Patroni 3.0: What's New and Future Plans

PGConf.DE 2023, Essen

Alexander Kukushkin

2023-06-27

About me

Alexander Kukushkin

Principal Software Engineer @Microsoft

The Patroni guy

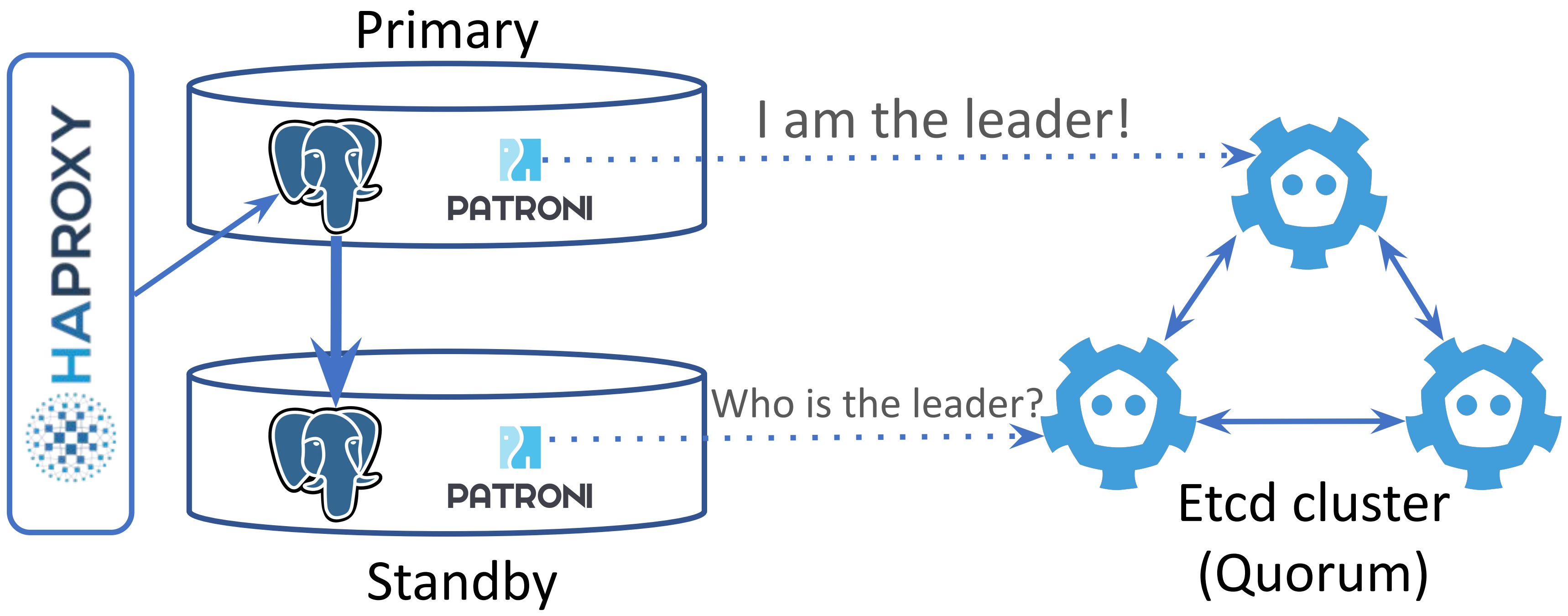
akukushkin@microsoft.com

Twitter: @cyberdemn

Agenda

- Brief introduction to automatic failover and Patroni
- New features
- Bug fixes
- Future plans
- Live Demo

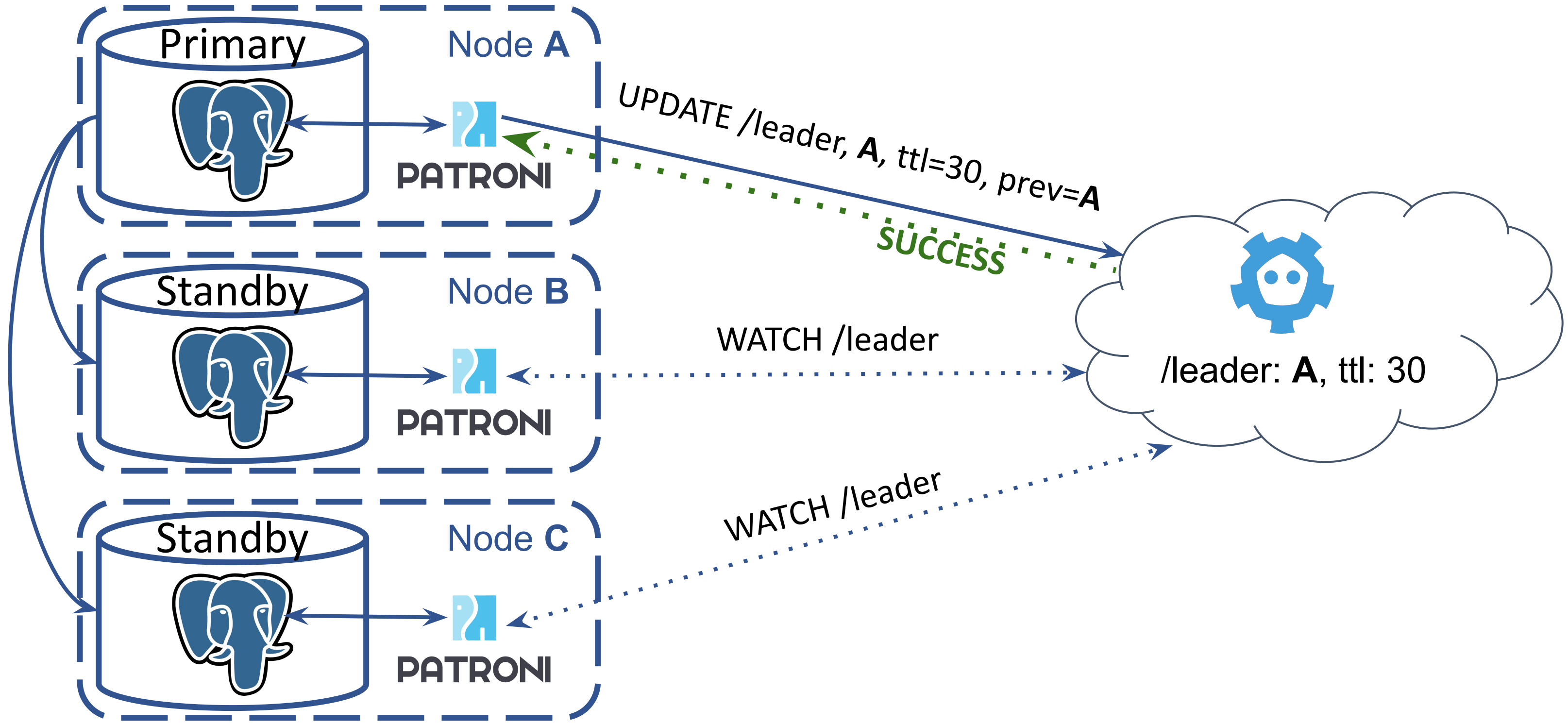
High availability with Patroni



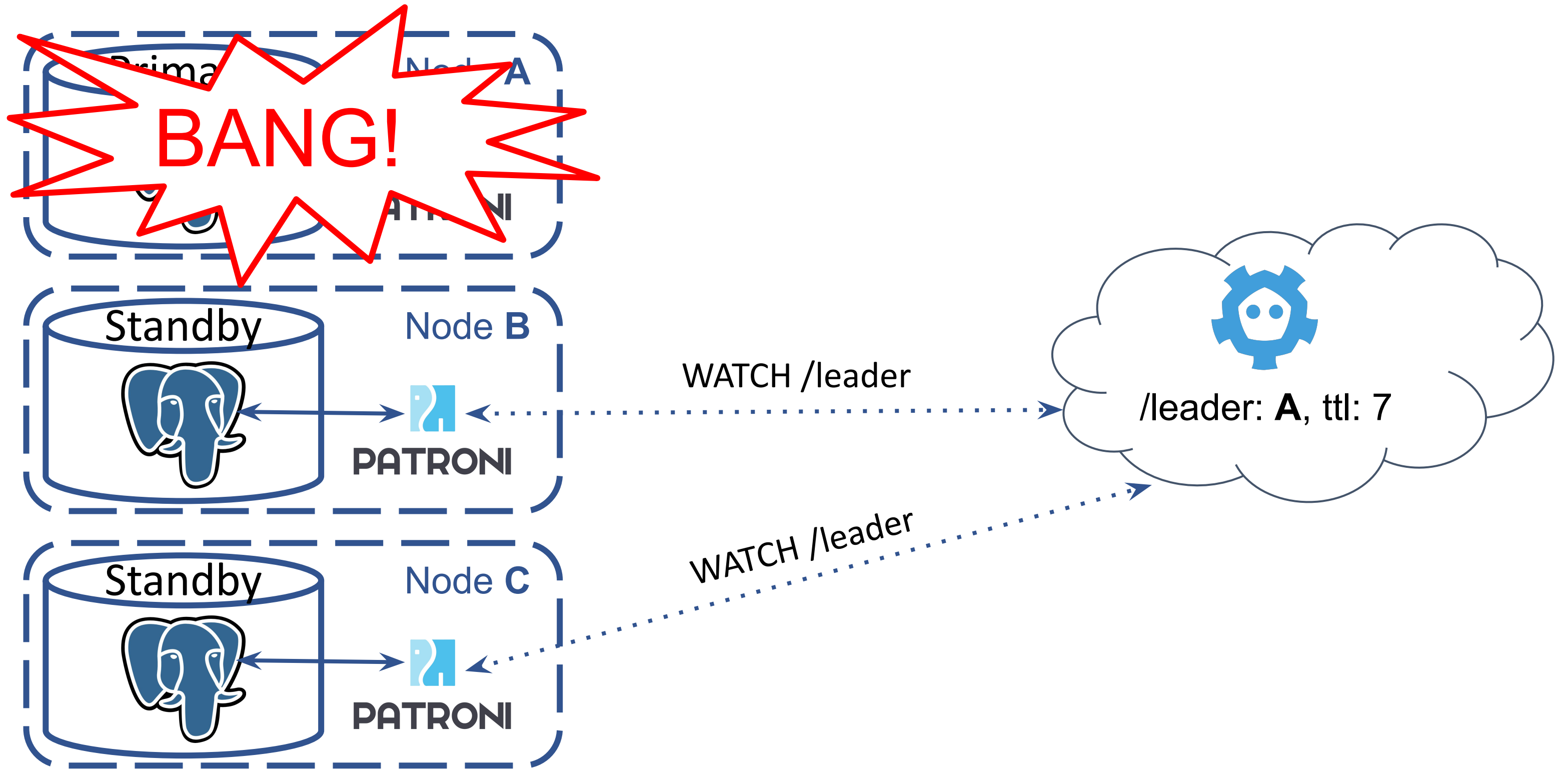
Distributed Configuration (Key-Value) Store

- Consul, Etcd (v2/v3), Zookeeper, Kubernetes API
- **Service Discovery**
  - Every Postgres node maintains a **key with its state**
  - **Leader key** points to the primary
- **Lease/Session/TTL** to expire data (i.e. leader key)
- **Atomic CAS** operations
- **Watches** for important keys (i.e. leader key)

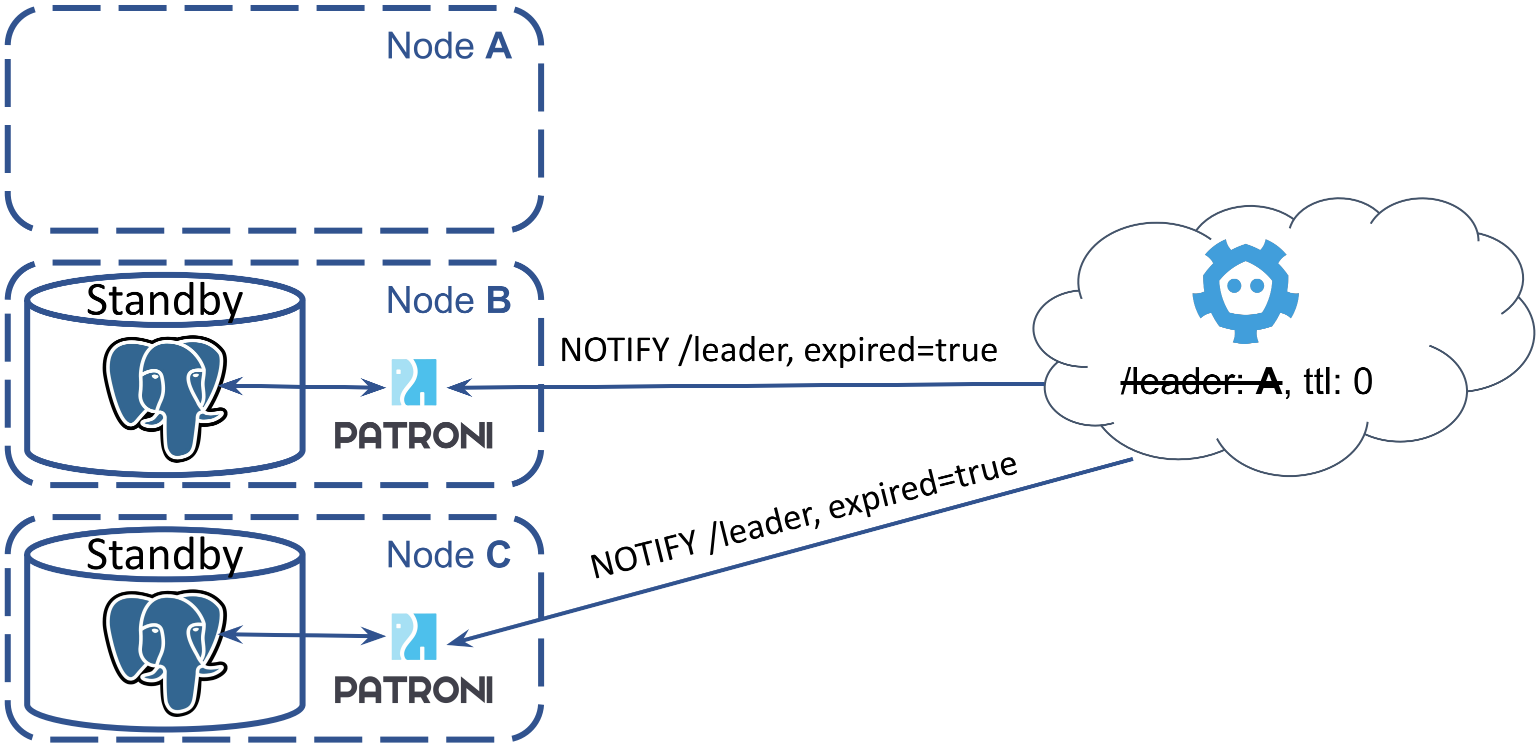
Patroni: Normal operation



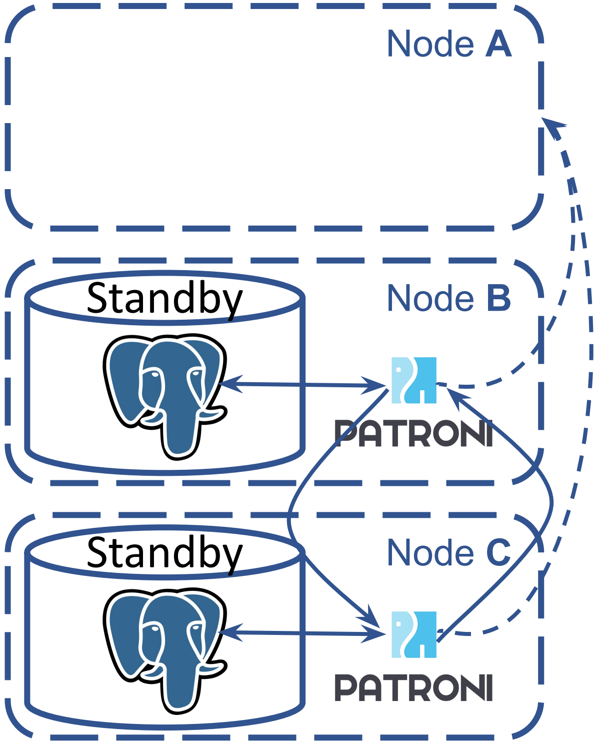
Patroni: primary dies, leader key holds



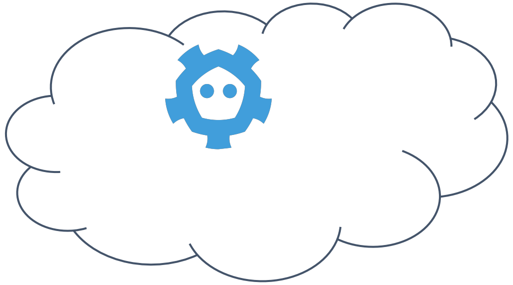
Patroni: leader key expires



Patroni: leader race

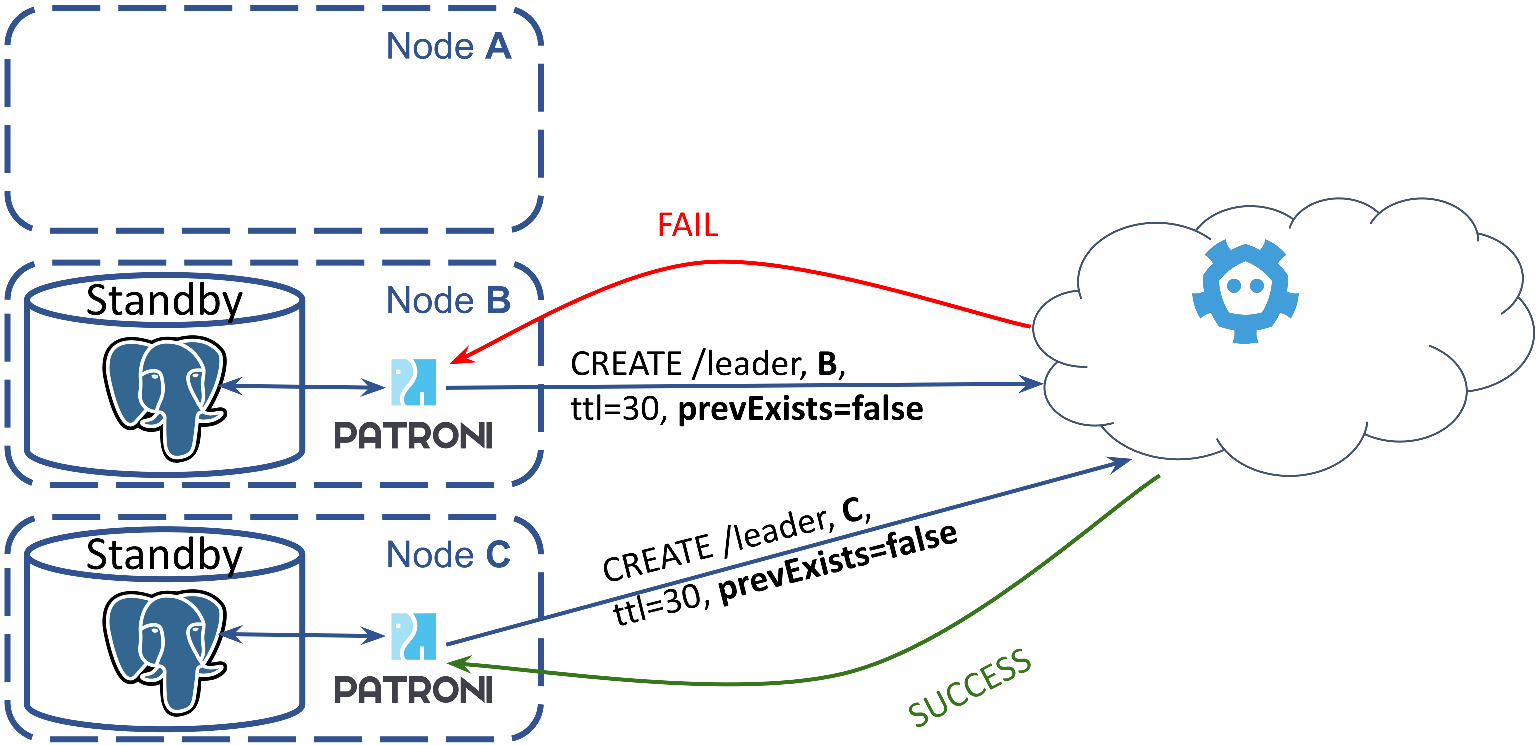


Node **B**:
GET http://A:8008/patroni -> **failed/timeout**
GET http://C:8008/patroni -> wal_lsn: **100**

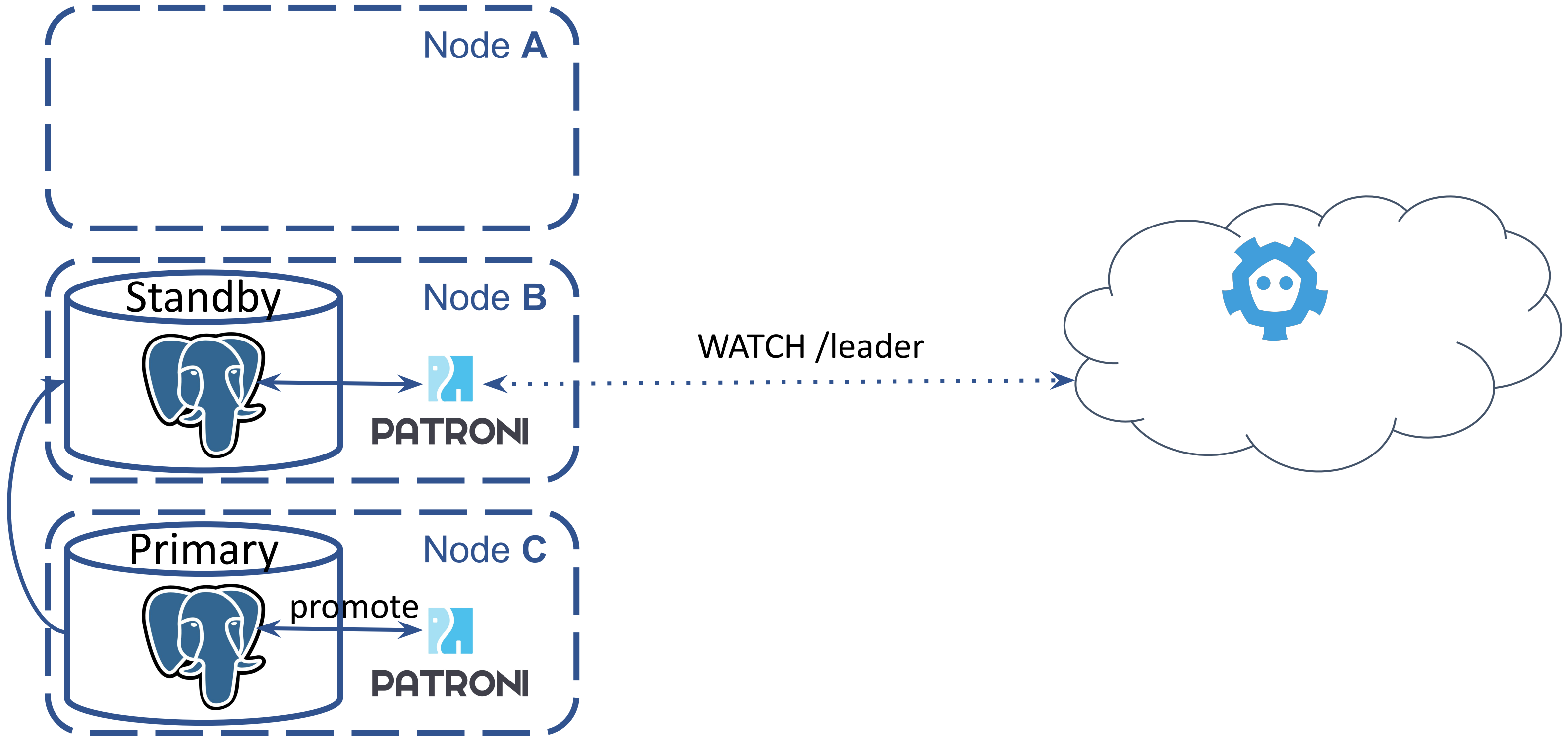


Node **C**:
GET http://A:8008/patroni -> **failed/timeout**
GET http://B:8008/patroni -> wal_lsn: **100**

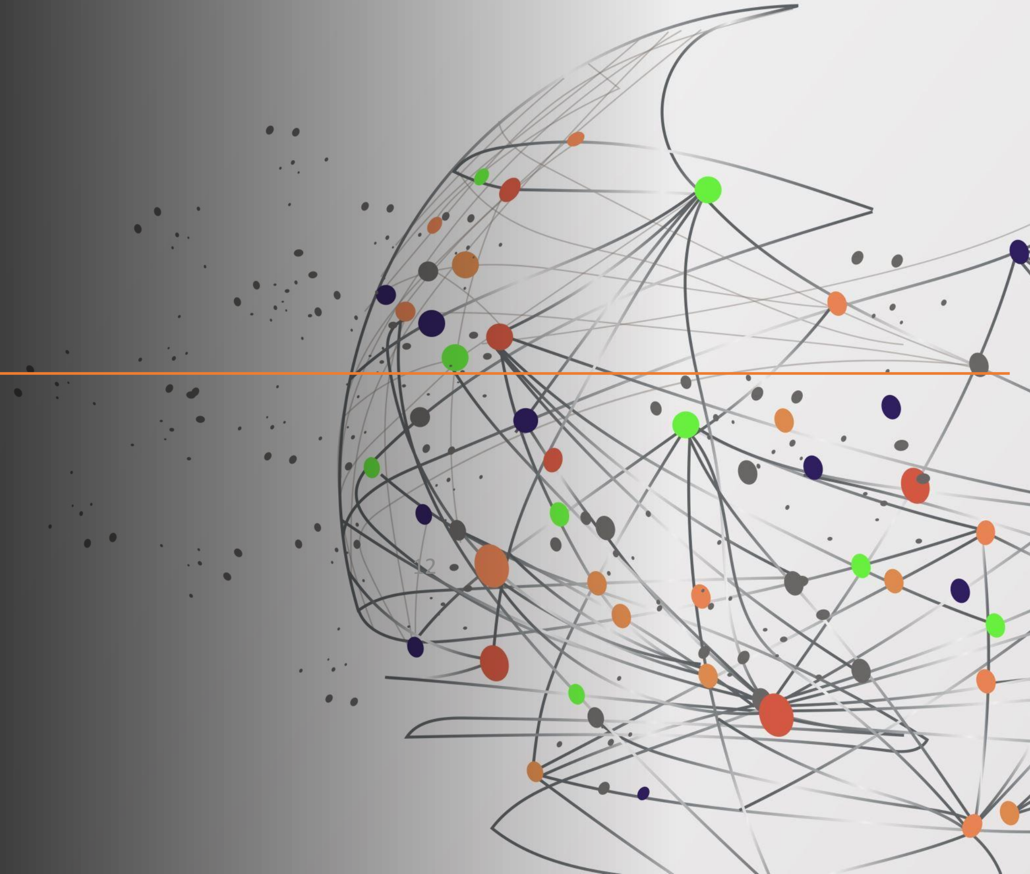
Patroni: leader race


Node A
Node B
Standby
PATRONI
Node C
Standby
PATRONI
FAIL
CREATE /leader, **B**, ttl=30, **prevExists=false**
CREATE /leader, **C**, ttl=30, **prevExists=false**
SUCCESS

Patroni: promote and continue replication



New Features



DCS Failsafe Mode

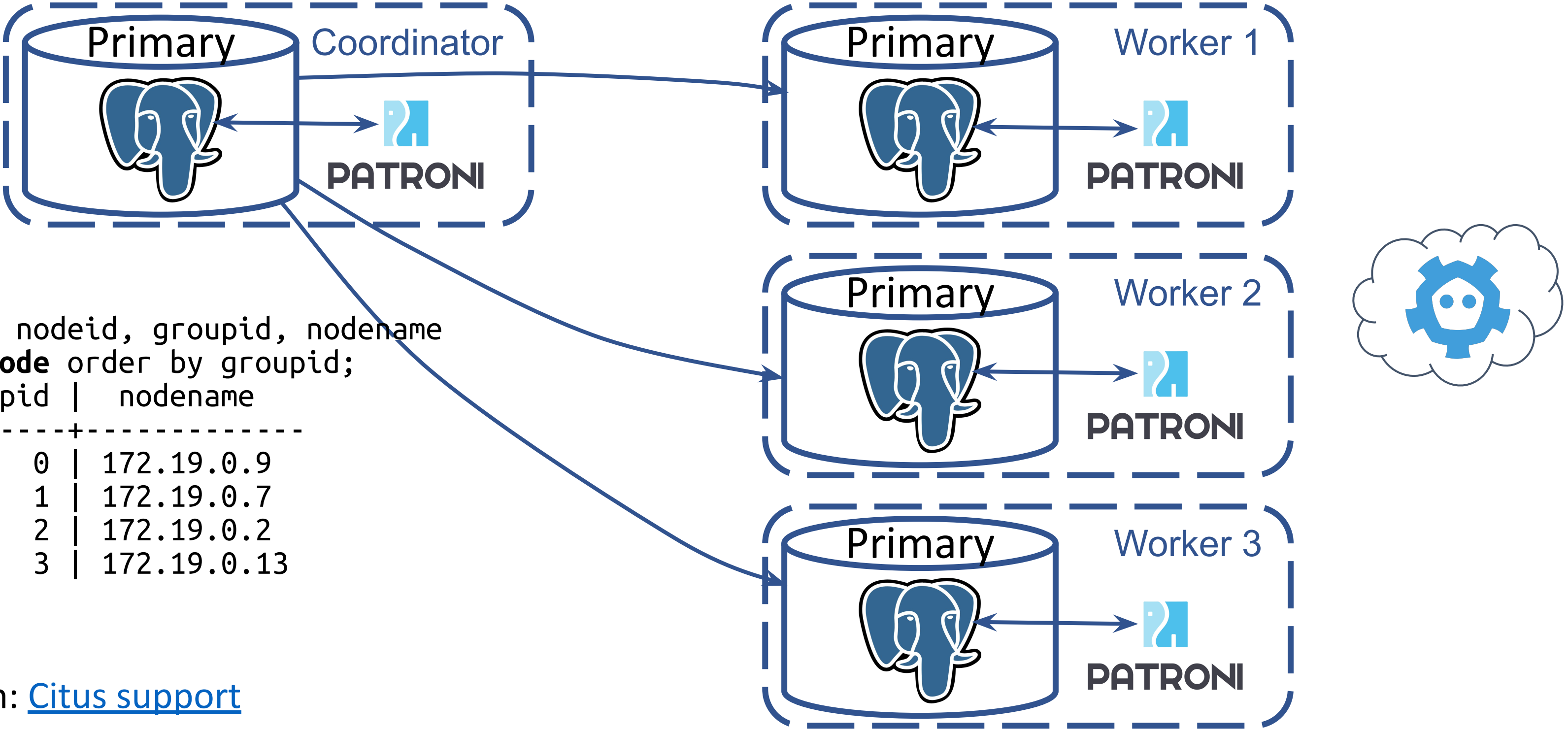
- **Case**: Postgres is running as primary only when Patroni can maintain leader lock in DCS
- **Before**: primary **is demoted** when lock can’t be updated
- **Now**: Patroni will keep primary **if all members** of the cluster **agree** with it

```
$ patronictl edit-config
---
+++
@@ -4,3 +4,4 @@
  use_pg_rewind: true
retry_timeout: 10
ttl: 30
+failsafe_mode: on

Apply these changes? [y/N]: y
Configuration changed
```

Documentation: DCS Failsafe Mode

Citus integration



```
citus=# SELECT nodeid, groupid, nodename
FROM pg_dist_node order by groupid;
 nodeid | groupid |  nodename
--------+---------+-------------
      1 |       0 | 172.19.0.9
      3 |       1 | 172.19.0.7
      2 |       2 | 172.19.0.2
      4 |       3 | 172.19.0.13
(4 rows)
```

Documentation: Citus support

Logical Failover Slots

- **Case**: logical replication slots are lost after failover.
- **Before**: don’t allow connections before logical slots are recreated
- **Now**: copy slots from the primary and use *pg_replication_slot_advance()* to keep logical slot ready.

```
$ patronictl edit-config
---
+++
@@ -1,6 +1,12 @@
 loop_wait: 10
 retry_timeout: 10
 ttl: 30
+permanent_slots:
+  my_slot:
+    database: testdb
+    plugin: test_decoding

Apply these changes?
Configuration changed
```

synchronous_mode improvements

- Support multiple synchronous standbys (**synchronous_node_count**) **@Krishna Sarabu**
  - Pick standby nodes based on replication lag (**maximum_lag_on_syncnode**)
  - Prefer nodes without **nofailover** tag
- Wait for standby to become really synchronous before exposing its name to DCS.

REST API improvements: security

- Limit available ciphers: **restapi.ciphers @Gunnar "Nick" Bluth**
- Encrypted TLS keys: **restapi.keyfile_password @Jonathan Katz**
  - See also **ctl.keyfile_password**
- Restrict incoming IPs: **restapi.allowlist** and **restapi.allowlist_include_members**

REST API improvements: endpoints

- **GET /metrics** – in Prometheus format **@Mark Mercado**, **@Michael Banck**
- **GET /readiness** and **GET /liveness** – useful on K8s
- Load-balancing based on user-defined tags: **@Arman Jafari Tehrani**
  - **GET /replica?lag=10MB&tag_key1=val1**
  - **GET /read-only?tag_key1=val1&tag_key2=val2**

Documentation: REST API

pg_rewind improvements

- Postgres v13+ supports **pg_rewind --restore-target-wal**
    - But, opt out **--restore-target-wal** on v13 and v14 if postgresql.conf if outside of $PGDATA (Debian/Ubuntu) **@Gunnar "Nick" Bluth**
- For older versions Patroni tries to fetch missing WALs when pg_rewind fails

pg_rewind improvements

- **Archive WAL’s** before calling pg_rewind on the old primary
    - pg_rewind might remove WAL’s even if they are needed for Postgres to start
- Fully support pg_rewind in a **standby cluster**
    - Make it possible to specify **multiple hosts** in the standby cluster configuration **@Michael Banck**

Configuration

- **Configuration directories @Floris van Nee**
    - YAML files (Patroni config) in a directory are loaded and applied in alphabetical order
- **Advanced validation** of PostgreSQL parameters
    - Discard unknown parameters or if the value isn’t correct.

General improvements

- **Removed** support of Python < 3.6
  - Introduced type hints!
  - Psycopg 3!
- **pre_promote** - run a script *<u>before</u>* **pg_ctl promote**
  - Abort if the exit code != 0
- **before_stop** - run a script *<u>before</u>* **pg_ctl stop @Le Duane**
  - pgbouncer PAUSE, terminate Debezium connections

Bug Fixes

TCP keepalives

- **Case**: Etcd v3 and K8s API are using long-polling connections for WATCH requests
  - response with infinite stream of chunked data
- **Before**: TCP connection could stay around even when the other side is gone
  - Stale data :(
- **Now**: bad sockets are detected/closed within TTL seconds

Sloooow execution and freezes of heart-beats

- **Case**: check presence of $PGDATA on every heart-beat
- **Before**: *os.listdir()*
  - Could be very sloooow when system is stressed
    - We have seen it taking more than TTL seconds
- **Now**: *first* check presence of **$PGDATA/global/pg_control** file

Sometimes broken switchover with Debezium

- **Case**: Postgres on stop waits until all WALs are streamed
    - Debezium doesn’t properly handle *keepalive* messages
- **Before**: Patroni keeps updating the leader key while Postgres is being stopped (indefinitely)
- **Now**: the leader key is removed when **pg_controldata** starts reporting “*shut down*” and there are nodes ready to fail over

What is coming Next?

Quorum based failover (aka Quorum Commit)

- **PostgreSQL v10+**: *synchronous_standby_names="ANY k (*)"*
  - Examples:
    1. "ANY 2 (node1,node2)",
    2. "ANY 2 (node1,node2,node3)"
- **Challenge**: figure out during failover whether the node was synchronous
  - Was the **node2** synchronous in the example **2**?

Quorum based failover: math

- synchronous_standby_names="ANY **2** (m2,m3,m4)"
- /sync: {leader: m1, sync: [m2,m3,m4], quorum: **1**}

- synchronous_standby_names="ANY **1** (m2,m3,m4)"
- /sync: {leader: m1, sync: [m2,m3,m4], quorum: **2**}

Quorum based failover: challenges

- How to change *synchronous_standby_names* and */sync* that we can always identify sync node?
- Example:
  - synchronous_standby_names="ANY **1**(m2,m3)"
  - /sync: {leader: m1, sync: [m2,m3], quorum: **1**}
  - Node m4 joins the cluster:
    1. change /sync to {leader: m1, sync: [m2,m3,**m4**], quorum: **2**}
    2. change synchronous_standby_names="ANY **1**(m2,m3,**m4**)"

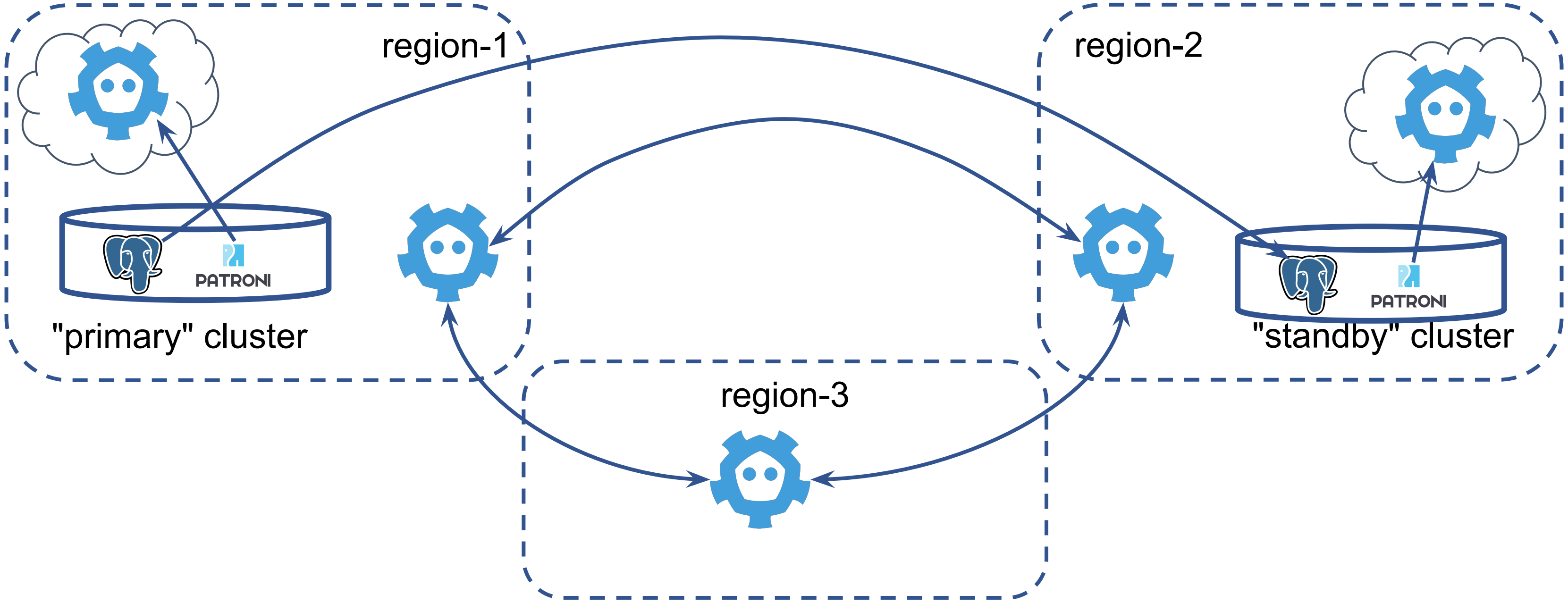
Integrate Patroni with pg_failover_slots

- https://github.com/EnterpriseDB/pg_failover_slots
- But Patroni already solved logical failover slots problem! Why?
  - Extension has mechanisms to wait for physical standbys before sending data to logical subscriber
    - **pg_failover_slots.standby_slot_names, pg_failover_slots.standby_slots_min_confirmed**
  - Works similar to synchronous_standby_names="ANY k (s1, s2, s3)"

Improve Citus support

- Manage **pg_dist_poolinfo,** to allow optional cross-node communication via **pgbouncer**
- Register replica nodes in **pg_dist_node**
  - for read scaling (easy)
  - to use them as failover targets (hard)

Multi-site Automatic Failover



Get rid of non-inclusive terminology

- role: ~~master~~ -> **primary**
  - Most of preparations are done in 3.0
    - If running something older, better to upgrade to 3.0.x first
- Kubernetes pod labels is a challenge
  - Migration will require temporary labels and 3 rolling upgrades

Live Demo!

Questions?