zalando

Patroni

HA PostgreSQL

made easy

PostgresConf US

Alexander Kukushkin

Oleksii Kliukin

Zalando SE

16-04-2018

Agenda

Architecture overview

Hands on: your first test cluster

Dynamic cluster configuration

REST endpoints and monitoring

Client connections

Advanced features

Custom extensions

Troubleshooting

zalando

PostgreSQL High Availability

- Shared storage solutions
  - DRDB + 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 replication
  - BDR, bucardo

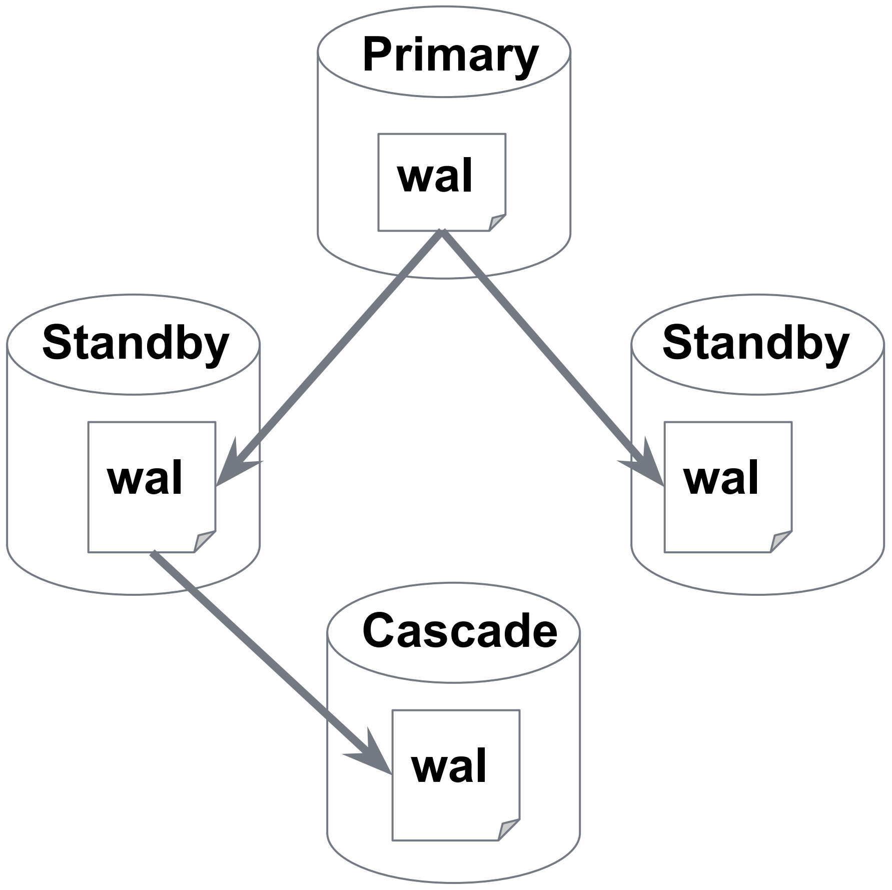
Physical single-master replication

Cons

- No partial replication
- Major versions much match
- Missing automatic failover

Pros

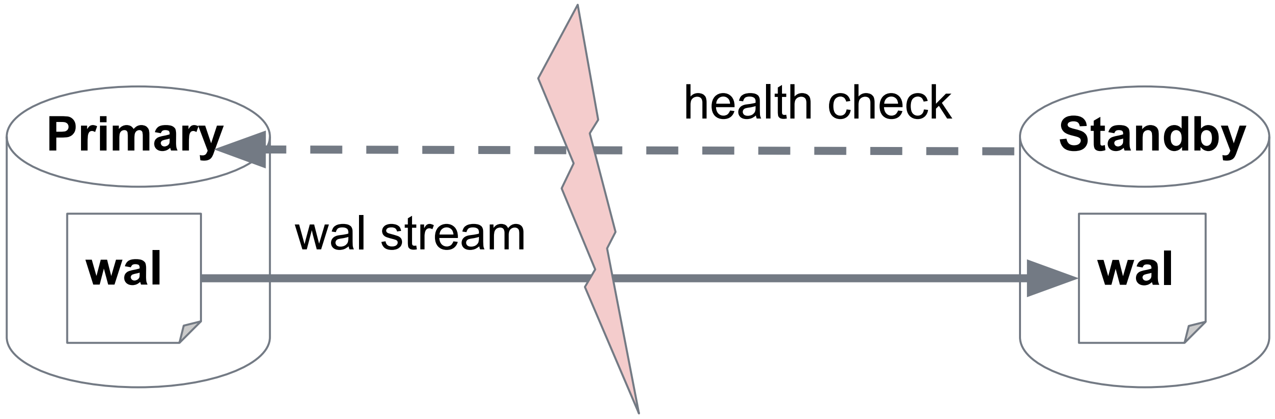
- Built-in since Postgres 9.0
- Minimal overhead
- Replicates everything
- Cascading replication
- Synchronous replication
- Takes advantage of streaming and WAL shipping



zalando

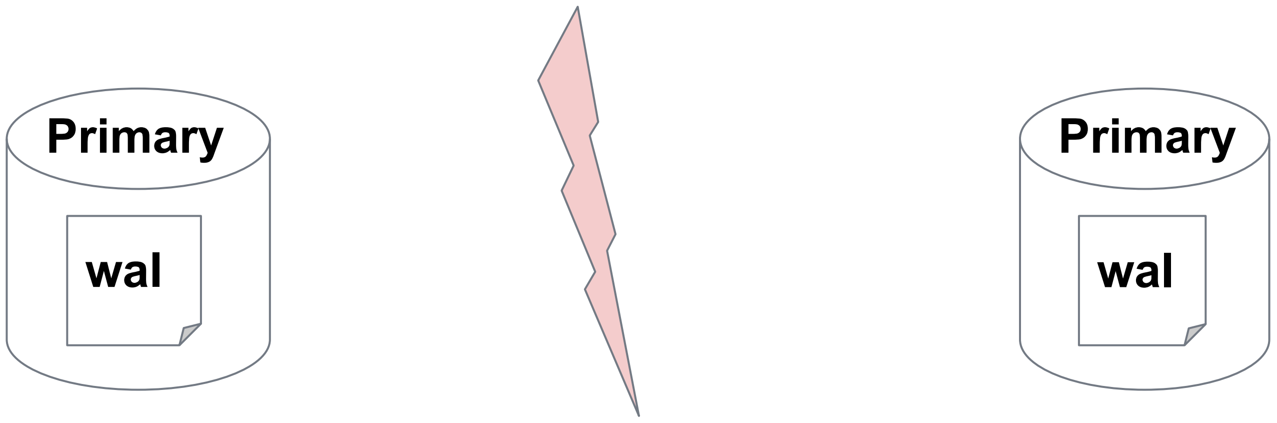
Automatic failover done wrong:
Running just two nodes

Run the **health check** from the standby and promote that standby when the health check indicates the primary failure



Automatic failover done wrong: running just two nodes

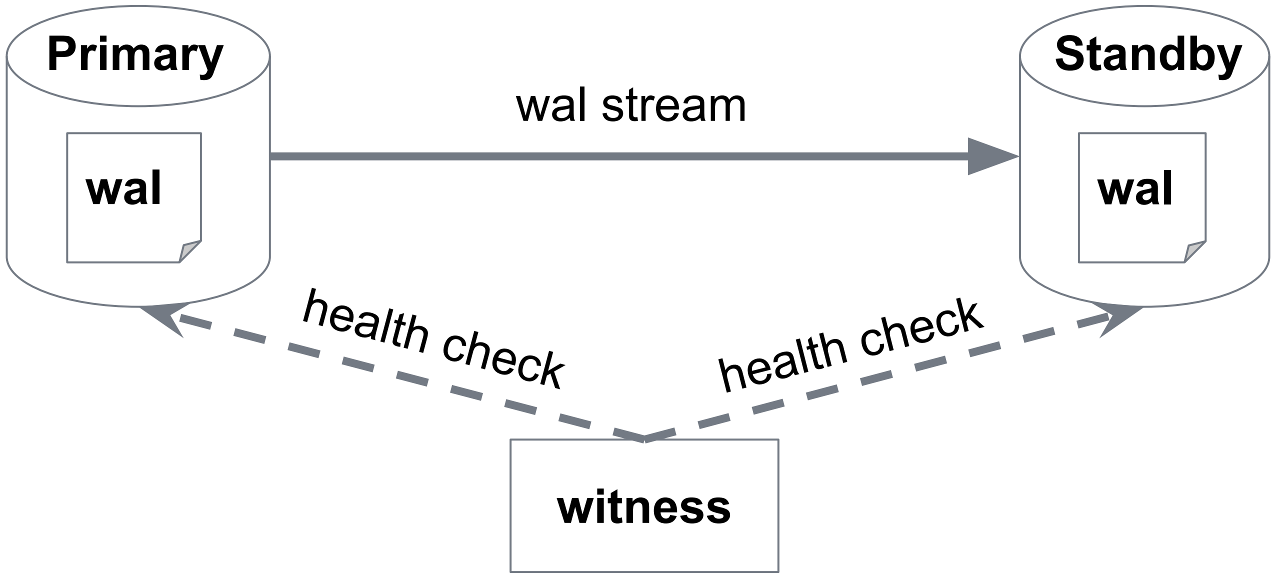
Split-brain!



zalando

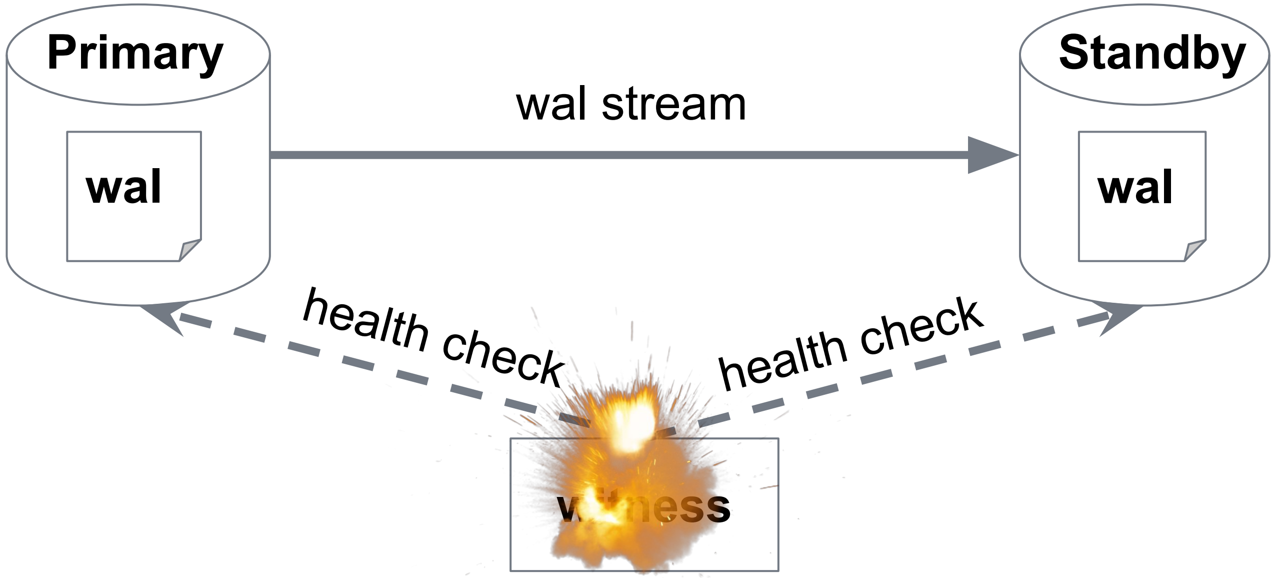
Automatic failover done wrong: Single witness node

What can possibly go wrong?



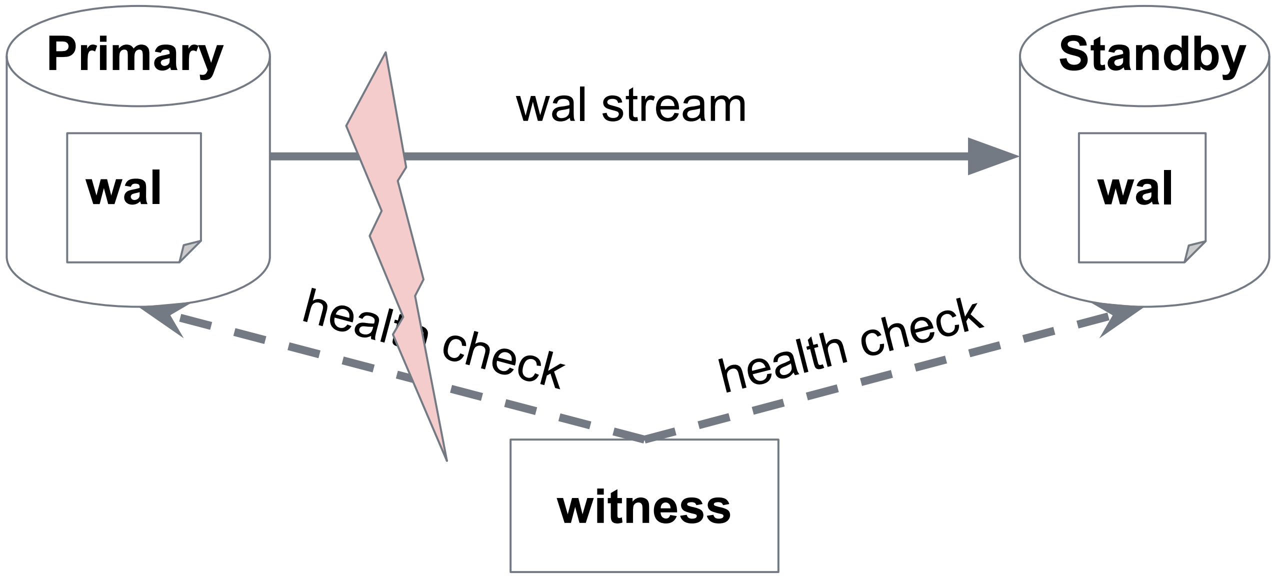
Automatic failover done wrong: Single witness node

Witness node dies



Automatic failover done wrong: Single witness node

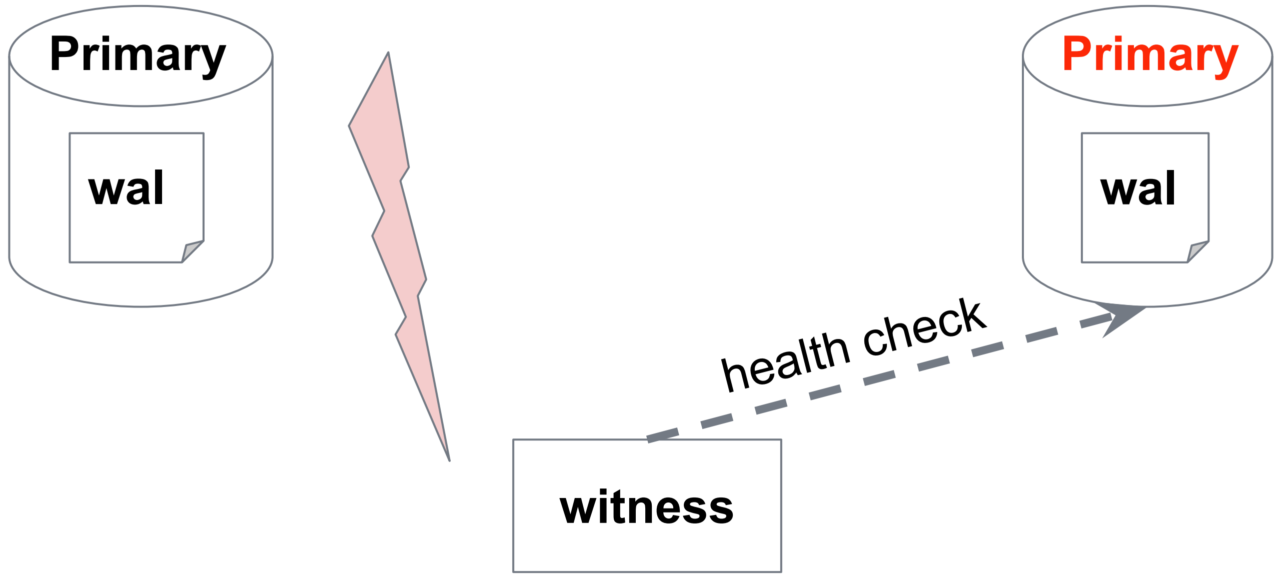
Or gets partitioned



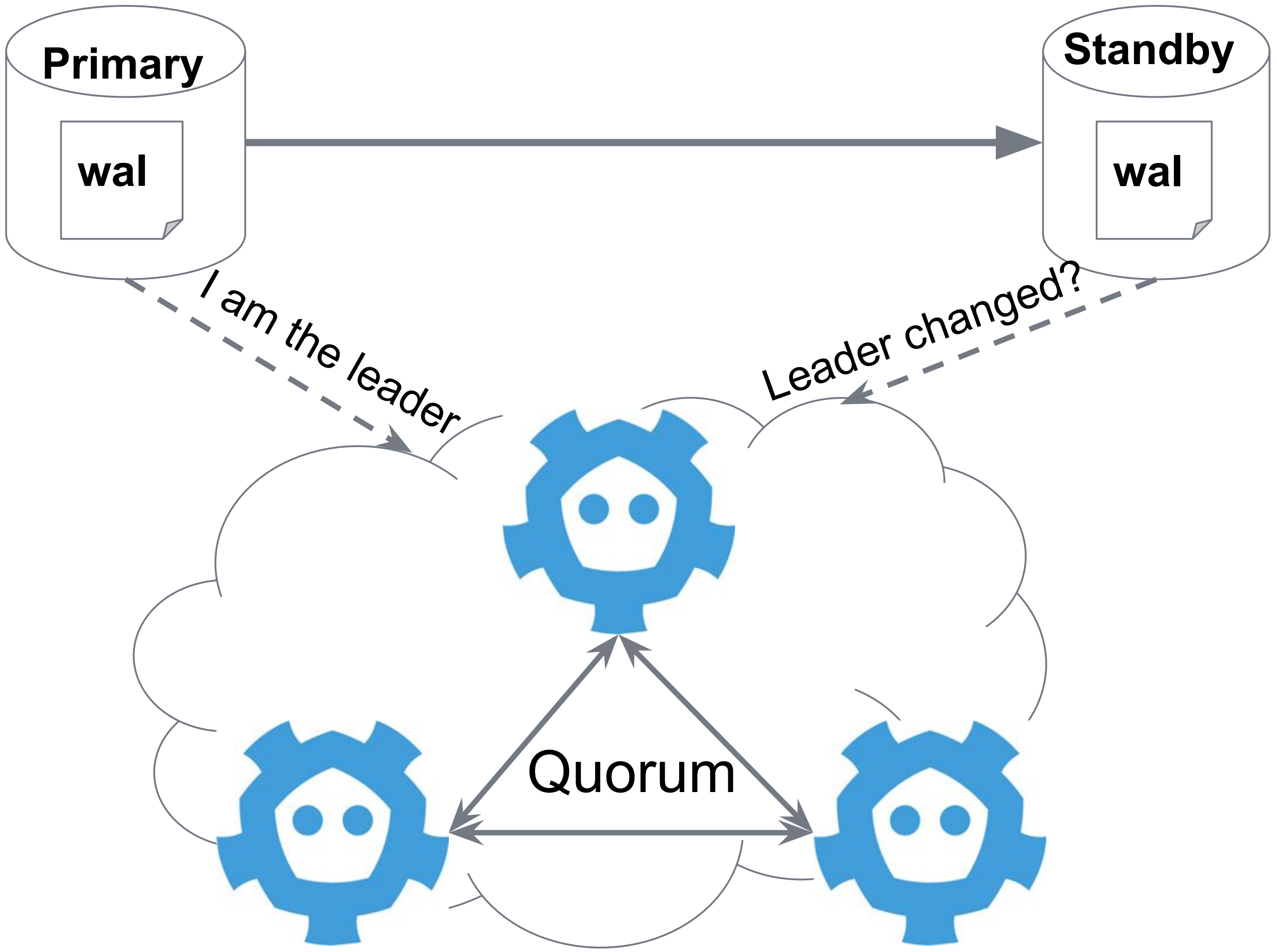
zalando

Automatic failover done wrong: Single witness node

Existing primary is running



Automatic failover done right



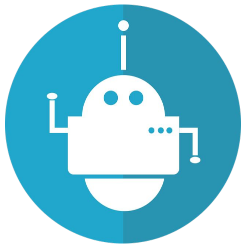
zalando

Automatic failover: the right way

- Leader elections among all members of the cluster
- Each member decides only for itself
- Cluster state stored in a consistent distributed storage
- Leader key changed via atomic CAS operations
- Client follow the new leader
- Fencing of non-cooperative or failed nodes

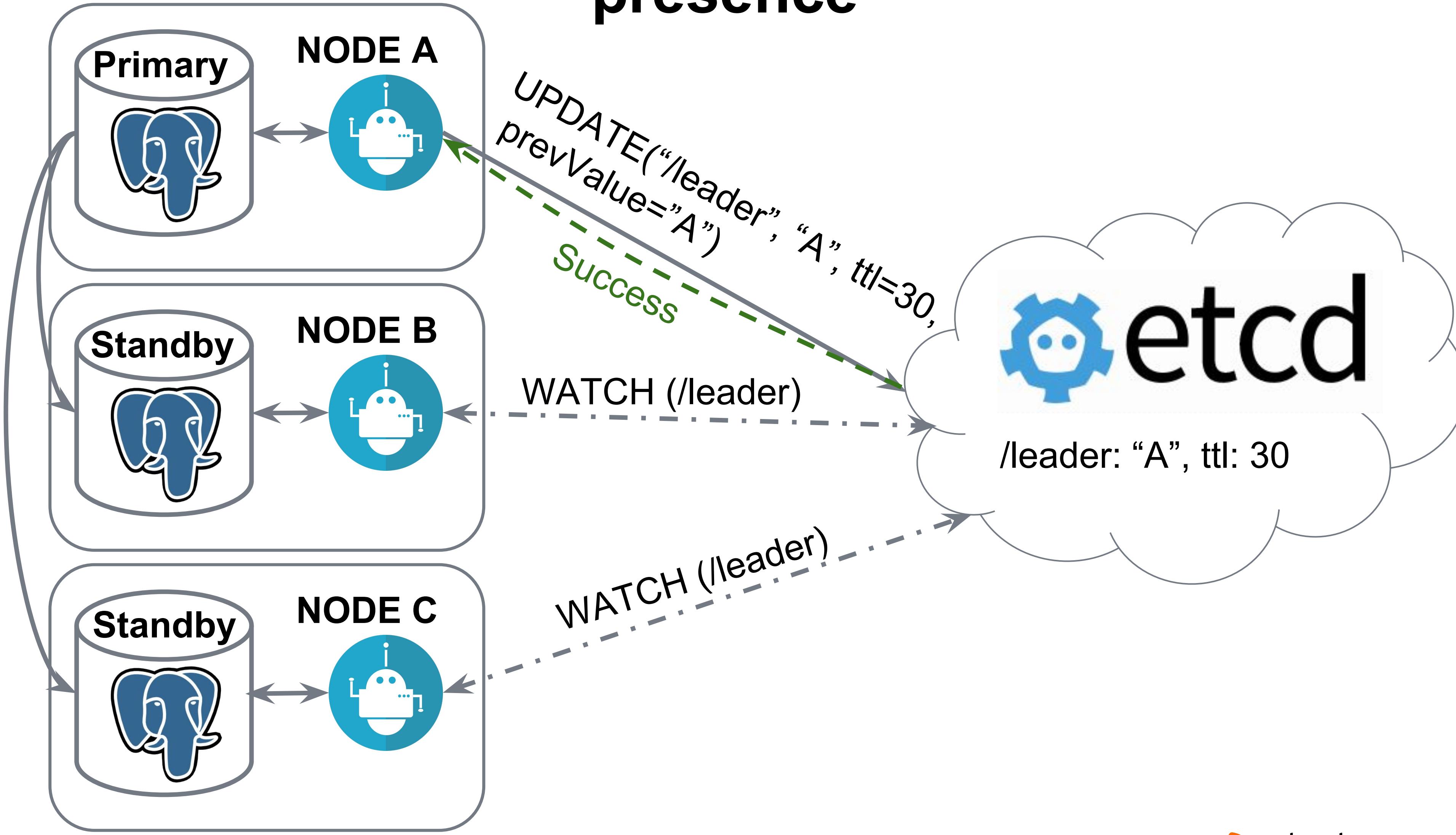
Bot pattern

- PostgreSQL cannot talk to Etcd directly
- Let’s employ a bot to manage PostgreSQL
- A bot should run alongside PostgreSQL
- A bot will talk to Etcd (or other DCS)
- A bot decides on promotion/demotion

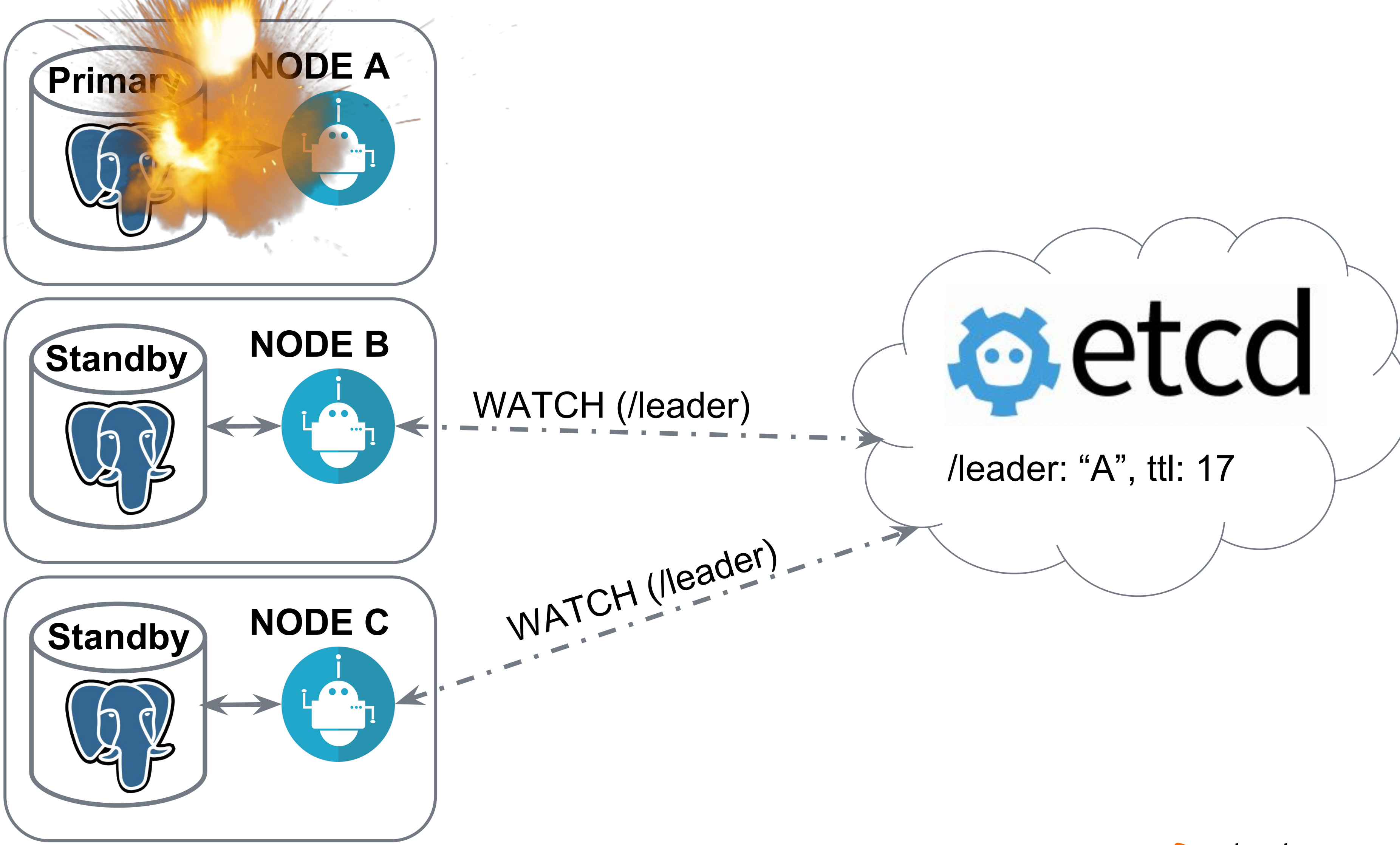


zalando

Bot pattern: master acknowledges its presence

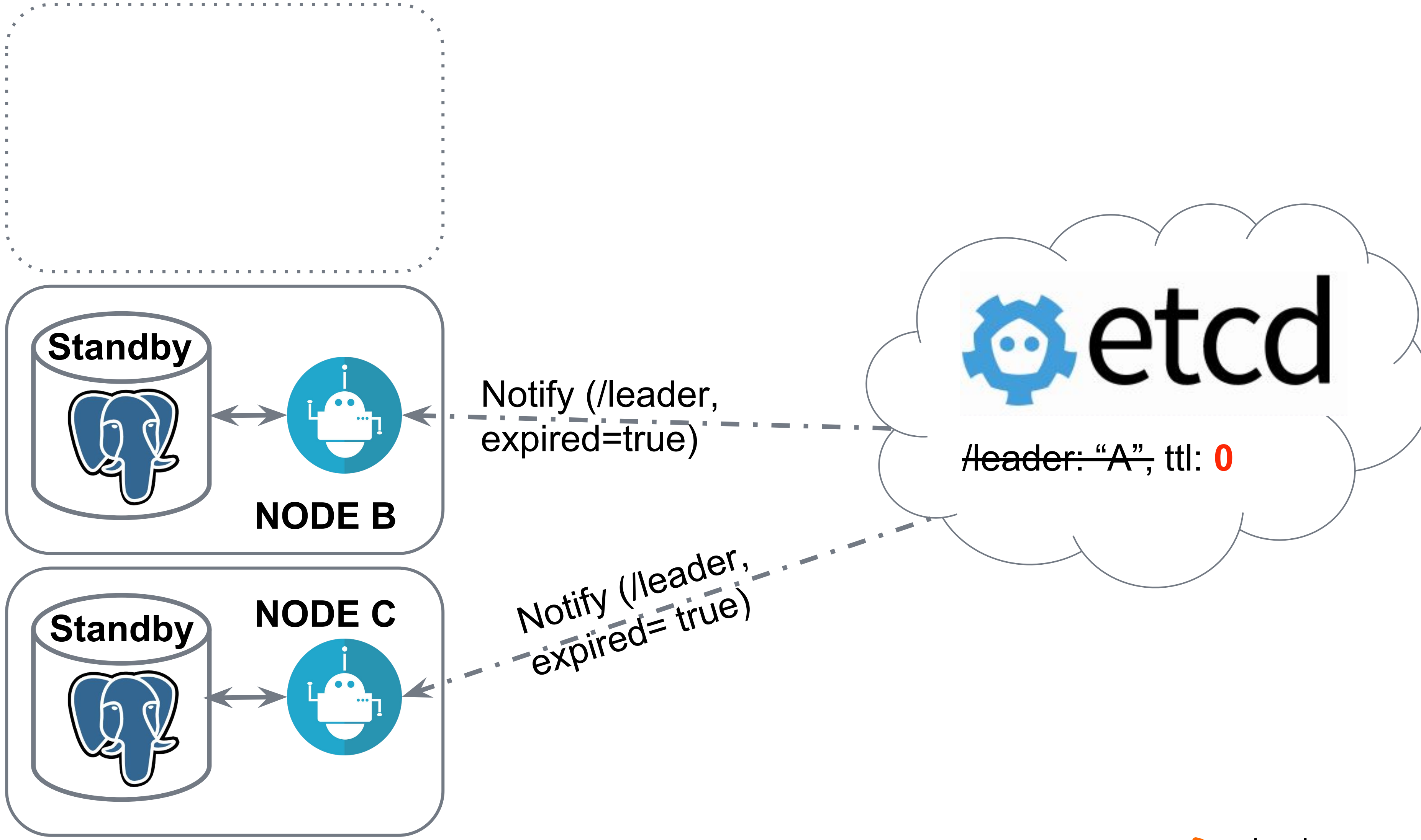


Bot pattern: master dies, leader key holds

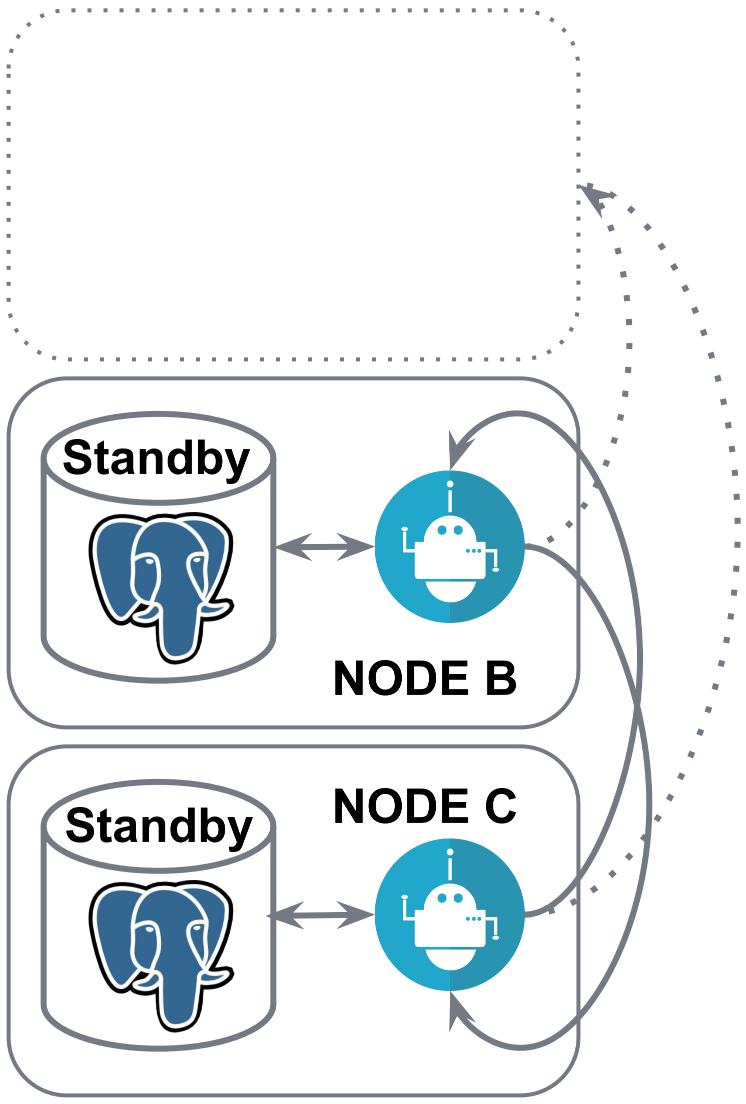


zalando

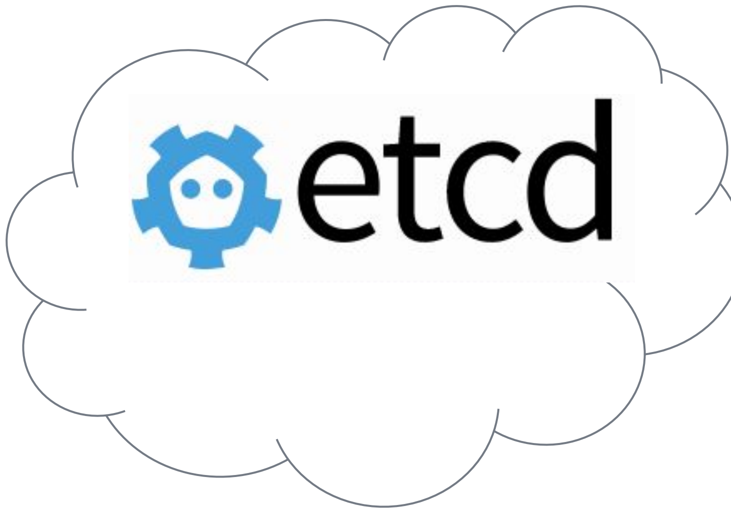
Bot pattern: leader key expires


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?



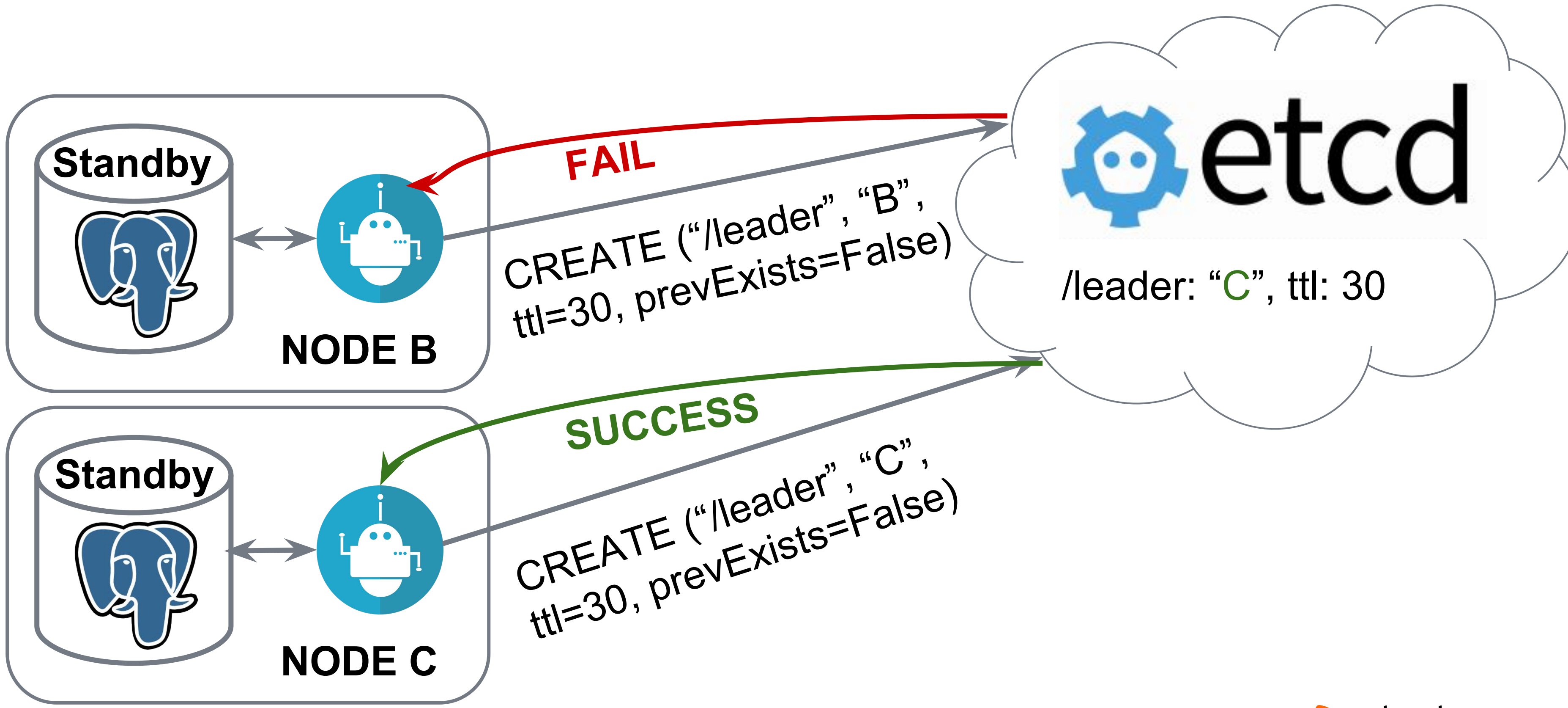
Node **B**:
GET A:8008/patroni -> **timeout**
GET C:8008/patroni -> wal_position: **100**



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

zalando

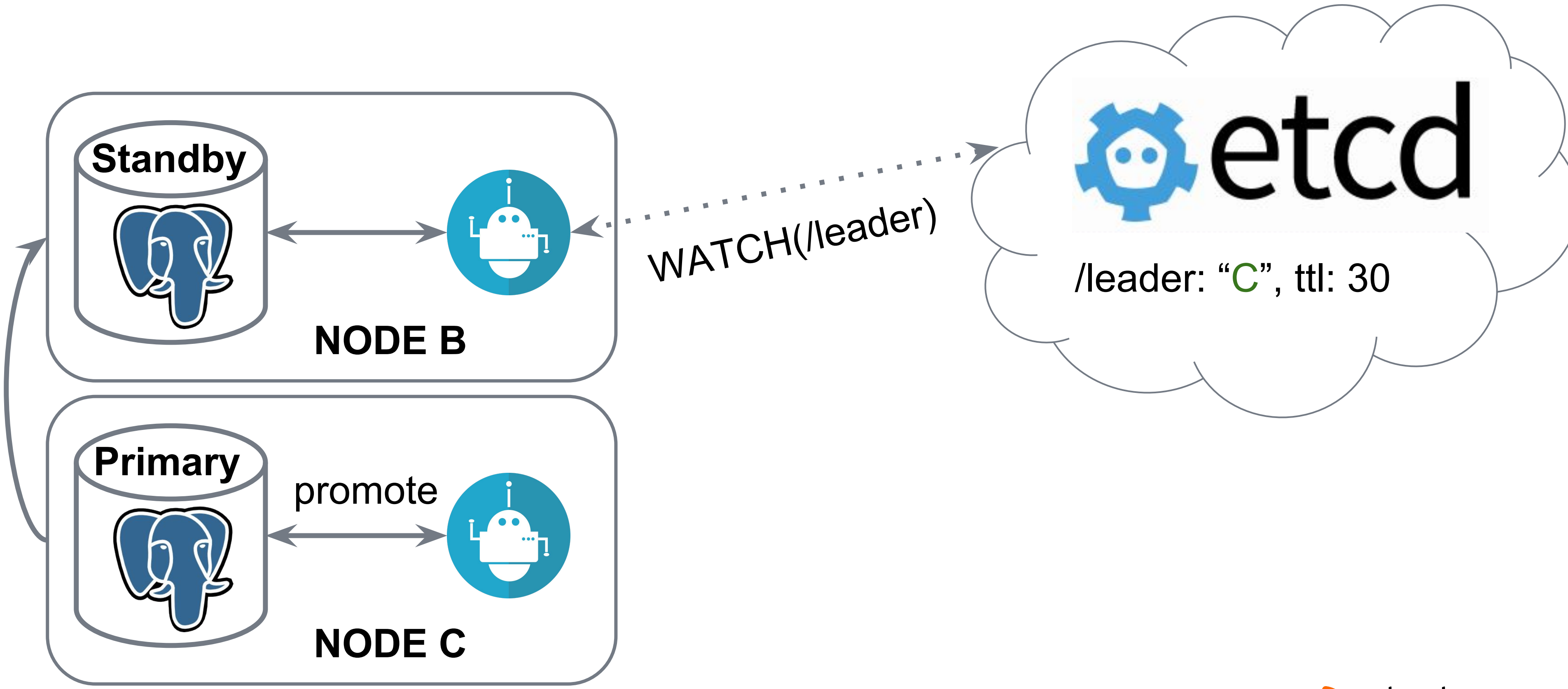
Bot pattern: leader race among equals



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)

zalando

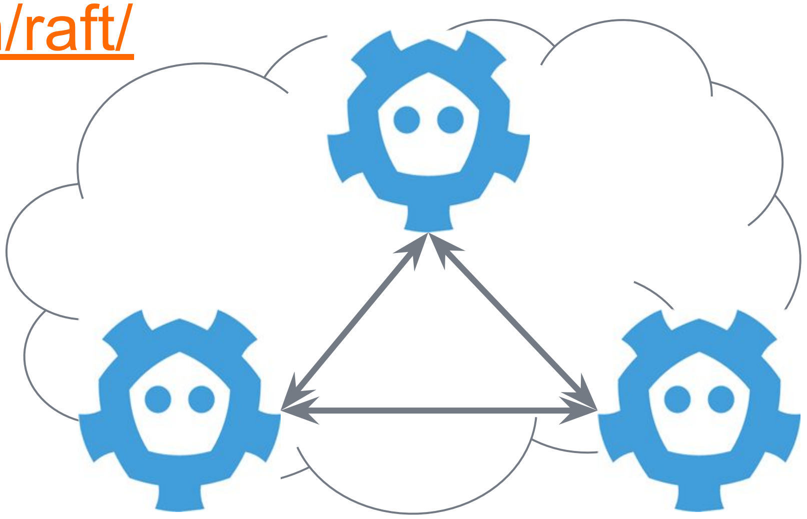
Bot pattern: promote and continue replication



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

Using docker

- install docker
- docker pull kliukin/patroni-training
- docker run -d --name patroni-training kliukin/patroni-training
- docker exec -ti patroni-training bash

```
postgres@f40a9391f810:~$ ls *.yml
postgres0.yml  postgres1.yml  postgres2.yml
```

zalando

(Optional) using vagrant

- install vagrant
- get the vagrantfile from https://github.com/alexeyklyukin/patroni-training
- vagrant up
- vagrant ssh

```
ubuntu@ubuntu-xenial:~$ sudo -iu postgres
postgres@ubuntu-xenial:~$ ls *.yml
postgres0.yml  postgres1.yml  postgres2.yml
```

Hands on: creating your first cluster with Patroni

```
$ patroni postgres0.yml

2018-01-18 13:29:06,714 INFO: Selected new
etcd server http://127.0.0.1:2379
2018-01-18 13:29:06,731 INFO: Lock owner:
None; I am postgresql0
2018-01-18 13:29:06,796 INFO: trying to
bootstrap a new cluster
...
Success. You can now start the database
server using:
    /usr/local/pgsql/bin/pg_ctl -D
data/postgresql0 -l logfile start
2018-01-18 13:29:13,115 INFO: initialized a
new cluster
2018-01-18 13:29:23,088 INFO: Lock owner:
postgresql0; I am postgresql0
2018-01-18 13:29:23,143 INFO: no action.  i
am the leader with the lock
```

```
$ patroni postgres1.yml

2018-01-18 13:45:02,479 INFO: Selected new
etcd server http://127.0.0.1:2379
2018-01-18 13:45:02,488 INFO: Lock owner:
postgresql0; I am postgresql1
2018-01-18 13:45:02,499 INFO: trying to
bootstrap from leader 'postgresql0'
2018-01-18 13:45:04,470 INFO: replica has
been created using basebackup
2018-01-18 13:45:04,474 INFO: bootstrapped
from leader 'postgresql0'
2018-01-18 13:45:07,211 INFO: Lock owner:
postgresql0; I am postgresql1
2018-01-18 13:45:07,212 INFO: does not
have lock
2018-01-18 13:45:07,440 INFO: no action.
i am a secondary and i am following a
leader
```

Patronictl output on success

```
$ patronictl list batman
+---------+-------------+-----------+--------+---------+-----------+
| Cluster | Member      | Host      |  Role  |  State  | Lag in MB |
+---------+-------------+-----------+--------+---------+-----------+
| batman  | postgresql0 | 127.0.0.1 | Leader | running |         0 |
| batman  | postgresql1 | 127.0.0.1 |        | running |         0 |
+---------+-------------+-----------+--------+---------+-----------+
```

zalando

Automatic failover

Failover happens when primary dies abruptly
We will simulate it by stopping Patroni

Don’t try this on your production databases

```
2018-01-18 16:04:15,869 INFO: Lock owner: postgresql0; I am
postgresql0
2018-01-18 16:04:15,908 INFO: no action.  i am the leader with
the lock
^Z
[1]+  Stopped                    patroni postgres0.yml

$ kill -9 %1
[1]+  Killed: 9                  patroni postgres0.yml
```

Replica promotion

```
2018-01-18 16:04:39,019 INFO: Lock owner: postgresql0; I am postgresql1
2018-01-18 16:04:39,019 INFO: does not have lock
2018-01-18 16:04:39,021 INFO: no action.  i am a secondary and i am following a
leader
2018-01-18 16:04:46,358 WARNING: request failed: GET
http://127.0.0.1:8008/patroni (HTTPConnectionPool(host='127.0.0.1', port=8008):
Max retries exceeded with url: /patroni (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at 0x109c92898>:
Failed to establish a new connection: [Errno 61] Connection refused',)))
2018-01-18 16:04:46,474 INFO: promoted self to leader by acquiring session lock
server promoting
2018-01-18 16:04:46.506 CET [36202] LOG:  received promote request
2018-01-18 16:04:46.506 CET [36209] FATAL:  terminating walreceiver process due
to administrator command
2018-01-18 16:04:46.508 CET [36202] LOG:  redo done at 0/3000028
2018-01-18 16:04:46.512 CET [36202] LOG:  selected new timeline ID: 2
2018-01-18 16:04:46.562 CET [36202] LOG:  archive recovery complete
2018-01-18 16:04:46.566 CET [36200] LOG:  database system is ready to accept
connections
2018-01-18 16:04:47,537 INFO: Lock owner: postgresql1; I am postgresql1
```

How does Patroni cope with split-brain

After the previous step we have two masters running
Let's make a divergence by writing to the unmanaged master.

Don't try this on your production databases

```
$ psql -h localhost -p 5432 -tA \
   -c "CREATE TABLE splitbrain();"
CREATE TABLE
```

zalando

Resume patroni and rejoin the former master

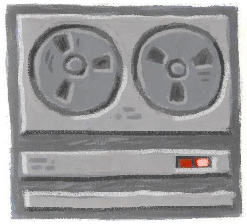
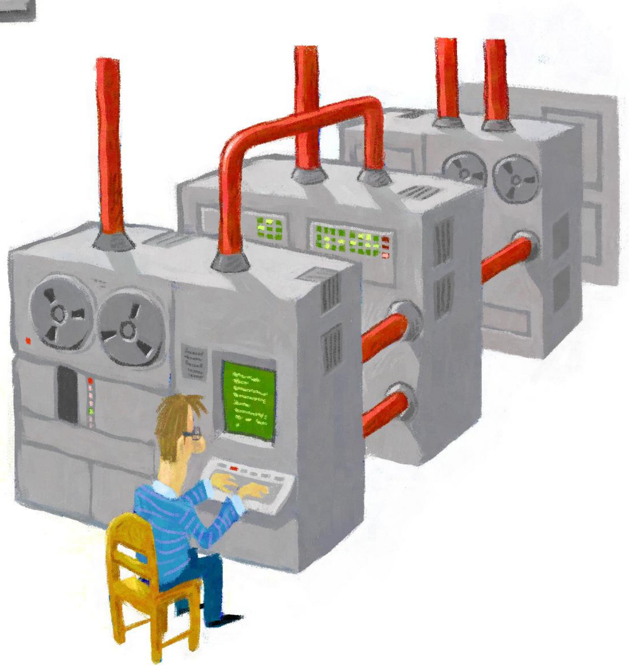
```
$ patroni postgres0.yml
2018-01-18 16:04:57,214 INFO: Selected new etcd server http://127.0.0.1:2379
2018-01-18 16:04:57,221 INFO: establishing a new patroni connection to the
postgres cluster
2018-01-18 16:04:57,344 INFO: Lock owner: postgresql1; I am postgresql0
2018-01-18 16:04:57,344 INFO: does not have lock
2018-01-18 16:04:57.370 CET [36179] LOG:  received immediate shutdown request
2018-01-18 16:04:57,384 INFO: demoting self because i do not have the lock and i
was a leader
2018-01-18 16:04:57.666 CET [36339] LOG:  entering standby mode
2018-01-18 16:04:57.669 CET [36339] LOG:  database system was not properly shut
down; automatic recovery in progress
2018-01-18 16:04:57,777 INFO: Lock owner: postgresql1; I am postgresql0
2018-01-18 16:04:57,777 INFO: does not have lock
2018-01-18 16:04:58,004 INFO: Local timeline=1 lsn=0/30175C0
2018-01-18 16:04:58,014 INFO: master_timeline=2
2018-01-18 16:04:58,014 INFO: master: history=1    0/3000060 no recovery target
specified
2018-01-18 16:04:58,155 INFO: running pg_rewind from user=postgres host=127.0.0.1
port=5433 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!
2018-01-18 16:04:59,490 INFO: starting as a secondary
```

Patronictl output

```
$ patronictl -c postgres0.yml list batman
+---------+-------------+-----------+--------+---------+-----------+
| Cluster | Member      | Host      |  Role  |  State  | Lag in MB |
+---------+-------------+-----------+--------+---------+-----------+
| batman  | postgresql0 | 127.0.0.1 |        | running |         0 |
| batman  | postgresql1 | 127.0.0.1 | Leader | running |         0 |
+---------+-------------+-----------+--------+---------+-----------+
```

Peek into etcd

```
$ etcdctl ls --recursive --sort -p /service/batman

/service/batman/config
/service/batman/history
/service/batman/initialize
/service/batman/leader
/service/batman/members/
/service/batman/members/postgresql0
/service/batman/members/postgresql1
/service/batman/optime/
/service/batman/optime/leader

$ etcdctl get /service/batman/leader
postgresql1

$ etcdctl get /service/batman/members/postgresql1
{"conn_url":"postgres://127.0.0.1:5433/postgres","api_url":"http://127.0.0.1:8009/patro
ni","state":"running","role":"master","xlog_location":50476648,"timeline":2}

$ etcdctl get /service/batman/history
[[1,50331744,"no recovery target specified","2018-01-18T16:04:46+01:00"]]
```

Let’s edit some configuration

Editing configuration with patronictl

```
$ patronictl -c postgres0.yml edit-config batman

"/tmp/batman-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
```

Editing configuration with patronictl

```
2018-01-18 14:19:06,352 INFO: Lock owner: postgresql1; I am postgresql0
2018-01-18 14:19:06,352 INFO: does not have lock
2018-01-18 14:19:06,360 INFO: no action.  i am a secondary and i am
following a leader
2018-01-18 14:19:16,355 INFO: Lock owner: postgresql1; I am postgresql0
2018-01-18 14:19:16,355 INFO: does not have lock
2018-01-18 14:19:16,368 INFO: no action.  i am a secondary and i am
following a leader
server signaled
2018-01-18 14:19:16.451 CET [28996] LOG:  received SIGHUP, reloading
configuration files
2018-01-18 14:19:16.461 CET [28996] LOG:  parameter "work_mem" changed to
"8MB"
2018-01-18 14:19:26,357 INFO: Lock owner: postgresql1; I am postgresql0
2018-01-18 14:19:26,357 INFO: does not have lock
2018-01-18 14:19:26,365 INFO: no action.  i am a secondary and i am
following a leader
```

Editing configuration with patronictl

```
$ patronictl edit-config batman

"/tmp/batman-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
```

Editing configuration with patronictl

```
$ patronictl list batman
+---------+-------------+-----------+--------+---------+-----------+-----------------+
| Cluster |    Member   |    Host   |  Role  |  State  | Lag in MB | Pending restart |
+---------+-------------+-----------+--------+---------+-----------+-----------------+
|  batman | postgresql0 | 127.0.0.1 |        | running |         0 |        *        |
|  batman | postgresql1 | 127.0.0.1 | Leader | running |         0 |        *        |
+---------+-------------+-----------+--------+---------+-----------+-----------------+
```

Editing configuration with patronictl

```
$ http http://127.0.0.1:8008
HTTP/1.0 503 Service Unavailable
...
{
    "database_system_identifier": "6512366775019348050",
    "patroni": {"scope": "batman", "version": "1.4"},
    "pending_restart": true,
    "postmaster_start_time": "2018-01-18 13:45:04.702 CET",
    "role": "replica",
    "server_version": 100000,
    "state": "running",
    "timeline": 2,
    "xlog": {
        "paused": false,
        "received_location": 50331968,
        "replayed_location": 50331968,
        "replayed_timestamp": null
    }
}
```

Editing configuration with patronictl

```
$ http http://127.0.0.1:8009
HTTP/1.0 200 OK
...
{
    "database_system_identifier": "6512366775019348050",
    "patroni": {"scope": "batman", "version": "1.4"},
    "pending_restart": true,
    "postmaster_start_time": "2018-01-18 13:44:44.764 CET",
...
    "role": "master",
    "server_version": 100000,
    "state": "running",
    "timeline": 2,
    "xlog": {
        "location": 50331968
    }
}
```

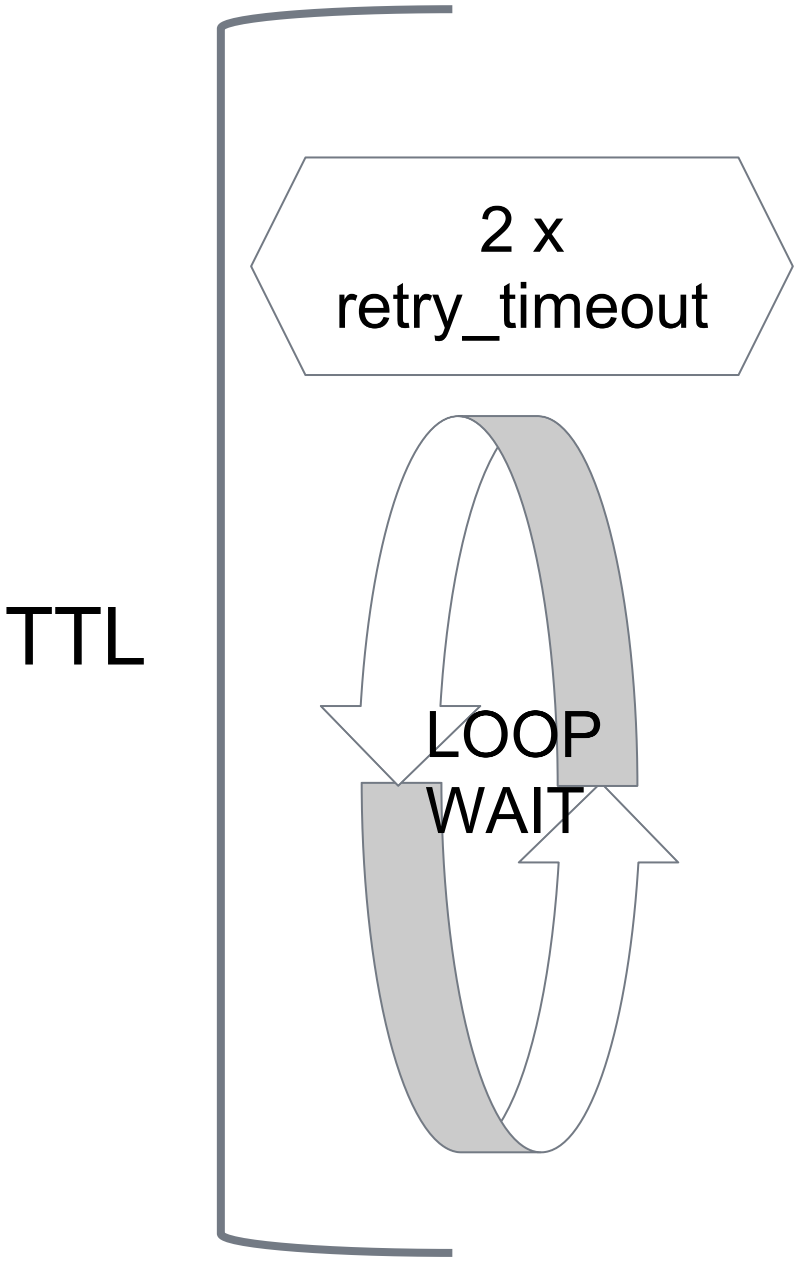
Editing configuration with patronictl

```
$ patronictl restart batman postgresql0
+---------+-------------+-----------+--------+---------+-----------+-----------------+
| Cluster |    Member   |    Host   |  Role  |  State  | Lag in MB | Pending restart |
+---------+-------------+-----------+--------+---------+-----------+-----------------+
|  batman | postgresql0 | 127.0.0.1 |        | running |         0 |        *        |
|  batman | postgresql1 | 127.0.0.1 | Leader | running |         0 |        *        |
+---------+-------------+-----------+--------+---------+-----------+-----------------+

Are you sure you want to restart members postgresql0? [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 postgresql0
```

Editing configuration with patronictl

```
$ psql -h localhost -p 5432 -U postgres -tqA \
   -c "SHOW max_connections"
101

…
$ psql -h localhost -p 5433 -U postgres -tqA \
   -c "SHOW max_connections"
100
```

```
ttl >= loop_wait +
 retry_timeout * 2
```



Changing TTL, loop_wait, retry_timeout

ttl >= loop_wait + retry_timeout * 2

```
$ patronictl edit-config batman
---
+++
@@ -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
```

zalando

Changing TTL, loop_wait, retry_timeout

```
2018-01-18 14:31:06,350 INFO: Lock owner: postgresql1; I am postgresql1
2018-01-18 14:31:06,364 INFO: no action.  i am the leader with the lock
2018-01-18 14:31:16,349 INFO: Lock owner: postgresql1; I am postgresql1
2018-01-18 14:31:16,362 INFO: no action.  i am the leader with the lock
2018-01-18 14:31:16,376 INFO: Lock owner: postgresql1; I am postgresql1
2018-01-18 14:31:16,392 INFO: no action.  i am the leader with the lock
2018-01-18 14:31:21,377 INFO: Lock owner: postgresql1; I am postgresql1
2018-01-18 14:31:21,392 INFO: no action.  i am the leader with the lock
2018-01-18 14:31:26,381 INFO: Lock owner: postgresql1; I am postgresql1
2018-01-18 14:31:26,396 INFO: no action.  i am the leader with the lock
```

Changing TTL, loop_wait, retry_timeout

ttl < loop_wait + retry_timeout * 2

```
$ patronictl edit-config batman
```

```
---
+++
@@ -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

```
2018-01-18 14:35:46,390 INFO: no action.  i am the leader with the
lock
2018-01-18 14:35:46,405 INFO: Lock owner: postgresql1; I am
postgresql1
2018-01-18 14:35:46,408 WARNING: Watchdog not supported because
leader TTL 5 is less than 2x loop_wait 10
2018-01-18 14:35:46,418 INFO: no action.  i am the leader with the
lock
2018-01-18 14:35:56,418 WARNING: Watchdog not supported because
leader TTL 5 is less than 2x loop_wait 10
2018-01-18 14:35:56,428 INFO: acquired session lock as a leader
2018-01-18 14:36:06,420 WARNING: Watchdog not supported because
leader TTL 5 is less than 2x loop_wait 10
2018-01-18 14:36:06,430 INFO: acquired session lock as a leader
```

Changing TTL, loop_wait, retry_timeout

ttl < loop_wait + retry_timeout * 2

```
2018-01-18 14:35:46,426 INFO: Lock owner: postgresql1; I am postgresql0
2018-01-18 14:35:46,426 INFO: does not have lock
2018-01-18 14:35:46,429 INFO: no action.  i am a secondary and i am
following a leader
2018-01-18 14:35:51,594 INFO: Got response from postgresql1
http://127.0.0.1:8008/patroni: b'{"state": "running",
"postmaster_start_time": "2018-01-18 13:44:44.764 CET", "role": "master",
"server_version": 100000, "xlog": {"location": 50331968}, "timeline": 2,
"replication": [{"usename": "replicator", "application_name": "postgresql1",
"client_addr": "127.0.0.1", "state": "streaming", "sync_state": "async",
"sync_priority": 0}], "database_system_identifier": "6512366775019348050",
"pending_restart": true, "patroni": {"version": "1.4", "scope": "batman"}}'
2018-01-18 14:35:51,680 WARNING: Master (postgresql1) is still alive
2018-01-18 14:35:51,683 INFO: following a different leader because i am not
the healthiest node
```

Change it back to original values

```
$ patronictl edit-config batman

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

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 /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

```
$ http http://127.0.0.1:8009/patroni
HTTP/1.0 200 OK
{
    "database_system_identifier": "6512366775019348050",
    "patroni": { "scope": "batman", "version": "1.4" },
    "postmaster_start_time": "2018-01-18 13:44:44.764 CET",
    "replication": [{
            "application_name": "postgresql0",
            "client_addr": "127.0.0.1",
            "state": "streaming",
            "sync_priority": 0,
            "sync_state": "async",
            "usename": "replicator"
        }],
    "role": "master",
    "server_version": 100000,
    "state": "running",
    "timeline": 2,
    "xlog": { "location": 50331968 }
}
```

GET /patroni on the replica

```
$ http http://127.0.0.1:8008/patroni
HTTP/1.0 200 OK
{
    "database_system_identifier": "6512366775019348050",
    "patroni": { "scope": "batman", "version": "1.4" },
    "postmaster_start_time": "2018-01-18 14:47:13.034 CET",
    "role": "replica",
    "server_version": 100000,
    "state": "running",
    "timeline": 2,
    "xlog": {
        "paused": false,
        "received_location": 50331648,
        "replayed_location": 50331968,
        "replayed_timestamp": null
    }
}
```

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).

Routing connections from clients

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

- Using callbacks:
  - on_start, on_stop, on_reload, on_restart, on_role_change,

- Using information from DCS (i.e. confd)

- HAProxy example: https://github.com/zalando/patroni/tree/master/extras/confd
  - Can be adopted to Pgbouncer
- Using jdbc:
`jdbc:postgresql://node1,node2,node3/postgres?targetServerType=master`
- libpq starting from PostgreSQL 10:
`postgresql://host1:port2,host2:port2/?target_session_attrs=read-write`

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

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.

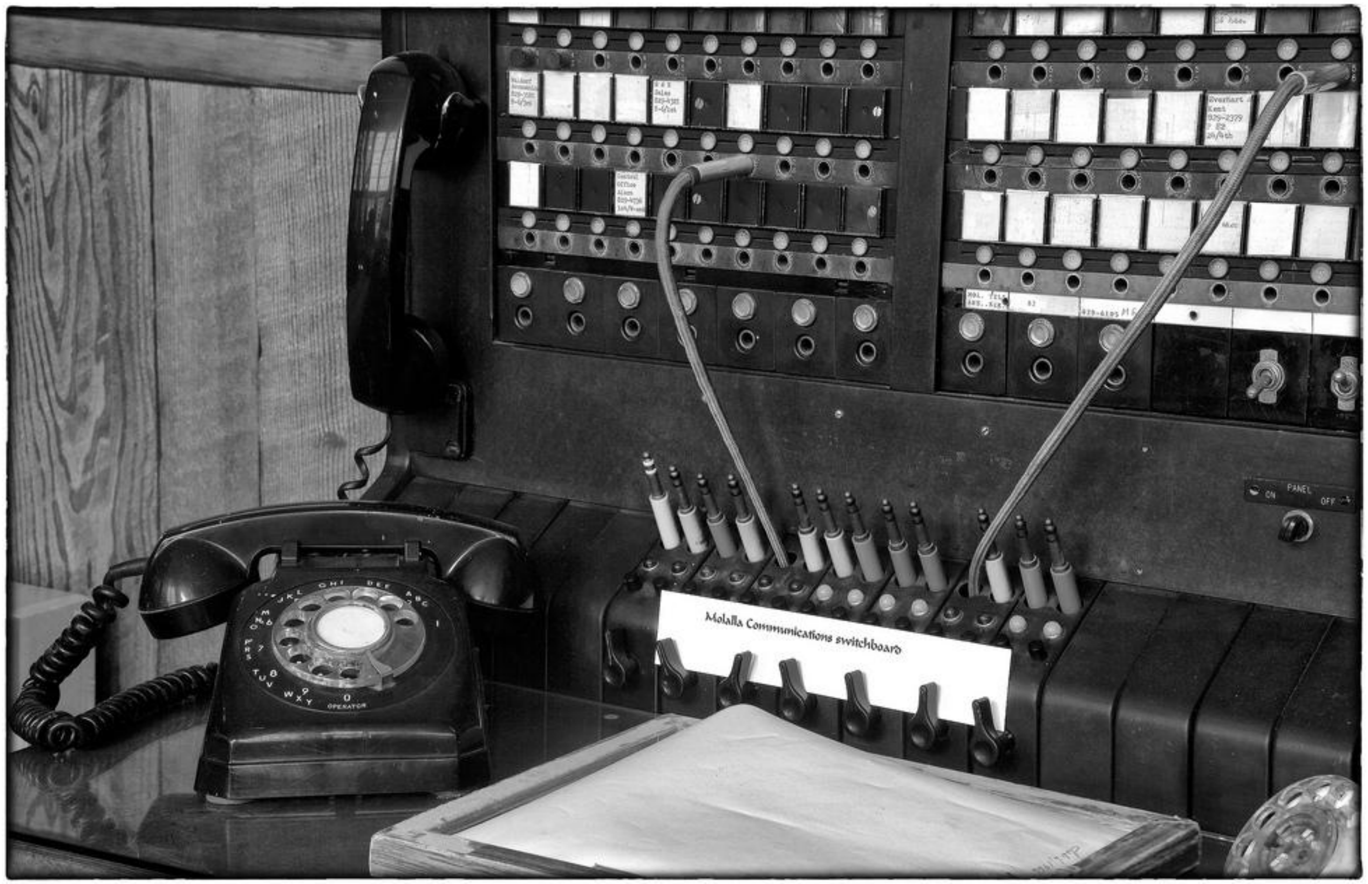
*** Tags are configured on every node individually**

Using replicatefrom to create cascading replication

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

HINT: look at postgres2.yml

Switchover and failover



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

```
$ patronictl switchover batman
Master [postgresql1]:
Candidate ['postgresql0'] []:
When should the switchover take place (e.g. 2015-10-01T14:30)
[now]:
Current cluster topology
+---------+-------------+-----------+--------+---------+-----------+
| Cluster | Member      | Host      |  Role  |  State  | Lag in MB |
+---------+-------------+-----------+--------+---------+-----------+
| batman  | postgresql0 | 127.0.0.1 |        | running |         0 |
| batman  | postgresql1 | 127.0.0.1 | Leader | running |         0 |
+---------+-------------+-----------+--------+---------+-----------+

Are you sure you want to switchover cluster batman, demoting current
master postgresql1? [y/N]: y
2018-01-18 16:22:12.21399 Successfully failed over to "postgresql0"
```

Switchover with patronictl (continue)

```
$ patronictl list batman
+---------+-------------+-----------+--------+---------+-----------+
| Cluster | Member      | Host      |  Role  |  State  | Lag in MB |
+---------+-------------+-----------+--------+---------+-----------+
| batman  | postgresql0 | 127.0.0.1 | Leader | running |         0 |
| batman  | postgresql1 | 127.0.0.1 |        | stopped |   unknown |
+---------+-------------+-----------+--------+---------+-----------+
```

```
$ patronictl list batman
+---------+-------------+-----------+--------+---------+-----------+
| Cluster | Member      | Host      |  Role  |  State  | Lag in MB |
+---------+-------------+-----------+--------+---------+-----------+
| batman  | postgresql0 | 127.0.0.1 | Leader | running |         0 |
| batman  | postgresql1 | 127.0.0.1 |        | running |         0 |
+---------+-------------+-----------+--------+---------+-----------+
```

Scheduled switchover

```
$ patronictl switchover batman
Master [postgresql0]:
Candidate ['postgresql1'] []:
When should the switchover take place (e.g. 2015-10-01T14:30)  [now]:
2018-01-18T16:27
Current cluster topology
+---------+-------------+-----------+--------+---------+-----------+
| Cluster | Member      | Host      |  Role  |  State  | Lag in MB |
+---------+-------------+-----------+--------+---------+-----------+
| batman  | postgresql0 | 127.0.0.1 | Leader | running |         0 |
| batman  | postgresql1 | 127.0.0.1 |        | running |         0 |
+---------+-------------+-----------+--------+---------+-----------+
Are you sure you want to switchover cluster batman, demoting current master
postgresql0? [y/N]: y
2018-01-18 16:26:35.45274 Switchover scheduled
+---------+-------------+-----------+--------+---------+-----------+
| Cluster | Member      | Host      |  Role  |  State  | Lag in MB |
+---------+-------------+-----------+--------+---------+-----------+
| batman  | postgresql0 | 127.0.0.1 | Leader | running |         0 |
| batman  | postgresql1 | 127.0.0.1 |        | running |         0 |
+---------+-------------+-----------+--------+---------+-----------+
 Switchover scheduled at: 2018-01-18T16:27:00+01:00
                   from: postgresql0
```

Scheduled restarts

```
$ patronictl restart batman postgresql1
+---------+-------------+-----------+--------+---------+-----------+
| Cluster | Member      | Host      |  Role  |  State  | Lag in MB |
+---------+-------------+-----------+--------+---------+-----------+
| batman  | postgresql0 | 127.0.0.1 |        | running |         0 |
| batman  | postgresql1 | 127.0.0.1 | Leader | running |         0 |
+---------+-------------+-----------+--------+---------+-----------+
Are you sure you want to restart members postgresql1? [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]:
2018-01-18T16:31:00
Success: restart scheduled on member postgresql1

$ patronictl list batman
+---------+-------------+-----------+--------+---------+-----------+---------------------------+
| Cluster | Member      | Host      |  Role  |  State  | Lag in MB | Scheduled restart         |
+---------+-------------+-----------+--------+---------+-----------+---------------------------+
| batman  | postgresql0 | 127.0.0.1 |        | running |         0 |                           |
| batman  | postgresql1 | 127.0.0.1 | Leader | running |         0 | 2018-01-18T16:31:00+01:00 |
+---------+-------------+-----------+--------+---------+-----------+---------------------------+
```

Scheduled restarts

```
2018-01-18 16:30:41,497 INFO: Awaiting restart at
2018-01-18T16:31:00+01:00 (in 19 seconds)
2018-01-18 16:30:41,507 INFO: no action.  i am the leader with the lock
2018-01-18 16:30:51,497 INFO: Lock owner: postgresql1; I am postgresql1
2018-01-18 16:31:00,003 INFO: Manual scheduled restart at
2018-01-18T16:31:00+01:00
2018-01-18 16:31:00,024 INFO: restart initiated
2018-01-18 16:31:00.234 CET [37661] LOG:  received fast shutdown request
CET
2018-01-18 16:31:00.372 CET [38270] FATAL:  the database system is
starting up
2018-01-18 16:31:00.386 CET [38267] LOG:  database system is ready to
accept connections
2018-01-18 16:31:00,627 INFO: Lock owner: postgresql1; I am postgresql1
2018-01-18 16:31:00,628 INFO: establishing a new patroni connection to
the postgres cluster
2018-01-18 16:31:00,770 INFO: no action.  i am the leader with the lock
```

Reinitialize (don’t repeat GitLab mistake)

```
$ patronictl reinit batman postgresql0
+---------+-------------+-----------+--------+---------+-----------+
| Cluster | Member      | Host      |  Role  |  State  | Lag in MB |
+---------+-------------+-----------+--------+---------+-----------+
| batman  | postgresql0 | 127.0.0.1 |        | running |       0.0 |
| batman  | postgresql1 | 127.0.0.1 | Leader | running |       0.0 |
+---------+-------------+-----------+--------+---------+-----------+
Are you sure you want to reinitialize members postgresql0? [y/N]: y
Success: reinitialize for member postgresql0

$ patronictl list batman
+---------+-------------+-----------+--------+------------------+-----------+
| Cluster | Member      | Host      |  Role  |      State       | Lag in MB |
+---------+-------------+-----------+--------+------------------+-----------+
| batman  | postgresql0 | 127.0.0.1 |        | creating replica |   unknown |
| batman  | postgresql1 | 127.0.0.1 | Leader |     running      |       0.0 |
+---------+-------------+-----------+--------+------------------+-----------+
```

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

zalando

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

```
$ patronictl pause batman --wait
'pause' request sent, waiting until it is recognized by all nodes
Success: cluster management is paused

$ patronictl list batman
+---------+-------------+-----------+--------+---------+-----------+
| Cluster | Member      | Host      |  Role  |  State  | Lag in MB |
+---------+-------------+-----------+--------+---------+-----------+
| batman  | postgresql0 | 127.0.0.1 |        | running |         0 |
| batman  | postgresql1 | 127.0.0.1 | Leader | running |         0 |
+---------+-------------+-----------+--------+---------+-----------+
 Maintenance mode: on

2018-01-19 15:51:43,908 INFO: Lock owner: postgresql1; I am postgresql1
2018-01-19 15:51:43,931 INFO: no action.  i am the leader with the lock
2018-01-19 15:51:46,864 INFO: Lock owner: postgresql1; I am postgresql1
2018-01-19 15:51:46,890 INFO: PAUSE: no action.  i am the leader with the lock
```

zalando

Pause mode (promoting another master)

```
$ pg_ctl -D data/postgresql0 promote
waiting for server to promote.... done
server promoted

2018-01-19 15:54:12.058 CET [81603] LOG:  received promote request
2018-01-19 15:54:12.058 CET [81638] FATAL:  terminating walreceiver process due to
administrator command
2018-01-19 15:54:12.062 CET [81603] LOG:  invalid record length at 0/3000060:
wanted 24, got 0
2018-01-19 15:54:12.062 CET [81603] LOG:  redo done at 0/3000028
2018-01-19 15:54:12.065 CET [81603] LOG:  selected new timeline ID: 2
2018-01-19 15:54:12.113 CET [81603] LOG:  archive recovery complete
2018-01-19 15:54:12.118 CET [81601] LOG:  database system is ready to accept
connections
2018-01-19 15:54:16,872 INFO: Lock owner: postgresql1; I am postgresql0
2018-01-19 15:54:16,872 INFO: does not have lock
2018-01-19 15:54:16,901 INFO: PAUSE: continue to run as master without lock
```

Pause mode (promoting another master)

```
$ http http://127.0.0.1:8008/master
HTTP/1.0 503 Service Unavailable
{
    "database_system_identifier": "6512774501076700824",
    "patroni": {
        "scope": "batman",
        "version": "1.4"
    },
    "pause": true,
    "postmaster_start_time": "2018-01-19 15:51:31.879 CET",
    "role": "master",
    "server_version": 100000,
    "state": "running",
    "timeline": 2,
    "xlog": {
        "location": 50332016
    }
}
```

Pause mode (resuming)

```
$ patronictl resume batman
Success: cluster management is resumed

2018-01-19 15:57:31,324 INFO: Lock owner: postgresql1; I am postgresql0
2018-01-19 15:57:31,324 INFO: does not have lock
2018-01-19 15:57:31.379 CET [81601] LOG:  received immediate shutdown
request
2018-01-19 15:57:31.380 CET [81720] WARNING:  terminating connection
because of crash of another server process
2018-01-19 15:57:31,805 INFO: Lock owner: postgresql1; I am postgresql0
2018-01-19 15:57:31,805 INFO: does not have lock
2018-01-19 15:57:32,021 INFO: Local timeline=2 lsn=0/3000170
2018-01-19 15:57:32,030 INFO: master_timeline=1
2018-01-19 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!
2018-01-19 15:57:33,560 INFO: Lock owner: postgresql1; I am postgresql0
2018-01-19 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

```
$ patronictl edit-config batman
---
+++
@@ -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

```
2018-01-19 16:33:11,329 INFO: Assigning synchronous standby
status to postgresql1
server signaled
2018-01-19 16:33:11.367 CET [81568] LOG:  received SIGHUP,
reloading configuration files
2018-01-19 16:33:11.380 CET [81568] LOG:  parameter
"synchronous_standby_names" changed to "postgresql1"
2018-01-19 16:33:13,377 INFO: Synchronous standby status
assigned to postgresql1
2018-01-19 16:33:13,385 INFO: no action.  i am the leader with
the lock
2018-01-19 16:33:13.993 CET [83425] LOG:  standby "postgresql1"
is now a synchronous standby with priority 1
2018-01-19 16:33:21,312 INFO: Lock owner: postgresql0; I am
postgresql0
```

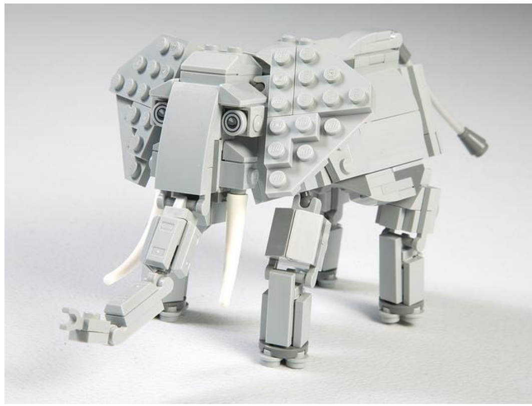
Synchronous replication REST endpoints

```
$ patronictl list  batman
+---------+-------------+-----------+--------------+---------+-----------+
| Cluster | Member      | Host      |     Role     |  State  | Lag in MB |
+---------+-------------+-----------+--------------+---------+-----------+
| batman  | postgresql0 | 127.0.0.1 | Sync standby | running |         0 |
| batman  | postgresql1 | 127.0.0.1 |    Leader    | running |         0 |
+---------+-------------+-----------+--------------+---------+-----------+
```

```
$ http OPTIONS http://127.0.0.1:8008/sync
HTTP/1.0 200 OK

$ http OPTIONS http://127.0.0.1:8008/async
HTTP/1.0 503 Service Unavailable
```

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
```

zalando

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=batman \
--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.

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="2018-01-19 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: "2018-01-19 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: batman # 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: postgresql0 # postgresql node name

restapi:
  # restapi configuration

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

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
```

Restapi configuration

```
restapi:
  listen: 0.0.0.0:8008 # address to listen to for REST API requests
  connect_address: 127.0.0.1:8008 # 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)
```

DCS configuration

```
etcd:
  host:         127.0.0.1:2379
# protocol:     http
# username:     etcd
# password:     v4rY$ecRetW0rd
# cacert:       /etc/ssl/ca.crt
# cert:         /etc/ssl/cert.crt
# key:          /etc/ssl/key.key

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

DCS 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
```

zalando

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: "2018-01-19 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: 127.0.0.1:5432 # 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"
```

zalando

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
```

zalando

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: postgresql1
```

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

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 postgresql0
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 'postgresql0'
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 "postgresql1" does not exist
2018-01-23 14:07:34.526 CET [28588] FATAL:  could not start WAL streaming: ERROR:
replication slot "postgresql1" 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 postgresql0
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 postgres0.yml
2018-01-23 14:53:27,878 CRITICAL: system ID
mismatch, node postgresql0 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
```

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

Thank you!