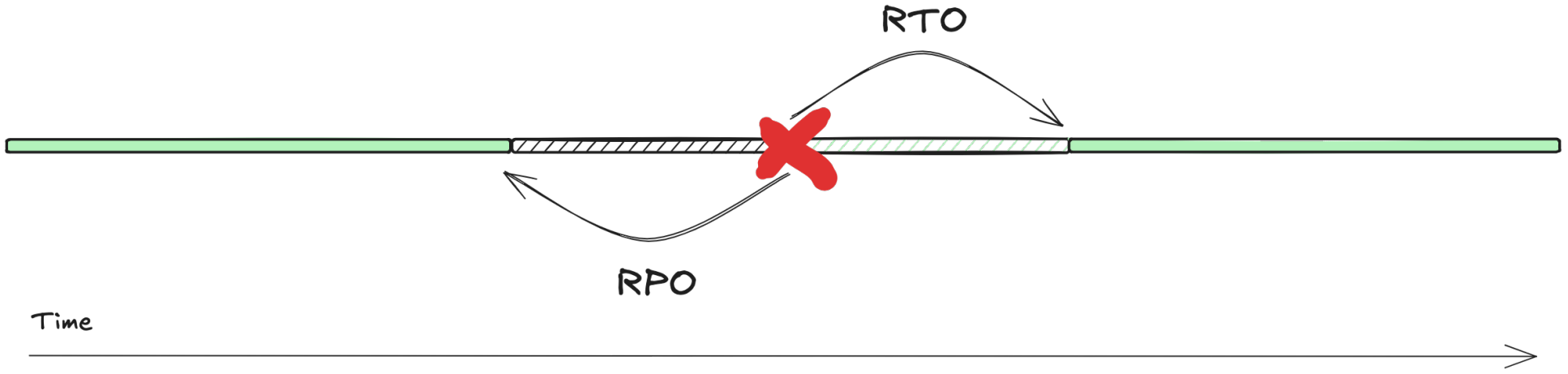
Anonymous

How High Should High Availability Be

- It is context-specific
- Uses some measure, frequently **uptime**
- High **Enough** Availability
- Related concepts:
  - RTO — how much time can I spend recovering
  - RPO — how much data can I afford to lose

https://github.com/nick-ivanov-edb/presentations

RTO and RPO visualised



https://github.com/nick-ivanov-edb/presentations

Database HA — part of the landscape

- Weakest link principle
- Other infrastructure components must play along
- The “nines” are often misleading
- RTO & RPO are probably better measures

https://github.com/nick-ivanov-edb/presentations

How to achieve HA

1. Build a super-reliable machine…
    a. …using redundant components
    b. …vulnerable to the site failure
2. Maintain multiple copies…
    a. …but struggle to keep them in sync
    b. …and pay for the stuff you rarely use
3. Combination of the above

https://github.com/nick-ivanov-edb/presentations

PostgreSQL HA options

- Physical backup
- Physical backup + WAL archive = PITR
- Physical replication
- Logical replication
- Bidirectional replication (“active-active”)

https://github.com/nick-ivanov-edb/presentations

Physical Backup and Restore

- pg_basebackup
  - Consistent snapshot copy of all database files
  - As fast/slow as copying files (locally or remotely)
  - Restore by copying the files back
- RPO is as good as your backup schedule
- RTO depends on data volume
- Can be appropriate for relatively static data

https://github.com/nick-ivanov-edb/presentations

Disk Volume Snapshots

- Provided by underlying storage or filesystem
  - Consistent snapshot copy of all database files
  - Must include WAL volume snapshot
  - Very fast
- RPO is as good as your backup schedule
- RTO usually short
- Can be appropriate for large volumes of relatively static data

Point In Time Recovery

- Improves physical backup
  - More flexible recovery
  - Better RPO
- RTO depends on the size and frequency of backups
- Management of recovery objects becomes more complicated
  - Tools help
    - Barman
    - pgBackRest

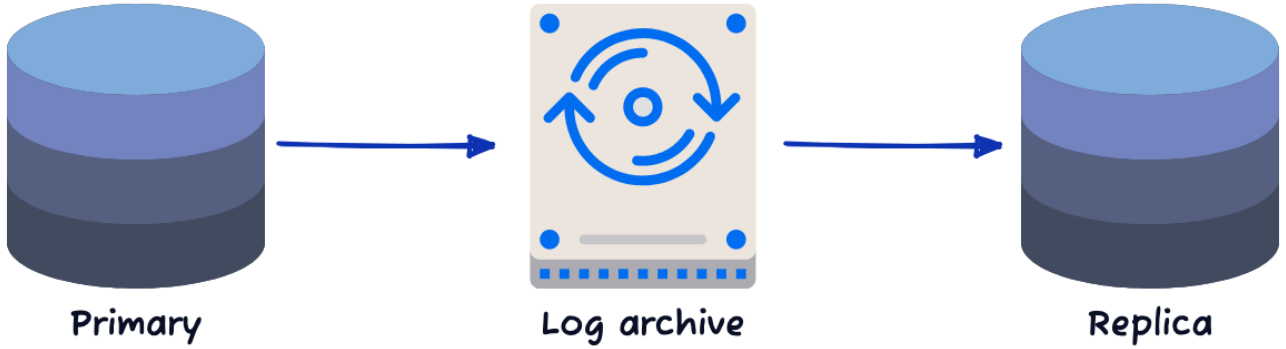
https://github.com/nick-ivanov-edb/presentations

Physical Replication

- "Endless recovery from a physical backup"
  - Log shipping – better RPO
    - Requires shared file store between the primary and replicas
  - Streaming – near zero RPO
    - Requires continuous connection from replicas to the primary
    - Replication slots track each replica's progress

https://github.com/nick-ivanov-edb/presentations

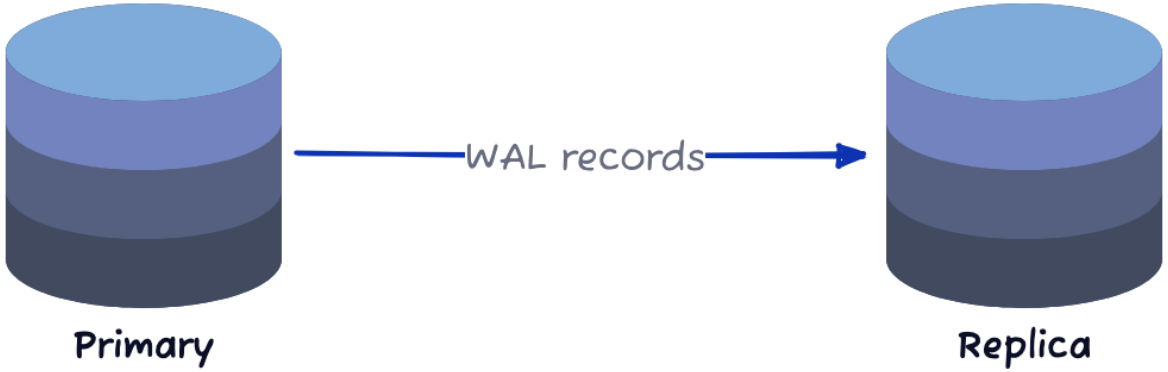
Log Shipping Replication



- WAL segments are archived when switched
- Replica retrieves from archive periodically
- Asynchronous process
- "Hot" standby can serve read-only workloads

https://github.com/nick-ivanov-edb/presentations

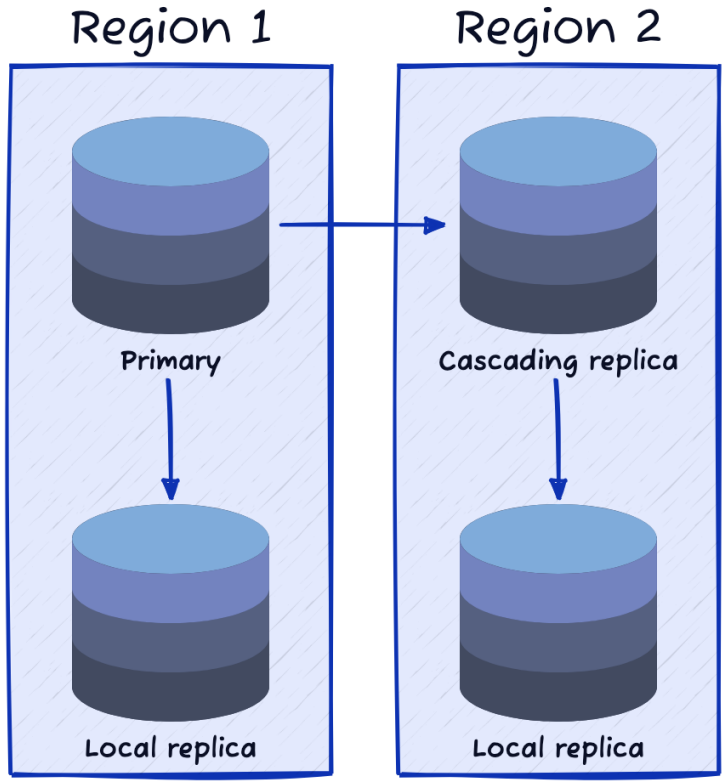
Streaming Replication



- WAL records are sent upon commit
- Asynchronous (default) or synchronous
  - Zero RPO possible
- Privileged connection (special REPLICATION privilege)

https://github.com/nick-ivanov-edb/presentations

Cascading Replication



- Reduced load on primary
- Reduced cross-region traffic
- Cascading replica becomes the new primary upon primary region failure

https://github.com/nick-ivanov-edb/presentations

Logical Replication

- Instead of WAL records — logical commands
- More flexible
  - Replicate a subset of tables, rows, or columns
- Can’t replicate DDL
- Not often used for HA *by itself*

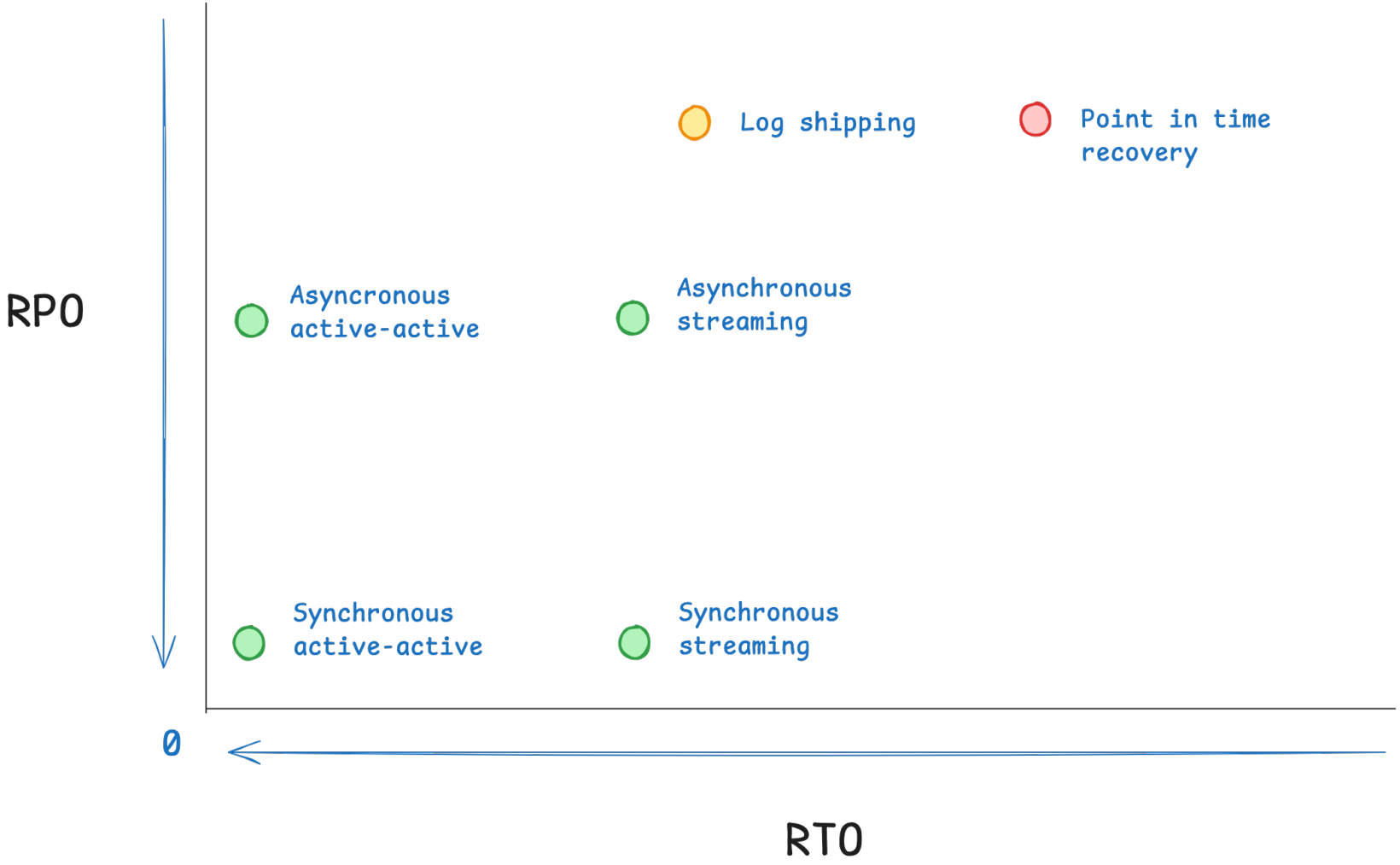
https://github.com/nick-ivanov-edb/presentations

Bidirectional replication solutions

- Using built-in logical replication
  - Possible but not practical
    - No DDL replication
    - No conflict resolution or even detection
    - Failovers difficult
- Extensions
  - pglogical2 (not maintained)
  - EDB Postgres Distributed (PGD)
  - PgEdge

https://github.com/nick-ivanov-edb/presentations

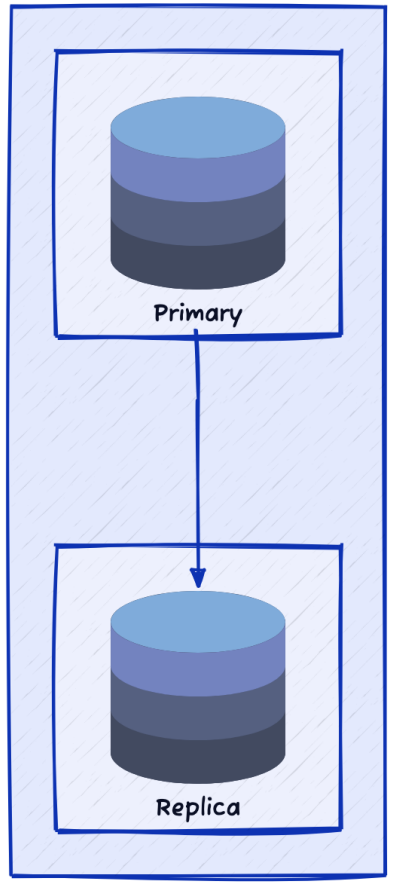
HA Options Compared



https://github.com/nick-ivanov-edb/presentations

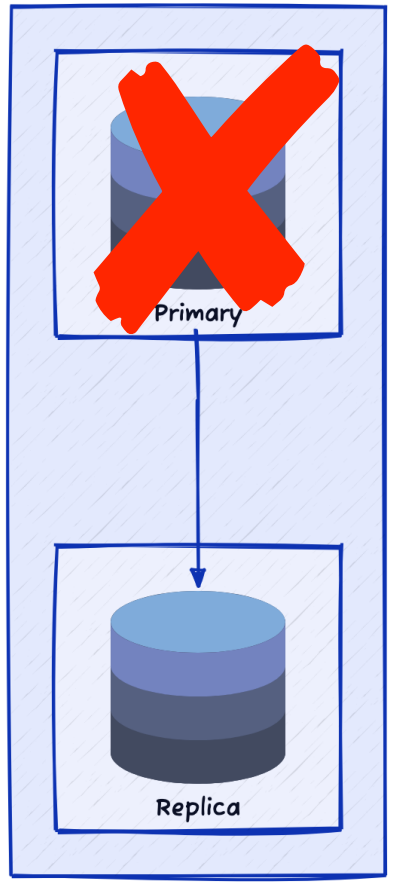
Deployment Examples

What's Missing?



https://github.com/nick-ivanov-edb/presentations

What's Missing?



https://github.com/nick-ivanov-edb/presentations

Failover Tasks

- Detect failure
- Promote best primary candidate
- Fence old primary
- Ensure client failover
- Point other replicas to new primary
- Admit old primary as replica

https://github.com/nick-ivanov-edb/presentations

Client Failover

- Driver failover
  - Configured with multiple target hostnames, round-robin
- Reconfigure proxy or pooler
- Virtual IP
- Third-party load balancer

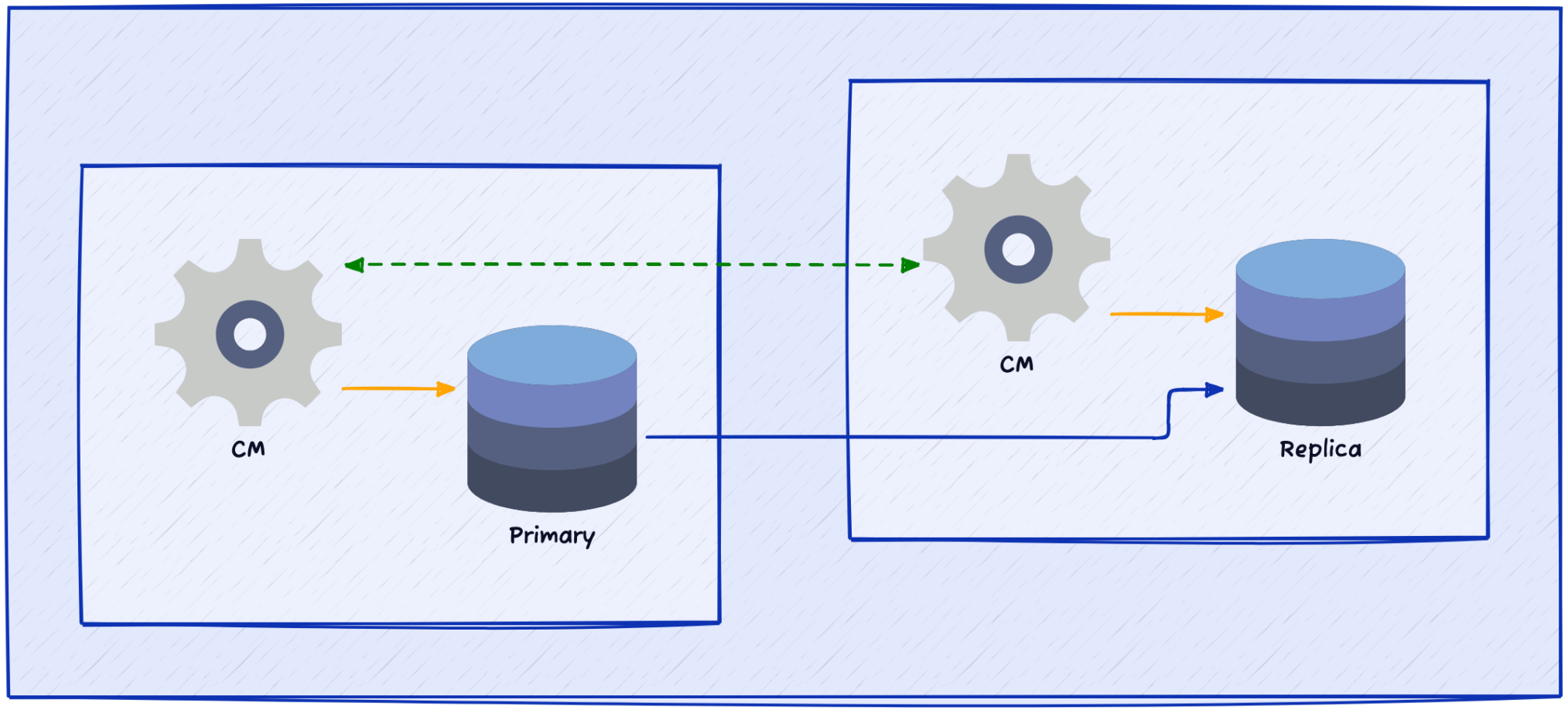
https://github.com/nick-ivanov-edb/presentations

Cluster Managers (Failover Managers)

- Replica cannot promote itself
- You are your cluster manager
- Automation is better

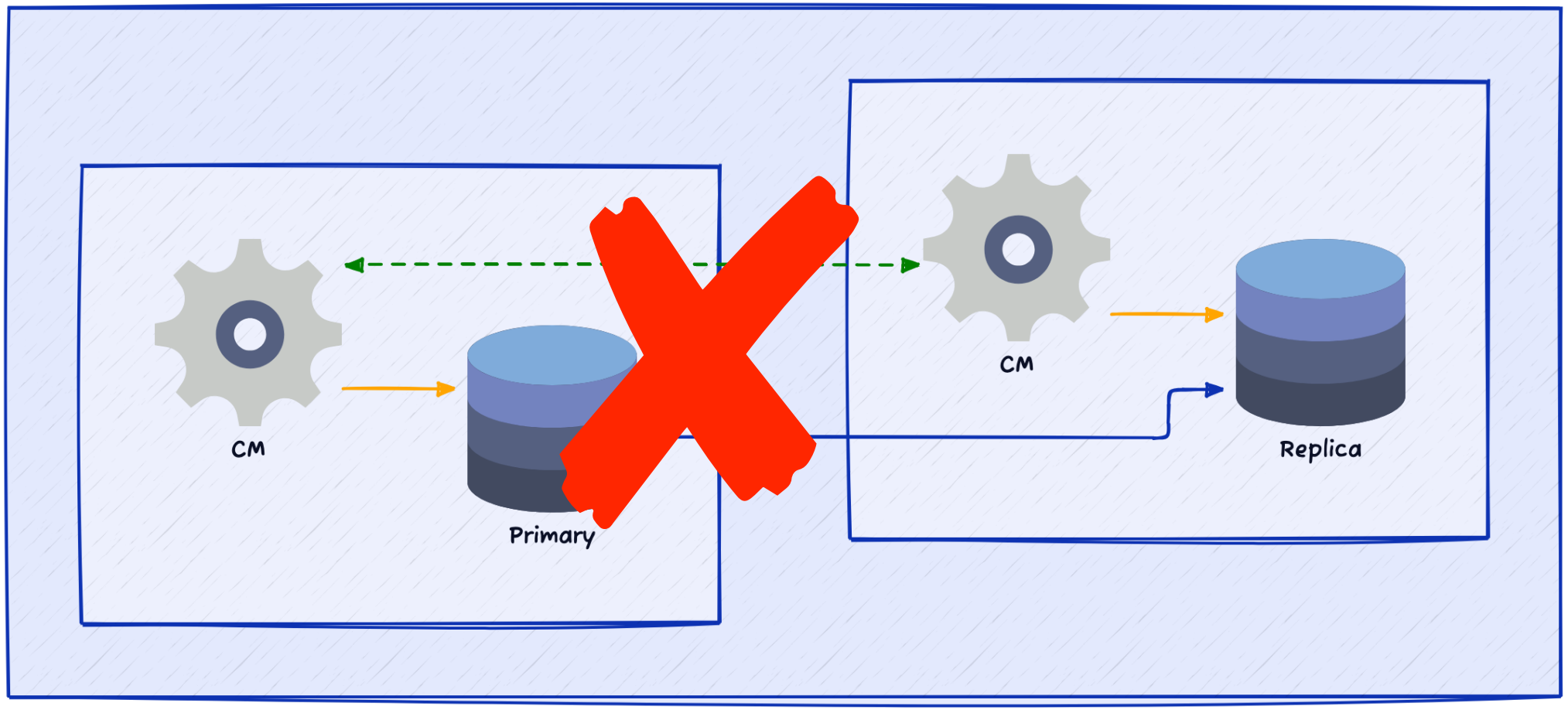
https://github.com/nick-ivanov-edb/presentations

Is Something Still Missing?



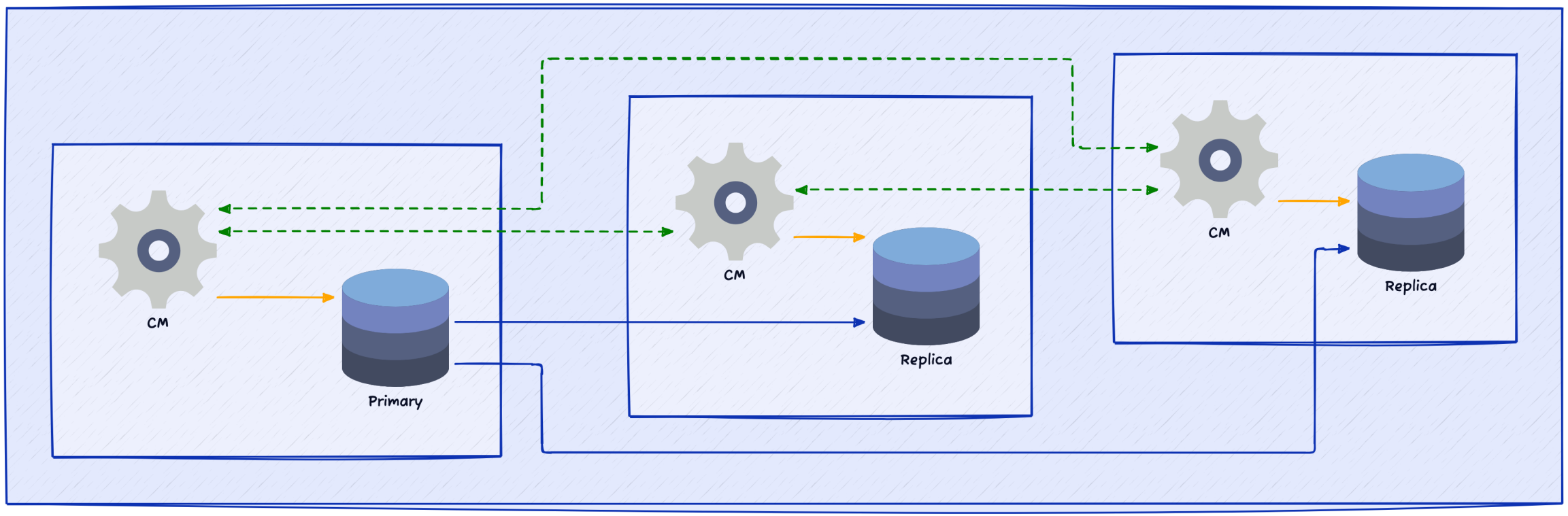
https://github.com/nick-ivanov-edb/presentations

Is Something Still Missing?



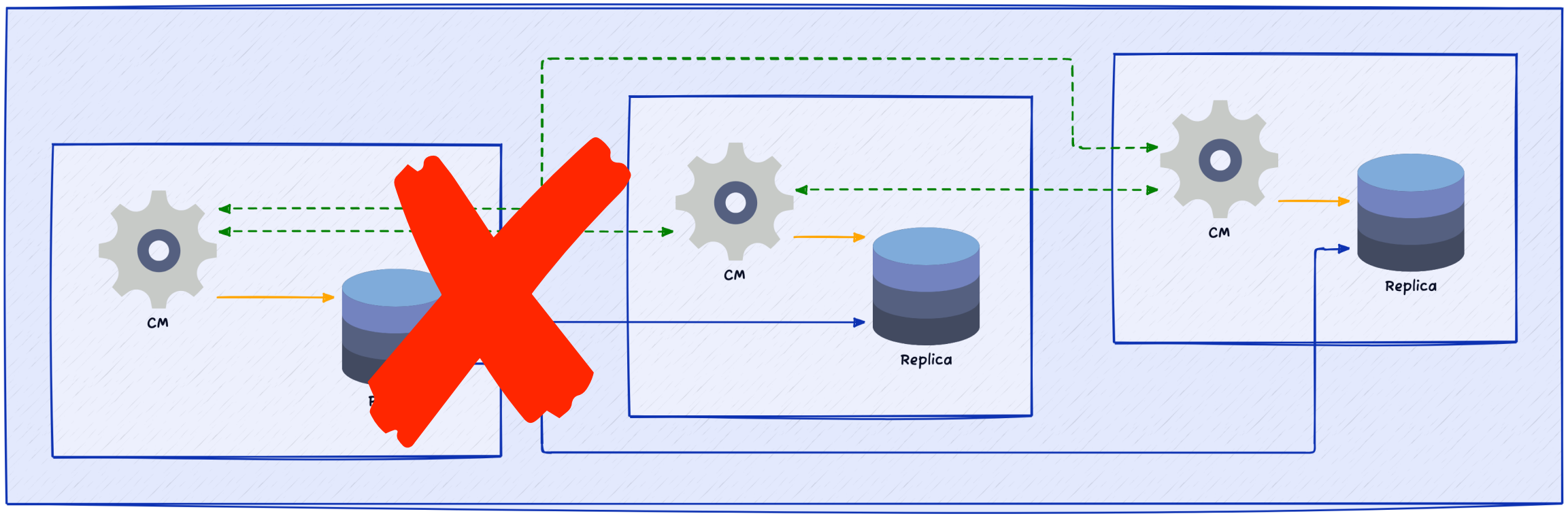
https://github.com/nick-ivanov-edb/presentations

Majority Rules



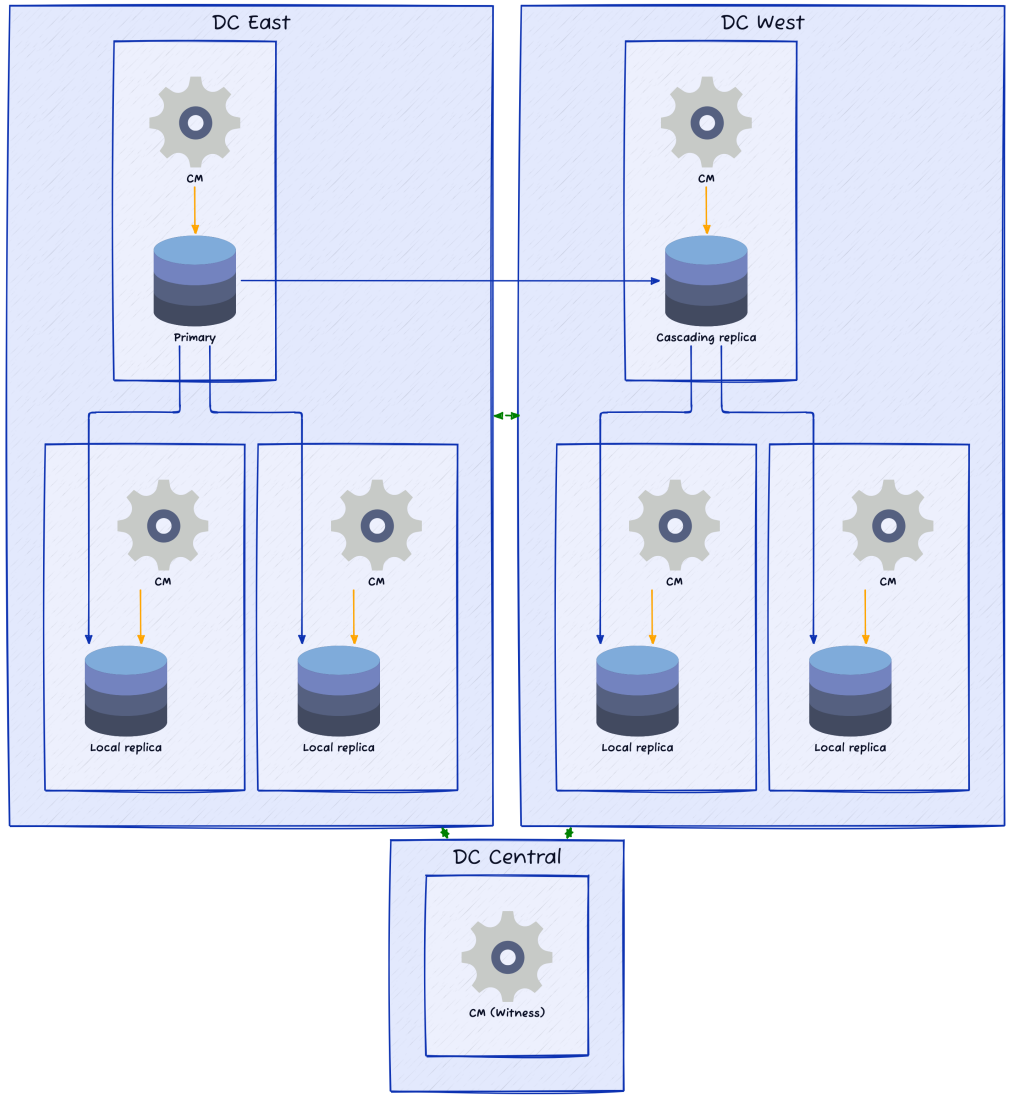
https://github.com/nick-ivanov-edb/presentations

Majority Rules



https://github.com/nick-ivanov-edb/presentations

Multiple Locations



CMs for Physical Replication Compared

	Patroni	repmgr	PAF (pg_auto_failover)	EFM (EDB Failover Manager)
Configuration store	External*	Internal	Internal	Internal
External monitor required	No	No	Yes	No
Cascading replication support	Yes	Yes	No	No
Auto DR	No	Yes	No	No
Auto recreate replica	Yes	No	Yes	No
Manage Postgres instance	Yes	No	Yes	Yes

https://github.com/nick-ivanov-edb/presentations

CMs for Physical Replication Compared

	Patroni	repmgr	PAF (pg_auto_failover)	EFM (EDB Failover Manager)
Proxy/pool integration	Callback	Callback	No	Yes
Virtual IP support	Callback	Callback	No	Yes
Interface	REST API, CLI	CLI, SQL	CLI, SQL	CLI
Language	Python	C	C	Java
Open source	Yes	Yes	Yes	No

https://github.com/nick-ivanov-edb/presentations

CMs for Logical Replication Compared

	EDB Postgres Distributed	pgEdge
Implementation type	PostgreSQL extension	PostgreSQL extension
Connection routing proxy	Yes	No
DDL replication	Yes	Yes
Commitment/consistency control	Async, sync, group commit, CAMO	Async, sync
Sequence support	galloc, snowflake	snowflake
Open source	No	Yes

https://github.com/nick-ivanov-edb/presentations

Things to Note

Replication Security

- SSL configuration of the Postgres instance
- Consider certificate authentication
- Use role with the REPLICATION privilege

https://github.com/nick-ivanov-edb/presentations

Physical Replication Performance

- Use same/similar hardware as the primary
- Ensure enough **max_worker_processes**, **max_wal_senders**, and **max_replication_slots**
- **wal_compression** is often beneficial

https://github.com/nick-ivanov-edb/presentations

Replication Stability

- Stopping a replica can break replication
- When *not* using replication slots: **wal_keep_size**
- When using slots: **max_slot_wal_keep_size**

Data Consistency with Synchronous Replication

- Different synchronous commit settings via
  - `synchronous_commit`
  - `synchronous_standby_names`
- Quorum vs Priority
  - `ANY n`
  - `FIRST n`
- Standby delay with hot standbys
  - `max_standby_*_delay`

https://github.com/nick-ivanov-edb/presentations

Monitoring Replication

- `pg_current_wal_lsn()` — current position on primary
- `pg_stat_replication.sent_lsn` — if it lags behind the current LSN, primary is struggling
  - WAL sender lag – possibly overloaded primary

```
select
 application_name,
 pg_size_pretty(pg_current_wal_lsn()-sent_lsn)
from pg_stat_replication
```


https://github.com/nick-ivanov-edb/presentations

Monitoring Replication

- Monitor replication slot status
  - Inactive slots will accumulate WAL files
- Monitor WAL disk space
  - Don’t run out of space
- Replication lag statistics are *approximate*

https://github.com/nick-ivanov-edb/presentations

Q&A

PASS24
Data Community Summit

Your feedback is important to us

Evaluate this session at:

www.PASSDataCommunitySummit.com/evaluation

Thank you

Nick Ivanov

nick.ivanov@enterprisedb.com

https://github.com/nick-ivanov-edb/presentations

https://www.linkedin.com/in/nick-ivanov-toronto/