zalando

Pump up your elephants with Patroni

PGDay.IT 2018

Lazise

ALEXANDER KUKUSHKIN

29-06-2018

ABOUT ME

Alexander Kukushkin

Database Engineer @ZalandoTech
Email: alexander.kukushkin@zalando.de
Twitter: @cyberdemn

zalando

AGENDA

PostgreSQL at Zalando

Brief introduction to automatic failover

Bot pattern and Patroni

Live-demo

Spilo & Patroni at Zalando

Managing clusters with Patronictl

WE BRING FASHION TO PEOPLE IN 17 COUNTRIES

17 markets

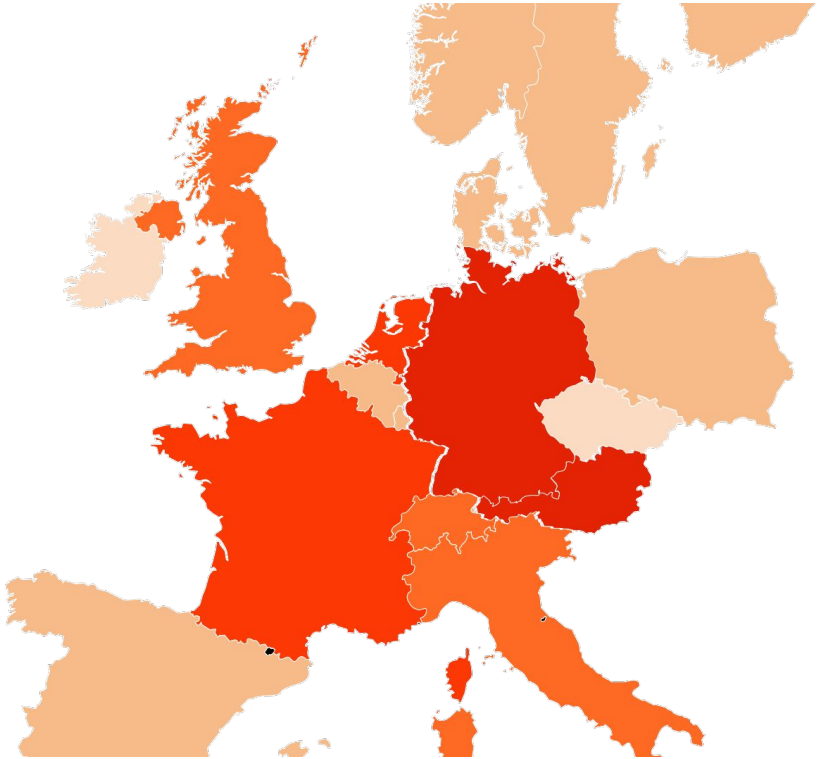
7 fulfillment centers

23 million active customers

4.5 billion € net sales 2017

200 million visits per month

15,000 employees in Europe



zalando

FACTS & FIGURES

> 300 databases
on premise

> 170
on AWS EC2

> 400
on K8S

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

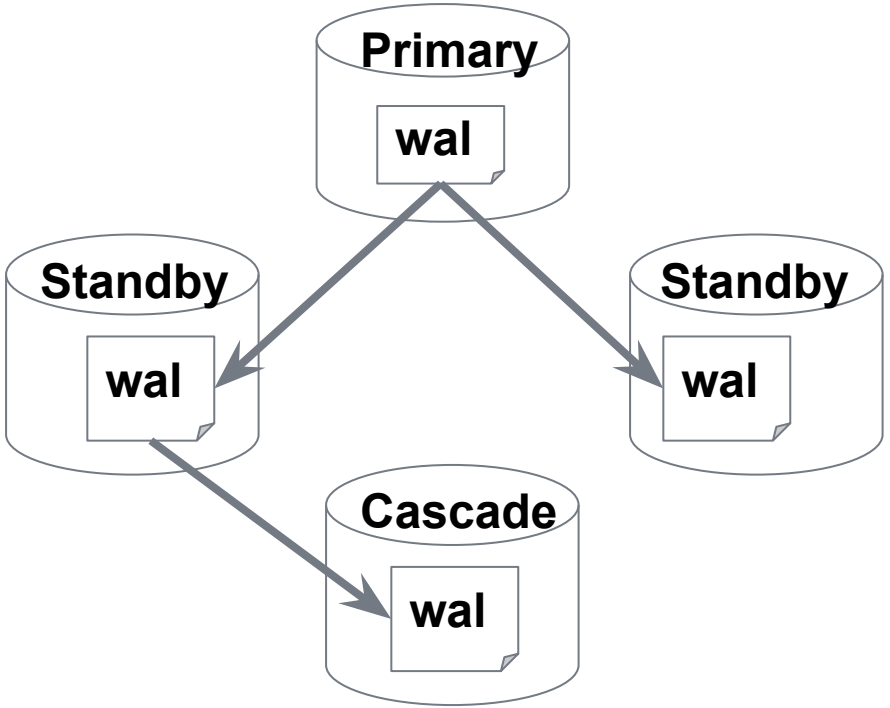
Physical single-master replication

Cons

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

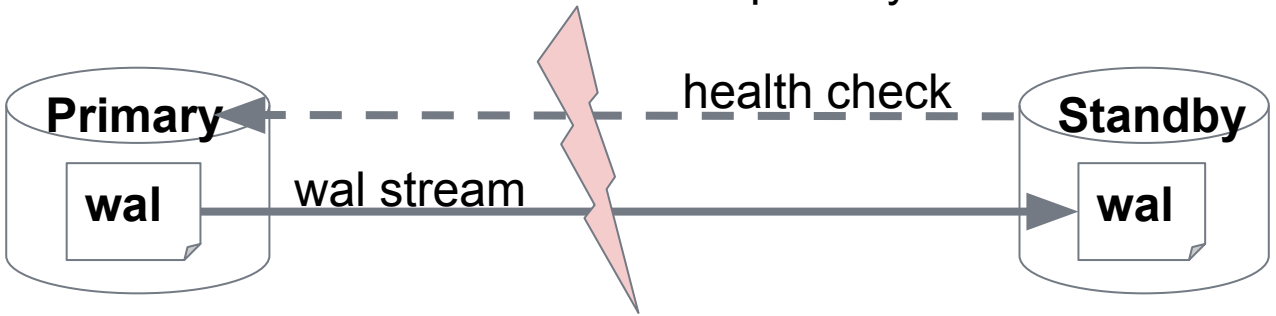
Pros

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



Automatic failover done wrong: Running just two nodes

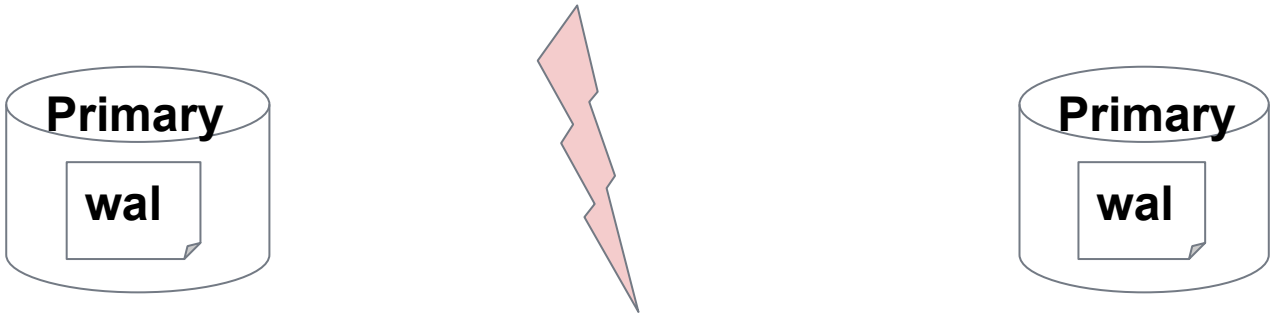
Run the **health check** from the standby and promote that standby when the health check indicates the primary failure



zalando

Automatic failover done wrong: running just two nodes

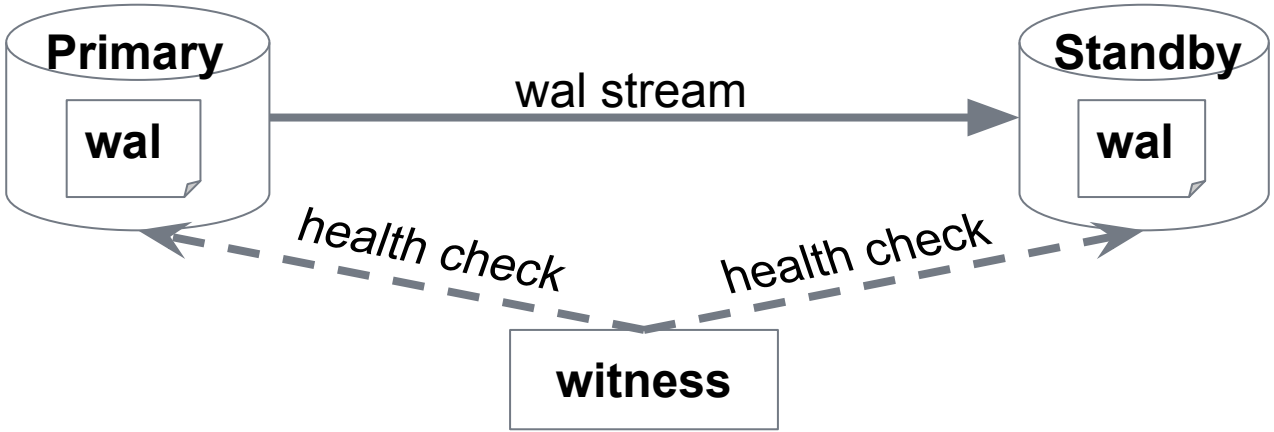
Split-brain!



zalando

Automatic failover done wrong: Single witness node

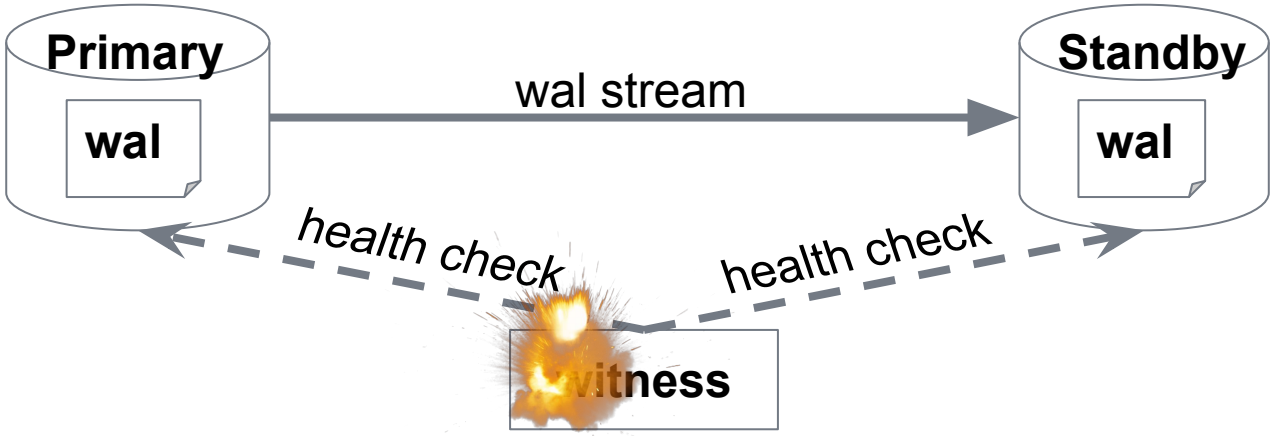
What can possibly go wrong?



zalando

Automatic failover done wrong: Single witness node

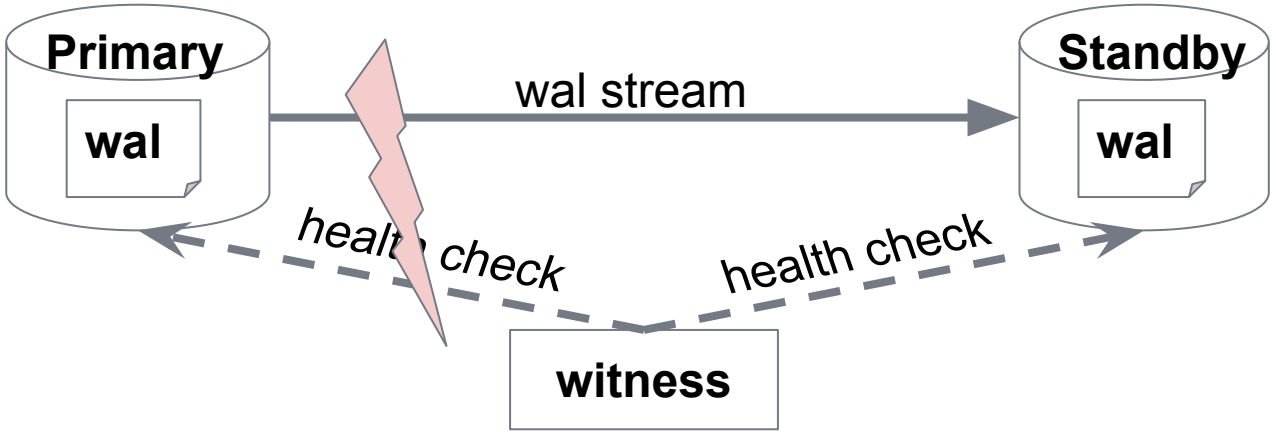
Witness node dies



zalando

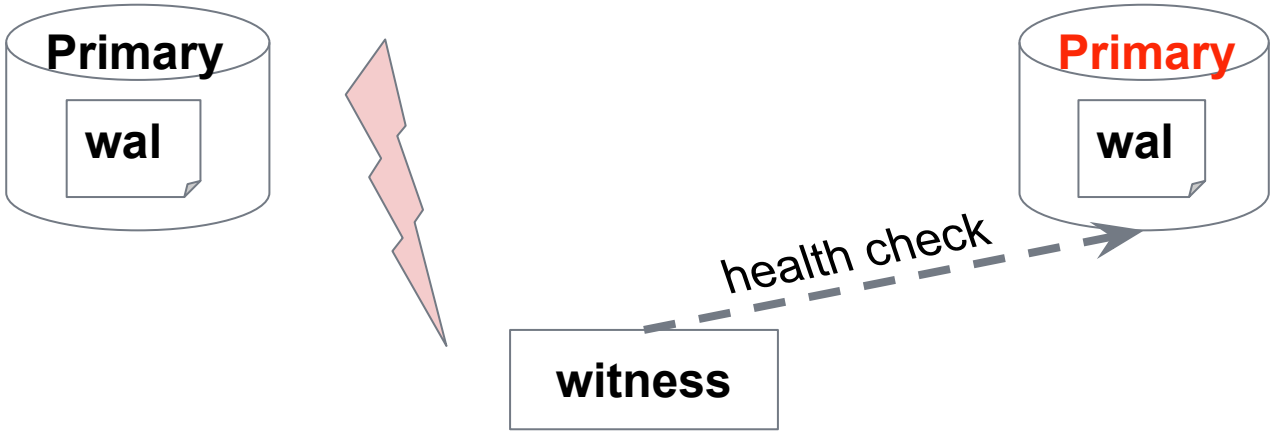
Automatic failover done wrong: Single witness node

Or gets partitioned

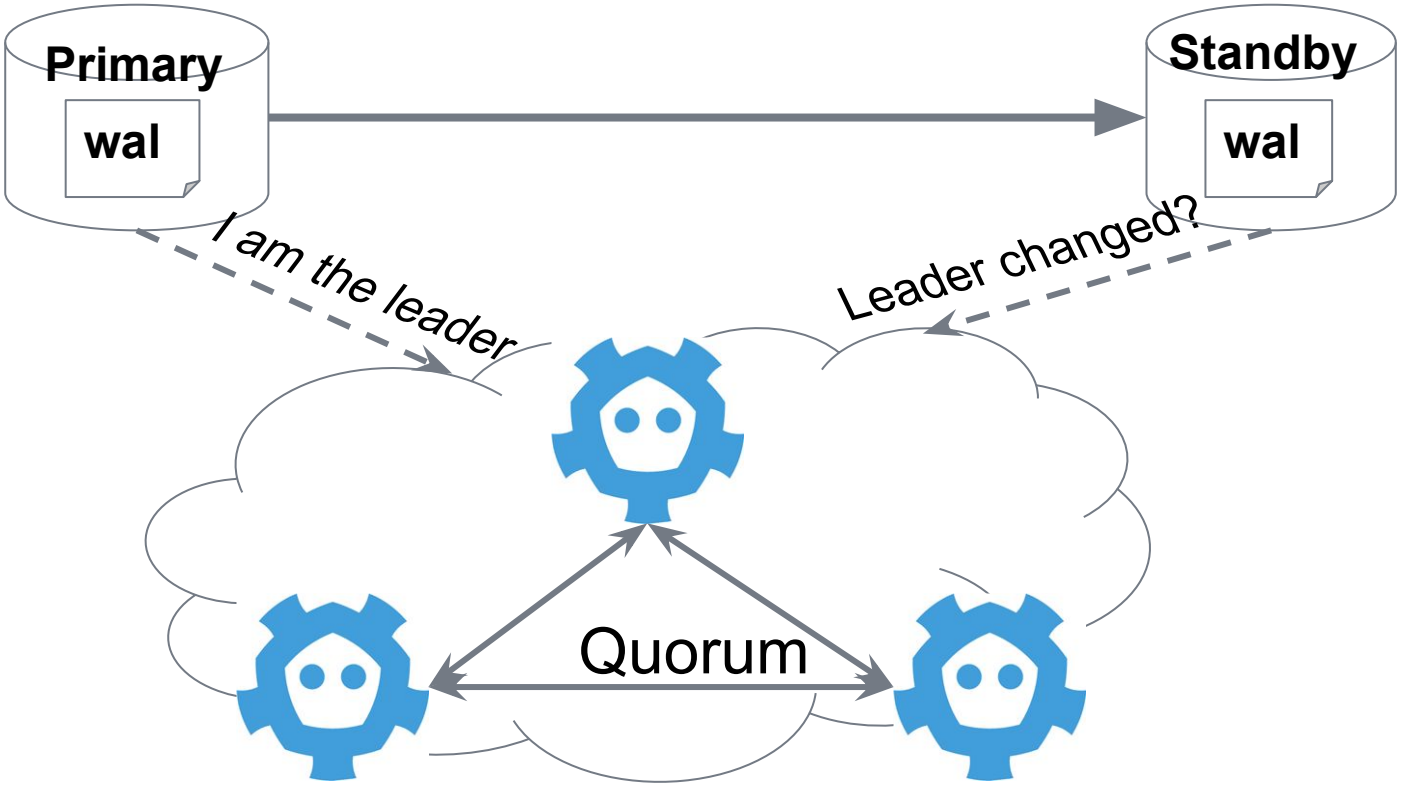


Automatic failover done wrong: Single witness node

Existing primary is running



Automatic failover done right

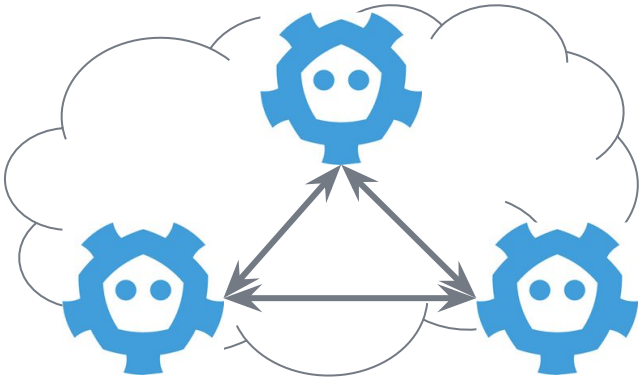


zalando

Etcd consistency store

- Distributed key-value store
- Implements RAFT
- Needs odd number of nodes (optimal: 3 or 5)

http://thesecretlivesofdata.com/raft/

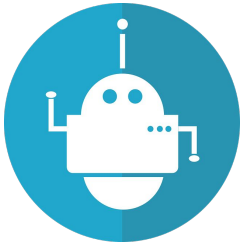


Automatic failover: the right way

- Cluster state stored in a consistent distributed storage
- Leader key changed via atomic CAS operations
- Leader elections among all members of the cluster
- Each member decides only for itself
- 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

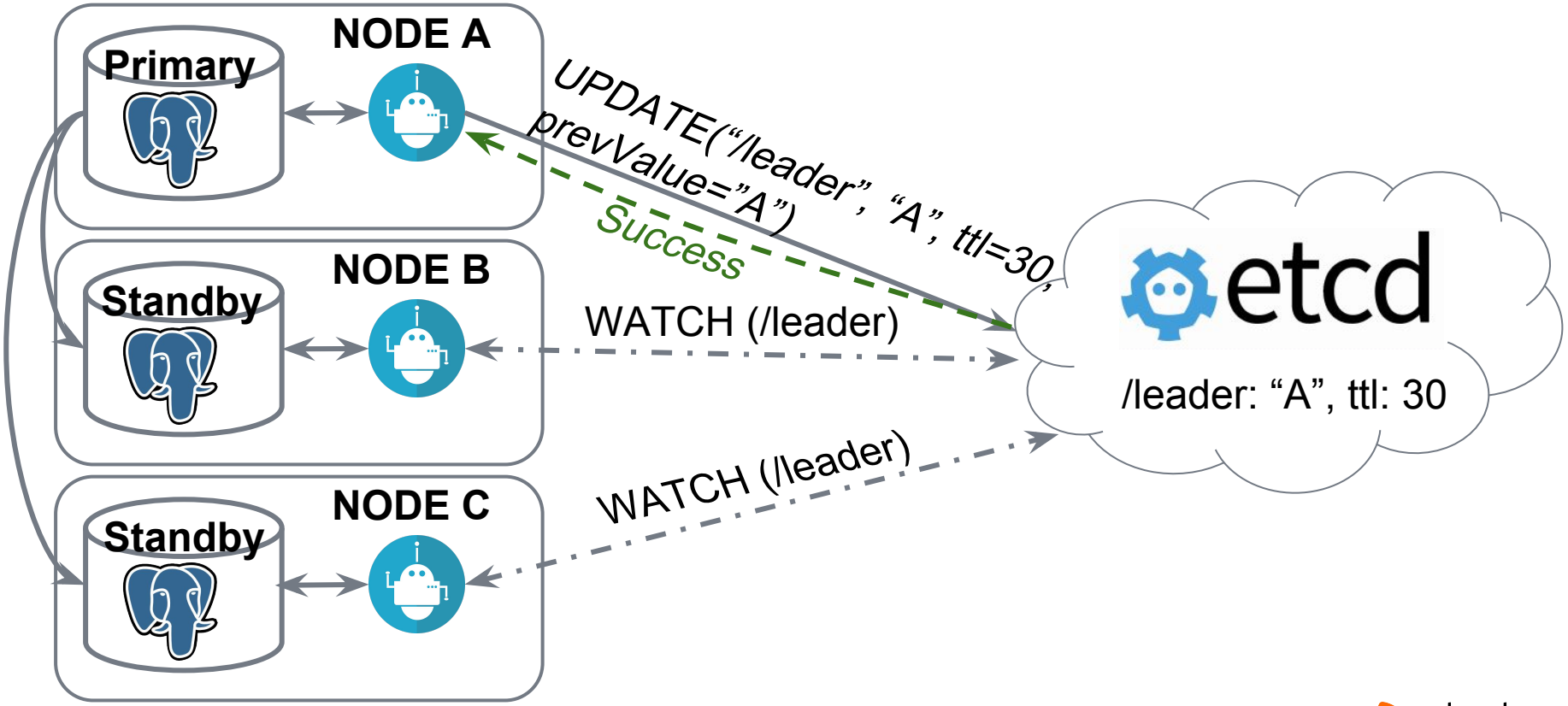


Don’t like Etcd?

No problem, Patroni also supports:

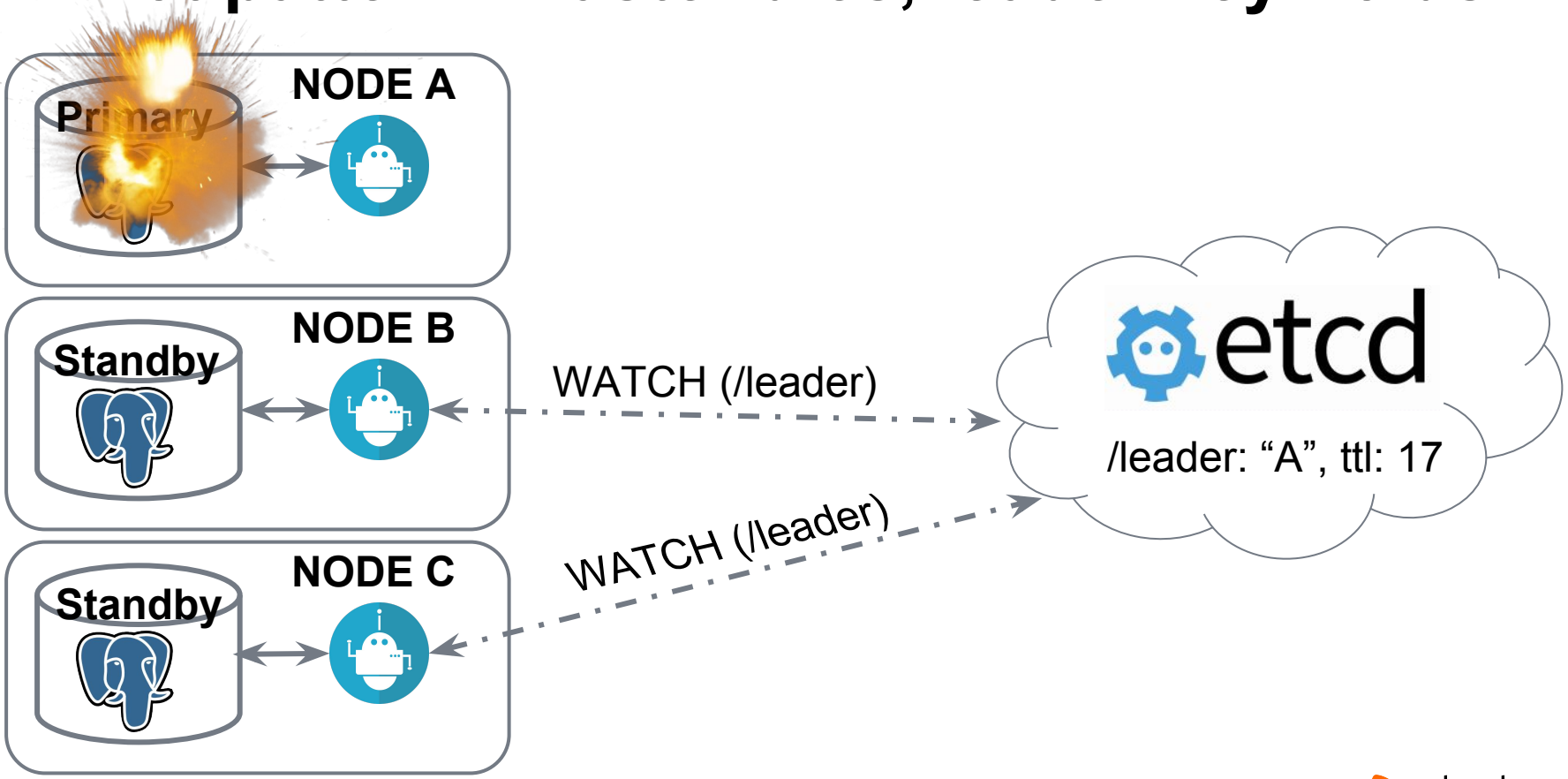
- ZooKeeper
- Consul
- Kubernetes API (kube-native deployment)

Bot pattern: leader alive



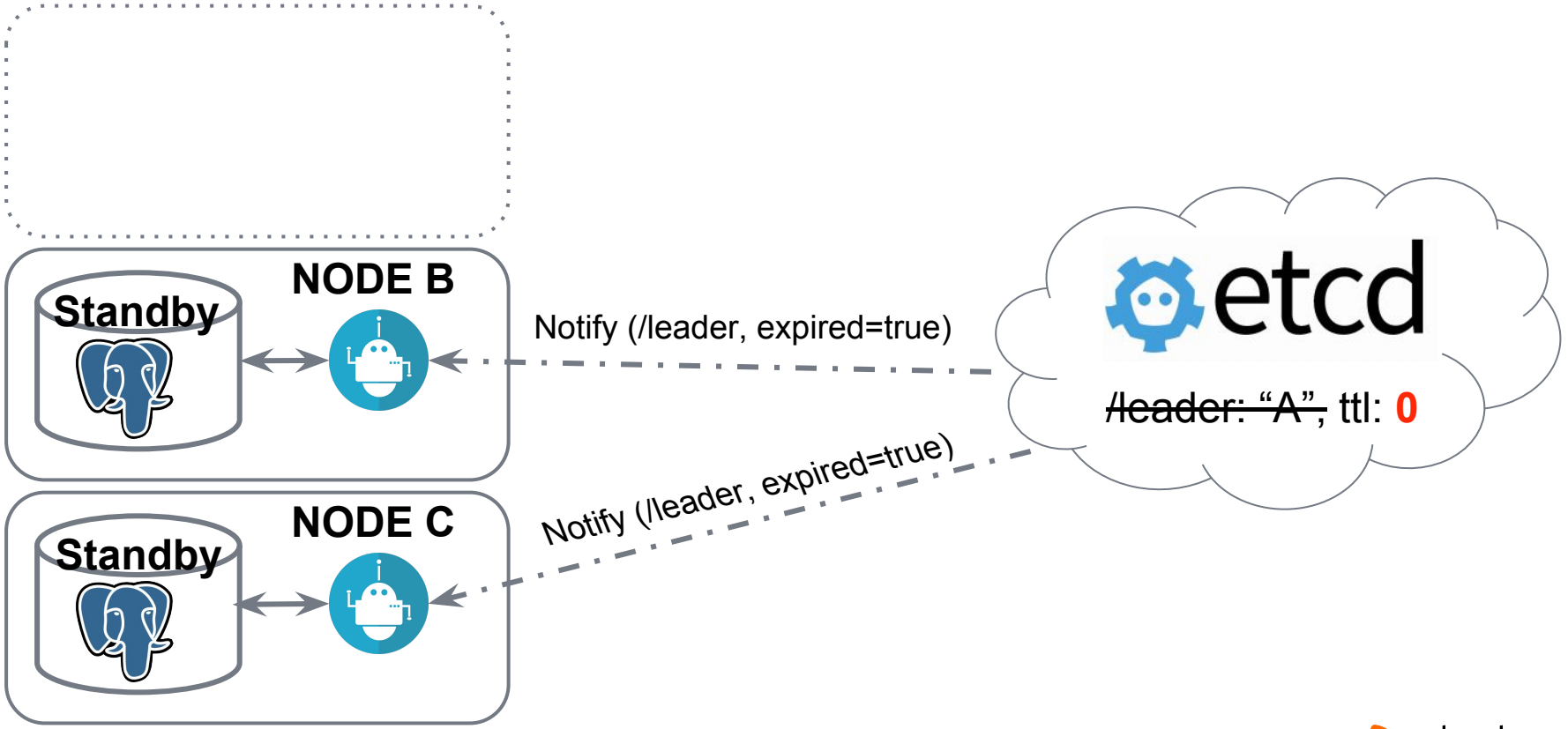
zalando

Bot pattern: master dies, leader key holds



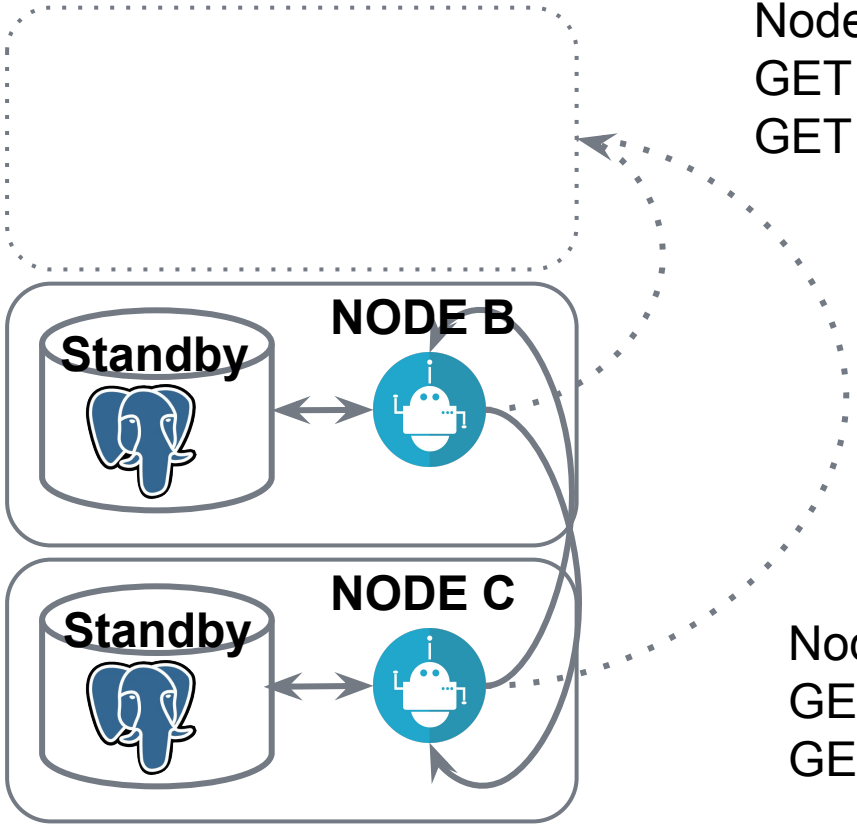
zalando

Bot pattern: leader key expires



zalando

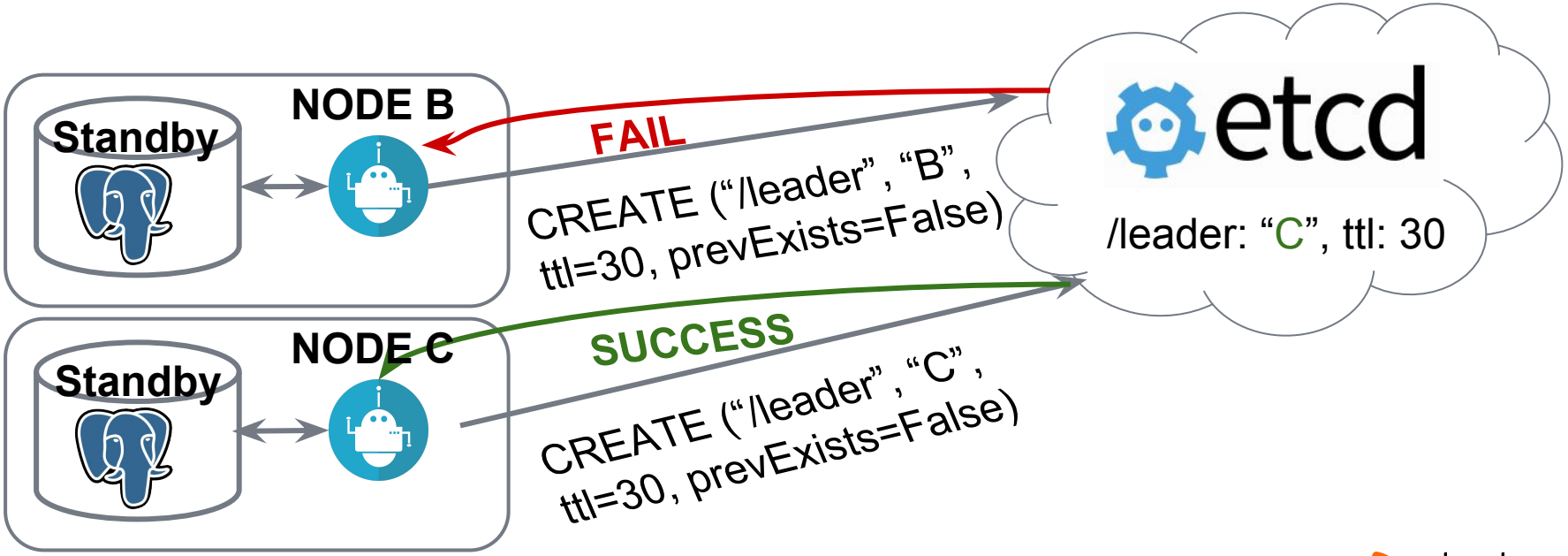
Bot pattern: who will be the next master?



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

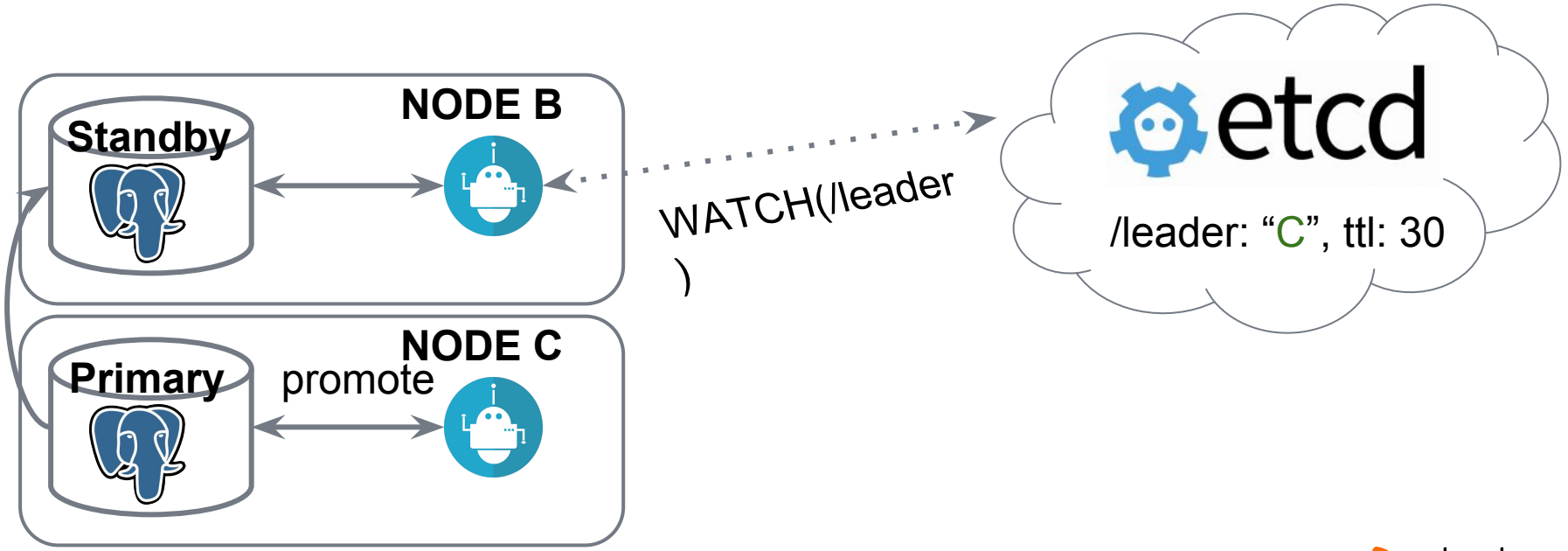
Node **C**:
GET A:8008/patroni -> **failed/timeout**
GET B:8008/patroni -> wal_position: **100**

Bot pattern: leader race among equals



zalando

Bot pattern: promote and continue replication



DCS structure

- /service/cluster-name/
  - config — {"postgresql":{"parameters":{"max_connections":300}}}
  - initialize — ”6303731710761975832” (database system identifier)
  - members/
    - *dbnode1* — {"role":"replica","state":"running”,"conn_url":"postgres://172.17.0.2:5432/postgres"}
    - *dbnode2* — {"role":"**master**","state":"running”,"conn_url":"postgres://172.17.0.3:5432/postgres"}
  - *leader* — ***dbnode2***
  - optime/
    - leader — “67393608” # ← absolute wal positition

Client traffic routing

- HAProxy + confd
  - confd to generate/update HAProxy config and restart/reload
  - HAProxy runs active health-checks against Patroni REST API to find the primary
  - use “*on-marked-down shutdown-sessions*” to close client connection when primary health-check fails
- pgbouncer + confd
  - confd to generate/update pgbouncer config and restart/reload
- callback scripts (on_start, on_stop, on_role_change) to move around Floating/Elastic IP
  - Beware of race conditions.See github.com/zalando/patroni/issues/536 for more details
- Vip-manager
  - https://github.com/cybertec-postgresql/vip-manager
- On Kubernetes:
  - Patroni updates **subsets** of master **Endpoint** with IP of master Pod.
  - Service + labelSelector for read-only load-balancing

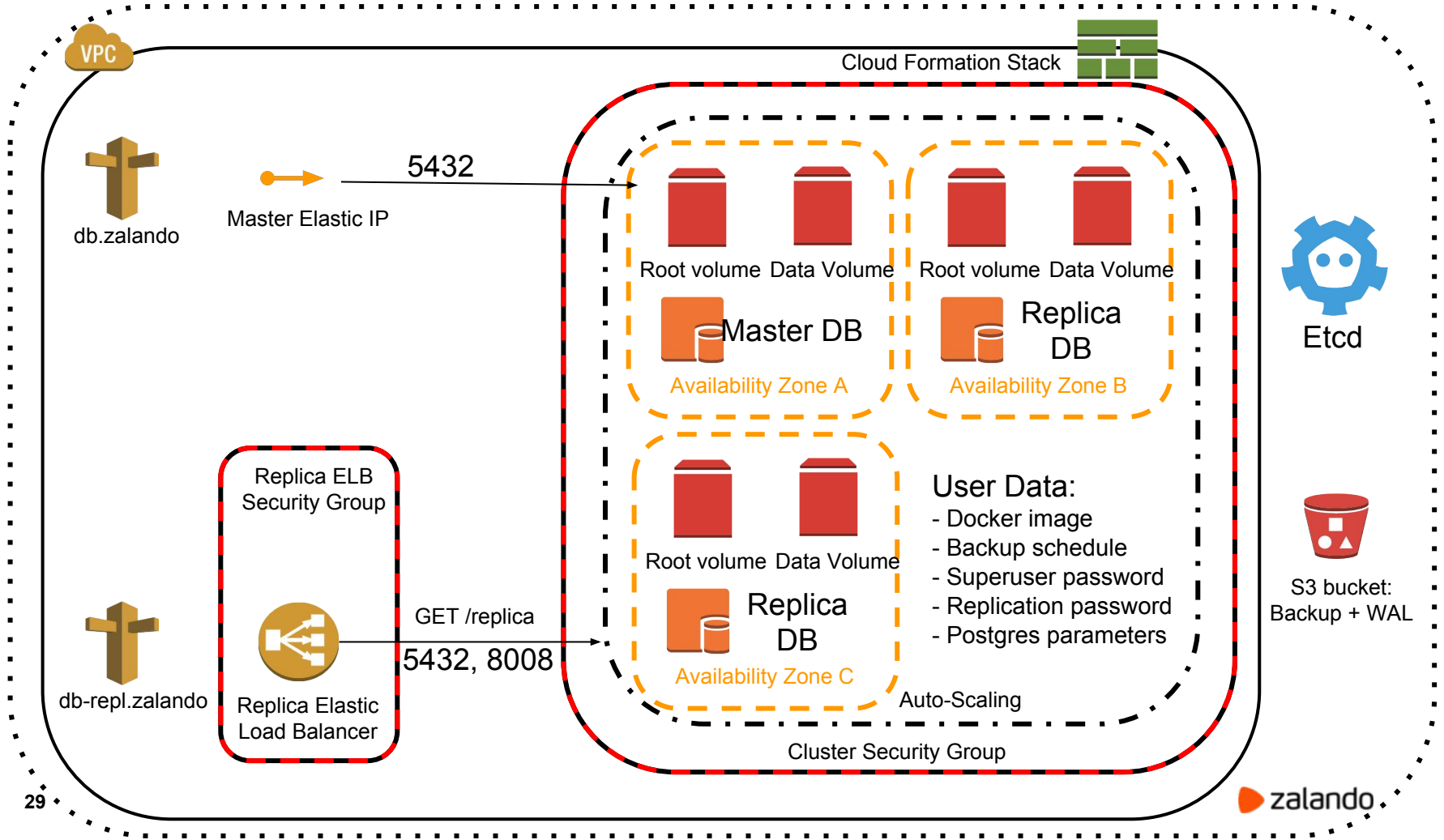
DEMO TIME

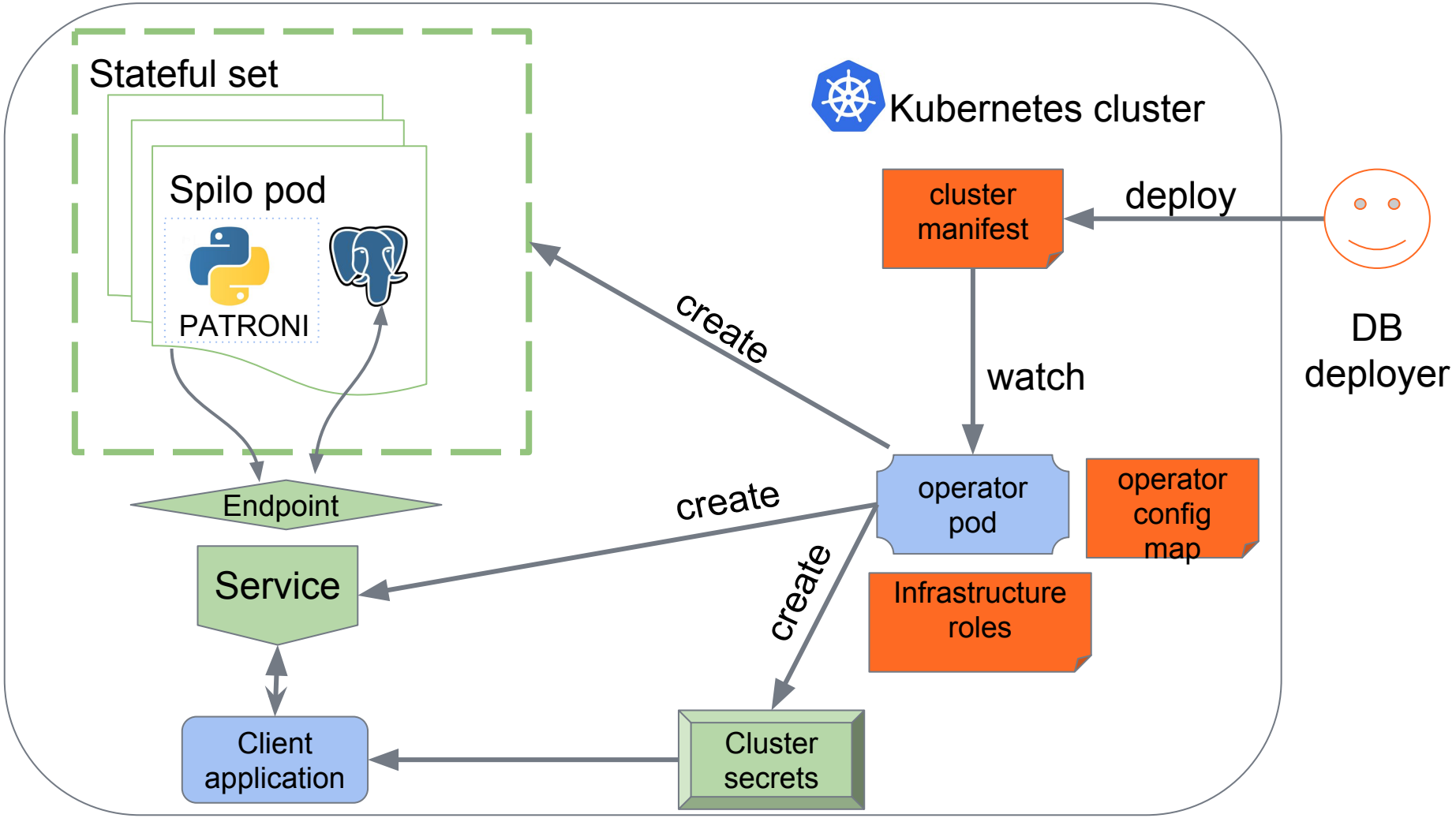
How we run Patroni

- On-premise
  - ZooKeeper cluster of three nodes as DCS
  - Primary + Replica in the same data-center

- On AWS
  - Etcd cluster of five nodes
  - Deployment with Cloudformation (2 or 3 EC2 in ASG, KMS to encrypt passwords, ElasticIP for traffic routing, ELB for accessing replicas)
  - One docker container per EC2 instance
  - All PostgreSQL clusters (> 170) are using the same Etcd cluster

- On Kubernetes
  - We have a few dozens of Kubernetes cluster
  - On every cluster we run postgres-operator
    - postgres-operator deploys Patroni clusters on-demand


VPC
Cloud Formation Stack
db.zalando
Master Elastic IP
5432
Root volume
Data Volume
Master DB
Availability Zone A
Root volume
Data Volume
Replica DB
Availability Zone B
Etcd
Replica ELB Security Group
Root volume
Data Volume
Replica DB
Availability Zone C
User Data:
- Docker image
- Backup schedule
- Superuser password
- Replication password
- Postgres parameters
S3 bucket: Backup + WAL
db-repl.zalando
GET /replica
5432, 8008
Replica Elastic Load Balancer
Auto-Scaling
Cluster Security Group
zalando


Stateful set
Spilo pod
PATRONI
Kubernetes cluster
cluster manifest
deploy
DB deployer
create
watch
operator pod
operator config map
Endpoint
create
Service
create
Infrastructure roles
Client application
Cluster secrets

Spilo: Docker with PostgreSQL and Patroni

- Docker image to deploy Patroni on AWS and Kubernetes
- All supported versions of PostgreSQL (9.3 - 10) in one image
- Patroni
- bunch of useful 3rd party extensions
- WAL-E
- cron
- script to build Patroni configuration from environment
- callback scripts
- pam-oauth2 - 1 hour OAuth tokens instead of password
- bg_mon - pg_view in browser

Patroni features

- Automated cluster deployment (initialization race)
- Automatic failover
- Synchronous and Cascading replication
- Linux watchdog
- Custom bootstrap (restore from basebackup vs. initdb)
- Custom replica creation methods (pg_basebackup vs. wal-e/Barman/pgBackRest/wal-g)
- REST API (status, health-check, reinit, restart, reload, switchover and so on)
- Callbacks (on_start, on_stop, on_restart, on_reload, on_role_change)
- Manual and Scheduled Failover/Switchover
- Scheduled restarts
- Cluster-wide dynamic configuration
- Pause (maintenance) mode
- Tags (nofailover, clonefrom, replicatefrom, noloadbalance, nosync)
- patronictl

patronictl

- list — List the Patroni members for a given cluster
- show-config — Show cluster configuration
- edit-config — Edit cluster configuration
- failover — Failover to a replica (when automatic failover didn’t happen due to replication lag)
- switchover — Switchover to a replica, can be also scheduled
- pause — Disable auto failover
- resume — Resume auto failover
- restart — Restart cluster member, can be also scheduled
- flush — Flush scheduled restarts
- reinit — Reinitialize cluster member

Most of the commands are interactive (asking questions and confirmations), but they also support “--force” argument, which is useful for scripting.

patronictl examples: pause

```
$ patronictl -c postgres0.yml pause batman --wait
'pause' request sent, waiting until it is recognized by all nodes
Success: cluster management is paused

$ patronictl -c postgres0.yml list # cluster_name (batman) is taken from config
+---------+-------------+-----------+--------+---------+-----------+
| Cluster |    Member   |    Host   |  Role  |  State  | Lag in MB |
+---------+-------------+-----------+--------+---------+-----------+
|  batman | postgresql0 | 127.0.0.1 | Leader | running |       0.0 |
|  batman | postgresql1 | 127.0.0.2 |        | running |       0.0 |
+---------+-------------+-----------+--------+---------+-----------+
Maintenance mode: on
```

patronictl examples: edit-config

```
$ patronictl -c postgres0.yml edit-config -p max_connections=200 -s synchronous_mode=on --force

---
+++
@@ -12,7 +12,9 @@
    log_rotation_age: 1d
    log_truncate_on_rotation: 'on'
    logging_collector: 'on'
+   max_connections: 200
    wal_level: logical
  use_pg_rewind: true
 retry_timeout: 10
+synchronous_mode: true
 ttl: 30

Configuration changed
```

patronictl examples: show-config

```
$ patronictl -c postgres0.yml show-config
loop_wait: 10
maximum_lag_on_failover: 1048576
pause: true
postgresql:
  parameters:
    archive_command: true
    archive_mode: 'on'
    max_connections: 200
    wal_level: logical
  use_pg_rewind: true
retry_timeout: 10
synchronous_mode: true
ttl: 30
```

patronictl examples: restart

```
$ patronictl -c postgres0.yml restart batman postgresql1 --force
+---------+------------+-----------+--------------+---------+-----------+-----------------+
| Cluster |    Member  |    Host   |     Role     |  State  | Lag in MB | Pending restart |
+---------+------------+-----------+--------------+---------+-----------+-----------------+
|  batman | postgresql0 | 127.0.0.1 |    Leader    | running |       0.0 |        *        |
|  batman | postgresql1 | 127.0.0.2 | Sync standby | running |       0.0 |        *        |
+---------+------------+-----------+--------------+---------+-----------+-----------------+
Success: restart on member postgresql1
$ patronictl -c postgres0.yml restart batman postgresql0  # interactive mode
+---------+------------+-----------+--------------+---------+-----------+-----------------+
| Cluster |    Member  |    Host   |     Role     |  State  | Lag in MB | Pending restart |
+---------+------------+-----------+--------------+---------+-----------+-----------------+
|  batman | postgresql0 | 127.0.0.1 |    Leader    | running |       0.0 |        *        |
|  batman | postgresql1 | 127.0.0.2 | Sync standby | running |       0.0 |                 |
+---------+------------+-----------+--------------+---------+-----------+-----------------+
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]: 2018-07-08T03:00UTC
Success: restart scheduled on member postgresql0
```

patronictl examples: scheduled switchover

```
$ patronictl -c postgres0.yml switchover  # interactive mode
Master [postgresql0]:
Candidate ['postgresql1'] []:
When should the switchover take place (e.g. 2015-10-01T14:30)  [now]: 2018-07-08T03:00UTC
Current cluster topology
+---------+-------------+-----------+--------------+---------+-----------+
| Cluster |    Member   |    Host   |     Role     |  State  | Lag in MB |
+---------+-------------+-----------+--------------+---------+-----------+
|  batman | postgresql0 | 127.0.0.1 |    Leader    | running |       0.0 |
|  batman | postgresql1 | 127.0.0.2 | Sync standby | running |       0.0 |
+---------+-------------+-----------+--------------+---------+-----------+
Are you sure you want to switchover cluster batman, demoting current master postgresql0? [y/N]: y
2018-06-03 16:18:47.41721 Switchover scheduled
+---------+-------------+-----------+--------------+---------+-----------+
| Cluster |    Member   |    Host   |     Role     |  State  | Lag in MB |
+---------+-------------+-----------+--------------+---------+-----------+
|  batman | postgresql0 | 127.0.0.1 |    Leader    | running |       0.0 |
|  batman | postgresql1 | 127.0.0.2 | Sync standby | running |       0.0 |
+---------+-------------+-----------+--------------+---------+-----------+
Switchover scheduled at: 2018-07-08T03:00:00+00:00
                   from: postgresql0
```

patronictl examples: reinit

```
$ patronictl -c postgres0.yml reinit batman postgresql1
+---------+-------------+-----------+--------------+---------+-----------+
| Cluster |    Member   |    Host   |     Role     |  State  | Lag in MB |
+---------+-------------+-----------+--------------+---------+-----------+
|  batman | postgresql0 | 127.0.0.1 |    Leader    | running |       0.0 |
|  batman | postgresql1 | 127.0.0.2 | Sync standby | running |       0.0 |
+---------+-------------+-----------+--------------+---------+-----------+
Are you sure you want to reinitialize members postgresql1? [y/N]: y
Success: reinitialize for member postgresql1

$ patronictl -c postgres0.yml list
+---------+-------------+-----------+--------+------------------+-----------+
| Cluster |    Member   |    Host   |  Role  |      State       | Lag in MB |
+---------+-------------+-----------+--------+------------------+-----------+
|  batman | postgresql0 | 127.0.0.1 | Leader |     running      |       0.0 |
|  batman | postgresql1 | 127.0.0.2 |        | creating replica |   unknown |
+---------+-------------+-----------+--------+------------------+-----------+
```

patronictl: tips & tricks

- patronictl can use the same config as Patroni
- requires direct access to the DCS and Patroni REST API.
  - please secure them (DCS and Patroni REST API)!
- cluster-name (scope) will be taken from the config if not specified in the command line
- every command has detailed help: "*patronictl command-name --help*"
- default place of patronictl config: "*~/.config/patroni/patronictl.yaml*"
  - create a symlink to avoid typing "*-c patroni.yaml*" all the time

LINKS

- Patroni: https://github.com/zalando/patroni
- Patroni Documentation: https://patroni.readthedocs.io
- Spilo: https://github.com/zalando/spilo
- Helm chart: https://github.com/kubernetes/charts/tree/master/incubator/patroni
- Postgres-operator: https://github.com/zalando-incubator/postgres-operator

We are hiring Database Engineers

jobs.zalando.com

Questions?

Thank you!