zalando

Patroni

Understanding and implementing PostgreSQL HA

PGCon 2019

Alexander Kukushkin

Oleksii Kliukin

28-05-2019

Agenda

Architecture overview

Hands on: your first test cluster

Client connections

Dynamic cluster configuration

REST endpoints and monitoring

Advanced features

Custom extensions

Troubleshooting

zalando

Do you need High Availability?

- Analyze your requirements (RPO, RTO, SLA)
- 99.99999% uptime is very expensive
- Keep it simple
- Avoid one-shot solutions

PostgreSQL High Availability

- Shared storage solutions
  - DRBD + LVM
- Trigger-based and logical replication
  - pglogical, bucardo, slony, londiste, built-in logical replication in PostgreSQL 10+
- Built-in physical single master replication
  - Starting from PostgreSQL 9.0
- Multi-master
  - BDR, Bucardo, Postgres Pro multimaster extension, Postgres XL, Postgres-XC

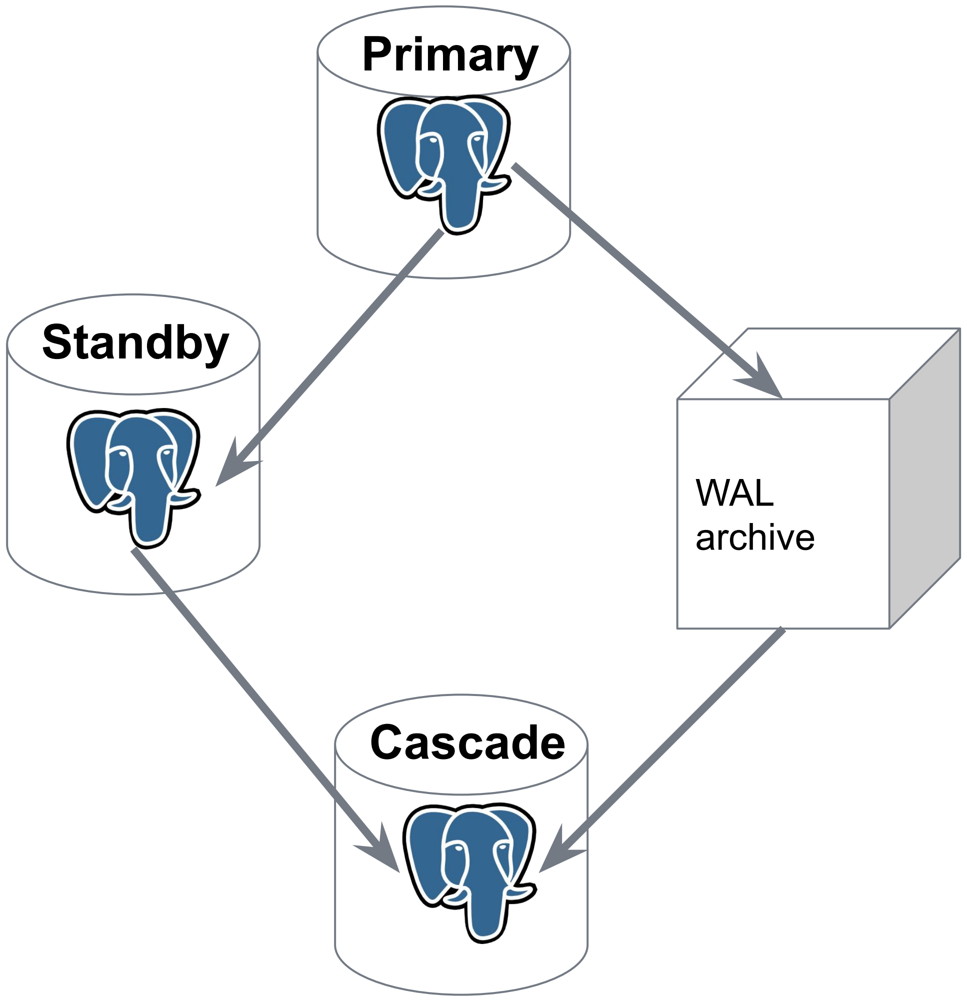
Physical single-master replication

Cons

- No partial replication
- Identical major Postgres versions
- No out-of-box automatic failover

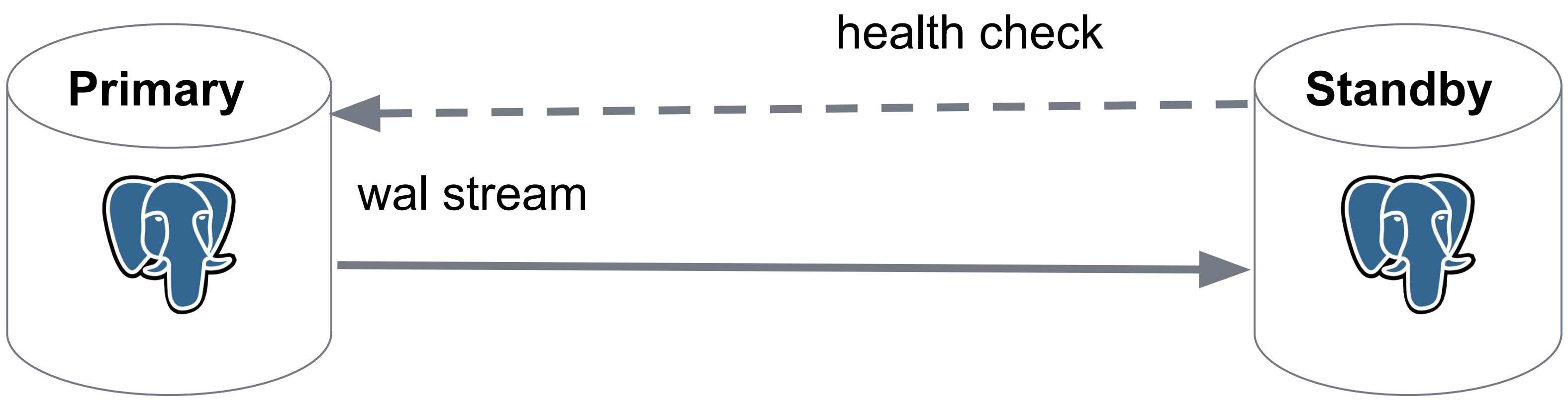
Pros

- Built-in since Postgres 9.0
- Minimal overhead
- Replicates everything
- Cascading replication
- Synchronous replication
- Complements WAL-shipping

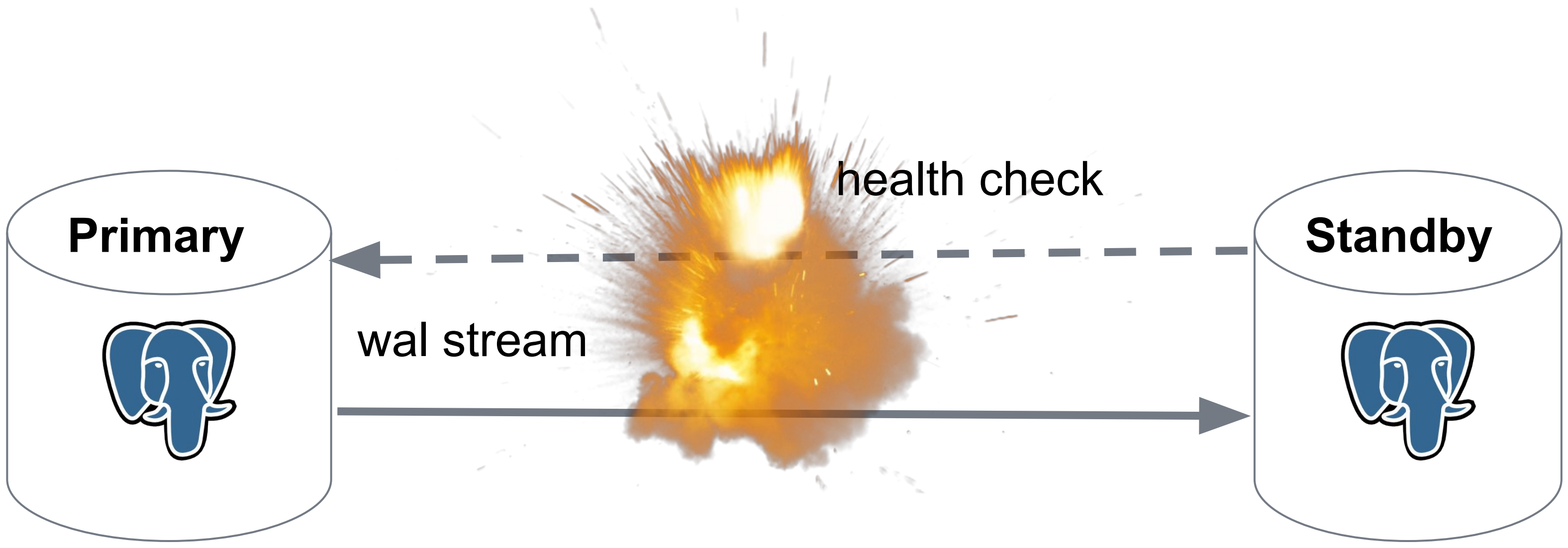


zalando

Unilateral failover decision without fencing

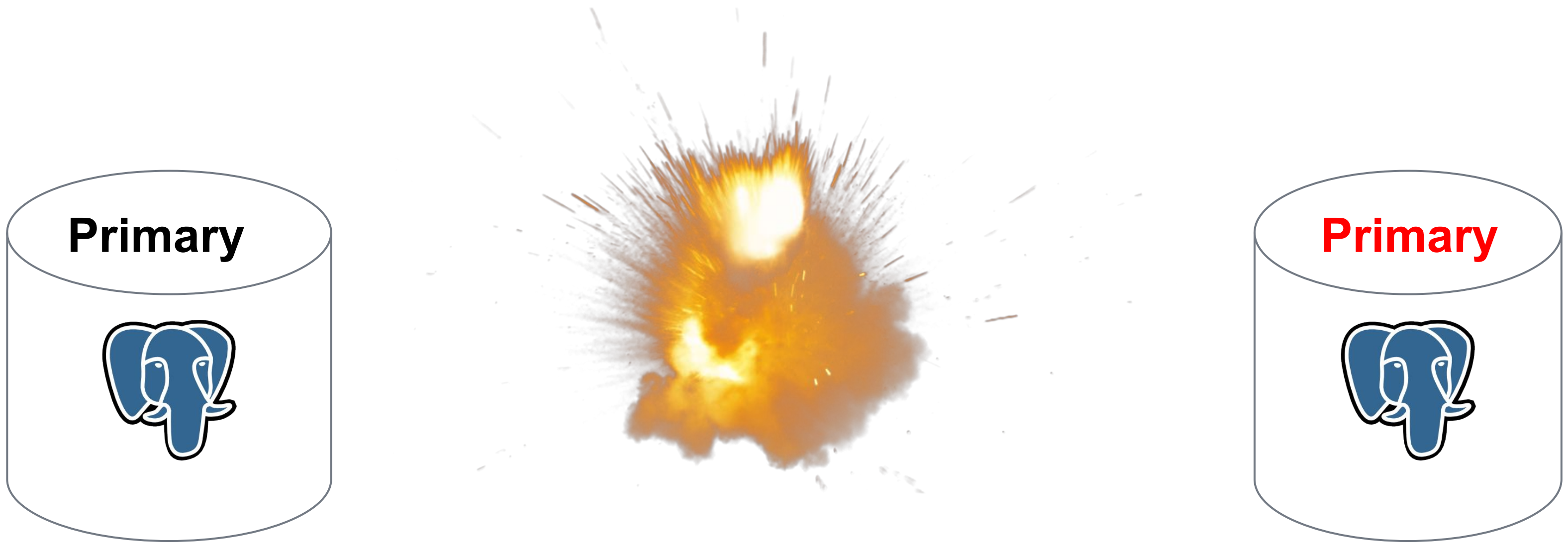


Unilateral failover decision without fencing



zalando

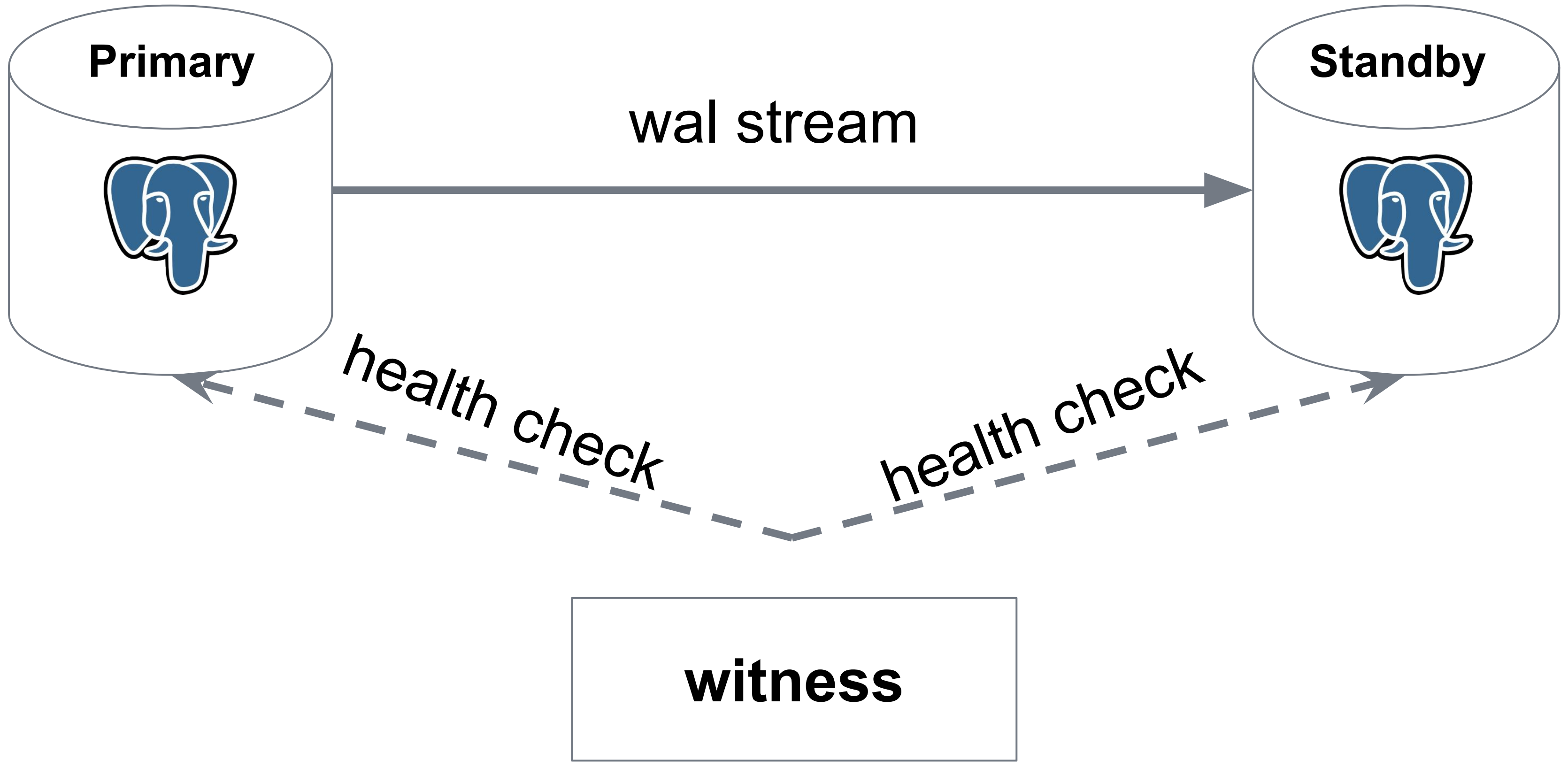
Unilateral failover decision without fencing



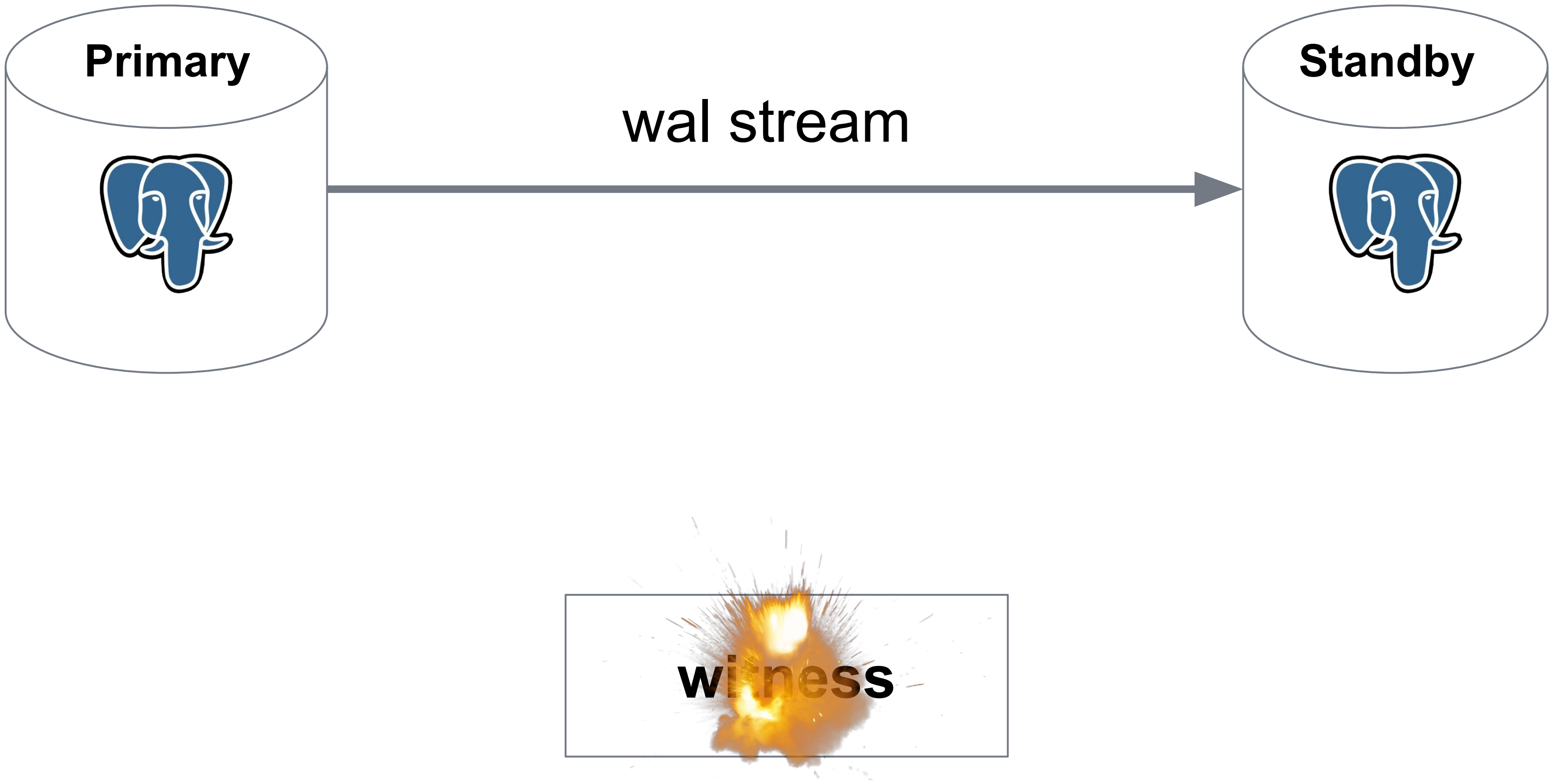
https://github.com/MasahikoSawada/pg_keeper

zalando

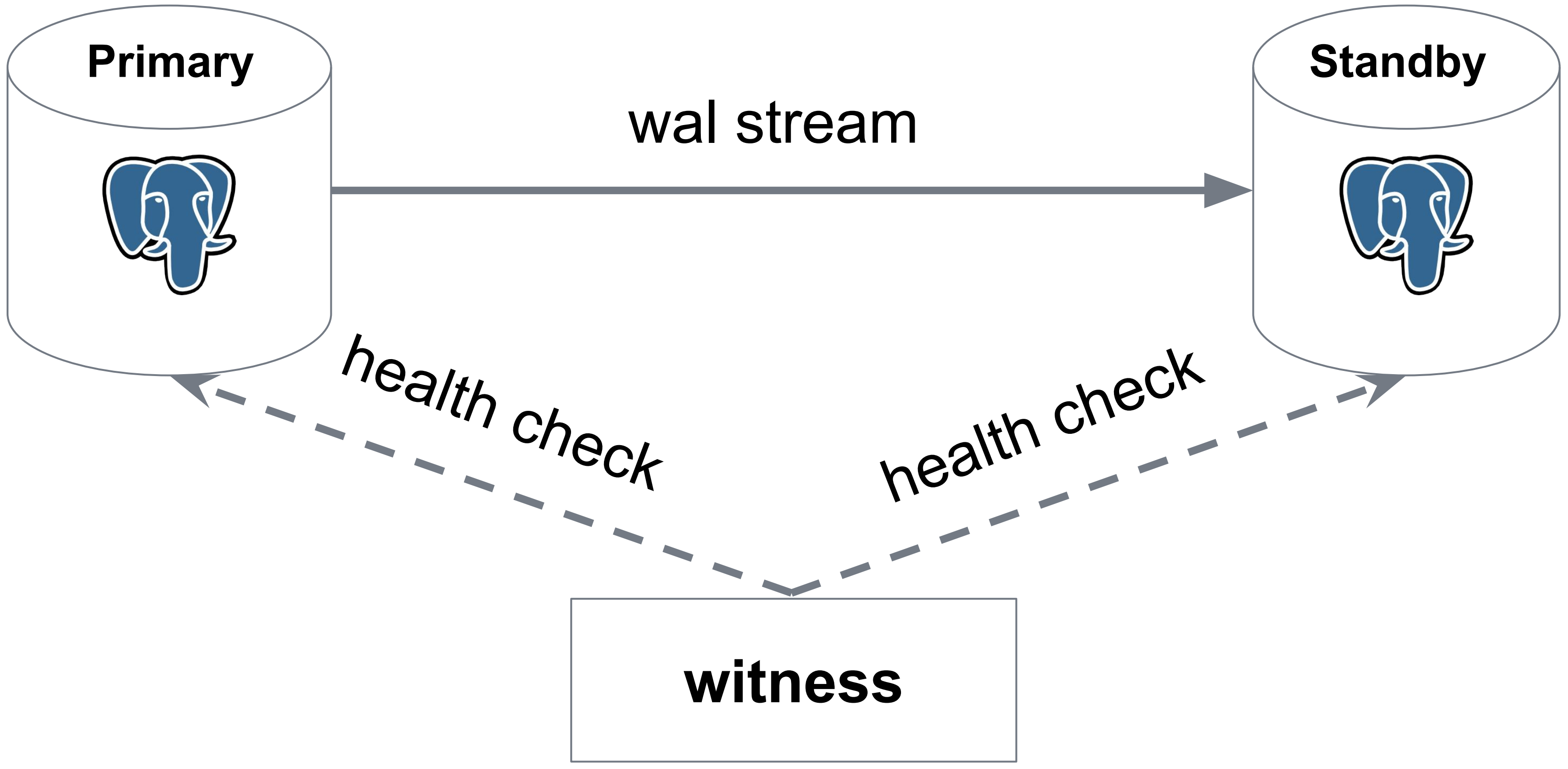
Single witness node to resolve partitions



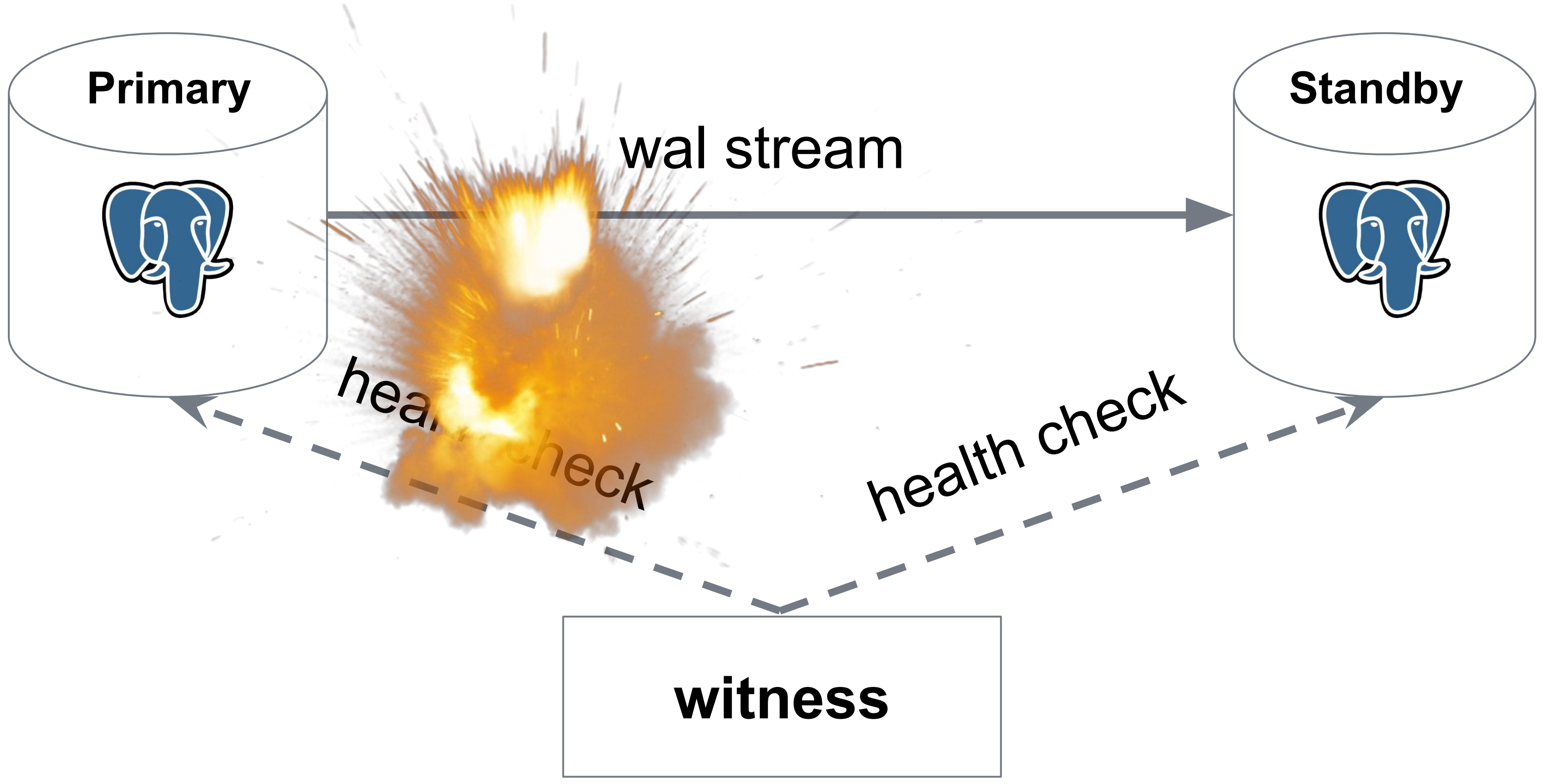
Single witness node as a point of failure



Single witness node w/o fencing

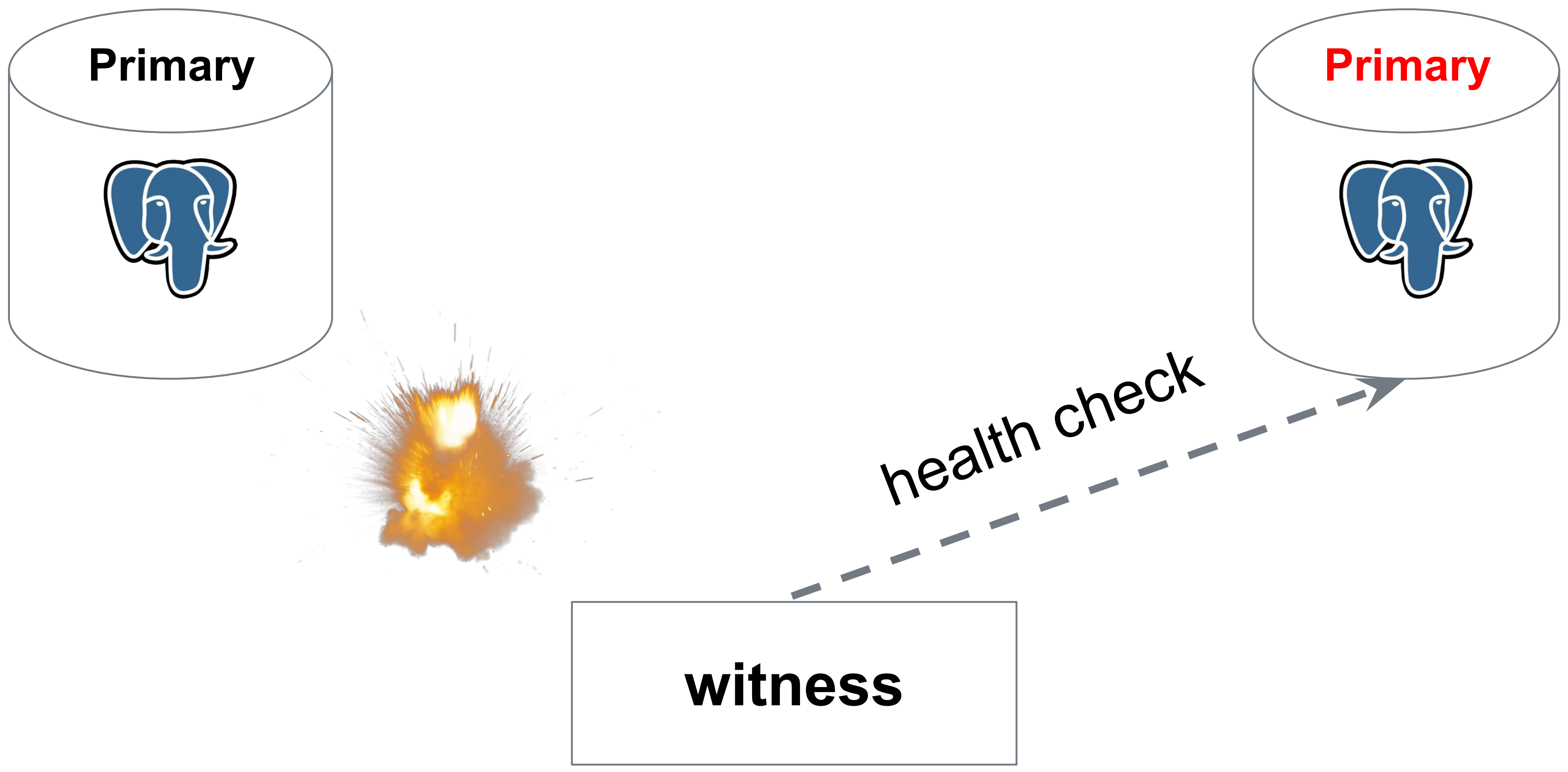


Single witness node without fencing

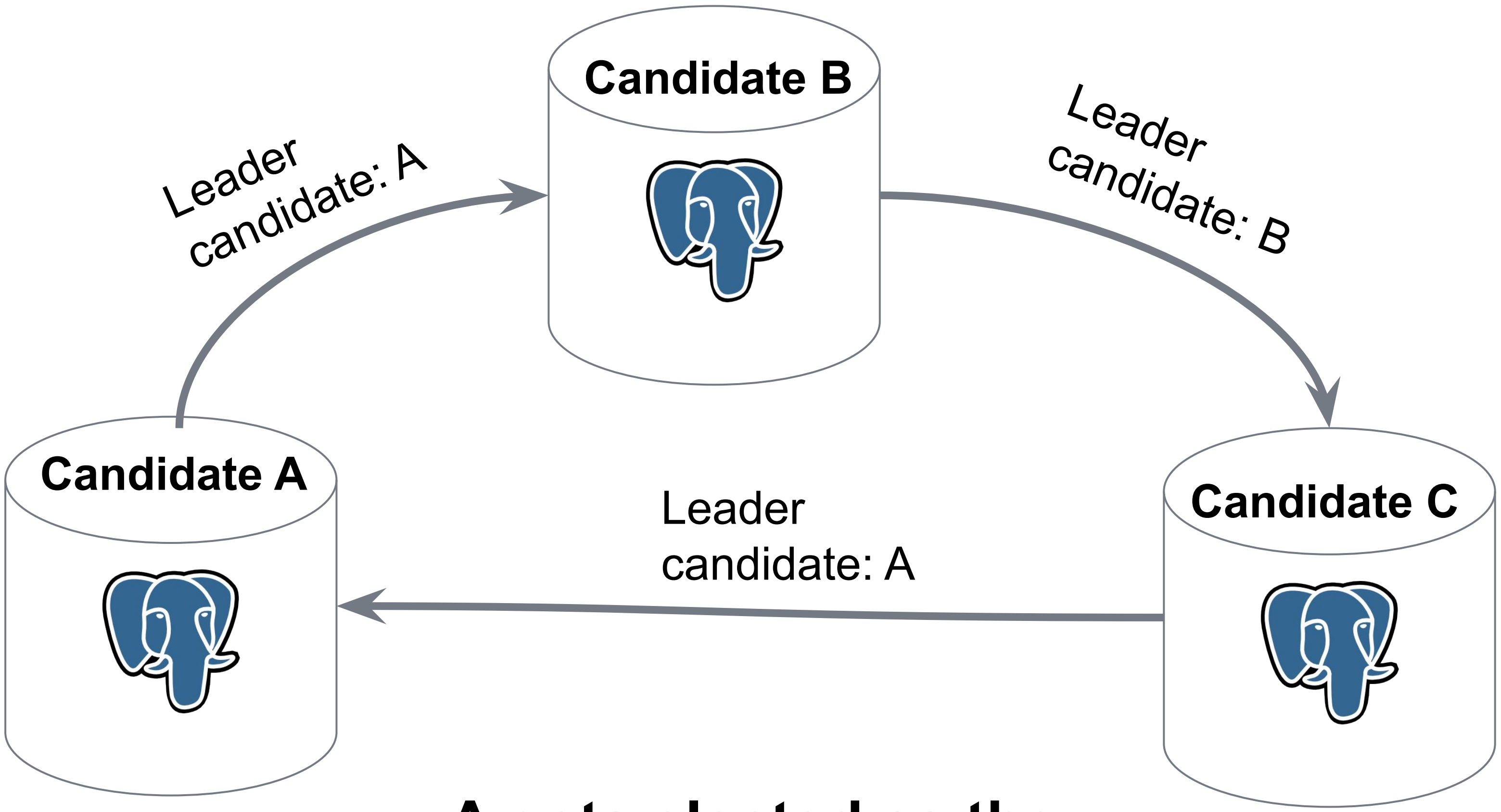


zalando

Single witness node without fencing



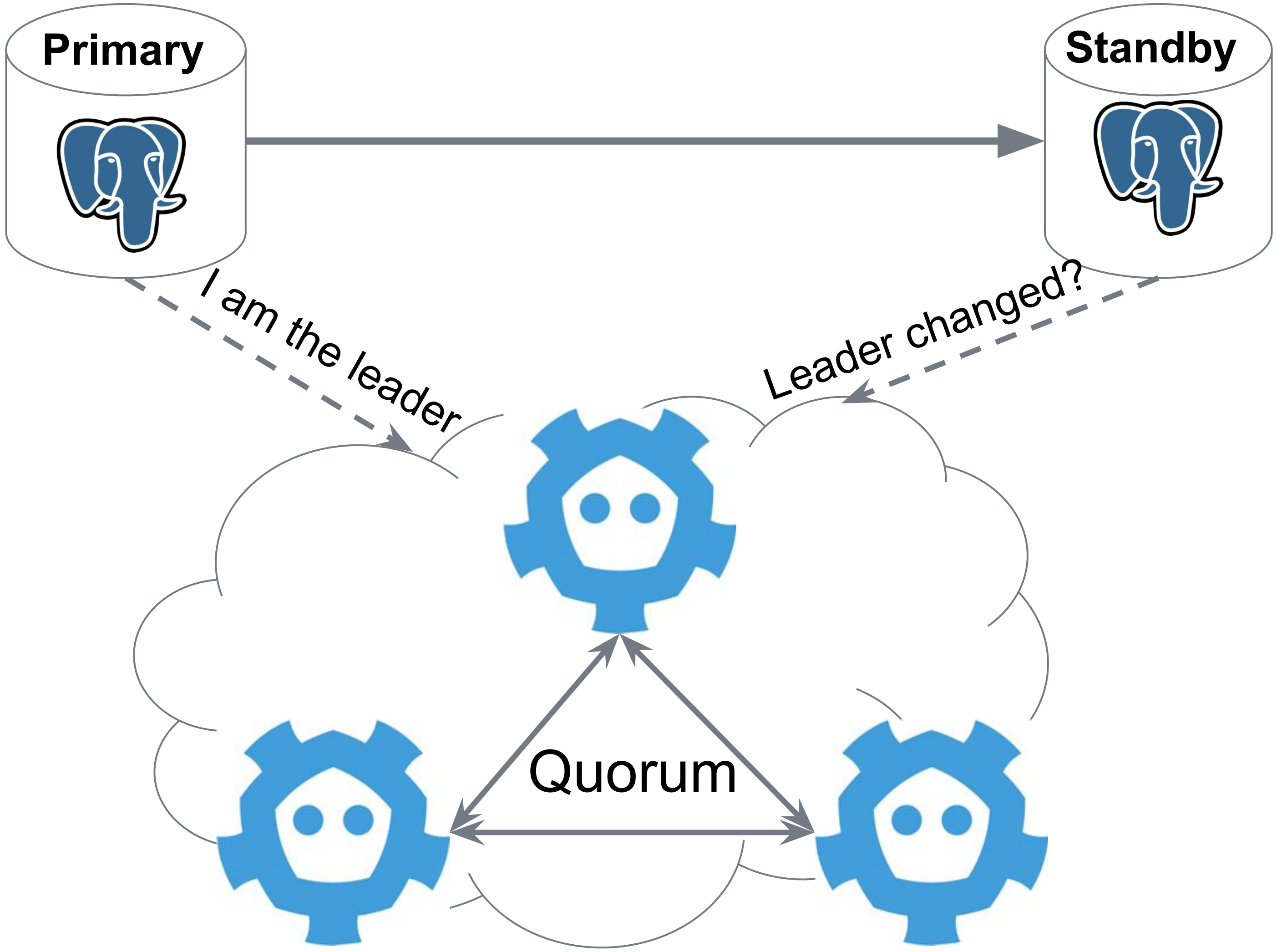
Leader election and quorum



A gets elected as the leader

zalando

Automatic failover: Patroni



Automatic failover: Patroni

- Leader race among healthy members of the cluster
- Each member decides only for itself
- Cluster state stored in a consistent distributed storage
- Leader key changes via atomic CAS operations
- Auto-fencing of non-cooperative or failed nodes

Avoiding split-brain

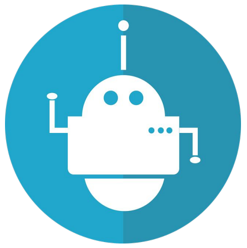
- Only one member can win the leader race
- Demote when leader key cannot be updated
- Patroni can ping the watchdog

DCS supporting Patroni

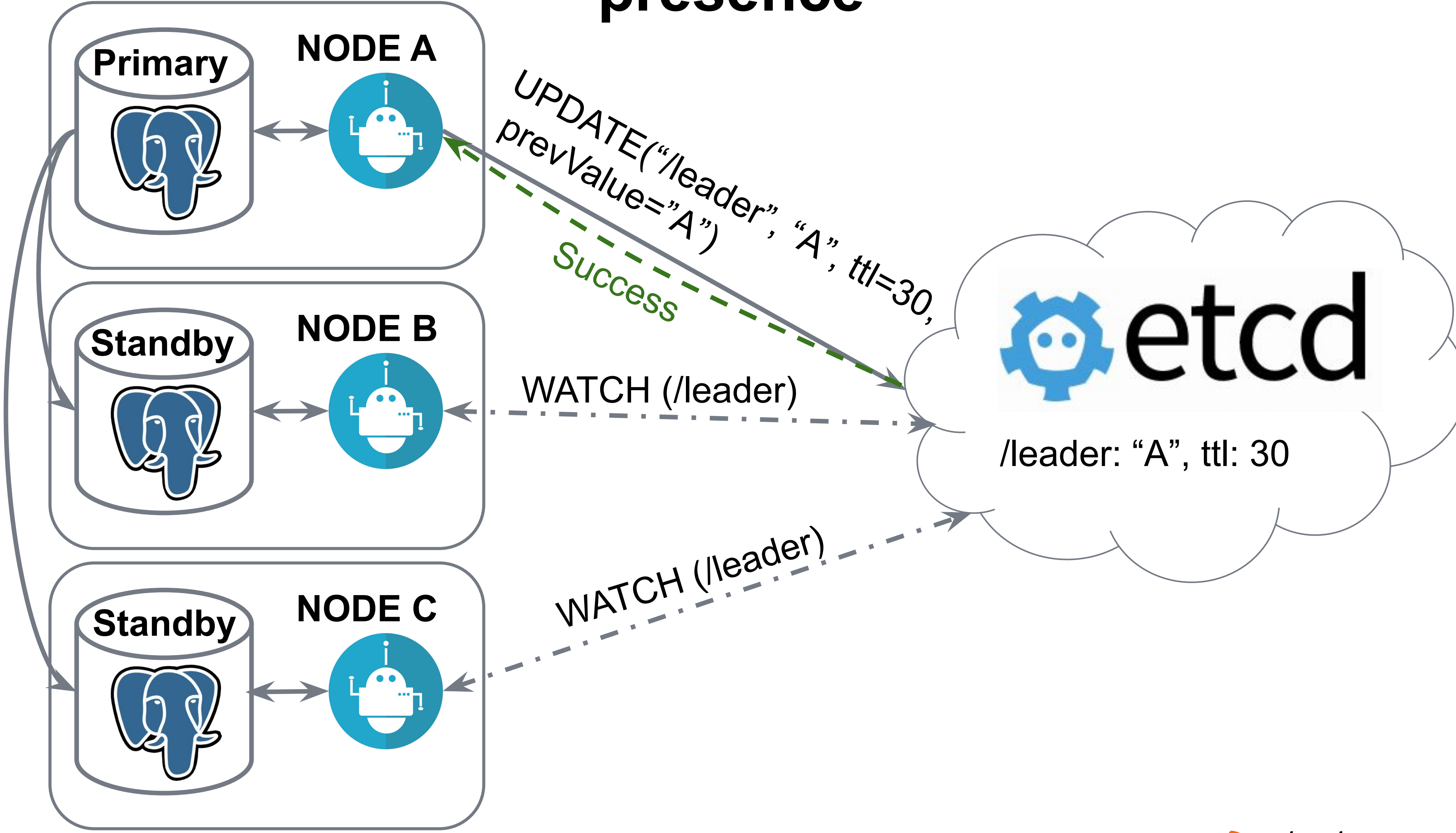
- Strongly-consistent distributed configuration store
- Text and lightweight structure (i.e. JSON)
- Less than 1KB of data per cluster
- Support for auto-expiring keys with TTLs and watches
- Examples: Etcd, Consul, Zookeeper.
- Derivatives of those (i.e. Kubernetes objects)

Bot pattern

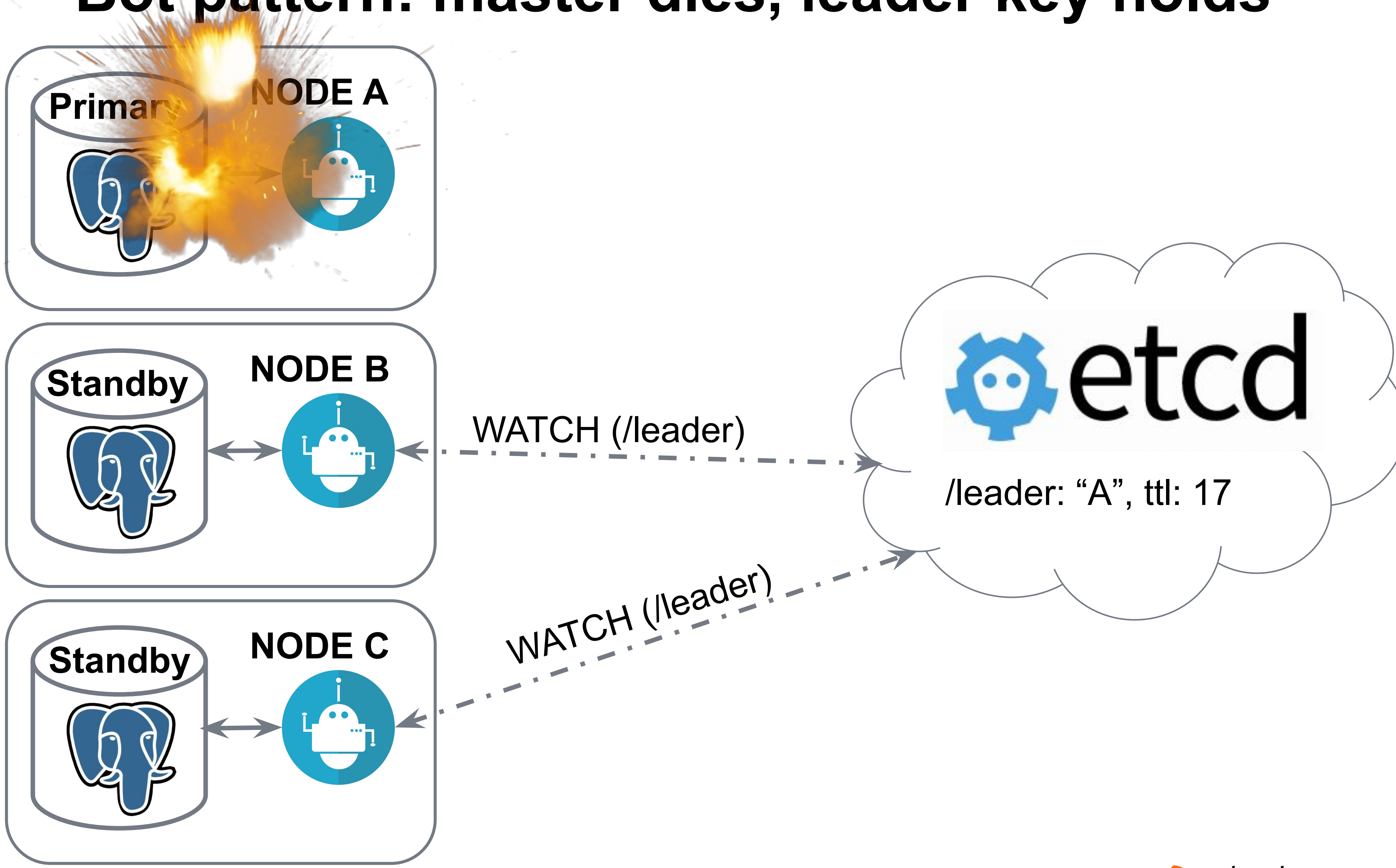
- PostgreSQL cannot talk to DCS (i.e Etcd) directly
- Let’s run a separate process alongside Postgres:
    - to manage PostgreSQL
    - to talk to DCS
    - to decide on promotion/demotion



Bot pattern: master acknowledges its presence

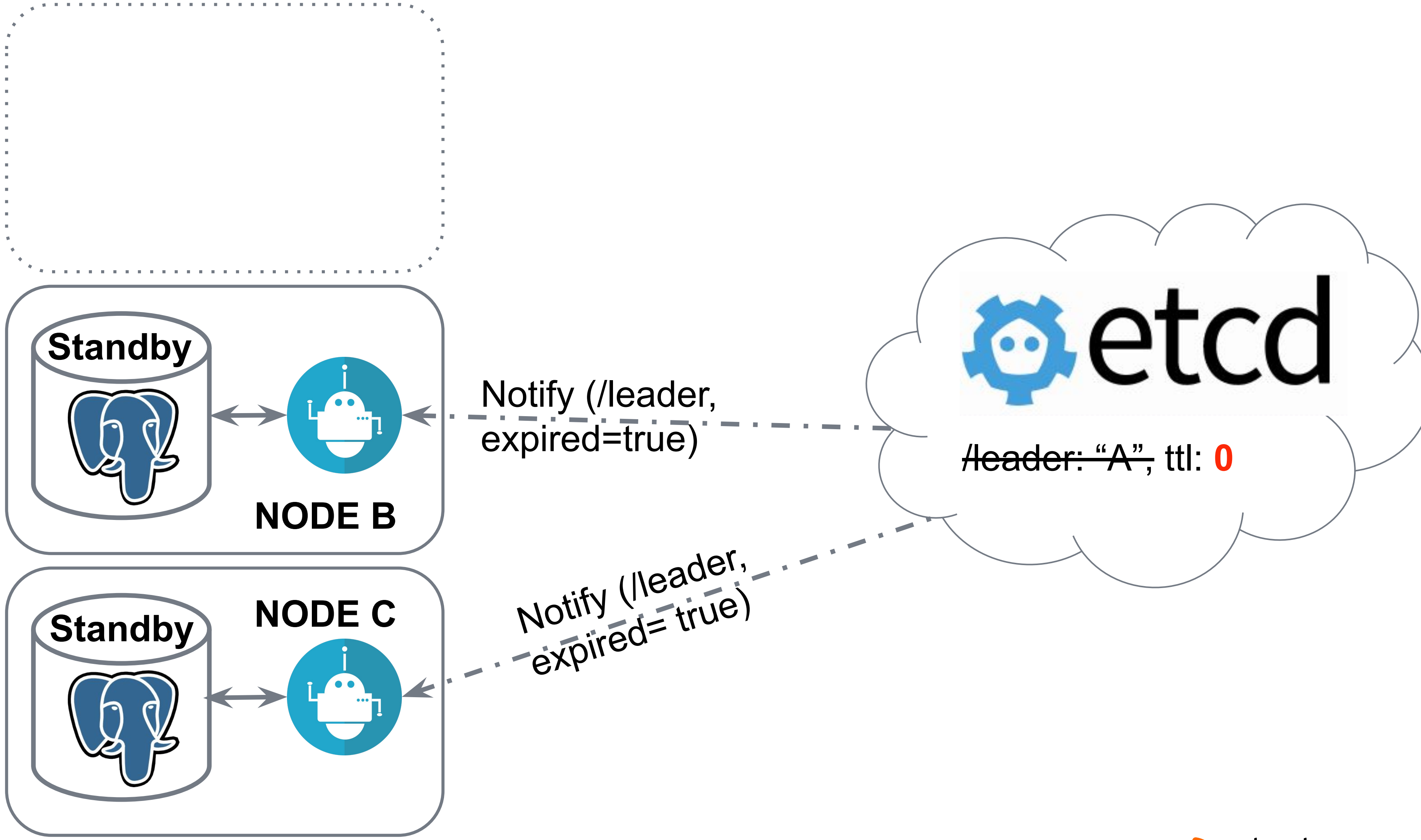


Bot pattern: master dies, leader key holds

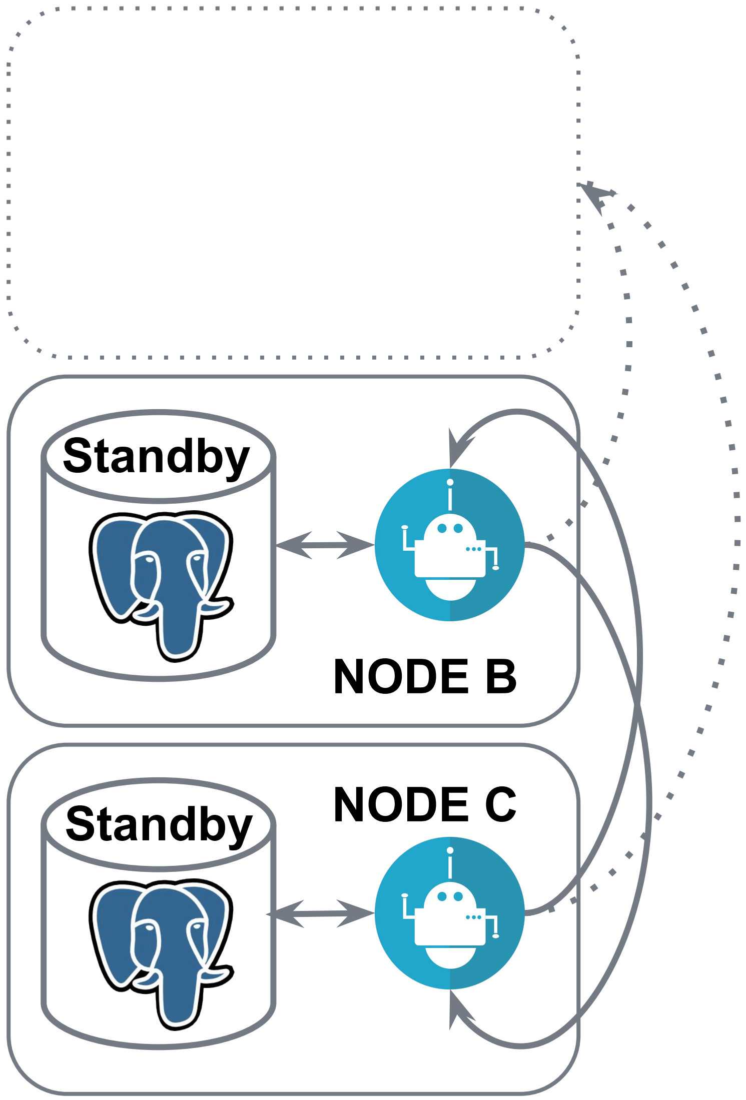


zalando

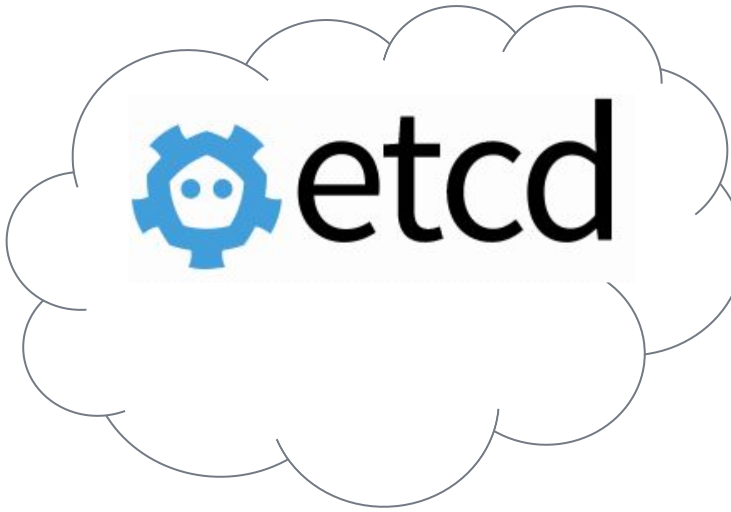
Bot pattern: leader key expires


Standby
NODE B
Notify (/leader, expired=true)
~~/leader: "A",~~ ttl: 0
etcd
Standby
NODE C
Notify (/leader, expired= true)

Bot pattern: who will be the next master?

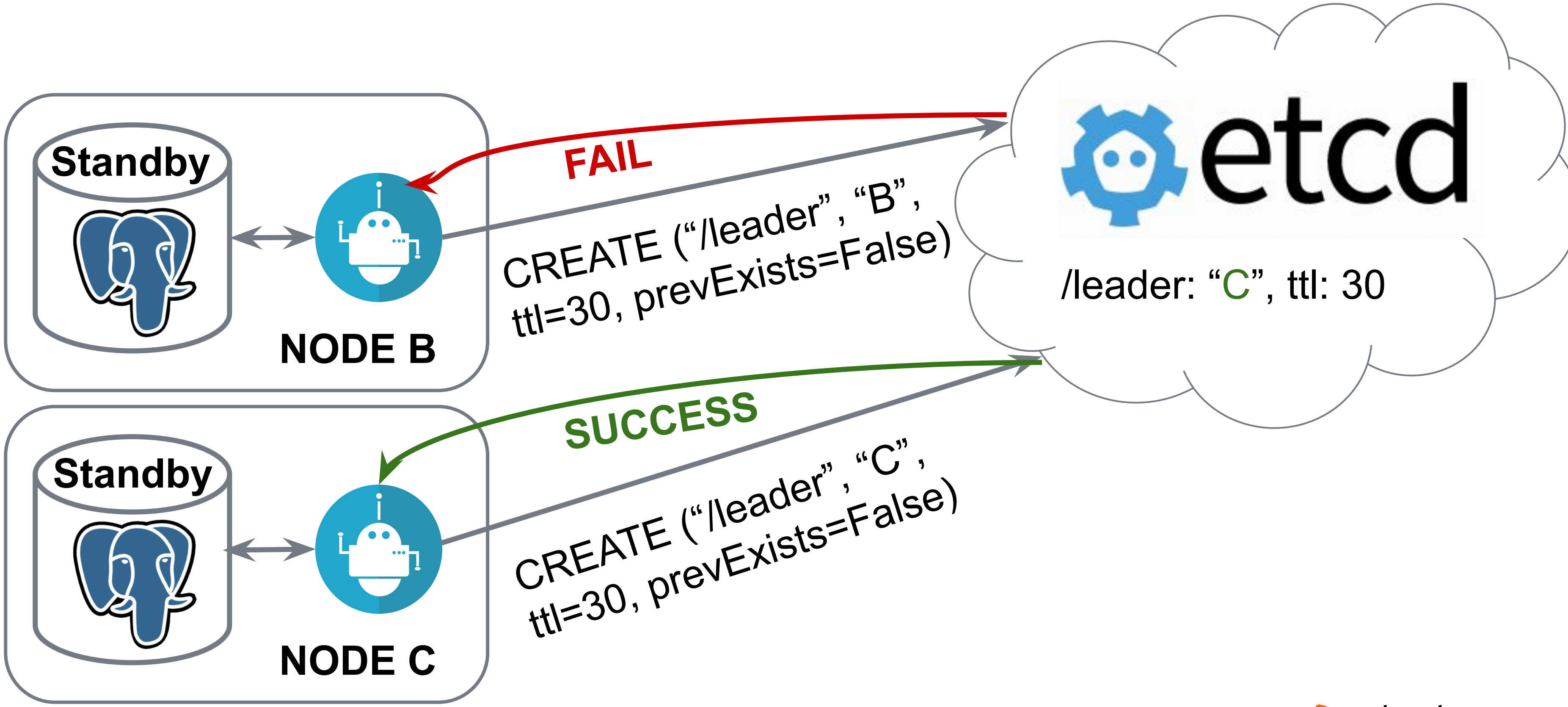


Node **B**:
GET A:8008/patroni -> **timeout**
GET C:8008/patroni -> wal_position: **100**

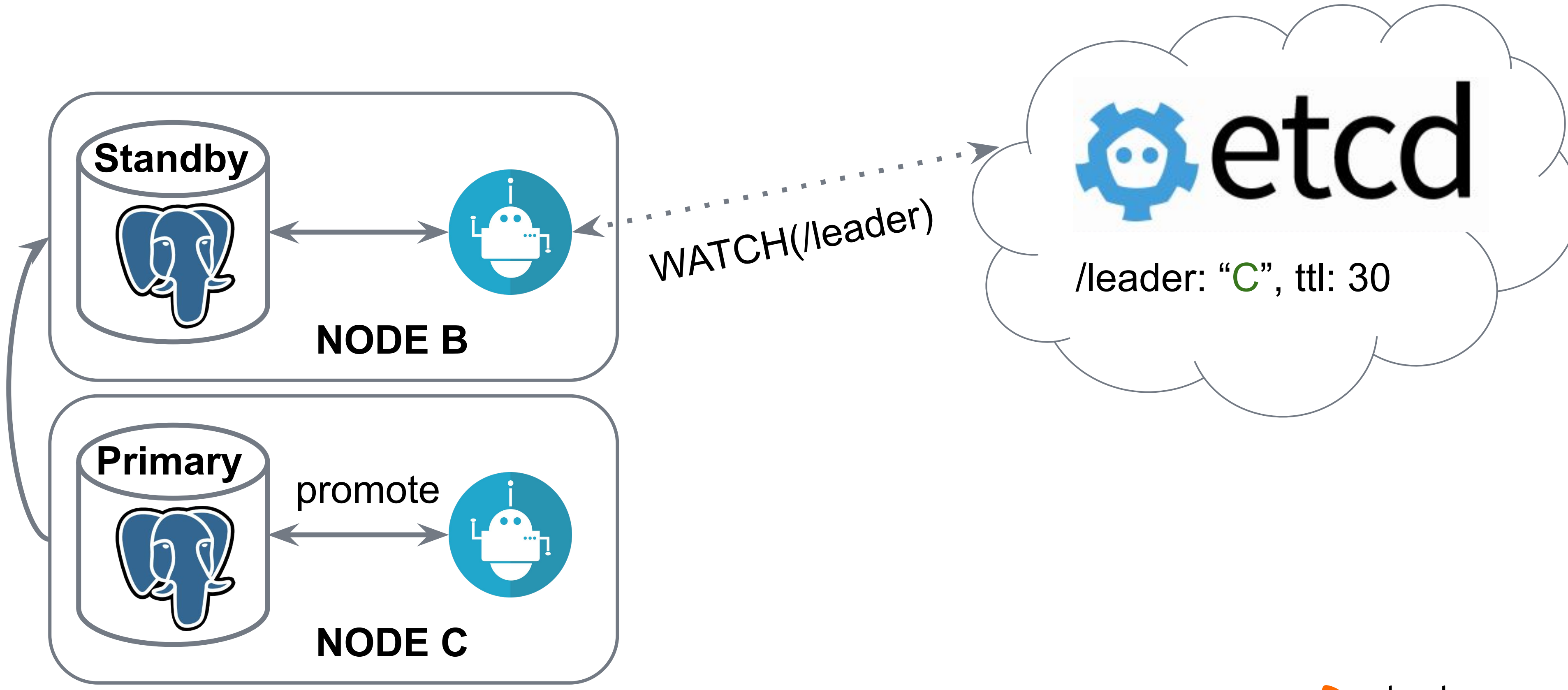


Node **C**:
GET A:8008/patroni -> **timeout**
GET B:8008/patroni -> wal_position: **100**

Bot pattern: leader race among equals


Standby
NODE B
FAIL
CREATE ("/leader", "B", ttl=30, prevExists=False)
etcd
/leader: "C", ttl: 30
SUCCESS
Standby
NODE C
CREATE ("/leader", "C", ttl=30, prevExists=False)

Bot pattern: promote and continue replication

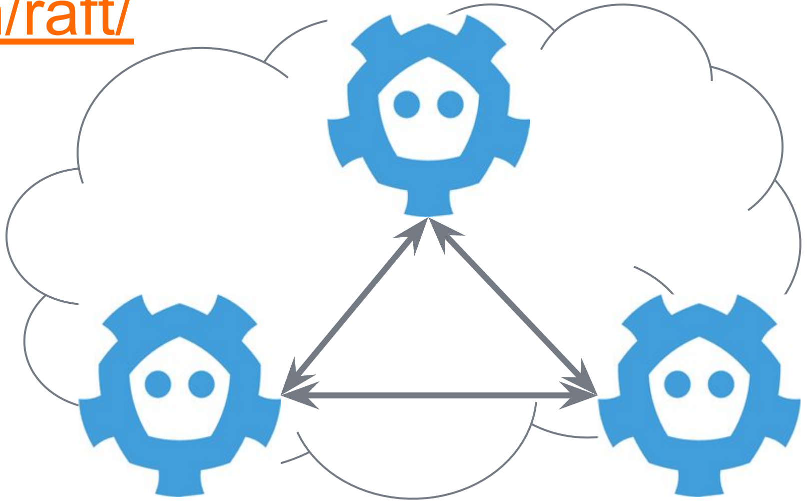


zalando

Etcd consistency store

- Distributed key-value store
- Implements RAFT
- Needs more than 2 nodes (optimal: odd number)

http://thesecretlivesofdata.com/raft/



zalando

Patroni

- Patroni implements bot pattern in Python
- Official successor of Compose Governor
- Developed in the open by Zalando and volunteers all over the world

https://github.com/zalando/patroni

zalando

Your First Patroni cluster

Training setup

- install docker and docker-compose
- Download docker-compose.yml from https://github.com/patroni-training/2019
- run [sudo] docker-compose up -d
- docker ps # list running containers
- docker logs -f container-name # check container logs
- docker exec -ti container-name bash # “ssh” into the container

Hands on: creating your first cluster with Patroni

```
$ docker logs demo-patroni1

2019-03-07 13:29:06,714 INFO: Selected new
etcd server http://127.0.0.1:2379
2019-03-07 13:29:06,731 INFO: Lock owner:
None; I am patroni1
2019-03-07 13:29:06,796 INFO: trying to
bootstrap a new cluster
…
Success. You can now start the database
server using:
    /usr/lib/postgresql/10/bin/pg_ctl -D
data/patroni1 -l logfile start
2019-03-07 13:29:13,115 INFO: initialized a
new cluster
2019-03-07 13:29:23,088 INFO: Lock owner:
patroni1; I am patroni1
2019-03-07 13:29:23,143 INFO: no action.  i
am the leader with the lock
```

```
$ docker logs demo-patroni2

2019-03-07 13:45:02,479 INFO: Selected new
etcd server http://127.0.0.1:2379
2019-03-07 13:45:02,488 INFO: Lock owner:
patroni1; I am patroni2
2019-03-07 13:45:02,499 INFO: trying to
bootstrap from leader 'patroni1'
2019-03-07 13:45:04,470 INFO: replica has
been created using basebackup
2019-03-07 13:45:04,474 INFO: bootstrapped
from leader 'patroni1'
2019-03-07 13:45:07,211 INFO: Lock owner:
patroni1; I am patroni2
2019-03-07 13:45:07,212 INFO: does not
have lock
2019-03-07 13:45:07,440 INFO: no action.
i am a secondary and i am following a
leader
```

Patronictl output on success

```
$ docker exec -ti demo-haproxy bash
postgres@haproxy:~$ patronictl list

+---------+----------+------------+--------+---------+----+-----------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB |
+---------+----------+------------+--------+---------+----+-----------+
|   demo  | patroni1 | 172.18.0.6 | Leader | running |  1 |         0 |
|   demo  | patroni2 | 172.18.0.7 |        | running |  1 |         0 |
|   demo  | patroni3 | 172.18.0.8 |        | running |  1 |         0 |
+---------+----------+------------+--------+---------+----+-----------+
```

Automatic failover

Failover happens when primary dies abruptly

Let's kill the docker container with the primary

```
$ docker logs -f demo-patroni2 # separate
$ docker logs -f demo-patroni3 # consoles

$ docker kill demo-patroni1
```

Replica promotion

```
2019-03-07 12:03:43,651 INFO: does not have lock
2019-03-07 12:03:43,666 INFO: no action.  i am a secondary and i am following a
leader
2019-03-07 12:03:52.758 UTC [36] FATAL:  could not receive data from WAL stream:
server closed the connection unexpectedly
         This probably means the server terminated abnormally
         before or while processing the request.

2019-03-07 12:04:13,884 INFO: Got response from patroni3
http://172.18.0.8:8008/patroni: b'{"timeline": 1, "cluster_unlocked": true,
"role": "replica", "patroni": {"scope": "demo", "version": "1.5.5"}, "state":
"running", "xlog": {"replayed_timestamp": null, "received_location": 67108960,
"replayed_location": 67108960}}'

2019-03-07 12:04:15,876 WARNING: Request failed to patroni1: GET
http://172.18.0.6:8008/patroni (HTTPConnectionPool(host='172.18.0.6', port=8008):
Max retries exceeded with url: /patroni (Caused by
ConnectTimeoutError(<requests.packages.urllib3.connection.HTTPConnection object
at 0x7fb2b4f39dd8>, 'Connection to 172.18.0.6 timed out. (connect timeout=2)')))

2019-03-07 12:04:15,897 INFO: promoted self to leader by acquiring session lock
server promoting
2019-03-07 12:04:15.913 UTC [29] LOG:  selected new timeline ID: 2
2019-03-07 12:04:16,924 INFO: Lock owner: patroni2; I am patroni2
2019-03-07 12:04:16,952 INFO: no action.  i am the leader with the lock
```

Patronictl output

```
postgres@haproxy:~$ patronictl list
+---------+----------+-------------+--------+---------+----+-----------+
| Cluster |  Member  |     Host    |  Role  |  State  | TL | Lag in MB |
+---------+----------+-------------+--------+---------+----+-----------+
|   demo  | patroni2 | 172.18.0.7  | Leader | running |  2 |         0 |
|   demo  | patroni3 | 172.18.0.8  |        | running |  2 |         0 |
+---------+----------+-------------+--------+---------+----+-----------+
```

Start the former master

```
$ docker start demo-patroni1
demo-patroni1
$ docker logs -f demo-patroni1

2019-03-07 12:14:33,823 INFO: Selected new etcd server http://etcd3:2379
2019-03-07 12:14:33,846 WARNING: Postgresql is not running.
2019-03-07 12:14:33,864 INFO: doing crash recovery in a single user mode
2019-03-07 12:14:34,111 WARNING: Postgresql is not running.
2019-03-07 12:14:34,146 INFO: running pg_rewind from patroni2
servers diverged at WAL location 0/4000060 on timeline 1
rewinding from last common checkpoint at 0/20001B0 on timeline 1
Done!
2019-03-07 12:14:34,760 INFO: starting as a secondary
2019-03-07 12:14:35,862 ERROR: postmaster is not running
2019-03-07 12:14:36,942 INFO: Lock owner: patroni2; I am patroni1
2019-03-07 12:14:36,954 INFO: starting as a secondary
2019-03-07 12:14:37.153 UTC [45] LOG:  entering standby mode
2019-03-07 12:14:37.171 UTC [51] LOG:  started streaming WAL from primary at
0/4000000 on timeline 2
2019-03-07 12:14:37.175 UTC [43] LOG:  database system is ready to accept read
only connections
localhost:5432 - accepting connections
2019-03-07 12:14:38,066 INFO: Lock owner: patroni2; I am patroni1
2019-03-07 12:14:38,066 INFO: does not have lock
```

Former master has joined the cluster

```
postgres@haproxy:~$ patronictl list
+---------+----------+------------+--------+---------+----+-----------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB |
+---------+----------+------------+--------+---------+----+-----------+
|   demo  | patroni1 | 172.18.0.6 |        | running |  2 |         0 |
|   demo  | patroni2 | 172.18.0.7 | Leader | running |  2 |         0 |
|   demo  | patroni3 | 172.18.0.8 |        | running |  2 |         0 |
+---------+----------+------------+--------+---------+----+-----------+
```

Peek into etcd

```
postgres@haproxy:~$ etcdctl ls --recursive --sort  /service/demo

/service/demo/config
/service/demo/history
/service/demo/initialize
/service/demo/leader
/service/demo/members
/service/demo/members/patroni1
/service/demo/members/patroni2
/service/demo/members/patroni3
/service/demo/optime
/service/demo/optime/leader

postgres@haproxy:~$ etcdctl get  /service/demo/leader
patroni2

postgres@haproxy:~$ etcdctl get /service/demo/members/patroni2
{"timeline":2,"role":"master","xlog_location":67253384,"state":"running","conn_url":"po
stgres://172.18.0.7:5432/postgres","api_url":"http://172.18.0.7:8008/patroni"}

postgres@haproxy:~$ etcdctl get /service/demo/history
[[1,67108960,"no recovery target specified","2019-03-07T12:04:15+00:00"]]
```

Etcd failure

```
$ docker kill demo-etcd1 demo-etcd2
$ docker logs demo-patroni1

2019-03-07 13:27:35,179 INFO: no action.  i am the leader with the lock
2019-03-07 13:27:45,152 INFO: Lock owner: patroni1; I am patroni1
2019-03-07 13:27:46,825 WARNING: Retrying (Retry(total=0, connect=None, read=None,
redirect=0)) after connection broken by
'ReadTimeoutError("HTTPConnectionPool(host='172.18.0.2', port=2379): Read timed out.
(read timeout=1.6666666666666667)",)': /v2/keys/service/demo/leader
2019-03-07 13:27:48,494 ERROR: Request to server http://172.18.0.2:2379 failed:
MaxRetryError('HTTPConnectionPool(host=\'172.18.0.2\', port=2379): Max retries exceeded
with url: /v2/keys/service/demo/leader (Caused by
ReadTimeoutError("HTTPConnectionPool(host=\'172.18.0.2\', port=2379): Read timed out.
(read timeout=1.6666666666666667)",))',)
2019-03-07 13:27:48,494 INFO: Reconnection allowed, looking for another server.
2019-03-07 13:27:54,695 ERROR: Machines cache is empty, no machines to try.
2019-03-07 13:27:54,698 ERROR: failed to update leader lock
2019-03-07 13:27:54.718 UTC [309] LOG:  received immediate shutdown request
2019-03-07 13:28:04,789 INFO: demoted self because failed to update leader lock in DCS
2019-03-07 13:28:04,791 INFO: closed patroni connection to the postgresql cluster
2019-03-07 13:28:04,792 WARNING: Loop time exceeded, rescheduling immediately.
```

Etcd failure (continue)

```
$ docker logs demo-patroni1

. . .

2019-03-07 13:28:04,796 INFO: Selected new etcd server http://etcd3:2379
2019-03-07 13:28:04,806 INFO: Lock owner: patroni1; I am patroni1
2019-03-07 13:28:04,865 INFO: postmaster pid=901
2019-03-07 13:28:04.909 UTC [904] FATAL:  the database system is starting up
2019-03-07 13:28:05.118 UTC [903] WARNING:  recovery command file "recovery.conf"
specified neither primary_conninfo nor restore_command
2019-03-07 13:28:05.118 UTC [903] LOG:  entering standby mode
2019-03-07 13:28:05.130 UTC [901] LOG:  database system is ready to accept read only
connections

postgres@haproxy:~$ etcdctl -o extended get /service/demo/leader
Key: /service/demo/leader
Created-Index: 484
Modified-Index: 825
TTL: -472
Index: 828

patroni1
```

Etcd recovery

```
$ docker start demo-etcd1 demo-etcd2
$ docker logs -f demo-patroni1

2019-03-07 13:36:55,674 ERROR: failed to update leader lock
2019-03-07 13:36:55,678 INFO: Selected new etcd server http://etcd3:2379
2019-03-07 13:36:56,174 INFO: not promoting because failed to update leader lock in DCS
2019-03-07 13:36:56,177 WARNING: Loop time exceeded, rescheduling immediately.
2019-03-07 13:36:56,182 INFO: Lock owner: patroni1; I am patroni1
2019-03-07 13:36:56,199 INFO: promoted self to leader because i had the session lock
2019-03-07 13:36:56,211 INFO: Lock owner: patroni1; I am patroni1
server promoting
2019-03-07 13:36:56,215 INFO: cleared rewind state after becoming the leader
2019-03-07 13:36:56.224 UTC [903] LOG:  received promote request
2019-03-07 13:36:56.224 UTC [903] LOG:  redo done at 0/4023530
2019-03-07 13:36:56.233 UTC [903] LOG:  selected new timeline ID: 4
2019-03-07 13:36:56,281 INFO: updated leader lock during promote
2019-03-07 13:36:56.313 UTC [903] LOG:  archive recovery complete
2019-03-07 13:36:56.356 UTC [901] LOG:  database system is ready to accept connections
2019-03-07 13:36:57,264 INFO: Lock owner: patroni1; I am patroni1
```

Routing connections from clients

- Typically via a middleware (proxy/connection pooler)
- Discover and adopt to the new cluster topology
- Not only connections to the primary
  - Replicas for balancing read-only queries
  - Only synchronous replicas to avoid anomalies
  - Exclude some replicas to minimize replication lag

zalando

Routing connections from clients

- REST API http status codes:
  - /master - {200: master, 503: replica}
  - /replica - {503: master, 200: replica}

- Callbacks:
  - on_start, on_stop, on_restart, on_role_change,

- Using information from DCS (i.e. confd, vip-manager)

- JDBC:
`jdbc:postgresql://node1,node2,node3/postgres?targetServerType=master`

- libpq starting from PostgreSQL 10:
`postgresql://host1:port2,host2:port2/?target_session_attrs=read-write`

- Consul services

HAProxy template for confd

```
postgres@haproxy:/etc/confd/templates$ cat haproxy.tmpl
global
    maxconn 100

listen master
    bind *:5000
    option httpchk OPTIONS /master
    http-check expect status 200
    default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
{{range gets "/members/*"}}    server {{base .Key}} {{$data := json
.Value}}{{base (replace (index (split $data.conn_url "/") 2) "@" "/" -1)}}
maxconn 100 check port {{index (split (index (split $data.api_url "/") 2) ":")
1}}
{{end}}

listen replicas
    bind *:5001
    option httpchk OPTIONS /replica
    http-check expect status 200
    default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
{{range gets "/members/*"}}    server {{base .Key}} {{$data := json
.Value}}{{base (replace (index (split $data.conn_url "/") 2) "@" "/" -1)}}
maxconn 100 check port {{index (split (index (split $data.api_url "/") 2) ":")
1}}
{{end}}
```

HAProxy template for confd

```
postgres@haproxy:/etc/confd/templates$ cat /etc/haproxy/haproxy.cfg
global
    maxconn 100

listen master
    bind *:5000
    option httpchk OPTIONS /master
    http-check expect status 200
    default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
    server patroni1 172.18.0.6:5432 maxconn 100 check port 8008
    server patroni2 172.18.0.7:5432 maxconn 100 check port 8008
    server patroni3 172.18.0.8:5432 maxconn 100 check port 8008

listen replicas
    bind *:5001
    option httpchk OPTIONS /replica
    http-check expect status 200
    default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
    server patroni1 172.18.0.6:5432 maxconn 100 check port 8008
    server patroni2 172.18.0.7:5432 maxconn 100 check port 8008
    server patroni3 172.18.0.8:5432 maxconn 100 check port 8008
```

Using callbacks

```
postgresql:
  callbacks:
    on_start: /etc/patroni/callback.sh
    on_stop: /etc/patroni/callback.sh
    on_role_change: /etc/patroni/callback.sh
```

Using callbacks

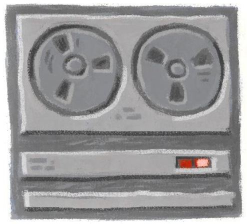
```
readonly cb_name=$1
readonly role=$2
readonly scope=$3
function usage() { echo "Usage: $0 <on_start|on_stop|on_role_change> <role> <scope>";
exit 1; }
case $cb_name in
   on_stop )
       remove_service_ip
       ;;
   on_start|on_role_change )
       [[ $role == 'master' ]] && add_service_ip || remove_service_ip
       ;;
   * )
       usage
       ;;
esac
```

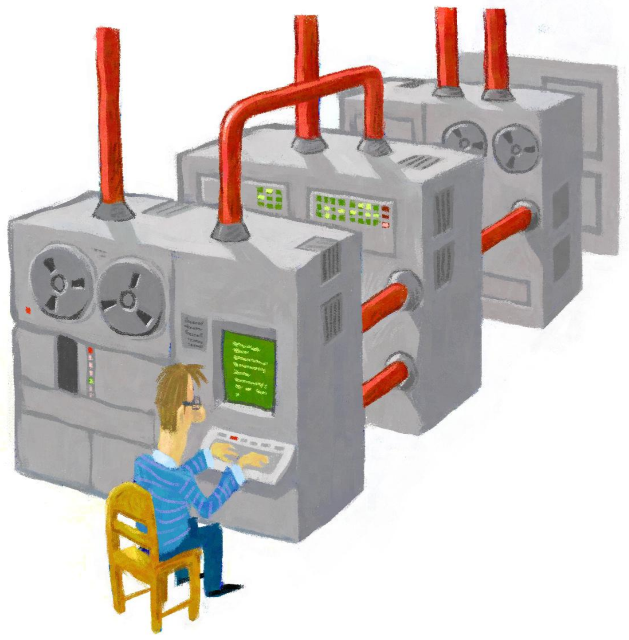
Using callbacks

Callbacks are executed asynchronously after successfully completing the actions that trigger them.

Beware of race conditions.
See https://github.com/zalando/patroni/issues/536 for more details

zalando





Let’s edit some configuration

Editing configuration with patronictl

```
postgres@haproxy:$ patronictl -c postgres0.yml edit-config

"/tmp/demo-config-lgtn6lbe.yaml" 8L, 146C written
---
+++
@@ -3,6 +3,7 @@
 postgresql:
   parameters:
     max_connections: 100
+    work_mem: 8MB
   use_pg_rewind: true
 retry_timeout: 10
 ttl: 30

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

zalando

Editing configuration with patronictl

```
2019-03-07 14:19:06,352 INFO: Lock owner: patroni2; I am patroni1
2019-03-07 14:19:06,352 INFO: does not have lock
2019-03-07 14:19:06,360 INFO: no action.  i am a secondary and i am
following a leader
2019-03-07 14:19:16,355 INFO: Lock owner: patroni2; I am patroni1
2019-03-07 14:19:16,355 INFO: does not have lock
2019-03-07 14:19:16,368 INFO: no action.  i am a secondary and i am
following a leader
server signaled
2019-03-07 14:19:16.451 CET [28996] LOG:  received SIGHUP, reloading
configuration files
2019-03-07 14:19:16.461 CET [28996] LOG:  parameter "work_mem" changed to
"8MB"
2019-03-07 14:19:26,357 INFO: Lock owner: patroni2; I am patroni1
2019-03-07 14:19:26,357 INFO: does not have lock
2019-03-07 14:19:26,365 INFO: no action.  i am a secondary and i am
following a leader
```

Editing configuration with patronictl

```
$ patronictl edit-config

"/tmp/demo-config-lgtn6lbe.yaml" 8L, 146C written
---
+++
@@ -2,7 +2,8 @@
 maximum_lag_on_failover: 1048576
 postgresql:
   parameters:
-    max_connections: 100
+    max_connections: 101
     work_mem: 8MB
   use_pg_rewind: true
 retry_timeout: 10
 ttl: 30

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

zalando

Editing configuration with patronictl

```
postgres@haproxy:~$ patronictl list

+---------+----------+------------+--------+---------+----+-----------+-----------------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB | Pending restart |
+---------+----------+------------+--------+---------+----+-----------+-----------------+
|   demo  | patroni1 | 172.18.0.6 | Leader | running |  4 |         0 |        *        |
|   demo  | patroni2 | 172.18.0.7 |        | running |  4 |         0 |        *        |
|   demo  | patroni3 | 172.18.0.8 |        | running |  4 |         0 |        *        |
+---------+----------+------------+--------+---------+----+-----------+-----------------+
```

Editing configuration with patronictl

```
postgres@haproxy:~$ curl -s 172.18.0.7:8008 | jq .
{
 "timeline": 4,
 "database_system_identifier": "6665617408057626651",
 "xlog": {
   "replayed_location": 67253880,
   "paused": false,
   "received_location": 67253880,
   "replayed_timestamp": null
 },
 "cluster_unlocked": false,
 "pending_restart": true,
 "role": "replica",
 "state": "running",
 "server_version": 100007,
 "postmaster_start_time": "2019-03-07 12:21:36.171 UTC",
 "patroni": {"scope": "demo","version": "1.5.5"}
}
```

Editing configuration with patronictl

```
postgres@haproxy:~$ patronictl restart demo patroni2
+---------+----------+------------+--------+---------+----+-----------+-----------------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB | Pending restart |
+---------+----------+------------+--------+---------+----+-----------+-----------------+
|   demo  | patroni1 | 172.18.0.6 | Leader | running |  4 |         0 |        *        |
|   demo  | patroni2 | 172.18.0.7 |        | running |  4 |         0 |        *        |
|   demo  | patroni3 | 172.18.0.8 |        | running |  4 |         0 |        *        |
+---------+----------+------------+--------+---------+----+-----------+-----------------+
Are you sure you want to restart members patroni2? [y/N]: y
Restart if the PostgreSQL version is less than provided (e.g. 9.5.2)  []:
When should the restart take place (e.g. 2015-10-01T14:30)  [now]:
Success: restart on member patroni2

postgres@haproxy:~$ patronictl list
+---------+----------+------------+--------+---------+----+-----------+-----------------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB | Pending restart |
+---------+----------+------------+--------+---------+----+-----------+-----------------+
|   demo  | patroni1 | 172.18.0.6 | Leader | running |  4 |         0 |        *        |
|   demo  | patroni2 | 172.18.0.7 |        | running |  4 |         0 |                 |
|   demo  | patroni3 | 172.18.0.8 |        | running |  4 |         0 |        *        |
+---------+----------+------------+--------+---------+----+-----------+-----------------+
```

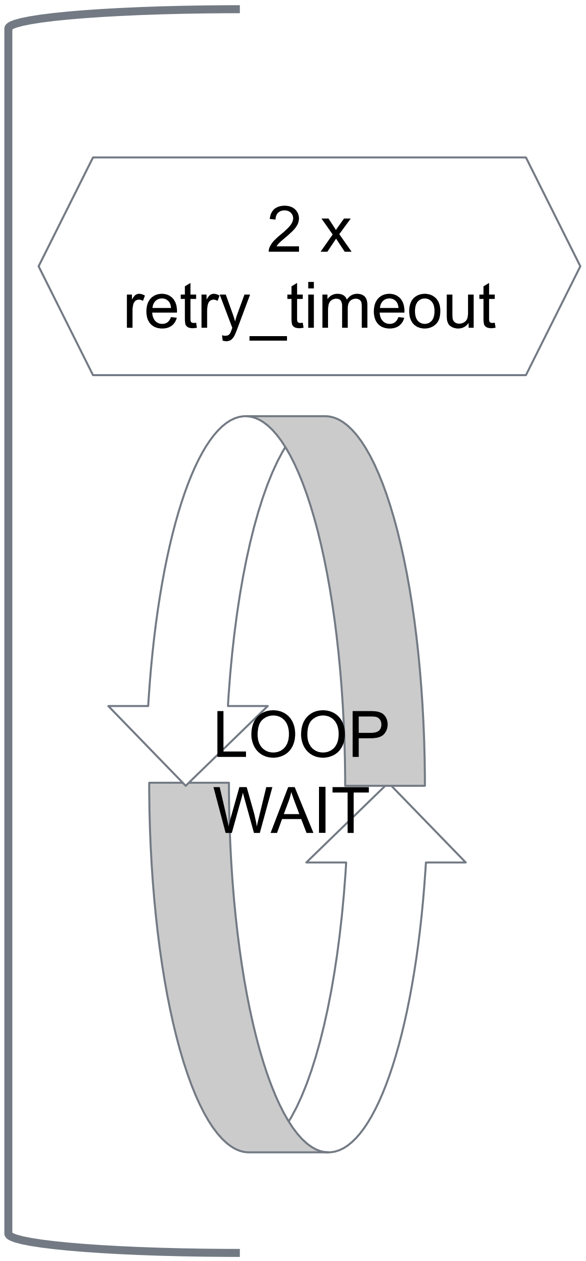
zalando

Editing configuration with patronictl

```
postgres@patroni1:~$ psql -tqA \
   -c "SHOW max_connections"
101

…
postgres@patroni2:~$ psql -tqA \
   -c "SHOW max_connections"
100
```

```
ttl > loop_wait +
retry_timeout * 2 +
[small gap]
```

TTL



Changing TTL, loop_wait, retry_timeout

- **ttl**
  - bigger - less “false-positives” due to temporary leader unavailability
  - smaller - faster detection of leader failure

- **loop_wait**
  - bigger - less CPU load
  - smaller - faster reaction to external events, i.e. appearance of new members, synchronous replica selection

- **retry_timeout**
  - smaller - faster detection of connectivity issues
  - bigger - less false positives due to intermittent connectivity failures.

Changing TTL, loop_wait, retry_timeout

`ttl >= loop_wait + retry_timeout * 2`

```
$ patronictl edit-config
---
+++
@@ -1,9 +1,9 @@
-loop_wait: 10
+loop_wait: 5
 maximum_lag_on_failover: 1048576
 postgresql:
   parameters:
     work_mem: 8MB
     max_connections: 101
   use_pg_rewind: true
-retry_timeout: 10
+retry_timeout: 27
-ttl: 30
+ttl: 60
```

Changing TTL, loop_wait, retry_timeout

```
2019-03-07 14:31:06,350 INFO: Lock owner: patroni2; I am patroni2
2019-03-07 14:31:06,364 INFO: no action.  i am the leader with the lock
2019-03-07 14:31:16,349 INFO: Lock owner: patroni2; I am patroni2
2019-03-07 14:31:16,362 INFO: no action.  i am the leader with the lock
2019-03-07 14:31:16,376 INFO: Lock owner: patroni2; I am patroni2
2019-03-07 14:31:16,392 INFO: no action.  i am the leader with the lock
2019-03-07 14:31:21,377 INFO: Lock owner: patroni2; I am patroni2
2019-03-07 14:31:21,392 INFO: no action.  i am the leader with the lock
2019-03-07 14:31:26,381 INFO: Lock owner: patroni2; I am patroni2
2019-03-07 14:31:26,396 INFO: no action.  i am the leader with the lock
```

zalando

Changing TTL, loop_wait, retry_timeout

ttl < loop_wait + retry_timeout * 2

```
postgres@haproxy:~$ patronictl edit-config

---
+++
@@ -1,4 +1,4 @@
-loop_wait: 5
+loop_wait: 10
 maximum_lag_on_failover: 1048576
 postgresql:
   parameters:
@@ -6,4 +6,4 @@
     max_connections: 101
   use_pg_rewind: true
 retry_timeout: 27
-ttl: 60
+ttl: 5
```

Changing TTL, loop_wait, retry_timeout

ttl < loop_wait + retry_timeout * 2

```
2019-03-07 14:35:46,390 INFO: no action.  i am the leader with the
lock
2019-03-07 14:35:46,405 INFO: Lock owner: patroni2; I am patroni2
2019-03-07 14:35:46,408 WARNING: Watchdog not supported because
leader TTL 5 is less than 2x loop_wait 10
2019-03-07 14:35:46,418 INFO: no action.  i am the leader with the
lock
2019-03-07 14:35:56,418 WARNING: Watchdog not supported because
leader TTL 5 is less than 2x loop_wait 10
2019-03-07 14:35:56,428 INFO: acquired session lock as a leader
2019-03-07 14:36:06,420 WARNING: Watchdog not supported because
leader TTL 5 is less than 2x loop_wait 10
2019-03-07 14:36:06,430 INFO: acquired session lock as a leader
```

Changing TTL, loop_wait, retry_timeout

ttl < loop_wait + retry_timeout * 2

```
2019-03-07 14:35:46,426 INFO: Lock owner: patroni2; I am patroni1
2019-03-07 14:35:46,426 INFO: does not have lock
2019-03-07 14:35:46,429 INFO: no action.  i am a secondary and i am
following a leader
2019-03-07 14:35:51,594 INFO: Got response from patroni2
http://172.18.0.7:8008/patroni: b'{"state": "running",
"postmaster_start_time": "2019-03-07 13:44:44.764 CET", "role": "master",
"server_version": 100007, "xlog": {"location": 50331968}, "timeline": 2,
"replication": [{"usename": "replicator", "application_name": "patroni2",
"client_addr": "172.18.0.6", "state": "streaming", "sync_state": "async",
"sync_priority": 0}], "database_system_identifier": "6512366775019348050",
"pending_restart": true, "patroni": {"version": "1.5.5", "scope": "demo"}}'
2019-03-07 14:35:51,680 WARNING: Master (patroni2) is still alive
2019-03-07 14:35:51,683 INFO: following a different leader because i am not
the healthiest node
```

Change it back to original values

```
postgres@haproxy:~$ patronictl edit-config

---
+++
@@ -11,5 +11,5 @@
     work_mem: 8MB
     max_connections: 101
   use_pg_rewind: true
-retry_timeout: 27
+retry_timeout: 10
-ttl: 5
+ttl: 30
```

Dynamic configuration

- Unified, stored in DCS
- Keep member's configuration identical
- Change configuration with REST API
- Apply to multiple nodes at once, auto reload
- Figure out if restart is necessary

Cluster-wide and local configuration

```
etcd: /config ->  {"postgresql":{"parameters":{"work_mem":"16MB"}}}

patroni.yaml:     postgresql:
                      parameters:
                          work_mem: 12MB

postgresql.conf   # Do not edit this file manually!
                  # It will be overwritten by Patroni!
                  include 'postgresql.base.conf'
                  work_mem = '12MB'

ALTER SYSTEM SET  work_mem TO '24MB';
$ psql -c "show work_mem"
 work_mem
----------
 24MB
(1 row)
```

Cluster-wide and local configuration

1. Patroni takes the contents of the **/config** key from DCS.
2. Most of the parameters can be redefined locally in the patroni.yaml **postgresql:** section. It allows to set parameters for this specific instance. One can use it to configure Patroni and PostgreSQL correctly on nodes that doesn’t have the same hardware specification.
3. ALTER SYSTEM SET overrides values set on the previous 2 steps. It is not recommended, since Patroni will not be aware of that changes and, for example, will not set the **pending_restart** flag.

Some argument, for instance, max_connections, max_locks_per_transaction, wal_level, max_wal_senders, max_prepared_transactions, max_replication_slots, max_worker_processes **cannot be redefined locally.**

Cluster-wide and local configuration

```
bootstrap: # is used only one-time, when the cluster is created
  dcs: # written to DCS /config on successful bootstrap,
       # applied on all nodes
    loop_wait: 5
    postgresql:
      max_connections: 142
```

Changing the bootstrap section in the Patroni configuration takes no effect once the cluster has been bootstrapped.

REST API and monitoring

REST API endpoints

GET /master or GET /

GET /replica

GET /patroni

GET, PUT, PATCH /config

POST /switchover, POST /failover

POST /restart, POST /reload

POST /reinitialize

GET /patroni is used by Patroni during failover in order to check if the master is running and compare the node's own WAL position with the one from other nodes.

GET /patroni on the master

```
postgres@patroni1:~$ curl -s http://localhost:8008/patroni | jq .
{
 "database_system_identifier": "6666696040414781469",
 "patroni": {"scope": "demo", "version": "1.5.5"},
 "timeline": 1,
 "xlog": {"location": 67108960},
 "replication": [
   {
     "application_name": "patroni2", "client_addr": "172.21.0.5",
     "sync_state": "async", "state": "streaming",
     "sync_priority": 0, "usename": "replicator"
   },
   {

     "application_name": "patroni3", "client_addr": "172.21.0.6",
     "sync_state": "async", "state": "streaming",
     "sync_priority": 0, "usename": "replicator"
   }
 ],
 "role": "master",
 "postmaster_start_time": "2019-03-10 09:45:33.770 UTC",
 "server_version": 100007,
 "state": "running",
 "cluster_unlocked": false
}
```

GET /patroni on the replica

```
postgres@patroni2:~$ curl -s http://localhost:8008/patroni | jq .
{
 "postmaster_start_time": "2019-03-10 09:45:45.483 UTC",
 "timeline": 1,
 "role": "replica",
 "patroni": {"scope": "demo", "version": "1.5.5"},
 "server_version": 100007,
 "database_system_identifier": "6666960404147814 69",
 "state": "running",
 "cluster_unlocked": false,
 "xlog": {
   "received_location": 67109184,
   "replayed_timestamp": null,
   "paused": false,
   "replayed_location": 67109184
 }
}
```

Monitoring PostgreSQL health

- PostgreSQL master is running
  - GET /master should return 200 for one and only one node

- PostgreSQL replicas are streaming
  - GET /patroni from the master should return replication: [{state: streaming} for all replica nodes]

- PostgreSQL is running
  - GET /patroni should return state:running for every node in the cluster

- PostgreSQL replicas is not lagging
  - GET /patroni received and replayed location on every replica should not be behind a certain threshold from the GET /patroni xlog: location from the master

Patroni API does not provide a way to discover all PostgreSQL nodes. This can be achieved by looking directly into the DCS, or using some features of the cloud provider (i.e. AWS labels, see https://github.com/zalando/patroni/blob/master/patroni/scripts/aws.py).

Using tags to modify behavior of individual nodes

- **nofailover** (true/false) - disable failover/switchover to the given node (node will not become a master)

- **noloadbalance** (true/false) - /replica always returns code 503

- **clonefrom** (true/false) - node adds itself to the list of origins for initializing new replicas. When at least one replica has this tag, cloning will always be performed from that replica if PostgreSQL is running there. When multiple replicas has it - the cloning origin is chosen randomly among one of them.

- **nosync** (true/false) - node will never become a synchronous replica

- **replicatefrom** (node name) - specify a node to replicate from. This can be used to implement a cascading replication. If the node is not suitable (doesn’t exist or not running PostgreSQL), the master will be chosen instead.

*** Tags are configured on every node individually**

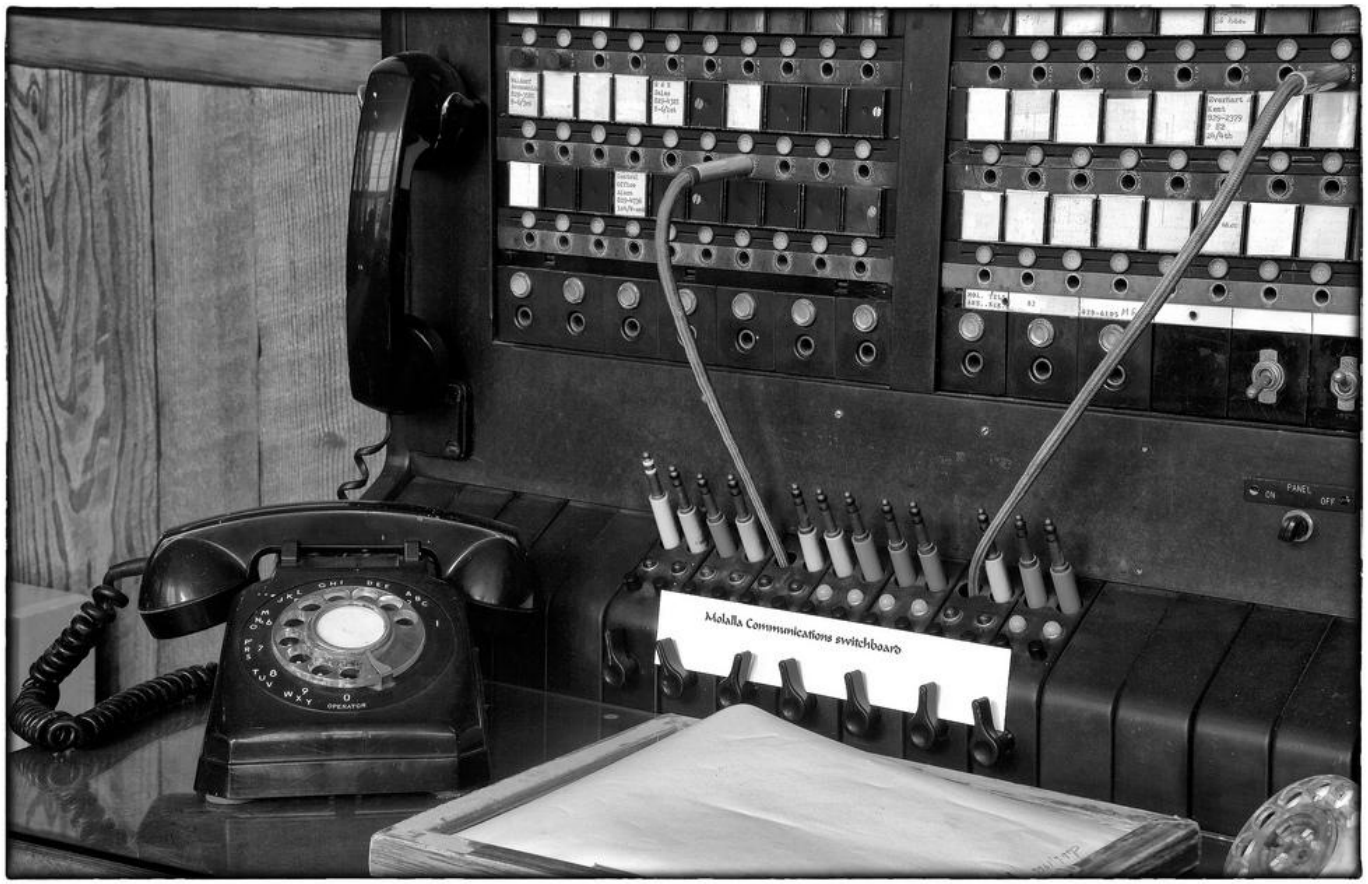
zalando

Using replicatefrom to create cascading replication

Use tags on the new node to create a cascading streaming replica.

HINT: look at postgres2.yml

Switchover, failover and more



Switchover and failover

- **Failover**: emergency promotion of a given node
  - automatic, when no leader is present in the cluster
  - manual, when automatic failover is not present or cannot decide on the new master

- **Switchover**: switch of the master role to a new node. Requires the presence of the master.

Switchover with patronictl

```
$ docker exec -ti demo-haproxy bash
postgres@haproxy:~$ patronictl switchover demo
Master [patroni2]:
Candidate ['patroni1', 'patroni3'] []: patroni1
When should the switchover take place (e.g. 2015-10-01T14:30)  [now]:
Current cluster topology
+---------+----------+------------+--------+---------+----+-----------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB |
+---------+----------+------------+--------+---------+----+-----------+
|   demo  | patroni1 | 172.21.0.5 |        | running |  1 |         0 |
|   demo  | patroni2 | 172.21.0.2 | Leader | running |  1 |         0 |
|   demo  | patroni3 | 172.21.0.4 |        | running |  1 |         0 |
+---------+----------+------------+--------+---------+----+-----------+
Are you sure you want to switchover cluster demo, demoting current master
patroni2? [y/N]: y
2019-03-08 11:49:50.23144 Successfully switched over to "patroni1"
```

Switchover with patronictl (continue)

```
postgres@haproxy:~$ patronictl list
+---------+----------+------------+--------+---------+----+-----------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB |
+---------+----------+------------+--------+---------+----+-----------+
|   demo  | patroni1 | 172.21.0.5 | Leader | running |  2 |         0 |
|   demo  | patroni2 | 172.21.0.2 |        | stopped |    |   unknown |
|   demo  | patroni3 | 172.21.0.4 |        | running |    |   unknown |
+---------+----------+------------+--------+---------+----+-----------+


postgres@haproxy:~$ patronictl list
+---------+----------+------------+--------+---------+----+-----------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB |
+---------+----------+------------+--------+---------+----+-----------+
|   demo  | patroni1 | 172.21.0.5 | Leader | running |  2 |         0 |
|   demo  | patroni2 | 172.21.0.2 |        | running |  2 |         0 |
|   demo  | patroni3 | 172.21.0.4 |        | running |  2 |         0 |
+---------+----------+------------+--------+---------+----+-----------+
```

Scheduled switchover

```
postgres@haproxy:~$ patronictl switchover demo
Master [patroni1]:
Candidate ['patroni2', 'patroni3'] []: patroni3
When should the switchover take place (e.g. 2015-10-01T14:30)  [now]: 2019-03-08T13:00
Current cluster topology
+---------+----------+------------+--------+---------+----+-----------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB |
+---------+----------+------------+--------+---------+----+-----------+
|   demo  | patroni1 | 172.21.0.5 | Leader | running |  2 |         0 |
|   demo  | patroni2 | 172.21.0.2 |        | running |  2 |         0 |
|   demo  | patroni3 | 172.21.0.4 |        | running |  2 |         0 |
+---------+----------+------------+--------+---------+----+-----------+
Are you sure you want to switchover cluster demo, demoting current master patroni1? [y/N]:
y
2019-03-08 11:55:41.95879 Switchover scheduled
+---------+----------+------------+--------+---------+----+-----------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB |
+---------+----------+------------+--------+---------+----+-----------+
|   demo  | patroni1 | 172.21.0.5 | Leader | running |  2 |         0 |
|   demo  | patroni2 | 172.21.0.2 |        | running |  2 |         0 |
|   demo  | patroni3 | 172.21.0.4 |        | running |  2 |         0 |
+---------+----------+------------+--------+---------+----+-----------+
Switchover scheduled at: 2019-03-08T12:00:00+00:00
                   from: patroni1
                     to: patroni3
```

Scheduled switchover

```
$ docker logs -f demo-patroni1

2019-03-08 11:58:41,989 INFO: no action.  i am the leader with the lock
2019-03-08 11:58:49,923 INFO: received switchover request with leader=patroni1
candidate=patroni3 scheduled_at=2019-03-08T12:00:00+00:00
2019-03-08 11:58:49,942 INFO: Lock owner: patroni1; I am patroni1
2019-03-08 11:58:49,955 INFO: Awaiting failover at 2019-03-08T12:00:00+00:00 (in 70 seconds)
2019-03-08 11:58:49,965 INFO: no action.  i am the leader with the lock
2019-03-08 11:58:59,943 INFO: Lock owner: patroni1; I am patroni1
. . .

2019-03-08 11:59:49,956 INFO: Awaiting failover at 2019-03-08T12:00:00+00:00 (in 10 seconds)
2019-03-08 11:59:49,971 INFO: no action.  i am the leader with the lock
2019-03-08 11:59:59,943 INFO: Lock owner: patroni1; I am patroni1
2019-03-08 12:00:00,000 INFO: Manual scheduled failover at 2019-03-08T12:00:00+00:00
2019-03-08 12:00:00,031 INFO: Got response from patroni3 http://172.21.0.4:8008/patroni:
b'{"database_system_identifier": "6665985748268707870", "postmaster_start_time": "2019-03-08
11:49:56.998 UTC", "xlog": {"received_location": 67110992, "paused": false, "replayed_loca
tion": 67110992, "replayed_timestamp": null}, "cluster_unlocked": false, "role": "replica",
"server_version": 100007, "state": "running", "patroni": {"version": "1.5.5", "scope": "demo"},
"timeline": 2}'
2019-03-08 12:00:00,125 INFO: manual failover: demoting myself
2019-03-08 12:00:00.264 UTC [29] LOG:  received fast shutdown request
```

Scheduled restarts

```
postgres@haproxy:~$ patronictl restart demo patroni2
+---------+----------+------------+--------+---------+----+-----------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB |
+---------+----------+------------+--------+---------+----+-----------+
|   demo  | patroni1 | 172.21.0.5 |        | running |  3 |         0 |
|   demo  | patroni2 | 172.21.0.2 |        | running |  3 |         0 |
|   demo  | patroni3 | 172.21.0.4 | Leader | running |  3 |         0 |
+---------+----------+------------+--------+---------+----+-----------+
Are you sure you want to restart members patroni2? [y/N]: y
Restart if the PostgreSQL version is less than provided (e.g. 9.5.2)  []:
When should the restart take place (e.g. 2015-10-01T14:30)  [now]:
2019-03-08T12:03
Success: restart scheduled on member patroni2
```

```
postgres@haproxy:~$ patronictl list
+---------+----------+------------+--------+---------+----+-----------+---------------------------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB |     Scheduled restart     |
+---------+----------+------------+--------+---------+----+-----------+---------------------------+
|   demo  | patroni1 | 172.21.0.5 |        | running |  3 |         0 |                           |
|   demo  | patroni2 | 172.21.0.2 |        | running |  3 |         0 | 2019-03-08T12:03:00+00:00 |
|   demo  | patroni3 | 172.21.0.4 | Leader | running |  3 |         0 |                           |
+---------+----------+------------+--------+---------+----+-----------+---------------------------+
```

Scheduled restarts

```
2019-03-08 12:02:41,447 INFO: Awaiting restart at 2019-03-08T12:03:00+00:00 (in 19 seconds)
2019-03-08 12:02:41,467 INFO: no action.  i am a secondary and i am following a leader
2019-03-08 12:02:51,448 INFO: Lock owner: patroni3; I am patroni2
2019-03-08 12:02:51,449 INFO: does not have lock
2019-03-08 12:03:00,007 INFO: Manual scheduled restart at 2019-03-08T12:03:00+00:00
2019-03-08 12:03:00,032 INFO: restart initiated
2019-03-08 12:03:00.142 UTC [532] LOG:  received fast shutdown request
2019-03-08 12:03:00.146 UTC [532] LOG:  aborting any active transactions
2019-03-08 12:03:00.146 UTC [540] FATAL:  terminating walreceiver process due to administrator
command
. . .
2019-03-08 12:03:00.301 UTC [668] FATAL:  the database system is starting up
2019-03-08 12:03:00.304 UTC [667] LOG:  redo starts at 0/4000A78
2019-03-08 12:03:00.305 UTC [667] LOG:  consistent recovery state reached at 0/4000B58
2019-03-08 12:03:00.305 UTC [667] LOG:  invalid record length at 0/4000B58: wanted 24, got 0
2019-03-08 12:03:00.305 UTC [665] LOG:  database system is ready to accept read only connections
localhost:5432 - accepting connections
2019-03-08 12:03:00.311 UTC [673] LOG:  started streaming WAL from primary at 0/4000000 on
timeline 3
2019-03-08 12:03:00,314 INFO: Lock owner: patroni3; I am patroni2
```

Reinitialize (don’t repeat GitLab mistake)

```
$ docker exec -ti demo-patroni1 bash
postgres@patroni1:~$ patronictl reinit demo patroni1
+---------+----------+------------+--------+---------+----+-----------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB |
+---------+----------+------------+--------+---------+----+-----------+
|   demo  | patroni1 | 172.21.0.5 |        | running |  1 |         0 |
|   demo  | patroni2 | 172.21.0.7 |        | running |  1 |         0 |
|   demo  | patroni3 | 172.21.0.3 | Leader | running |  1 |         0 |
+---------+----------+------------+--------+---------+----+-----------+
Are you sure you want to reinitialize members patroni1? [y/N]: y
Success: reinitialize for member patroni1
postgres@patroni1:~$ patronictl list
+---------+----------+------------+--------+------------------+----+-----------+
| Cluster |  Member  |    Host    |  Role  |      State       | TL | Lag in MB |
+---------+----------+------------+--------+------------------+----+-----------+
|   demo  | patroni1 | 172.21.0.5 |        | creating replica |    |   unknown |
|   demo  | patroni2 | 172.21.0.7 |        |     running      |  1 |         0 |
|   demo  | patroni3 | 172.21.0.3 | Leader |     running      |  1 |         0 |
+---------+----------+------------+--------+------------------+----+-----------+
```

https://about.gitlab.com/2017/02/10/postmortem-of-database-outage-of-january-31/

Pause mode

Pause mode is useful for performing maintenance on the PostgreSQL cluster or DCS.

- The mode is cluster-wide (all nodes or no nodes)
- Takes up to loop_wait seconds for a node to be paused
- Nodes might not be paused simultaneously
- Automatic failover is disabled
- No automatic read-only mode when DCS is not accessible
- PostgreSQL is not shut down when Patroni is stopped
- PostgreSQL is not started automatically when shut down
- PostgreSQL master will update the leader key (or acquire it if it is not taken)

However

- New replicas can be created
- Manual switchover/failover works

Pause mode

```
postgres@haproxy:~$ patronictl pause demo --wait
'pause' request sent, waiting until it is recognized by all nodes
Success: cluster management is paused

postgres@haproxy:~$ patronictl list
+---------+----------+------------+--------+---------+----+-----------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB |
+---------+----------+------------+--------+---------+----+-----------+
|   demo  | patroni1 | 172.21.0.5 |        | running |  1 |         0 |
|   demo  | patroni2 | 172.21.0.7 |        | running |  1 |         0 |
|   demo  | patroni3 | 172.21.0.3 | Leader | running |  1 |         0 |
+---------+----------+------------+--------+---------+----+-----------+
Maintenance mode: on

$ docker logs -f demo-patroni3
2019-03-07 15:51:43,908 INFO: Lock owner: patroni3; I am patroni3
2019-03-07 15:51:43,931 INFO: no action.  i am the leader with the lock
2019-03-07 15:51:46,864 INFO: Lock owner: patroni3; I am patroni3
2019-03-07 15:51:46,890 INFO: PAUSE: no action.  i am the leader with the lock
```

Pause mode (promoting another master)

```
$ docker exec -ti demo-patroni2 bash
postgres@patroni2:~$ pg_ctl promote
waiting for server to promote.... done
server promoted

$ docker logs -f demo-patroni2

2019-03-07 15:54:12.058 CET [81603] LOG:  received promote request
2019-03-07 15:54:12.058 CET [81638] FATAL:  terminating walreceiver process due to
administrator command
2019-03-07 15:54:12.062 CET [81603] LOG:  invalid record length at 0/3000060:
wanted 24, got 0
2019-03-07 15:54:12.062 CET [81603] LOG:  redo done at 0/3000028
2019-03-07 15:54:12.065 CET [81603] LOG:  selected new timeline ID: 2
2019-03-07 15:54:12.113 CET [81603] LOG:  archive recovery complete
2019-03-07 15:54:12.118 CET [81601] LOG:  database system is ready to accept
connections
2019-03-07 15:54:16,872 INFO: Lock owner: patroni3; I am patroni2
2019-03-07 15:54:16,872 INFO: does not have lock
2019-03-07 15:54:16,901 INFO: PAUSE: continue to run as master without lock
```

Pause mode (promoting another master)

```
postgres@patroni2:~$ patronictl list
+---------+----------+------------+--------+---------+----+-----------+
| Cluster |  Member  |    Host    |  Role  |  State  | TL | Lag in MB |
+---------+----------+------------+--------+---------+----+-----------+
|   demo  | patroni1 | 172.21.0.5 |        | running |  1 |         0 |
|   demo  | patroni2 | 172.21.0.7 |        | running |  2 |           |
|   demo  | patroni3 | 172.21.0.3 | Leader | running |  1 |         0 |
+---------+----------+------------+--------+---------+----+-----------+
Maintenance mode: on
```

Pause mode (promoting another master)

```
postgres@patroni2:~$ curl -v http://172.21.0.7:8008/master
< HTTP/1.0 503 Service Unavailable
{
  "timeline": 2,
  "database_system_identifier": "6665930822459834398",
  "cluster_unlocked": false,
  "patroni": {
    "scope": "demo",
    "version": "1.5.5"
  },
  "xlog": {
    "location": 67109456
  },
  "role": "master",
  "postmaster_start_time": "2019-03-08 08:16:16.010 UTC",
  "server_version": 100007,
  "state": "running",
  "pause": true
}
```

zalando

Pause mode (resuming)

```
postgres@haproxy:~$ patronictl resume demo
Success: cluster management is resumed

2019-03-07 15:57:31,324 INFO: Lock owner: patroni3; I am patroni2
2019-03-07 15:57:31,324 INFO: does not have lock
2019-03-07 15:57:31.379 CET [81601] LOG:  received immediate shutdown
request
2019-03-07 15:57:31.380 CET [81720] WARNING:  terminating connection
because of crash of another server process
2019-03-07 15:57:31,805 INFO: Lock owner: patroni3; I am patroni2
2019-03-07 15:57:31,805 INFO: does not have lock
2019-03-07 15:57:32,021 INFO: Local timeline=2 lsn=0/3000170
2019-03-07 15:57:32,030 INFO: master_timeline=1
2019-03-07 15:57:32,158 INFO: running pg_rewind from user=postgres
host=127.0.0.1 port=5432 dbname=postgres sslmode=prefer sslcompression=1
servers diverged at WAL location 0/3000060 on timeline 1
rewinding from last common checkpoint at 0/2000060 on timeline 1
Done!
2019-03-07 15:57:33,560 INFO: Lock owner: patroni3; I am patroni2
2019-03-07 15:57:33,563 INFO: starting as a secondary
```

Synchronous replication

- **synchronous_mode**: true/false
  Cluster-wide settings. Patroni will choose one of the replicas and set it to be the synchronous one. Information about the synchronous replica is kept in DCS. When the master dies patroni fails over only to the synchronous replica (if it exists). Manual failover is possible to a non-synchronous one. If no replica can be set to synchronous - the synchronous replication is disabled, favoring availability over durability.

- **synchronous_mode_strict**: true/false
  Works the same as a synchronous mode, but if no replicas can be set to synchronous - the synchronous mode is retained and the master will not accept any writes (*) until another synchronous replica is available, resulting in no data loss

* - setting synchronous_commit to local or off per transaction will disable that guarantee on a given transaction.

Synchronous replication

```
postgres@haproxy:~$ patronictl edit-config demo
---
+++
@@ -3,5 +3,6 @@
 postgresql:
   parameters: null
   use_pg_rewind: true
+synchronous_mode: true
 retry_timeout: 10
 ttl: 30


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

Synchronous replication

```
2019-03-07 16:33:11,329 INFO: Assigning synchronous standby
status to patroni1
server signaled
2019-03-07 16:33:11.367 CET [81568] LOG:  received SIGHUP,
reloading configuration files
2019-03-07 16:33:11.380 CET [81568] LOG:  parameter
"synchronous_standby_names" changed to "patroni1"
2019-03-07 16:33:13,377 INFO: Synchronous standby status
assigned to patroni1
2019-03-07 16:33:13,385 INFO: no action.  i am the leader with
the lock
2019-03-07 16:33:13.993 CET [83425] LOG:  standby "patroni1" is
now a synchronous standby with priority 1
2019-03-07 16:33:21,312 INFO: Lock owner: patroni1; I am
patroni1
```

Synchronous replication REST endpoints

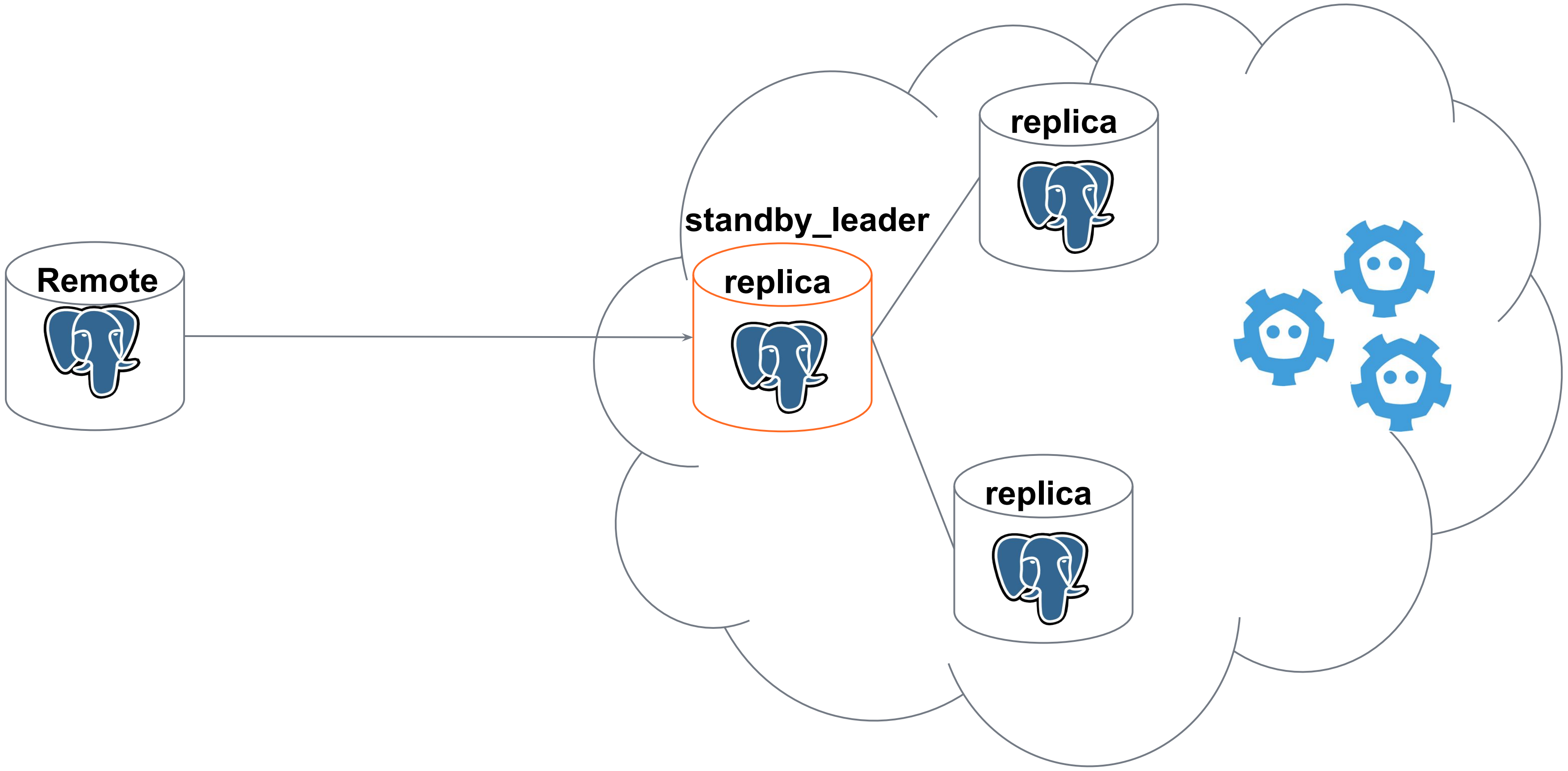
```
postgres@haproxy:~$ patronictl list
+---------+----------+------------+----------------+---------+----+-----------+
| Cluster |  Member  |    Host    |      Role      |  State  | TL | Lag in MB |
+---------+----------+------------+----------------+---------+----+-----------+
|   demo  | patroni1 | 172.21.0.5 | Sync standby   | running |  1 |         0 |
|   demo  | patroni2 | 172.21.0.7 |                | running |  1 |         0 |
|   demo  | patroni3 | 172.21.0.3 |     Leader     | running |  1 |         0 |
+---------+----------+------------+----------------+---------+----+-----------+

$ curl -w '%{http_code}\n' -X OPTIONS http://172.21.0.5:8008/sync
200
$ curl -w '%{http_code}\n' -X OPTIONS http://172.21.0.5:8008/async
503
$ curl -w '%{http_code}\n' -X OPTIONS http://172.21.0.7:8008/sync
503
$ curl -w '%{http_code}\n' -X OPTIONS http://172.21.0.7:8008/async
200
```

zalando

Standby cluster

- Run all Patroni cluster members as standbys
- ‘Standby leader’ replicates from the specified host:port or using restore_command
- ‘Primary cluster’ knows nothing about the standby one.
- Standby cluster can be switched to active by removing the *standby_cluster* parameter from the **/config** key in DCS (`patronictl edit-config`).
- No automatic ‘promotion’ of the standby cluster

Standby cluster



zalando

Permanent replication slots

- Postgres supports physical and logical slots
- Patroni creates slots on the cluster member automatically
    - When **use_slots** is enabled (default)
    - For every replica of the given member
    - All other slots are dropped
- Permanent replication slots to keep a given slot
- Also recreates the slot on failover
    - No client connections until Patroni creates all logical slots

zalando

Standby cluster and permanent slots

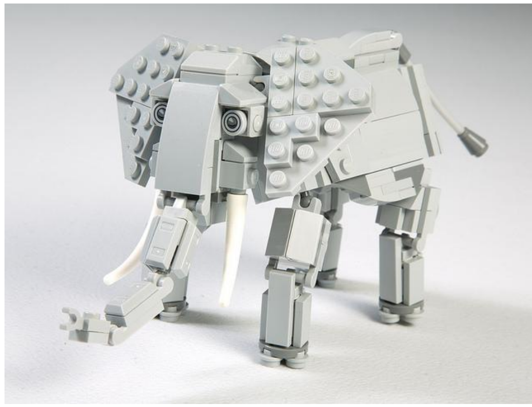
- ‘Primary’ replica streams from an external Postgres node
- Remote node doesn’t create slots automatically
- Slot can be created manually on the remote side
- Slot can be specified in the standby cluster configuration
- When the remote node is part of the active Patroni cluster, the slot for the standby cluster should be configured as permanent.

Standby cluster and permanent slots

```
postgres@haproxy:~$ patronictl edit-config
---
+++
@@ -1,8 +1,16 @@
loop_wait: 10
maximum_lag_on_failover: 1048576
+slots:
+  test_physical:
+    type: physical
+  test_logical:
+    type: logical
+    database: postgres
+    plugin: test_decoding
postgresql:
  parameters:
    max_connections: 100
+    wal_level: logical
  pg_hba:
  - local all all trust
  - host replication all all md5

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

Extensibility

- **Callbacks**
  - client routing and server monitoring
- **Custom replica creation** methods
  - create replicas **in the existing cluster** with methods other than pg_basebackup (i.e wal-e, rsync)
- **Custom bootstrap** methods
  - initialize **first node in the cluster** with a custom script (by default initdb is used)
  - useful to implement PITR or clone existing clusters
- **post_bootstrap** script
  - called after bootstrapping of the new cluster. If they return non-zero - bootstrap is cancelled. One can populate a database or create initial users from that script.



Custom replica creation

```
postgresql:
 create_replica_method:
   - wal_e
   - basebackup
 wal_e:
   command: /bin/wale_restore
   envdir: /etc/env.d/wal-e
   threshold_megabytes: 4096
   threshold_backup_size_percentage: 30
   use_iam: 1
   retries: 2
   no_master: 1
```

Custom replica creation

```
wal_e:
   command: /bin/wale_restore # script to call
   no_master: 1 # whether to call it to
                # initialize the replica w/o
                # the master
   # following arguments are method-specific
   envdir: /etc/env.d/wal-e
   use_iam: 1
   retries: 2
```

Custom replica creation

```
wal_e:
  command: /bin/wale_restore
  no_master: 1
  envdir: /etc/env.d/wal-e
  use_iam: 1
  retries: 2
```

```
# Replica creation command:
 /bin/wale_restore \
--scope=demo \
--datadir=/home/postgres/pgdata \
--role=replica \
--connstring=”postgres://postgres@l
ocalhost:5432/postgres” \
--no_master=1 \
--envdir=/etc/env.d/wal-e \
--use-iam=1 \
--retries=2
```

Custom replica creation

- command is called for new replicas only when the cluster is already present in DCS
- if method defines **no_master** - script will be called even when there is no master (i.e. restore from the WAL archive)
- command must return 0 only on success
- when multiple methods are specified - they are executed one by one until the first successful one, when no success - repeat on the next iteration of the HA loop.
- basebackup is used when no methods are specified, can be added explicitly with `basebackup` method name.

Custom bootstrap

Override default initdb with a custom command to create new cluster. Examples: clone an existing one, recover to a point in time.

zalando

initdb with arguments

```
bootstrap:
  initdb:
  - encoding: UTF8
  - data-checksums
  - auth-host: md5
  - auth-local: trust
```

Custom bootstrap

```
bootstrap:
  method: clone_with_wale
  clone_with_wale:
    command: python3 /clone_with_s3.py --envdir
"/etc/env.d/clone/wal-e"
--recovery-target-time="2019-03-07 00:00:18.349 UTC"
    recovery_conf:
      restore_command: envdir
"/etc/env.d/clone/wal-e" wal-e wal-fetch "%f" "%p"
      recovery_target_timeline: latest
      recovery_target_action: promote
      recovery_target_time: "2019-03-07 00:00:18.349
UTC"
      recovery_target_inclusive: false
```

Custom bootstrap

- only one method allowed (initdb or custom)

- by default initdb is called

- **/initialize** lock is acquired before the method is called
  - only one custom bootstrap script runs at a given time
  - on success Patroni starts PostgreSQL node produced by the script and waits until the node becomes the master (pg_is_in_recovery() == false)

- on failure - the data directory is wiped out and **/initialize** lock is released

- after the successful bootstrap a **post_boostrap** script is called

- if **post_boostrap** script fails - the actions are the same as when the bootstrap fails.

post_bootstrap

```
bootstrap:
  post_bootstrap: /post_bootstrap.sh

$ cat /post_bootstrap.sh
#!/bin/bash
echo "\c template1
CREATE EXTENSION pg_stat_statements;
CREATE ROLE admin;" \
| psql -d $1 # $1 - connection string to the newly created
master.
```

Patroni configuration

```
scope: demo # cluster name, must be the same for all node in the given cluster
#namespace: /service/ # namespace (key prefix) in DCS, default value is /service
name: patroni1 # postgresql node name

log:
  # logging configuration

restapi:
  # restapi configuration

ctl:
  # some configuration for patronictl

etcd:
  # etcd configuration (can also be consul, zoookeeper or kubernetes in
corresponding sections).
```

Patroni configuration

```
bootstrap:
  # configuration applied once during the cluster bootstrap

postgresql:
  # postgres-related node-local configuration

watchdog:
  # how Patroni interacts with the watchdog

tags:
  # map of tags: nofailover, noloadbalance, nosync, replicatefrom, clonefrom
```

Logging configuration

```
log:
# level: INFO  # the logging level. Could be DEBUG, INFO, WARNING, ERROR, CRITICAL
# format: '%(asctime)s %(levelname)s: %(message)s'  # log formatting string
# dateformat: '%Y-%m-%d %H:%M:%S'  # date formatting string
# dir: /where/to/write/patron/logs  # if not specified, write logs to stderr
# file_size: 50000000  # 50MB maximum size of the log file before rotate
# file_num: 10  # keep history of 10 files
# loggers:  # This section allows redefining logging level per python module
#   patroni.postmaster: WARNING
#   urllib3: DEBUG
```

Restapi & Ctl configuration

```
restapi:
  listen: 0.0.0.0:8008 # address to listen to for REST API requests
  connect_address: 172.21.0.6:8008 # address to connect to this node from other
                                  # nodes, also stored in DCS
# certfile: /etc/ssl/certs/ssl-cert-snakeoil.pem  # certificate for SSL connection
# keyfile: /etc/ssl/private/ssl-cert-snakeoil.key # keyfile for SSL connection
# authentication:                # username and password for basic auth.
#   username: admin              # Used for all data modifying operations
#   password: secret             # (POST, PATCH, PUT)

ctl:
# insecure: false  # Do not validate restapi certificates on members
# the following two parameters providing you a mean to validate member certificates
# certfile: /etc/ssl/certs/ssl-cert-snakeoil.pem
# cacert: /etc/ssl/private/ssl-cacert-snakeoil.pem
```

Etcd configuration

```
etcd:
  host:          127.0.0.1:2379
# hosts:        ['etcd1:2379', 'etcd2:2379', 'etcd3:2379']
# use_proxies: false
# protocol:     http
# username:     etcd
# password:     v4rY$ecRetW0rd
# cacert:       /etc/ssl/ca.crt
# cert:         /etc/ssl/cert.crt
# key:          /etc/ssl/key.key
# url:           http://user:passwd@host:port
# proxy:         http://user:passwd@host:port
```

zalando

Consul configuration

```
consul:
  host:        127.0.0.1:8500
  checks:      []  # avoid using the default serfHealth check
# scheme:      http
# token:       abcd1234
# verify:      true
# cacert:      /etc/ssl/ca.crt
# cert:        /etc/ssl/cert.crt
# key:         /etc/ssl/key.key
# dc:          default

# If the register_service is set to true, Patroni will
# register nodes in Consul service 'scope'
# with the tag current role (master, replica or standby_leader)
# register_service: false

# how often Consul should check health by calling Patroni REST API
# service_check_interval: 5s
```

zalando

ZooKeeper & Exhibitor configuration

```
zookeeper:
  hosts:
    - host1:port1
    - host2:port2
    - host3:port3

exhibitor:
  hosts:
    - host1
    - host2
    - host3
  poll_interval: 300 # interval to update topology from Exhibitor
  port: 8181         # Exhibitor port (not ZooKeeper!)
```

zalando

Bootstrap configuration

```
bootstrap:
  dcs: # this content is written into the `/config` key after bootstrap succeeded
    loop_wait: 10
    ttl: 30
    retry_timeout: 10
    maximum_lag_on_failover: 10485760
#   master_start_timeout: 300
#   synchronous_mode: false
#   synchronous_mode_strict: false
    postgresql:
      use_pg_rewind: true
      use_slots: true
#     parameters:  # These parameters could be changed only globally (via DCS)
#       max_connections: 100
#       max_wal_senders: 10
#       max_prepared_transactions: 0
#       max_locks_per_transaction: 64
#       max_replication_slots: 10
#       max_worker_processes: 8
      pg_hba:
        - local   all         all          trust
        - hostssl all         all     all md5
        - hostssl replication standby all md5
```

Bootstrap configuration (continue)

```
bootstrap:
  method: my_bootstrap_method
  my_bootstrap_method:
    command: /usr/local/bin/my_bootstrap_script.sh
#   recovery_conf:
#     restore_command: /usr/local/bin/my_restore_command.sh
#     recovery_target_timeline: latest
#     recovery_target_action: promote
#     recovery_target_time: "2019-03-07 00:00:18.349 UTC"
#     recovery_target_inclusive: false

  post_bootstrap: /usr/local/bin/my_post_bootstrap_command.sh
```

Postgresql configuration

```
postgresql:
  use_unix_socket: true # how Patroni will connect to the local postgres
  listen: 0.0.0.0:5432
  connect_address: 172.21.0.6:5432 # how this node can be accessed from outside
  data_dir: /home/postgres/pgroot/pgdata
  bin_dir: /usr/lib/postgresql/10/bin # where the postgres binaries are located
  authentication:
    superuser:
      username: postgres
      password: SeCrEtPaS$WoRd
    replication:
      username: standby
      password: sTaNdByPaS$WoRd

  parameters:
    shared_buffers: 8GB
    unix_socket_directories: /var/run/postgresql

# recovery_conf:
#   restore_command: /usr/local/bin/my_restore_command.sh "%f" "%p"
```

Postgresql configuration (continue)

```
postgresql:
  callbacks:
    on_start: /usr/local/bin/my_callback.sh
    on_stop: /usr/local/bin/my_callback.sh
    on_role_change: /usr/local/bin/my_callback.sh
  create_replica_method:
    - custom_backup
    - basebackup
  custom_backup:
    command: /usr/local/bin/restore_cluster.sh
    retries: 2
    no_master: 1
  basebackup:  # custom arguments to pg_basebackup
    max-rate: 10M
    waldir: /home/postgres/pgroot/pgwal
```

Watchdog and tags configuration

```
watchdog:
  mode: automatic  # Allowed values: off, automatic, required
  device: /dev/watchdog

  # Watchdog will be triggered 5 seconds before the leader expiration
  safety_margin: 5

tags:
    nofailover: false
    noloadbalance: false
    clonefrom: true
#   nosync: true
#   replicatefrom: patroni2
```

zalando

Additional ways of configuring Patroni

- Patroni can also be configured with environment varibles described at https://patroni.readthedocs.io/en/latest/ENVIRONMENT.html

- Environment variables take priority over the corresponding parameters listed in the configuration file.

- One can pass a complete Patroni configuration in the PATRONI_CONFIGURATION environment variable. If it is present - no other sources of configuration are considered.

Troubleshooting

zalando

DCS is not accessible

```
$ patroni postgres0.yml

2018-01-23 14:00:07,211 INFO: Selected new etcd server http://127.0.0.1:2379
2018-01-23 14:00:07,212 WARNING: Retrying (Retry(total=1, connect=None,
read=None, redirect=0, status=None)) after connection broken by
'NewConnectionError('<urllib3.connection.HTTPConnection object at
0x7f27e4524b90>: Failed to establish a new connection: [Errno 111] Connection
refused',)': /v2/machines
2018-01-23 14:00:07,212 WARNING: Retrying (Retry(total=0, connect=None,
read=None, redirect=0, status=None)) after connection broken by
'NewConnectionError('<urllib3.connection.HTTPConnection object at
0x7f27e4524cd0>: Failed to establish a new connection: [Errno 111] Connection
refused',)': /v2/machines
2018-01-23 14:00:07,213 ERROR: Failed to get list of machines from
http://127.0.0.1:2379/v2: MaxRetryError("HTTPConnectionPool(host='127.0.0.1',
port=2379): Max retries exceeded with url: /v2/machines (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at
0x7f27e4524dd0>: Failed to establish a new connection: [Errno 111] Connection
refused',))",)
2018-01-23 14:00:07,213 INFO: waiting on etcd
2018-01-23 14:00:12,218 INFO: Selected new etcd server http://127.0.0.1:2379
```

Patroni can’t find PostgreSQL binaries

```
$ patroni postgres0.yml

2018-01-23 14:04:52,284 INFO: Selected new etcd server http://127.0.0.1:2379
2018-01-23 14:04:52,291 INFO: Lock owner: None; I am patroni1
2018-01-23 14:04:52,299 INFO: trying to bootstrap a new cluster
2018-01-23 14:04:52,301 ERROR: Exception during execution of long running task bootstrap
Traceback (most recent call last):
  File "/home/akukushkin/git/patroni/patroni/async_executor.py", line 97, in run
    wakeup = func(*args) if args else func()
  File "/home/akukushkin/git/patroni/patroni/postgresql.py", line 1556, in bootstrap
    return do_initialize(config) and self._configure_server_parameters() and self.start()
  File "/home/akukushkin/git/patroni/patroni/postgresql.py", line 537, in _initdb
    ret = self.pg_ctl('initdb', *options)
  File "/home/akukushkin/git/patroni/patroni/postgresql.py", line 283, in pg_ctl
    return subprocess.call(pg_ctl + ['-D', self._data_dir] + list(args), **kwargs) == 0
  File "/usr/lib/python3.5/subprocess.py", line 557, in call
    with Popen(*popenargs, **kwargs) as p:
  File "/usr/lib/python3.5/subprocess.py", line 947, in __init__
    restore_signals, start_new_session)
  File "/usr/lib/python3.5/subprocess.py", line 1551, in _execute_child
    raise child_exception_type(errno_num, err_msg)
FileNotFoundError: [Errno 2] No such file or directory: 'pg_ctl'
2018-01-23 14:04:52,308 INFO: removing initialize key after failed attempt to bootstrap the
cluster
```

Not really an error, will disappear after “loop_wait” seconds

```
$ patroni postgres1.yml

2018-01-23 14:07:34,295 INFO: bootstrapped from leader 'patroni1'
2018-01-23 14:07:34,373 INFO: postmaster pid=28577
2018-01-23 14:07:34.381 CET [28577] LOG:  listening on IPv4 address "127.0.0.1", port
5433
2018-01-23 14:07:34.396 CET [28577] LOG:  listening on Unix socket "./.s.PGSQL.5433"
2018-01-23 14:07:34.430 CET [28579] LOG:  database system was interrupted; last known up
at 2018-01-23 14:07:33 CET
2018-01-23 14:07:34.431 CET [28580] FATAL:  the database system is starting up
localhost:5433 - rejecting connections
2018-01-23 14:07:34.438 CET [28582] FATAL:  the database system is starting up
localhost:5433 - rejecting connections
2018-01-23 14:07:34.487 CET [28579] LOG:  entering standby mode
2018-01-23 14:07:34.501 CET [28579] LOG:  redo starts at 0/2000028
2018-01-23 14:07:34.507 CET [28579] LOG:  consistent recovery state reached at 0/20000F8
2018-01-23 14:07:34.508 CET [28577] LOG:  database system is ready to accept read only
connections
2018-01-23 14:07:34.522 CET [28586] FATAL:  could not start WAL streaming: ERROR:
replication slot "patroni2" does not exist
2018-01-23 14:07:34.526 CET [28588] FATAL:  could not start WAL streaming: ERROR:
replication slot "patroni2" does not exist
localhost:5433 - accepting connections
```

Wrong initdb config options

```
$ patroni postgres0.yml

2018-01-23 14:13:23,292 INFO: Selected new etcd server http://127.0.0.1:2379
2018-01-23 14:13:23,309 INFO: Lock owner: None; I am patroni1
2018-01-23 14:13:23,318 INFO: trying to bootstrap a new cluster
/usr/lib/postgresql/10/bin/initdb: option '--data-checksums' doesn't allow an argument
Try "initdb --help" for more information.
pg_ctl: database system initialization failed
2018-01-23 14:13:23,345 INFO: removing initialize key after failed attempt to bootstrap the
cluster
```

```
--- a/postgres0.yml
+++ b/postgres0.yml
@@ -43,7 +43,7 @@ bootstrap:
   # some desired options for 'initdb'
   initdb: # Note: It needs to be a list (some options need values, others are switches)
   - encoding: UTF8
-  - data-checksums: true
+  - data-checksums

   pg_hba: # Add following lines to pg_hba.conf after running 'initdb'
   - host replication replicator 127.0.0.1/32 md5
```

Badly formatted yaml

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

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

```
ERROR:  DO $$
    BEGIN
        SET local synchronous_commit = 'local';
        PERFORM * FROM pg_authid WHERE rolname = 'admin';
        IF FOUND THEN
            ALTER ROLE "admin" WITH - C R E A T E R O L E   - C R E A T E D B LOGIN PASSWORD
'admin';
        ELSE
            CREATE ROLE "admin" WITH - C R E A T E R O L E   - C R E A T E D B LOGIN PASSWORD
'admin';
        END IF;
    END;
    $$
```

Cluster was initialized during install of postgres packages

```
# node1

$ sudo apt-get install postgresql
$ sudo pip install patroni[etcd]
$ cat /etc/patroni.yaml
...
postgresql:
  data_dir: /var/lib/postgresql/10/main
...

$ patroni /etc/patroni.yaml
2018-01-23 14:50:54,342 INFO: Selected new
etcd server http://127.0.0.1:2379
2018-01-23 14:50:54,347 INFO: establishing
a new patroni connection to the postgres
cluster
2018-01-23 14:50:54,364 INFO: acquired
session lock as a leader
```

```
# node2

$ sudo apt-get install postgresql
$ sudo pip install patroni[etcd]
$ cat /etc/patroni.yaml
...
postgresql:
  data_dir: /var/lib/postgresql/10/main
...

$ patroni.py patroni.yaml
2018-01-23 14:53:27,878 CRITICAL: system ID
mismatch, node patroni1 belongs to a different
cluster: 6497216458191333666 != 6497220080226496012
2018-01-23 14:53:28.373 CET [30508] LOG:  received
fast shutdown request
2018-01-23 14:53:28.418 CET [30508] LOG:  database
system is shut down
2018-01-23 14:53:28,426 INFO: Lock owner: node1; I
am node2
```

Can we avoid demoting primary when DCS is not available?

- DCS is a source of truth about running leader
- DCS keeps information about the cluster topology
- Generally not possible to tell failed connection from DCS from the failure of DCS itself
- DCS could be inaccessible for some nodes but not others
- Without DCS it is not possible to figure out if there is another leader already running
- To be on the safe side Patroni demotes Postgres

Can I have high availability with 2 nodes/2 datacenters?

- DCS operates a quorum, decision by the majority of nodes (50% + 1)

- Quorum of 2 is 2.

- With 50% of nodes are down, there is no quorum.

Useful links

- Patroni - https://github.com/zalando/patroni
- Web-based searchable documentation: https://patroni.readthedocs.io
- Spilo - a docker image based on Patroni: https://github.com/zalando/spilo

zalando

Thank you!