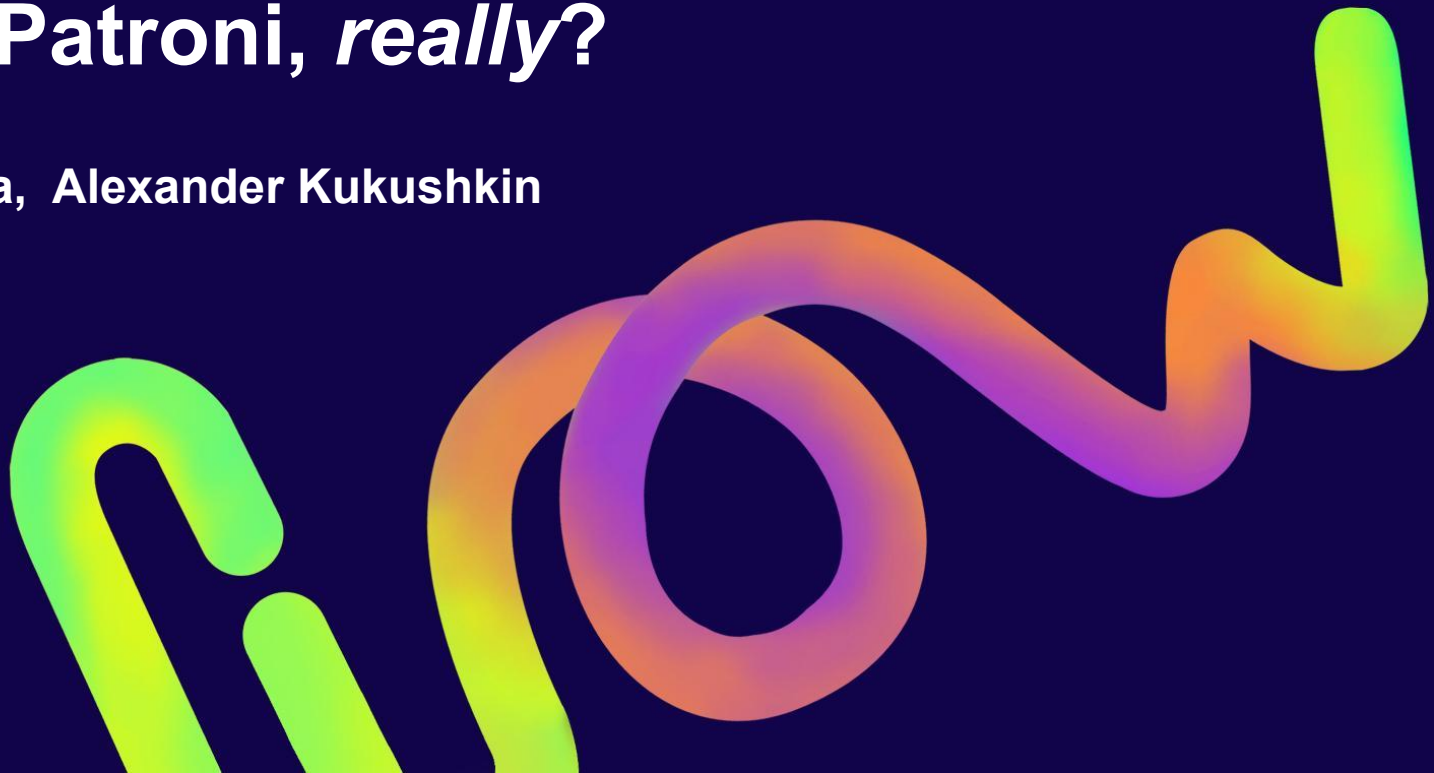


What is Patroni, *really*?

Polina Bungina, Alexander Kukushkin



About us

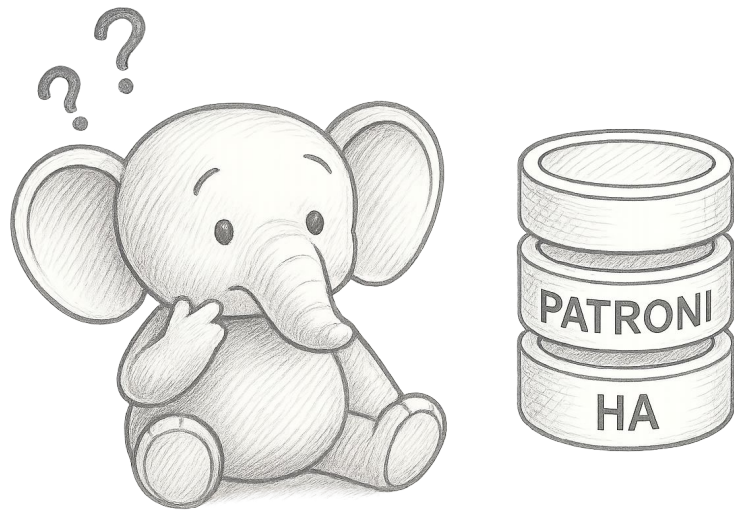
Alexander Kukushkin

- Principal Software Engineer @Microsoft
- The Patroni guy
- akukushkin@microsoft.com

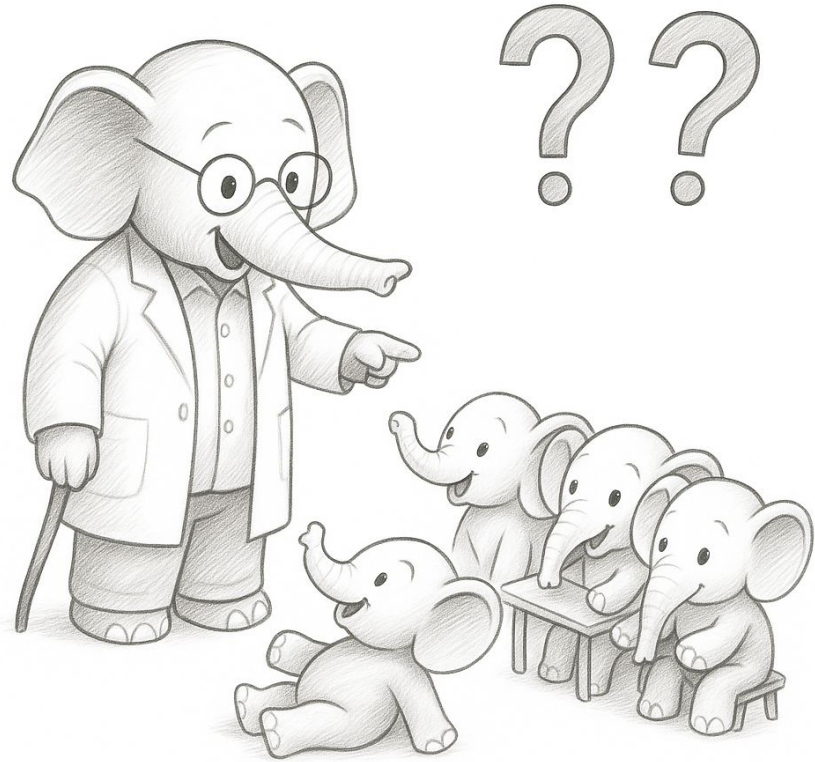
Polina Bungina

- Software Engineer @Zalando
- polina.bungina@zalando.de

- **What is it, really?**
- **Automatic failover done wrong**
- **Patroni overview**
 - **how it works?**
 - **notable features**

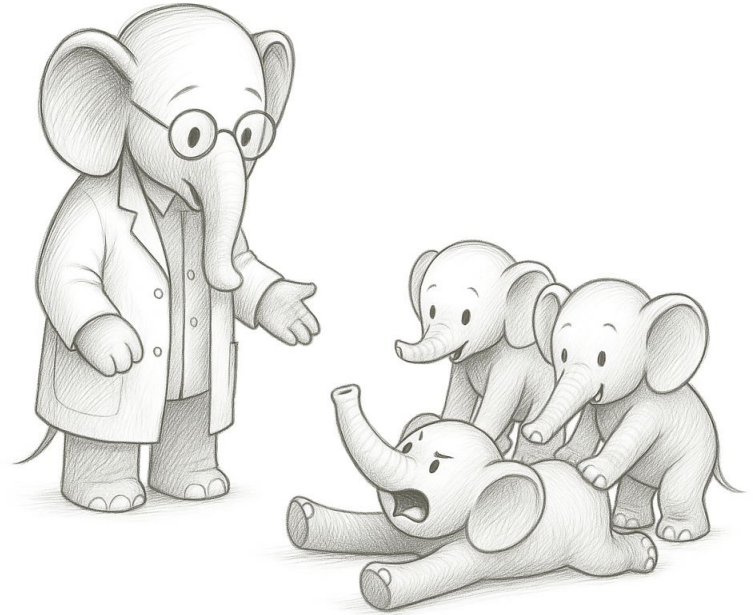


What is it, really?



- Originated from Governor project by Compose, in 2015
- Main functions:
 - Automatic failover
 - Cluster creation and initial setup
 - Cluster management
 - ~ Monitoring

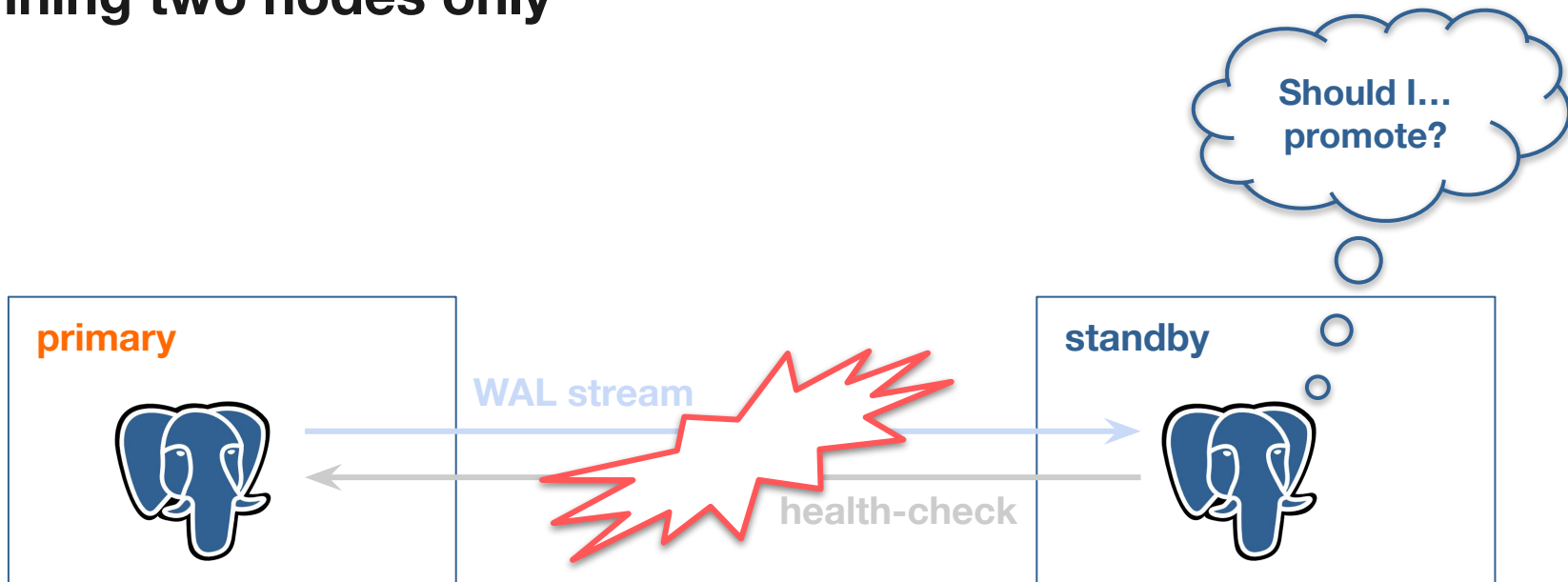
Automatic failover done wrong



Running two nodes only



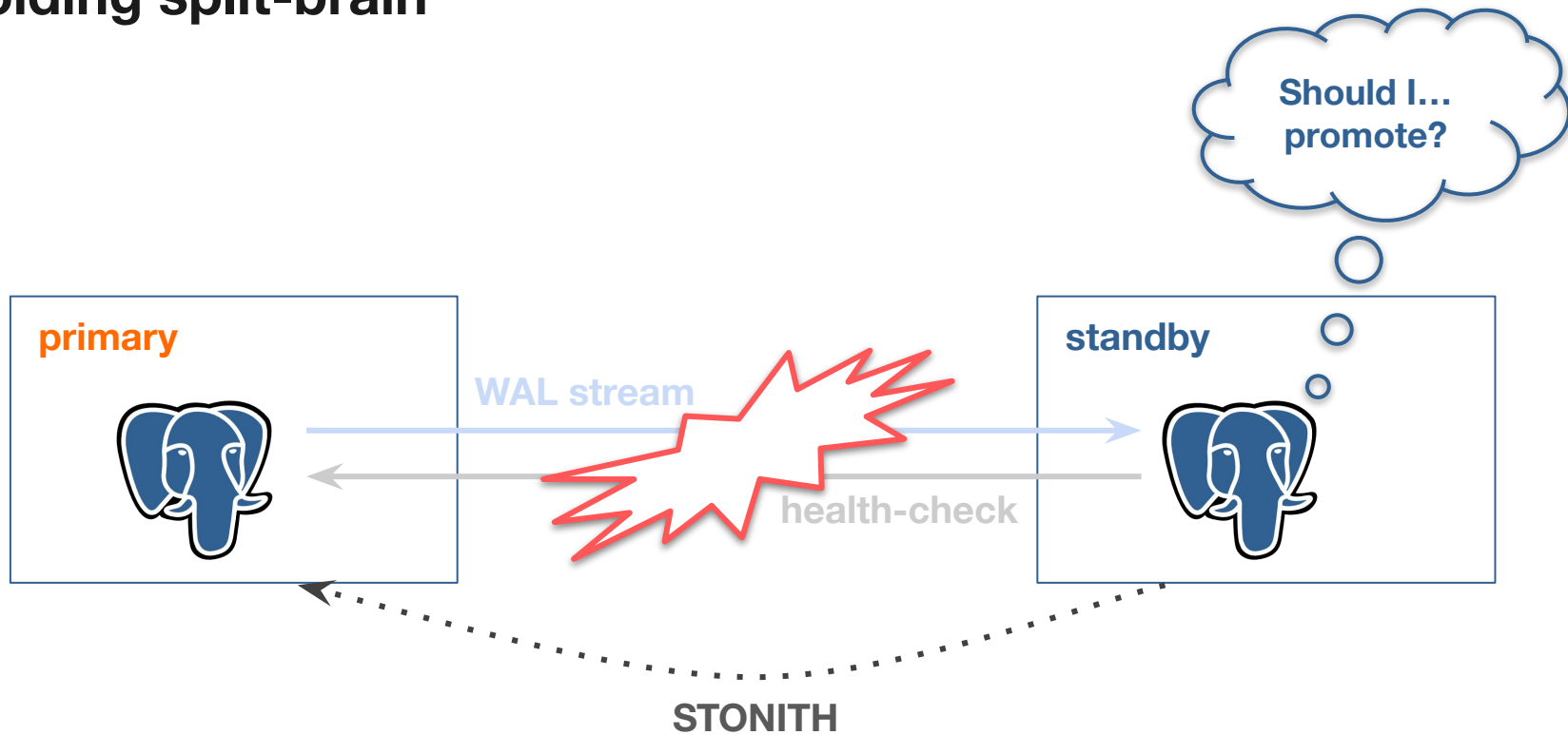
Running two nodes only



Running two nodes only



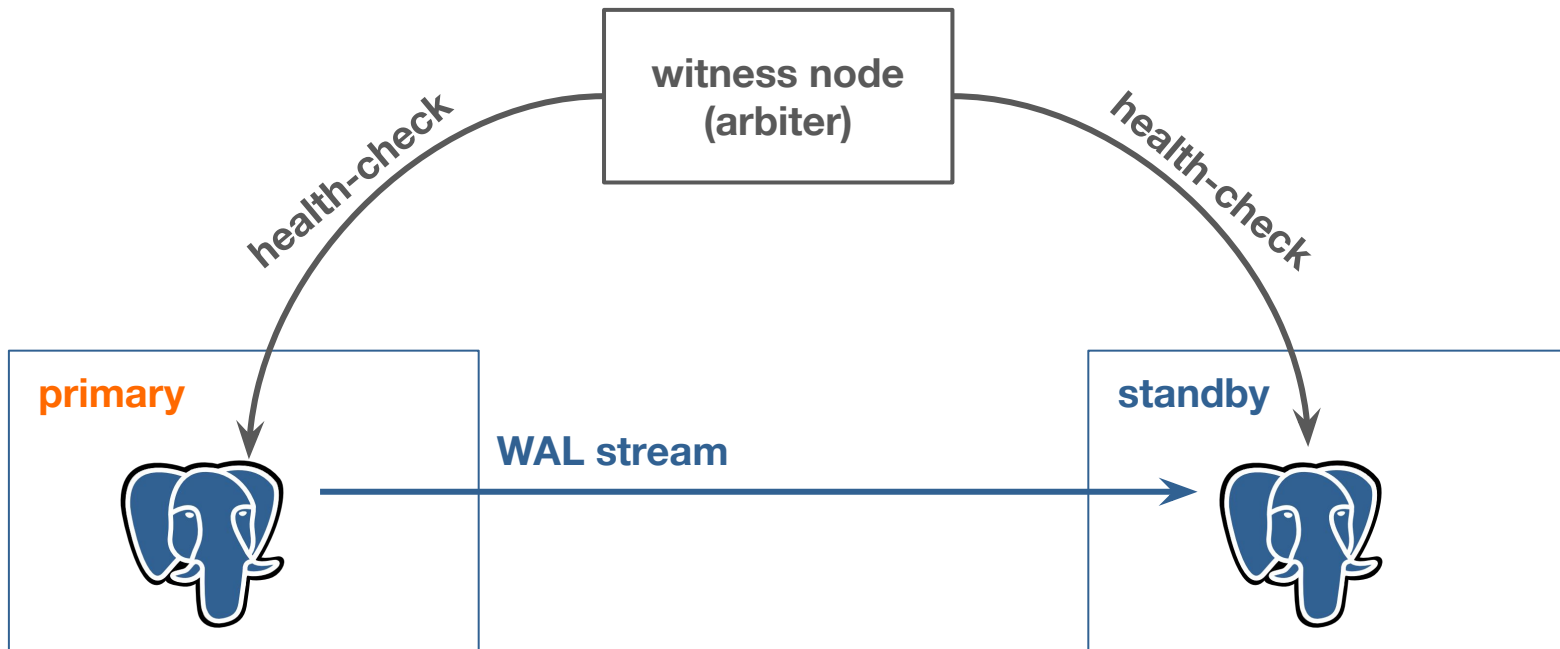
Avoiding split-brain



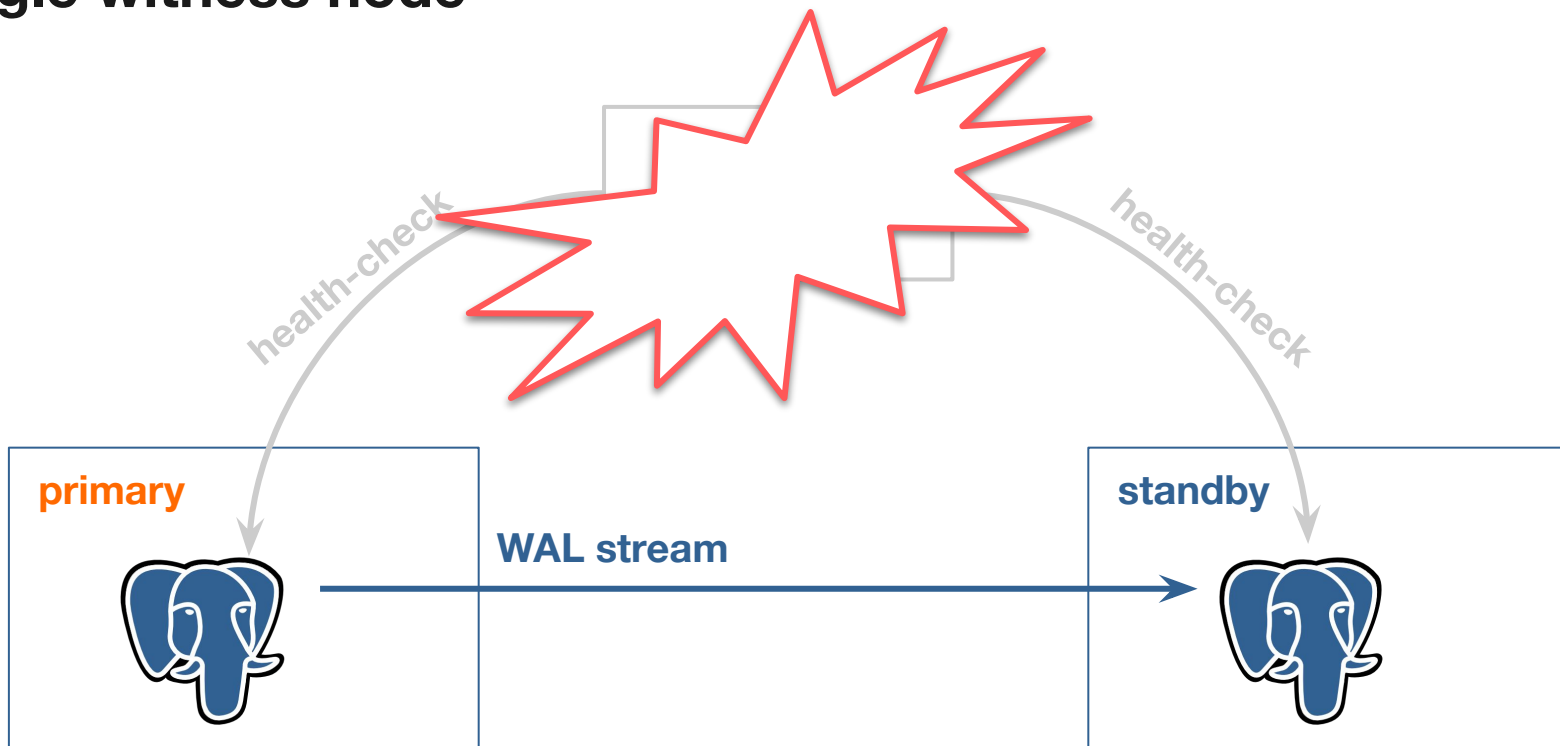
Avoiding split-brain

- STONITH (shoot the other node in the head)
- Must use a secondary network
- Almost impossible to get it right

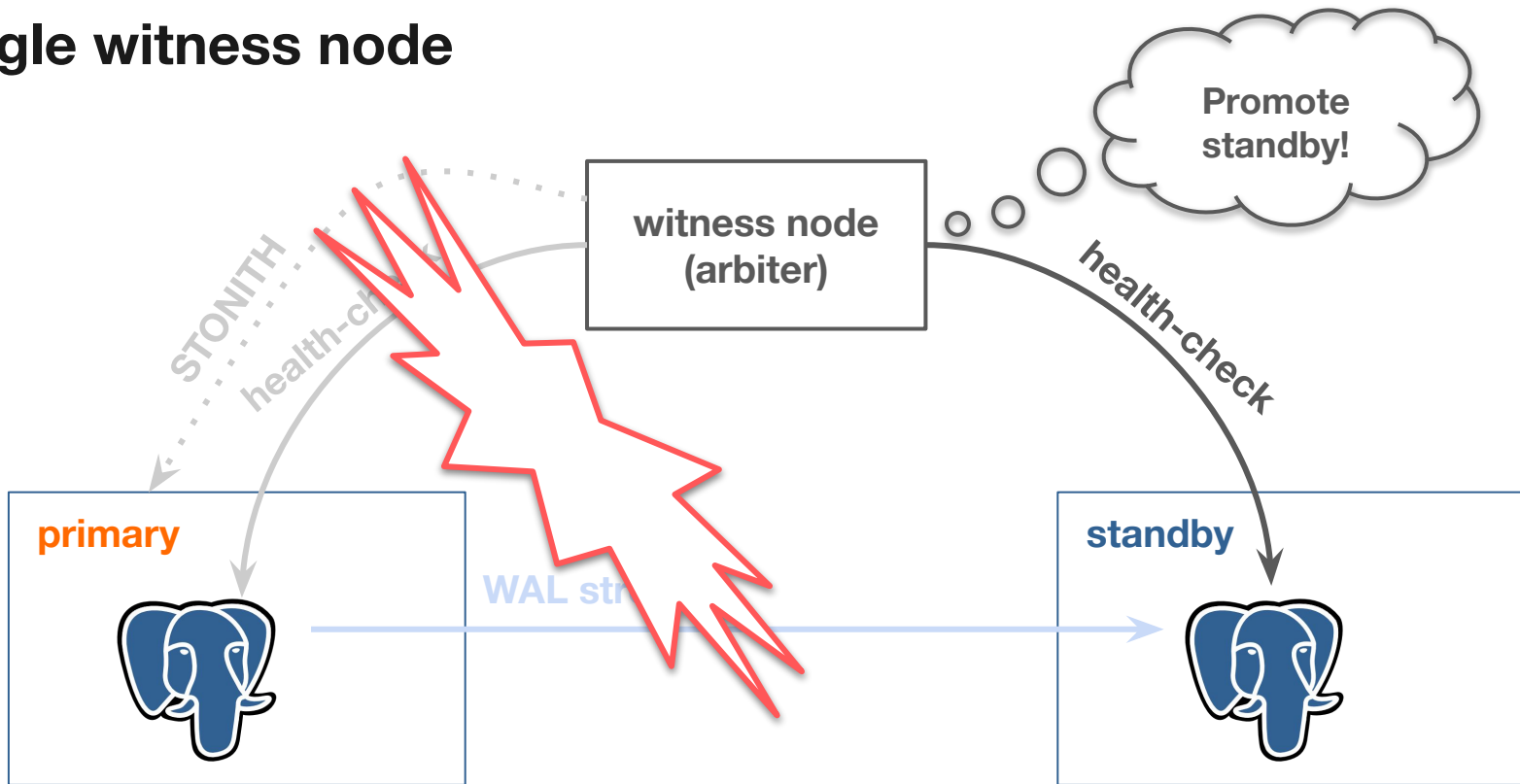
Single witness node



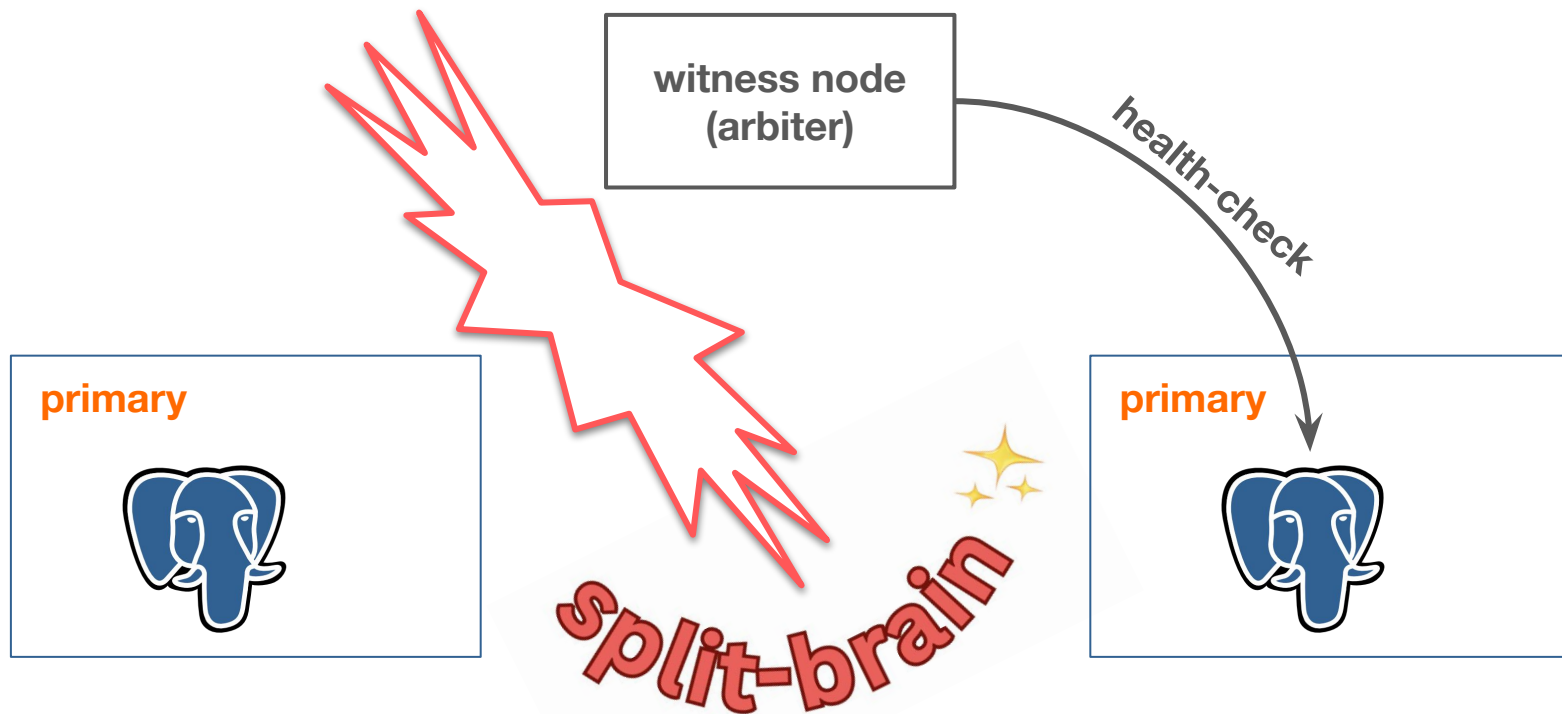
Single witness node



Single witness node



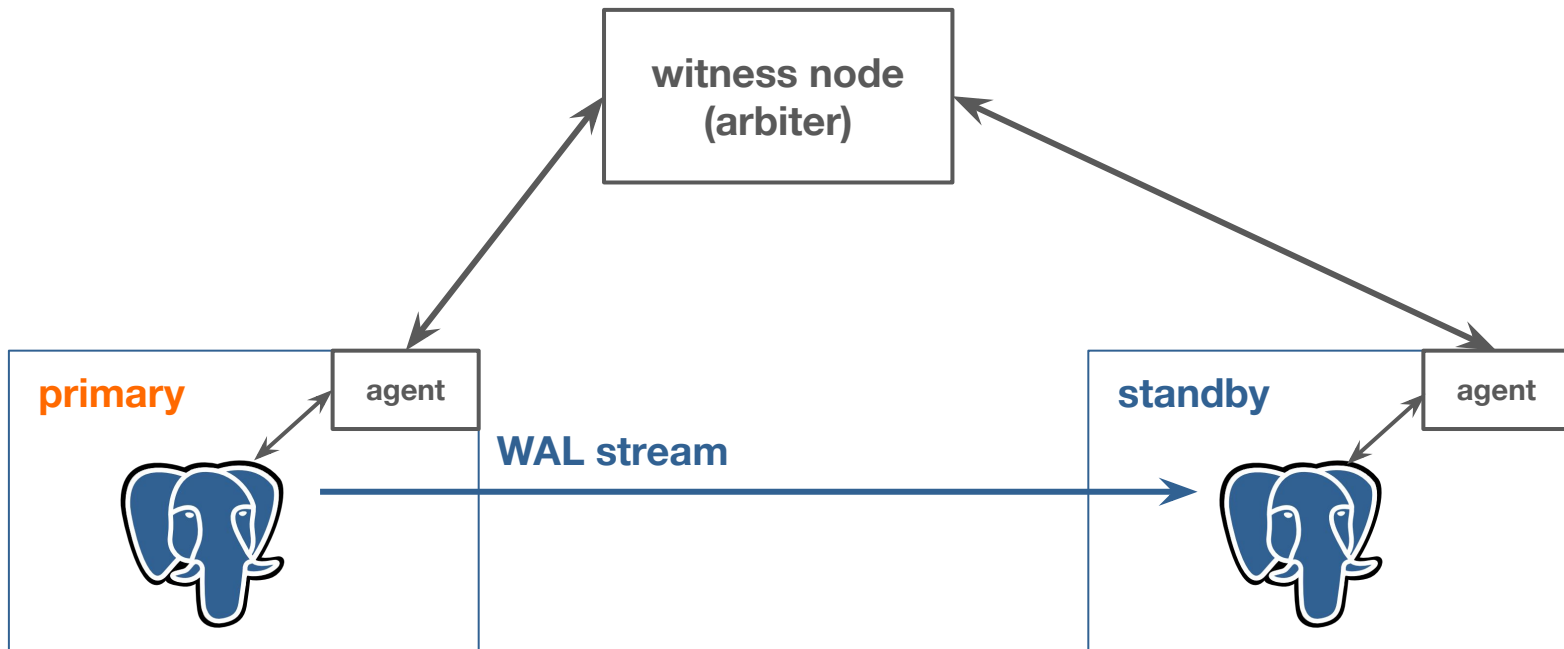
Single witness node



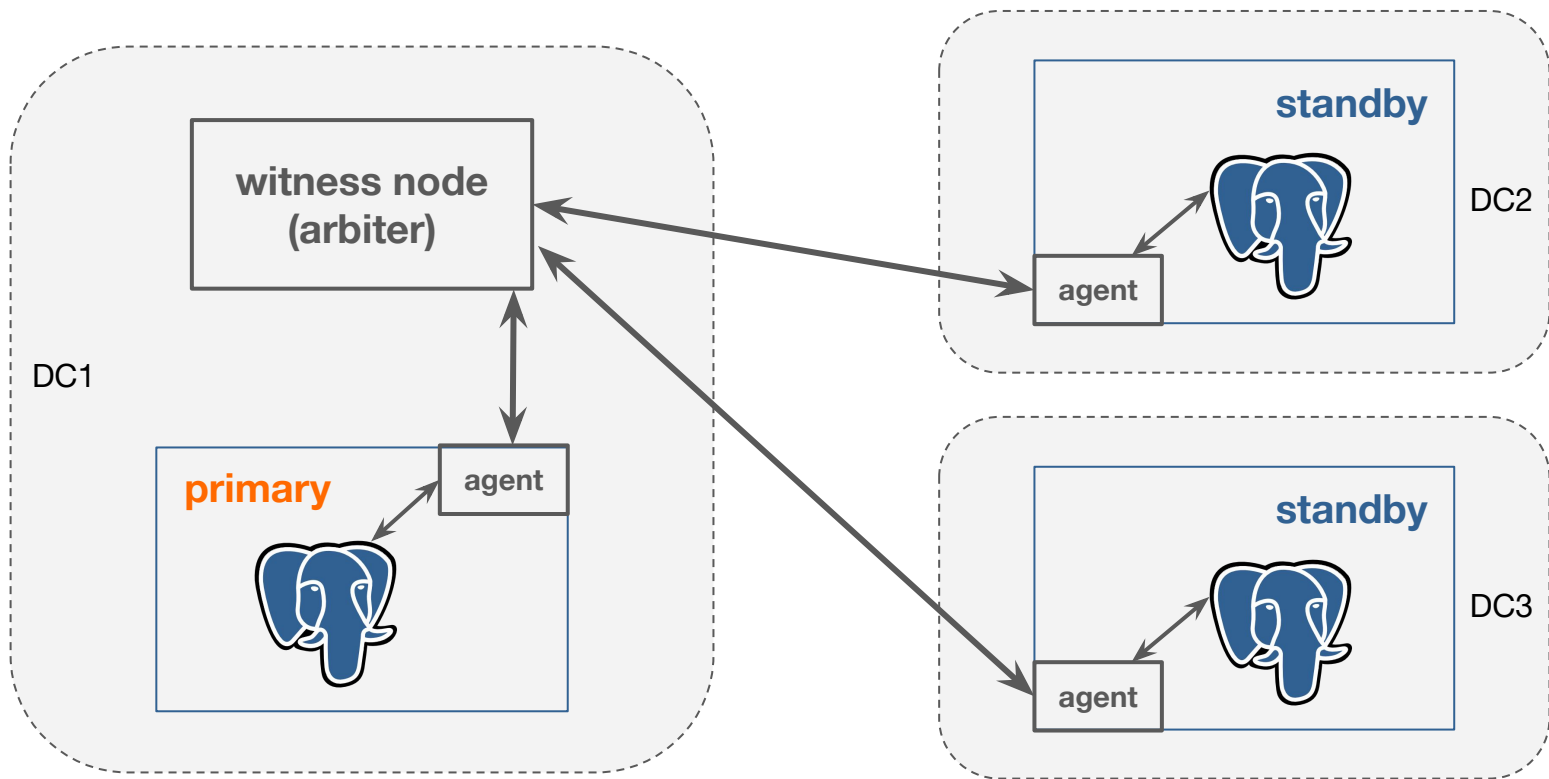
Things to consider

- Think about network partition
- Prevent split-brain → fencing
 - STONITH
 - Shut down
 - Kill old connections, re-configure proxy
 - Self-fencing (locally)
- Watchdog

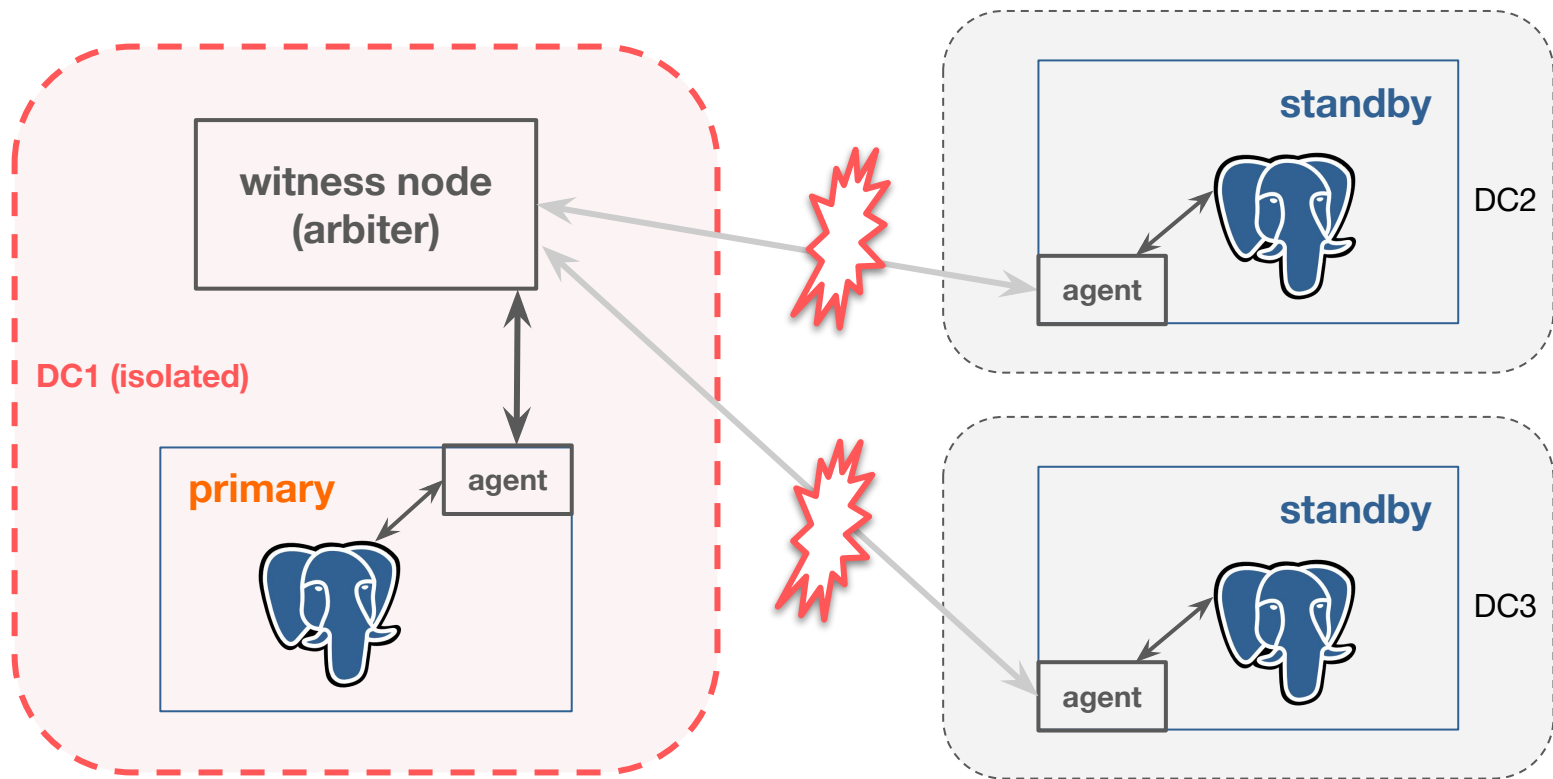
Local agents



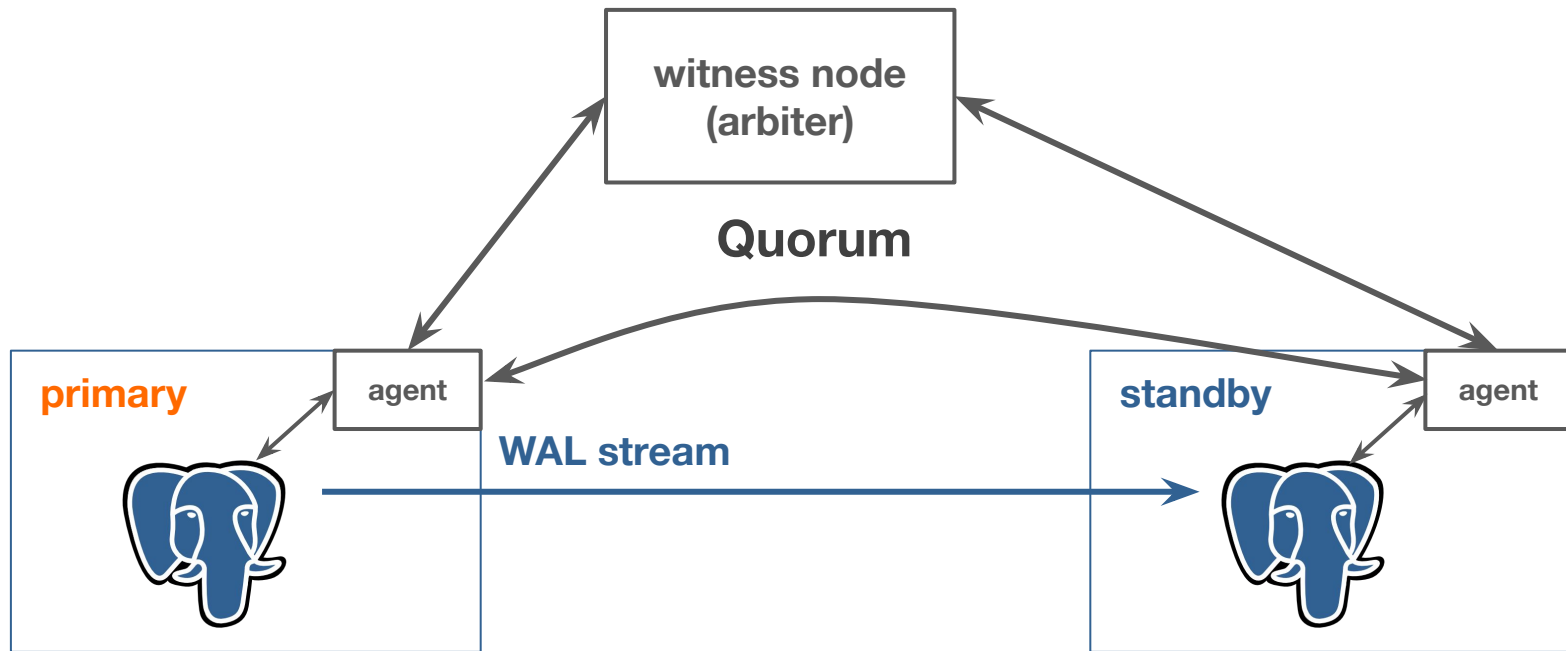
Local agents



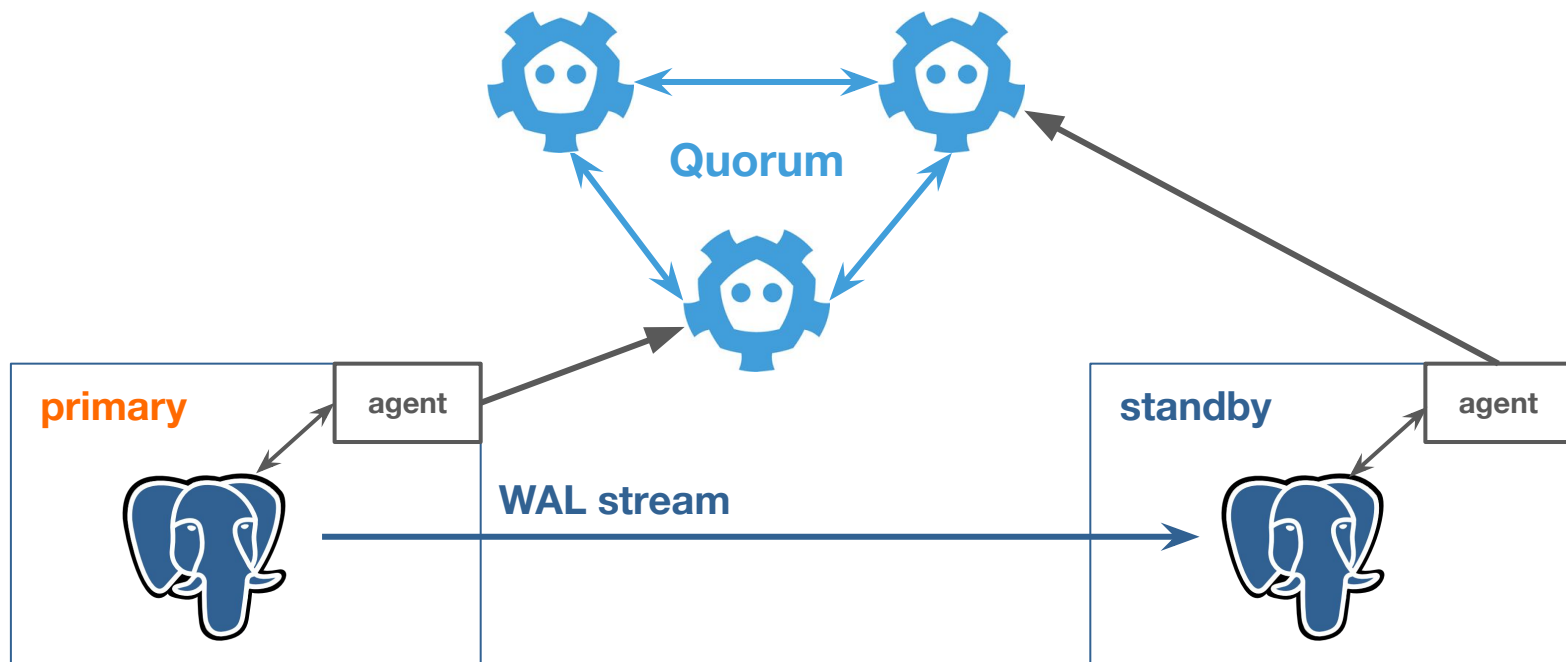
Local agents



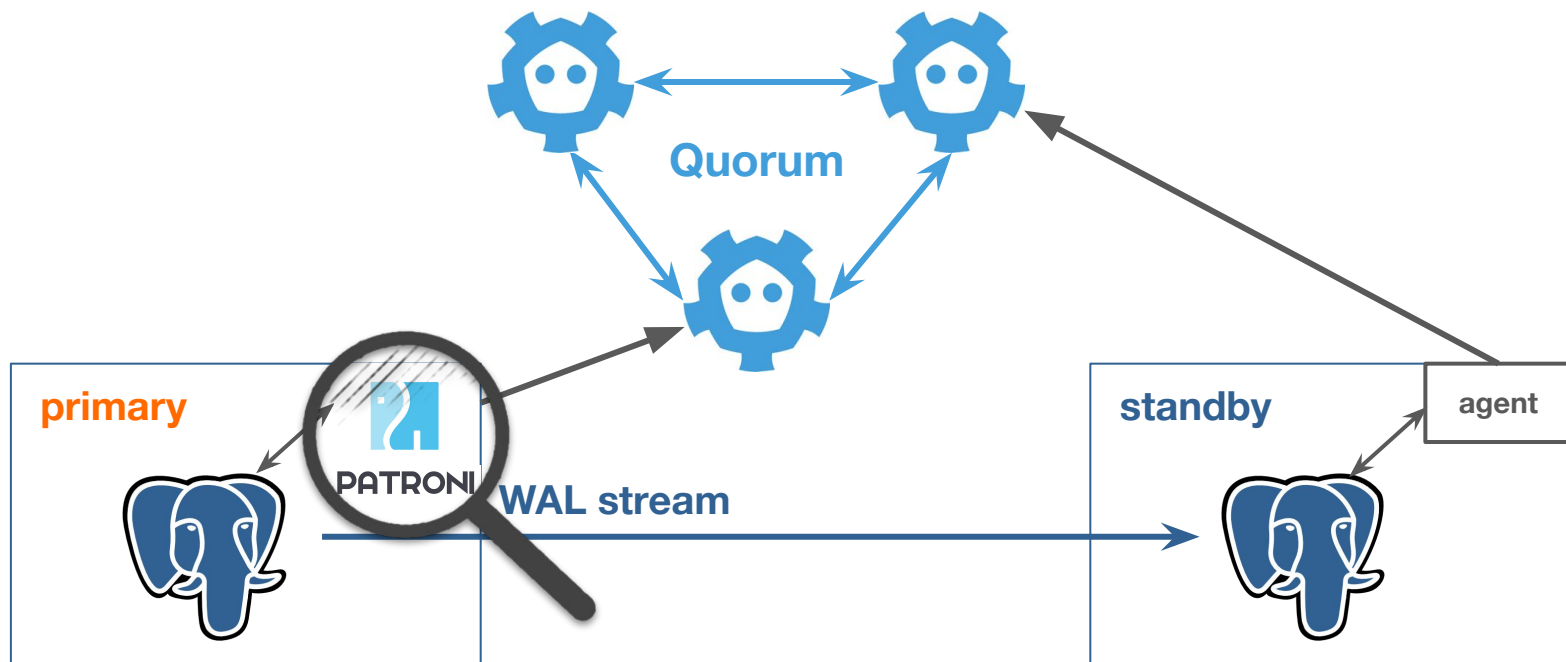
But how to do it right?



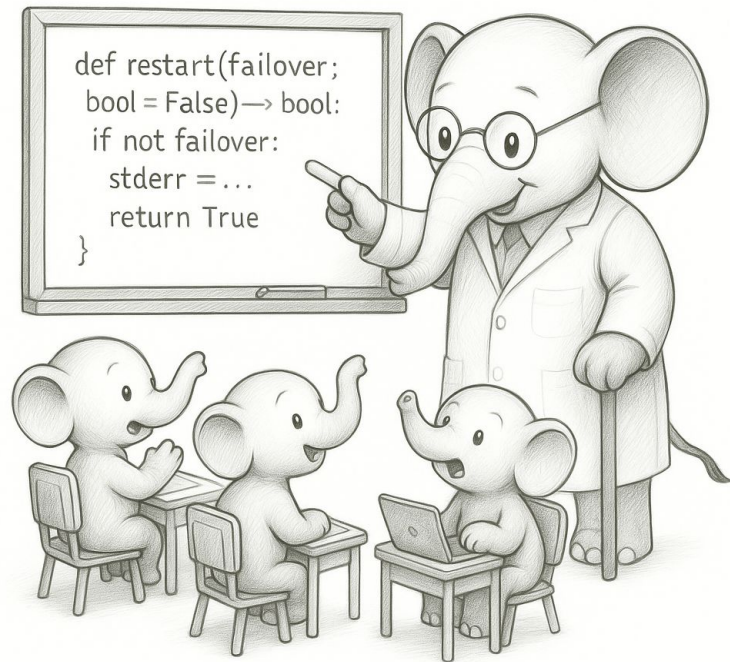
But how to do it right?



But how to do it right?



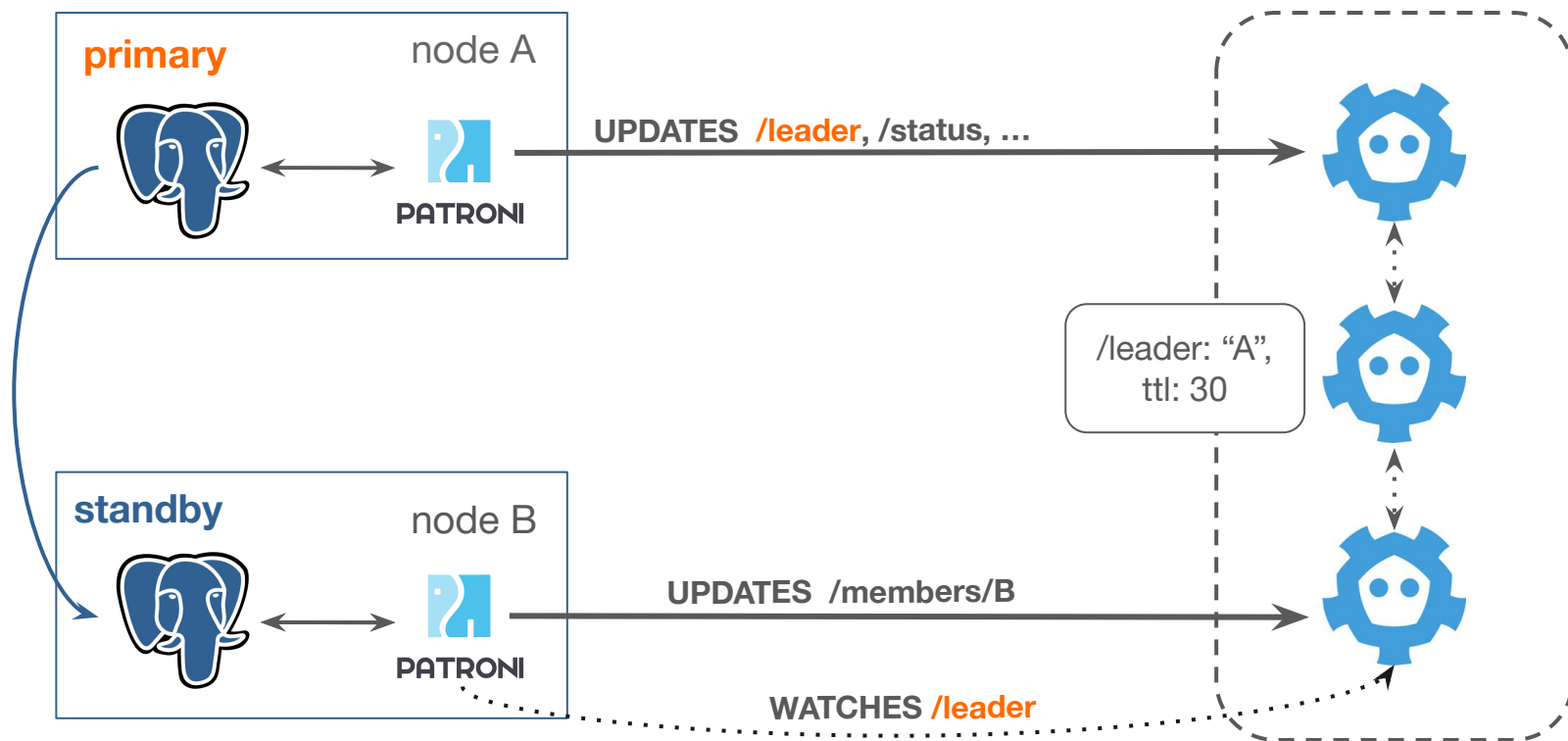
Patroni: how it works?



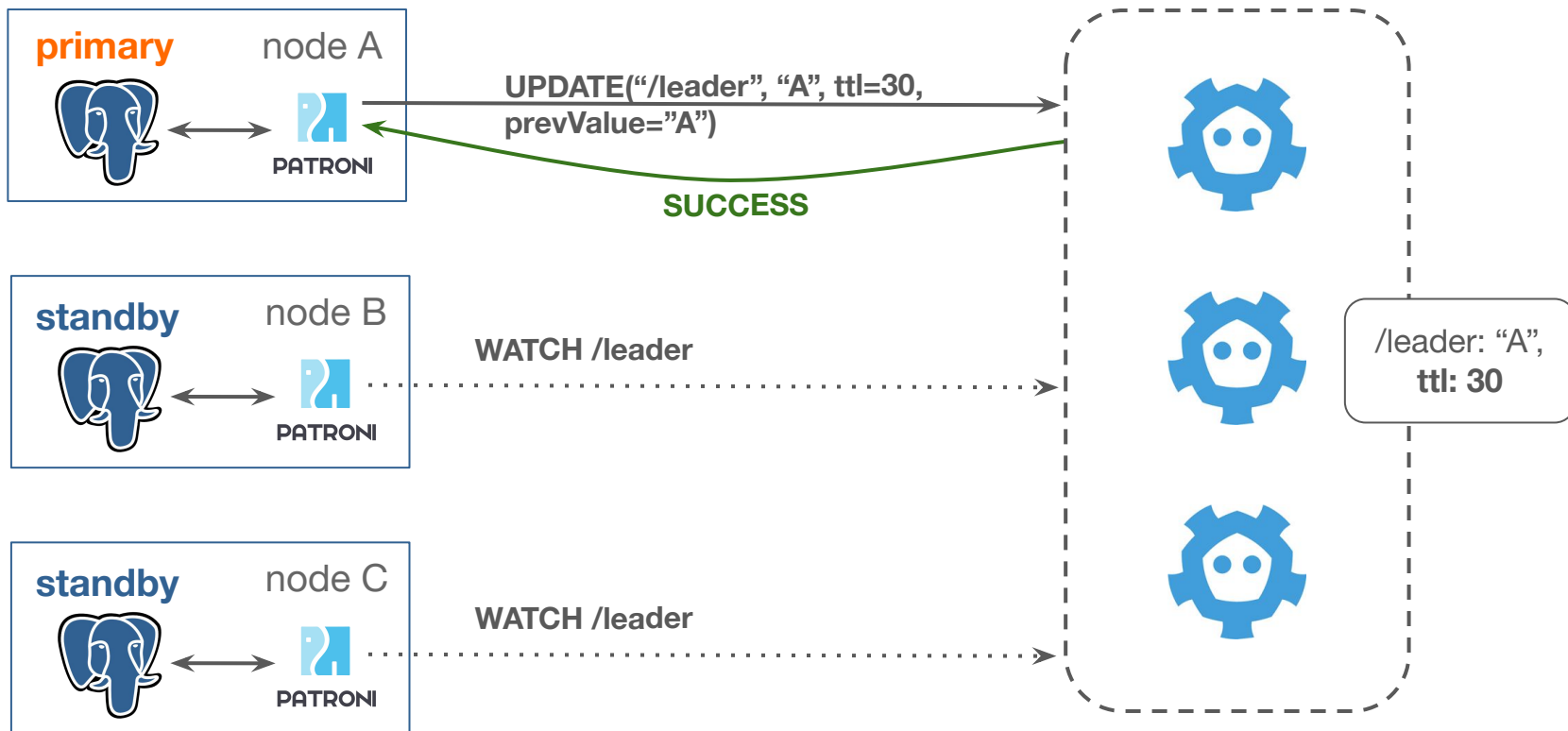
General idea

- State stored in Distributed Configuration Store (**DCS**)
 - Etcd, ZooKeeper, Consul, Kubernetes control-plane
- **Built-in distributed consensus** (RAFT, Zab)
- Key-value store
- **Atomic CAS** (compare-and-swap) **operations**
- **Lease/Session/TTL** to expire data
 - */leader, /members/**
- **Watches** for keys

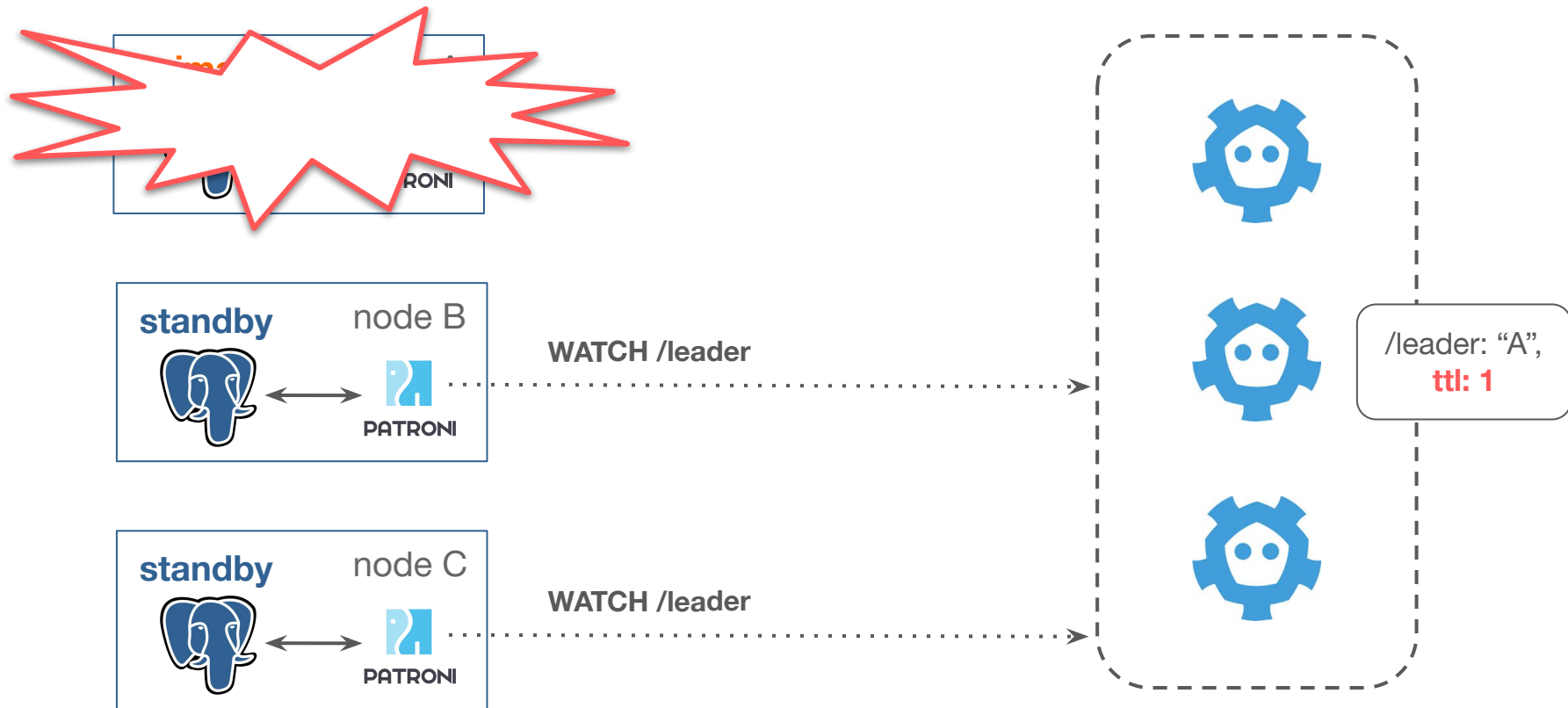
Patroni overview



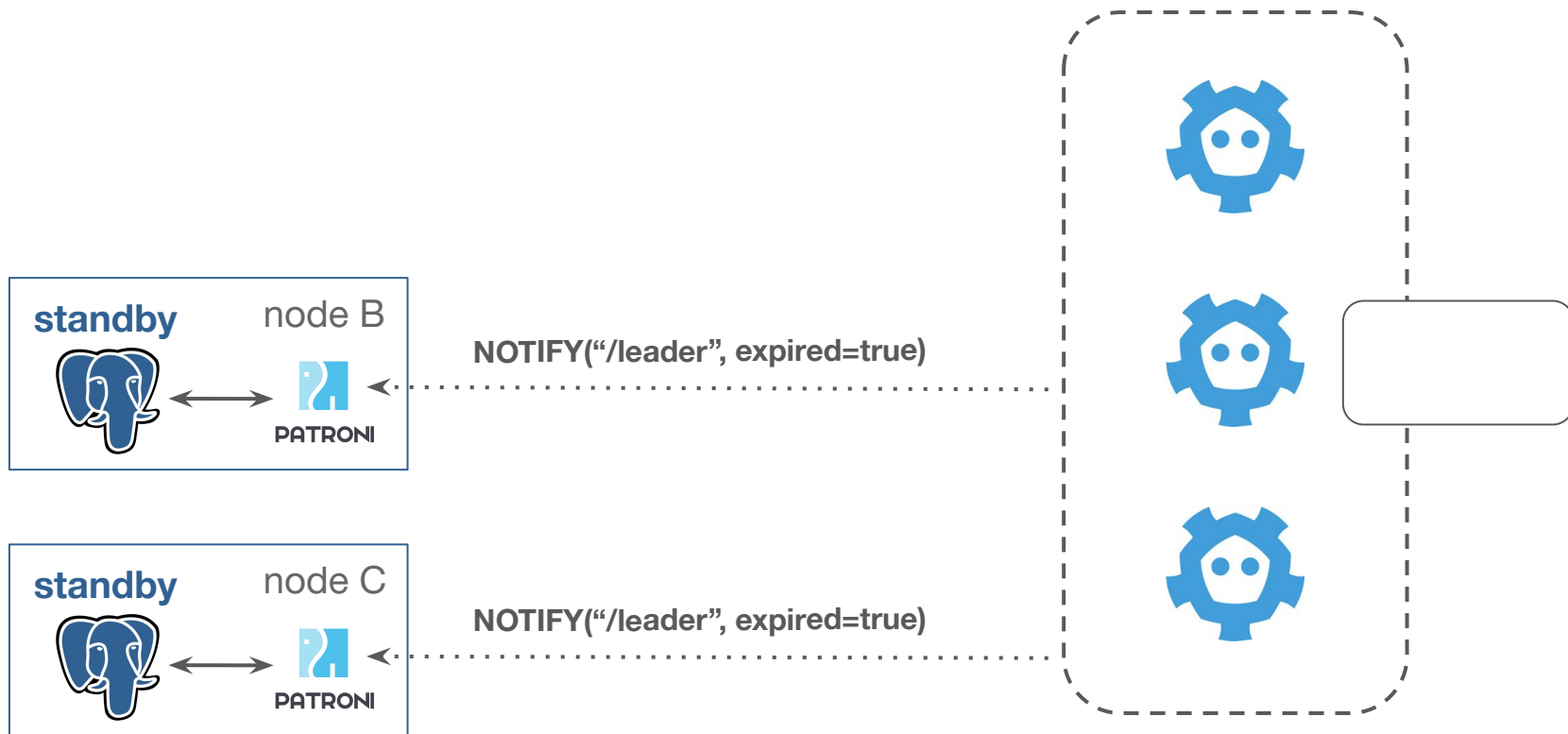
Leader race



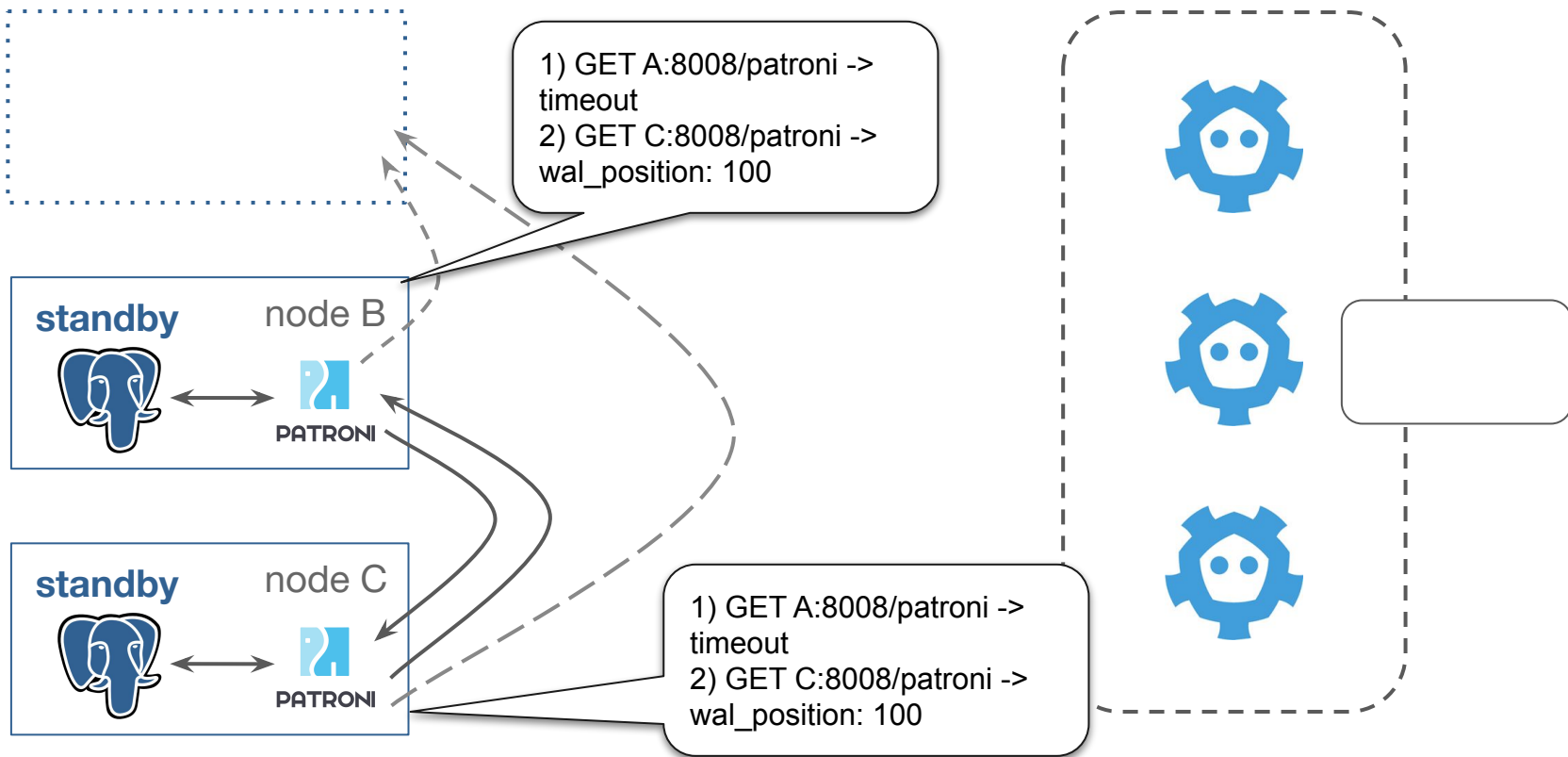
Leader race



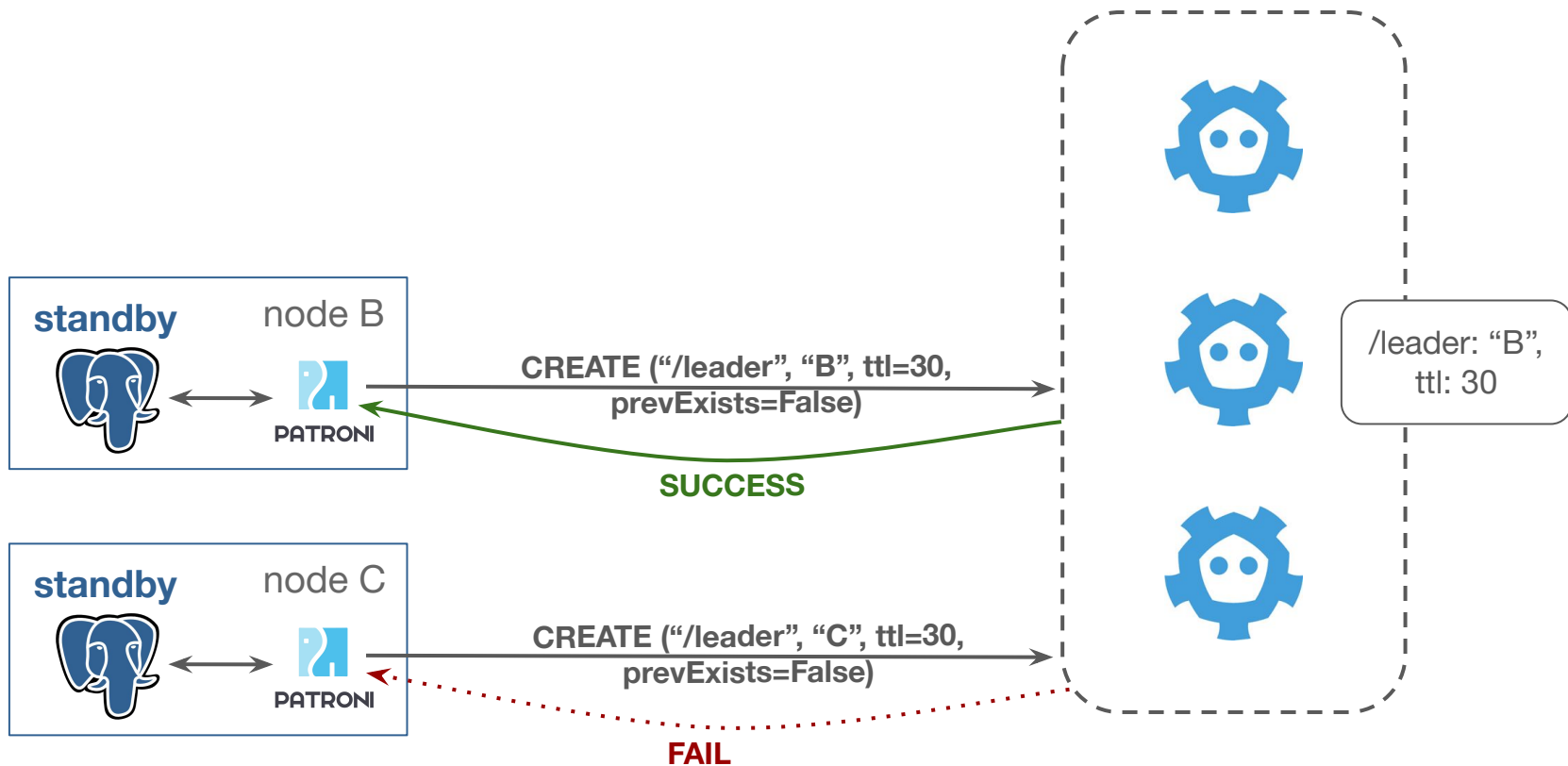
Leader race



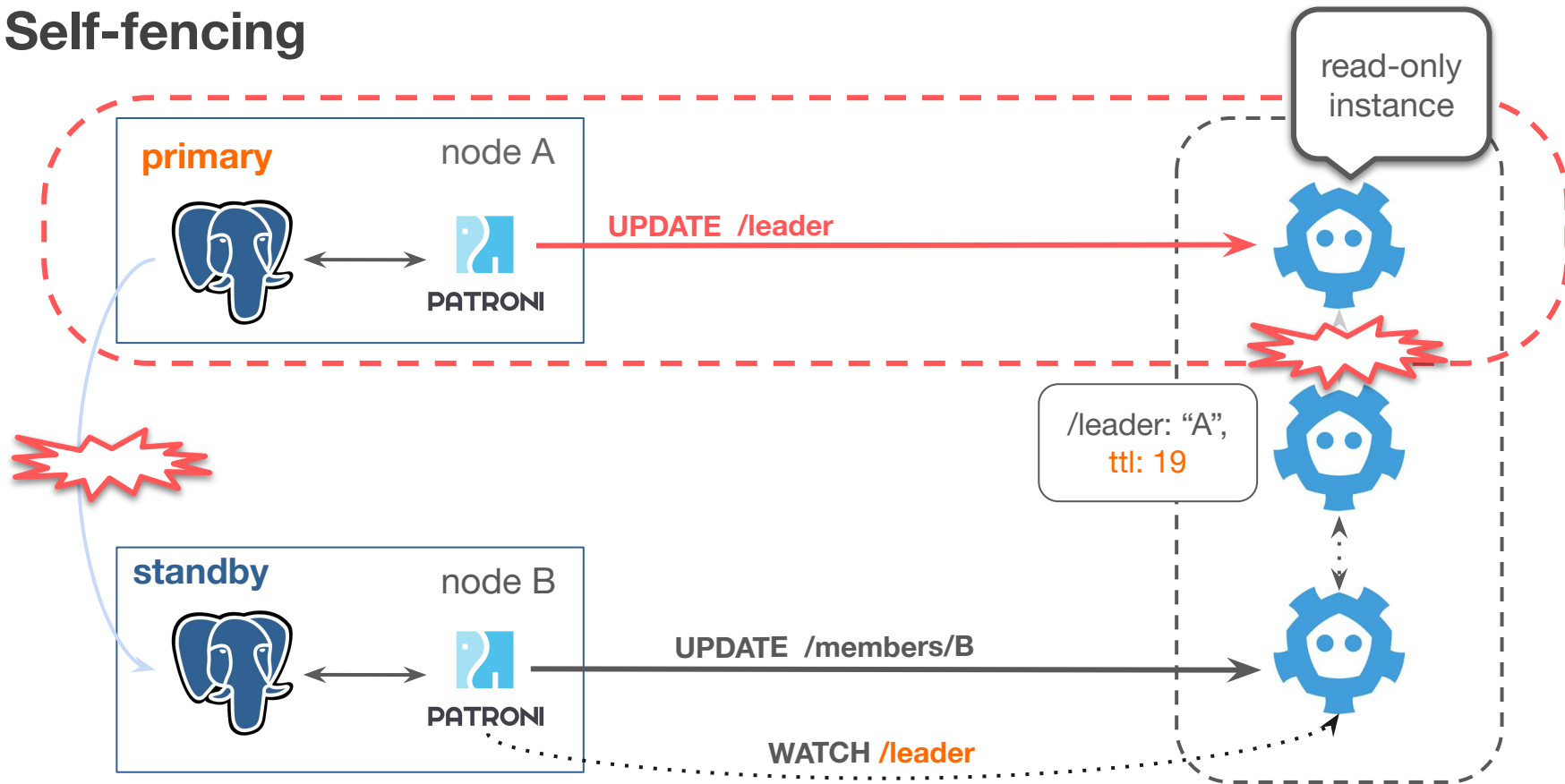
Leader race



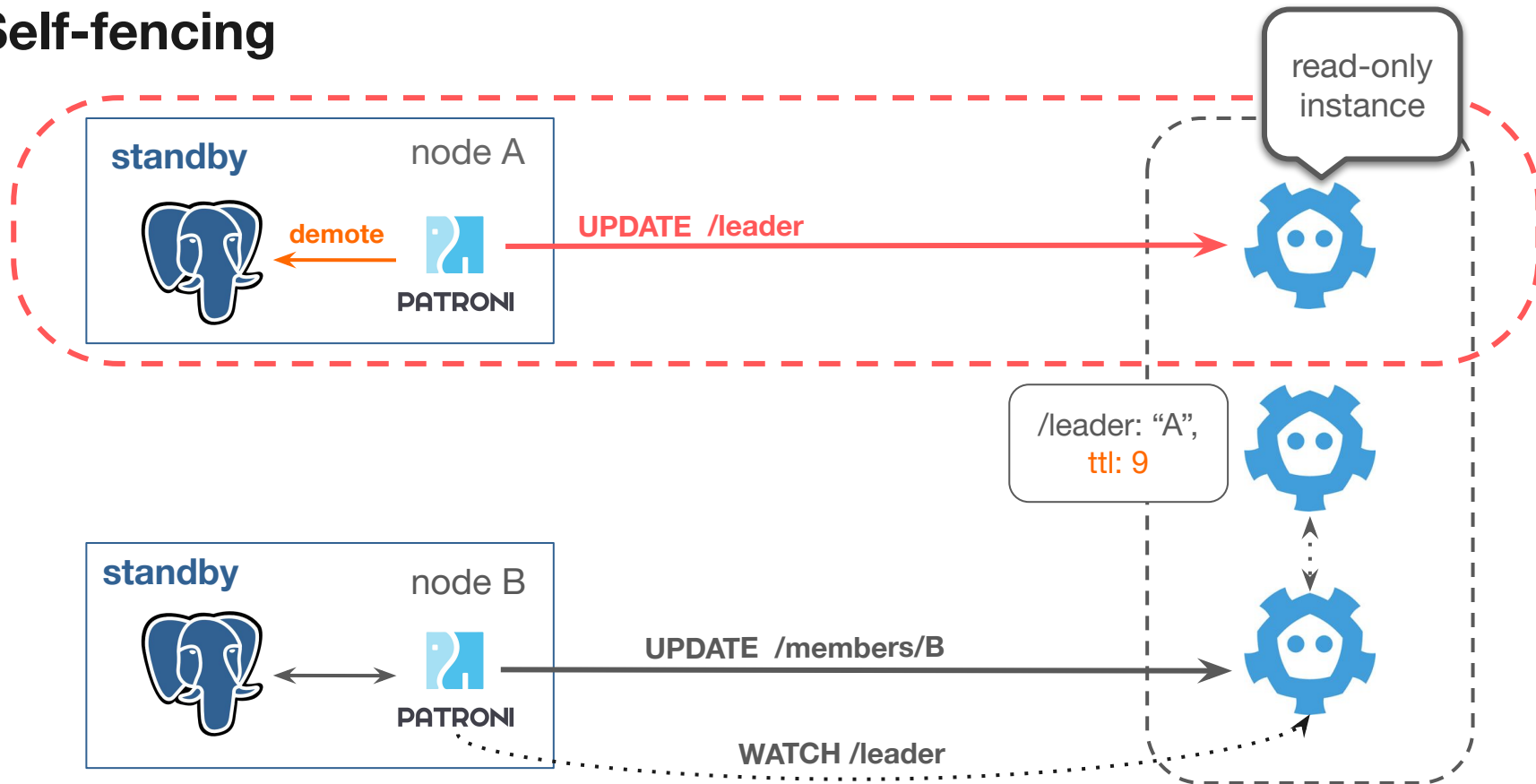
Leader race



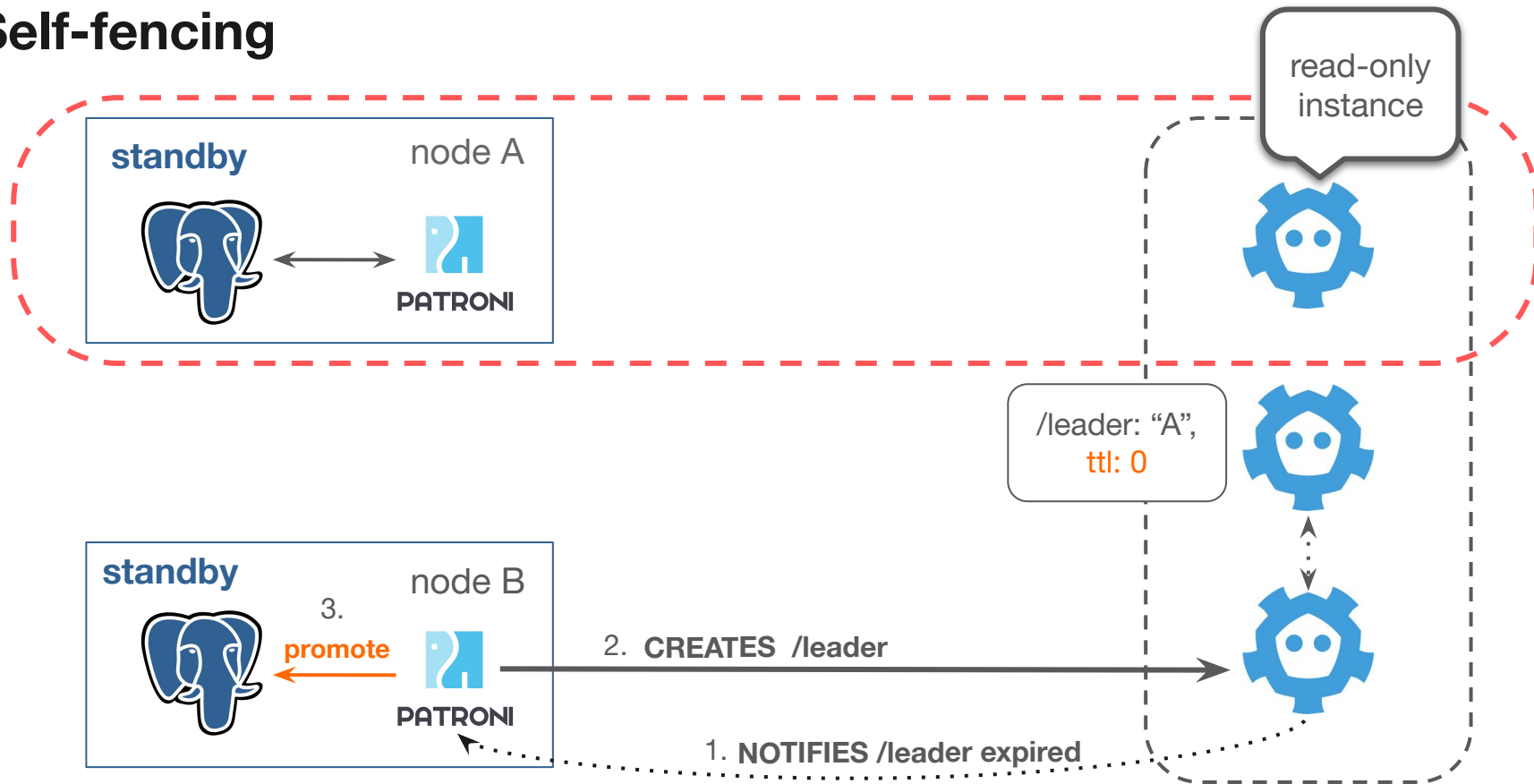
Self-fencing



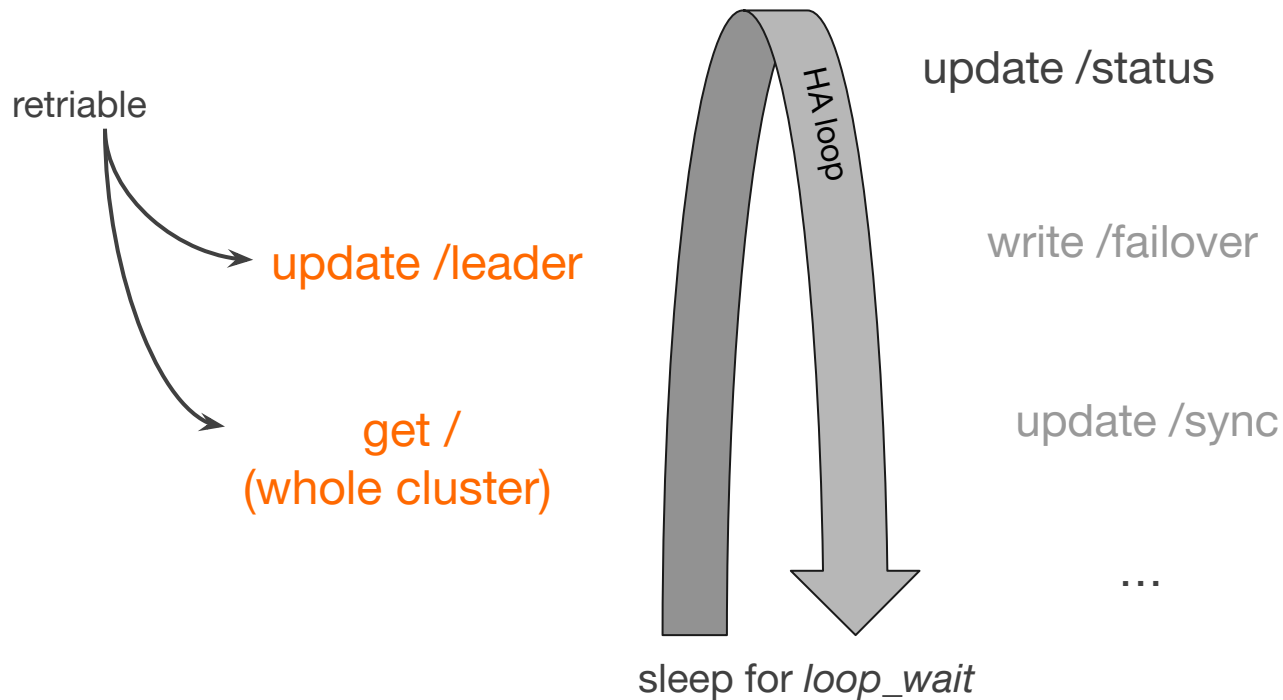
Self-fencing



Self-fencing



Communication with DCS – leader



ttl, loop_wait, retry_timeout

$$\begin{array}{ccc} (10) & & (10) & & (30) \\ & \nearrow & & \nwarrow & \\ \text{loop_wait} + 2 * \text{retry_timeout} & \leq & \text{ttl} \end{array}$$

get /
(whole cluster)

update /leader

Data stored in DCS

```
$ etcdctl get --keys-only --prefix /service/demo
```

<code>/service/demo/config</code>	<code>/* global (dynamic) configuration */</code>
<code>/service/demo/initialize</code>	<code>/* cluster identifier */</code>
<code>/service/demo/leader</code>	<code>/* who is the primary? */</code>
<code>/service/demo/members/patroni1</code>	<code>/* discovery */</code>
<code>/service/demo/members/patroni2</code>	
<code>/service/demo/members/patroni3</code>	
<code>/service/demo/status</code>	
<code>/service/demo/history</code>	<code>/* failover history */</code>
<code>/service/demo/failover</code>	<code>/* manual failover/switchover */</code>
<code>/service/demo/sync</code>	<code>/* synchronous mode */</code>

Data stored in DCS

data retrieved
from DCS

```
$ patronictl list
```

```
+ Cluster: demo (7497665970948870167) -----+-----+-----+
| Member    | Host          | Role    | State      | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| patroni1  | 172.18.0.2    | Leader  | running    | 1  |           |
| patroni2  | 172.18.0.7    | Replica | streaming  | 1  | 0         |
| patroni3  | 172.18.0.3    | Replica | streaming  | 1  | 0         |
+-----+-----+-----+-----+-----+-----+
```

Data stored in DCS

```
$ etcdctl get --print-value-only --prefix /service/demo/leader
```

```
patroni1
```

```
$ etcdctl get --print-value-only --prefix /service/demo/initialize
```

```
7497665970948870167
```

Data stored in DCS

```
$ etcdctl get --keys-only --prefix /service/demo/members/patroni2

{
  "conn_url": "postgres://172.18.0.7:5432/postgres",
  "api_url": "http://172.18.0.7:8008/patroni",
  "state": "running",
  "role": "replica",
  "version": "4.0.5",
  "xlog_location": 67425896, /* max(receive_lsn or 0, replay_lsn or 0) */
  "replication_state": "streaming",
  "timeline": 1
}
```

Data stored in DCS

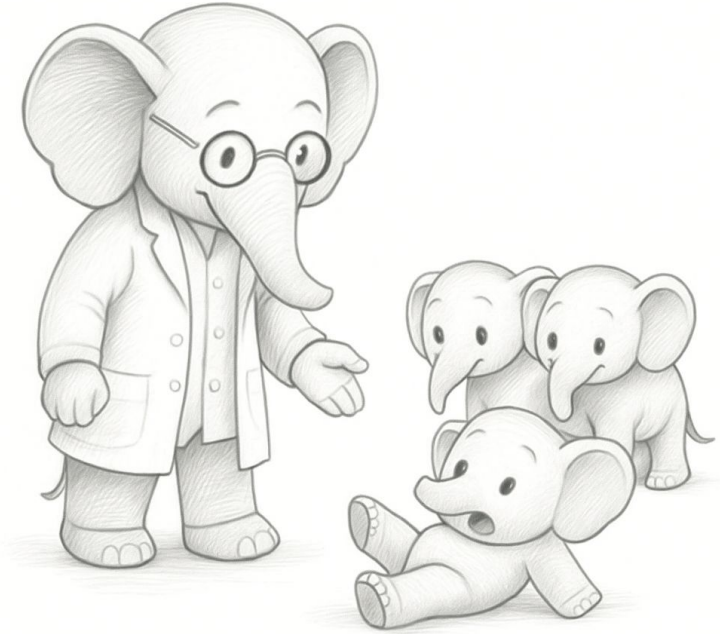
```
$ etcdctl get --print-value-only --prefix /service/demo/status
{
  "optime": 67425896, /* pg_current_wal_flush_lsn() */
  "slots": {
    "patroni2": 67425896,
    "patroni3": 67425896,          /* members slots */
    "patroni1": 67425896,
    "my_logical_slot": 67425700    /* permanent slots */
  },
  "retain_slots": [
    "patroni1",
    "patroni2",          /* member_slots_ttl */
    "patroni3"
  ]
}
```


Data stored in DCS

```
$ etcdctl get --print-value-only --prefix /service/demo/config
{
  "loop_wait": 10,
  "ttl": 30,
  "retry_timeout": 10,
  "maximum_lag_on_failover": 1048576,
  "postgresql": {
    "parameters": {
      "max_connections": 100
    },
    "use_pg_rewind": true
  },
  "synchronous_mode": "quorum"
}
```

/ applied to all members (global) */*

What else?



Notable features

- **Standby cluster** – running cascading replication to a remote datacenter (region) [\[docs\]](#)
- **Synchronous mode** – manage “*synchronous_standby_names*” to enable synchronous replication whenever there are healthy standbys available [\[docs\]](#)
- **Quorum-based failover** – reduce latencies, compensating higher latency of replicating to one synchronous standby by other standbys [\[docs\]](#)
- **DCS failsafe mode** – survive temporary DCS outages without primary demotion [\[docs\]](#) [\[slides\]](#)
- **Citus support** [\[docs\]](#) [\[article\]](#)

More links

- [Patroni – Postgres.FM](#) podcast
- [Patroni tutorial](#) (A bit outdated but still good)
- [Step-by-step Patroni cooking guide](#) talk slides
- [Official documentation](#) (Read the docs! No, seriously...)
- [Changelog](#) (new features and bugfixes)
- [Patroni](#) channel in the [PostgreSQL Slack](#)

Thank you!

