Patroni: PostgreSQL High Availability made easy

Alexander Kukushkin, Oleksii Kliukin
Zalando SE
Percona Live Amsterdam 2016

zalando

PERCONA LIVE EUROPE AMSTERDAM

About us

Alexander Kukushkin

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

Oleksii Kliukin

Database Engineer @ZalandoTech
Email: oleksii.kliukin@zalando.de
Twitter: @hintbits

Zalando

- ~ 3 bn EUR revenue
- ~ 160 m visits/month
- 65% visits from mobile devices
- > 170 databases
- > 1300 tech employees
- We are hiring!

zalando

Radical agility and autonomous teams

Organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations

Conway's Law

Cloud databases

- Rapid deployments
- Commodity hardware (cattle vs pets)
- Standard configuration and automatic tuning

Existing automatic failover solutions

- Promote a replica when the master is not responding
  - Split brain/potentially many masters
- Use one monitor node to make decisions
  - Monitor node is a single point of failure
  - Former master needs to be killed (STONITH)
- Use multiple monitor nodes
  - Distributed consistency problem

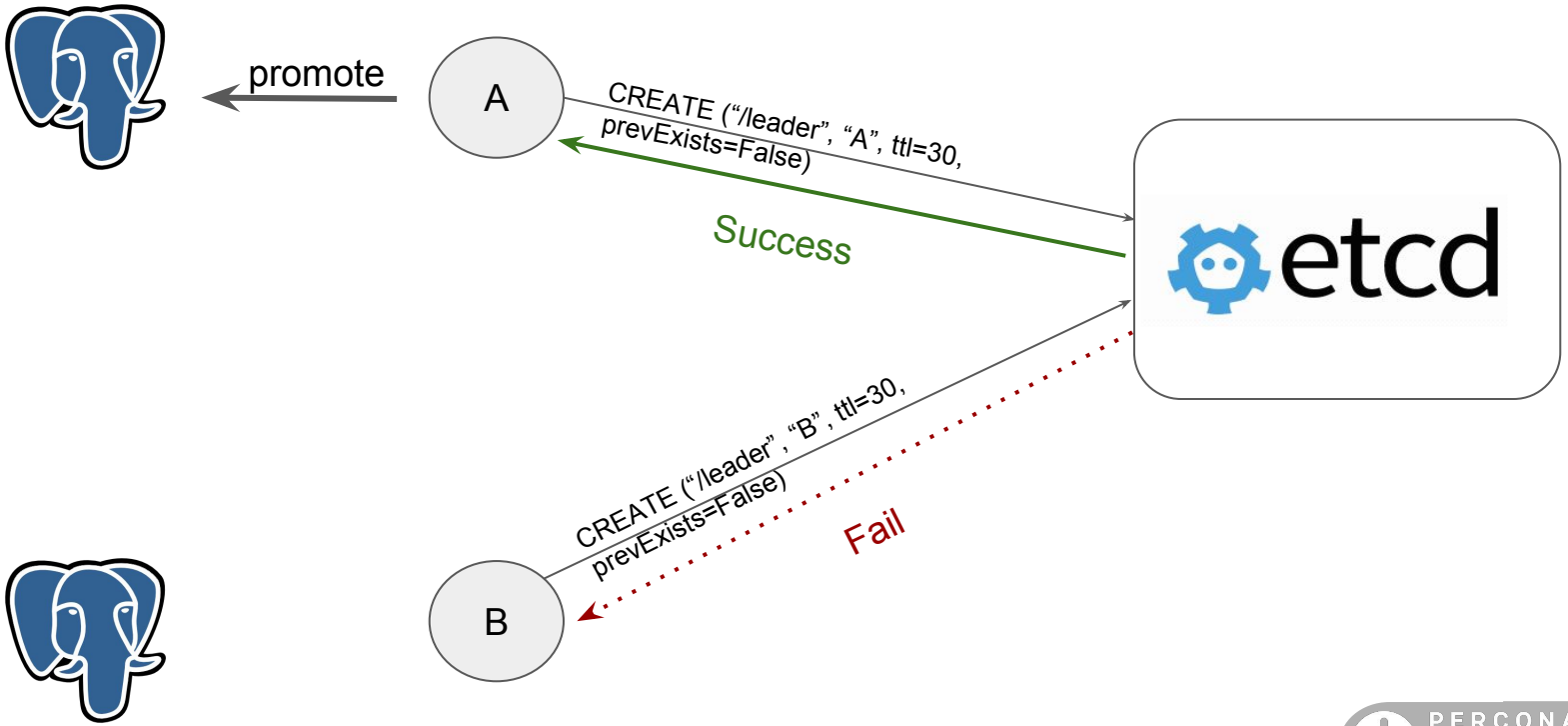
Distributed consistency problem

https://www.flickr.com/photos/kevandotorg

Patroni approach: use DCS

- Distributed configuration system (DCS): Etcd, Zookeeper or Consul
- Built-in distributed consensus (RAFT, Zab)
- Session/TTL to expire data (i.e. master key)
- Key-value storage for cluster information
- Atomic operations (CAS)
- Watches for important keys

Leader race


promote
A
CREATE ("/leader", "A", ttl=30, prevExists=False)
Success
etcd
CREATE ("/leader", "B", ttl=30, prevExists=False)
Fail
B

- /service/cluster/
  - config
  - initialize
  - members/
    - dbnode1
    - dbnode2
  - leader
  - optime/
    - leader
  - failover

Keys that never expire

- initialize
  - "key": "/service/testcluster/initialize",

    "value": "6303731710761975832",
- leader/optime
  - "key": "/service/testcluster/optime/leader",

    "value": "67393608",
- config
  - "key": "/service/testcluster/config",

    "value": "{\"postgresql\":{\"parameters\":{\"synchronous_standby_names\":\"*\"}}}",

Keys with TTL

- leader
  - "key": "/service/testcluster/leader",

    "value": "dbnode2",

    "ttl": 22
- members
  - "key": "/service/testcluster/members/dbnode2",

    “value": "{\"role\":\"master\",\"state\":\"running\",

    \"conn_url\":\"postgres://172.17.0.3:5432/postgres\",

    \"api_url\":\"http://172.17.0.3:8008/patroni\",\"xlog_location\":67393608}",

    "ttl": 22

Bootstrapping of a new cluster

- Initialization race
- initdb by a winner of an initialization race
- Waiting for the leader key by the rest of the nodes
- Bootstrapping of non-leader nodes (pg_basebackup)

Event loop of a running cluster (master)

- Update the leader key or demote if update failed
- Write the leader/optime (xlog position)
- Update the member key
- Add/delete replication slots for other members

Event loop of a running cluster (replica)

- Update the member key
- Check that the cluster has a leader
  - Check recovery.conf points to the correct leader
  - Join the leader race if a leader is not present
- Add/delete replication slots for cascading replicas

Leader race

- Check whether the member is the healthiest
  - Evaluate its xlog position against all other members
- Try to acquire the leader lock
- Promote itself to become a master after acquiring the lock

LIVE DEMO

Patroni features

- Manual and Scheduled Failover
- Attach the old master with pg_rewind
- Customizable replica creation methods
- Dynamic configuration
- Pause (maintenance) mode
- patronictl

Dynamic configuration

- Ensure identical configuration of the following parameters on all members:
  - ttl, loop_wait, retry_timeout, maximum_lag_on_failover
  - wal_level, hot_standby
  - max_connections, max_prepared_transactions, max_locks_per_transaction, max_worker_processes, track_commit_timestamp, wal_log_hints
  - wal_keep_segments, max_replication_slots
- Change Patroni/PostgreSQL configuration dynamically
- Inform the user that PostgreSQL needs to be restarted (pending_restart flag)
- Store parameters in DCS and apply to all members

Building HA PostgreSQL based on Patroni

- Client traffic routing
  - patroni callbacks
  - conf.d + haproxy, pgbouncer
- Backup and recovery
  - WAL-E, barman
- Monitoring
  - Nagios, zabbix, zmon

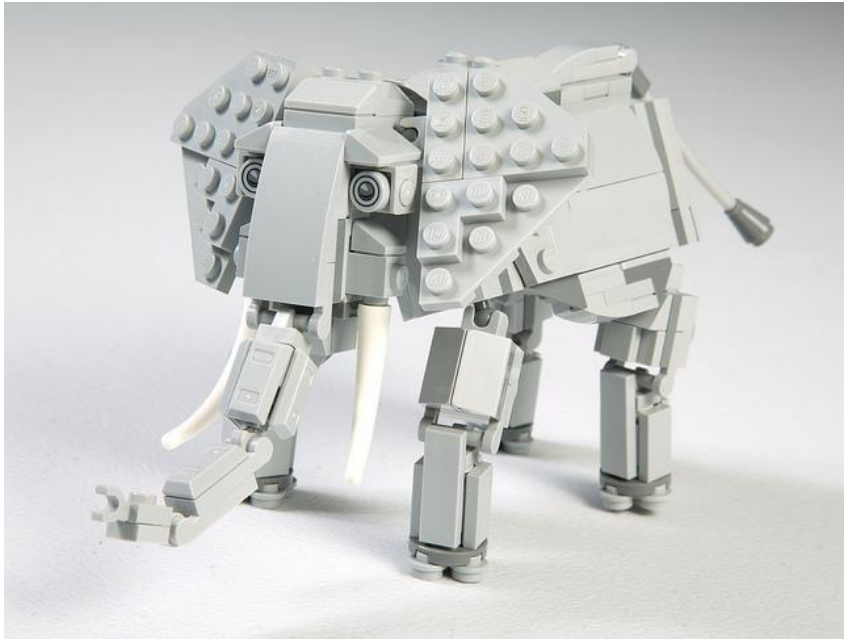
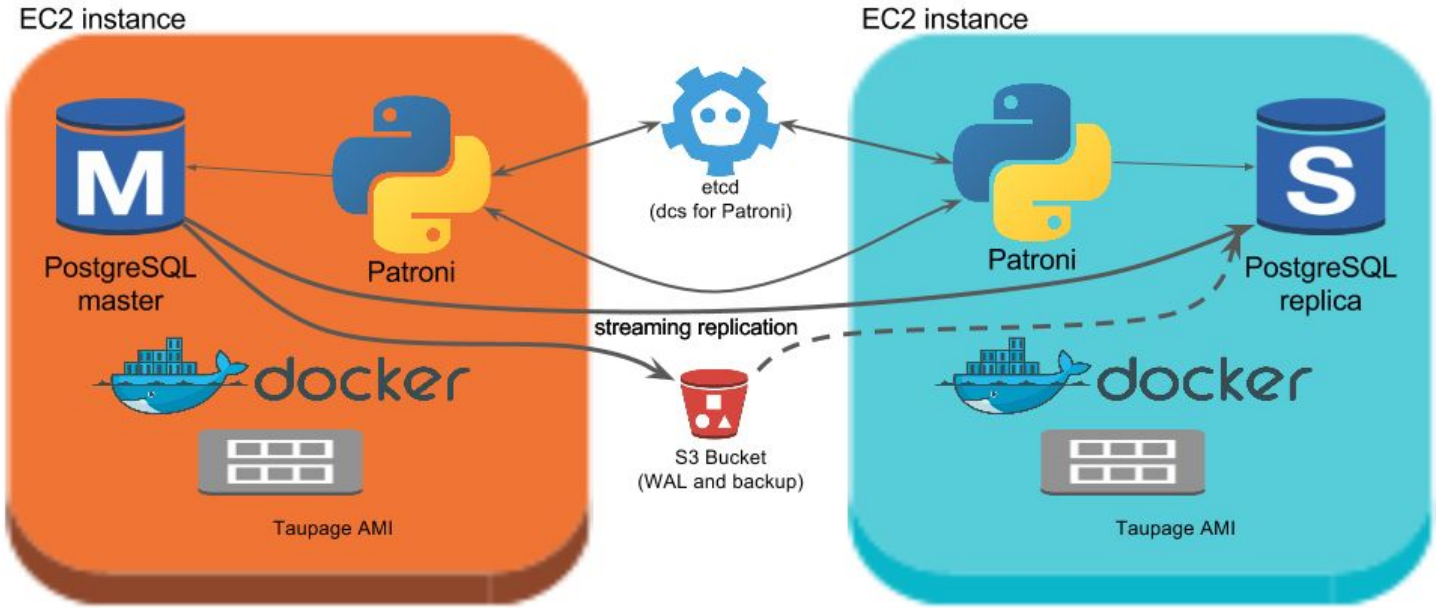
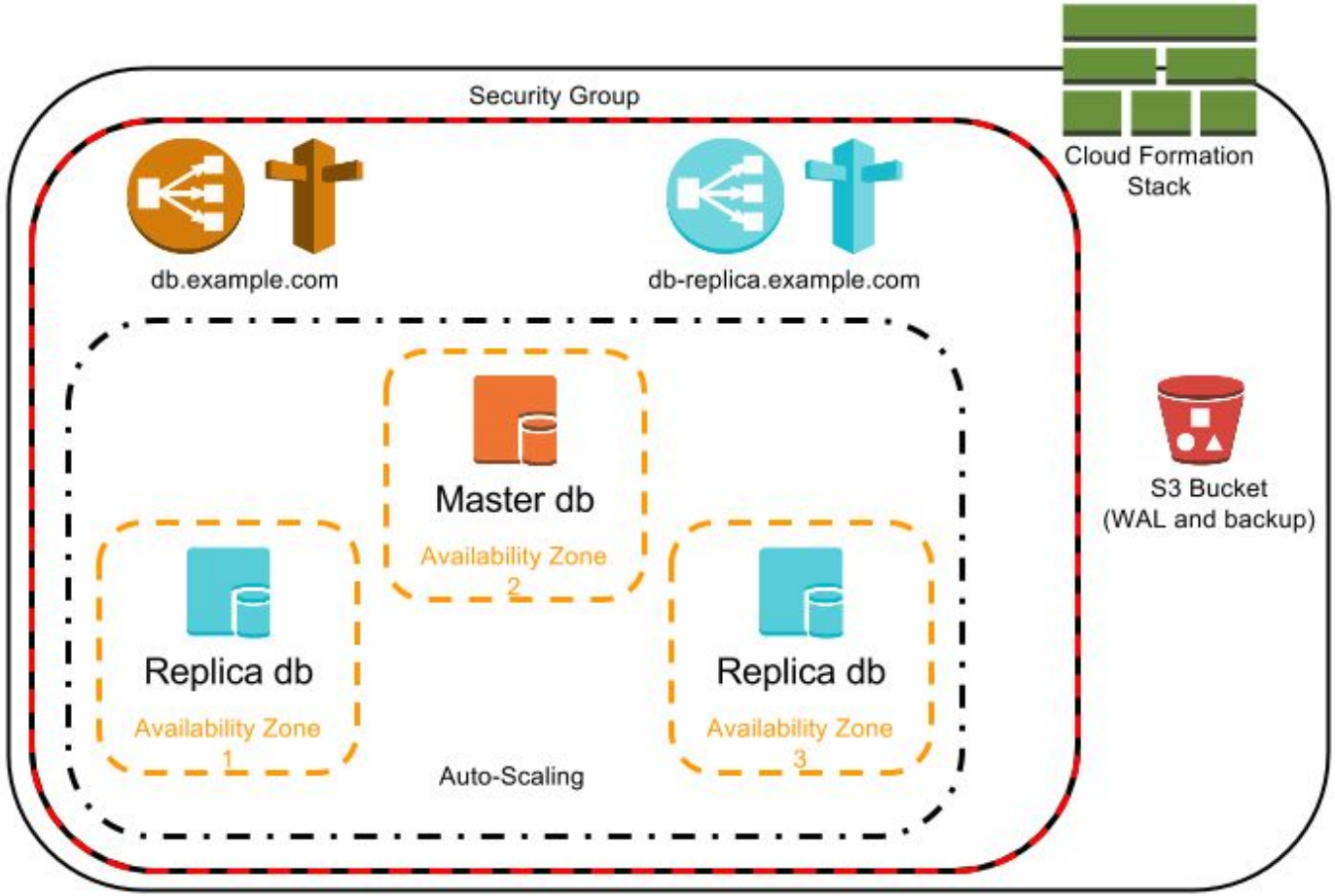


Image by flickr user https://www.flickr.com/photos/brickset/

Spilo: Patroni + Docker + WAL-E + AWS/K8S




Security Group
Cloud Formation Stack
db.example.com
db-replica.example.com
Master db
Availability Zone 2
S3 Bucket (WAL and backup)
Replica db
Availability Zone 1
Replica db
Availability Zone 3
Auto-Scaling

When should the master demote itself?

- Chances of data loss vs write availability
- Avoiding too many master switches (retry_timeout, loop_wait, ttl)
- 2 x retry_timeout + loop_wait < ttl
- Zookeeper and Consul session duration quirks

Choosing a new master

- Reliability/performance of the host or connection
  - nofailover tag
- XLOG position
  - highest xlog position = the best candidate
  - xlog > leader/optime - maximum_lag_on_failover
    - maximum_lag_on_failover > size of WAL segment (16MB) for disaster recovery

Attaching the old master back as a replica

- Diverged timelines after the former master crash
- pg_rewind
  - use_pg_rewind
  - remove_data_directory_on_rewind_failure

Thank you!

https://github.com/zalando/patroni

zalando

PERCONA LIVE EUROPE AMSTERDAM

Useful links

- Spilo: https://github.com/zalando/spilo
- Confd: http://www.confd.io
- Etcd: https://github.com/coreos/etcd
- RAFT: http://thesecretlivesofdata.com/raft/

Rate Our Session!