Отказоустойчивый

PostgreSQL кластер с Patroni

Александр Кукушкин

PgConf.Russia 2017, Москва

zalando

ABOUT ME

Alexander Kukushkin

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

zalando

ZALANDO AT A GLANCE

~3.6 billion EURO
net sales 2016

~165 million visits per month

~20 million
active customers

~200,000
product choices

>1,500
brands

15
countries

>12,000
employees in Europe

50%
return rate across all categories

zalando

BERLIN

ZALANDO TECHNOLOGY

BERLIN

DORTMUND

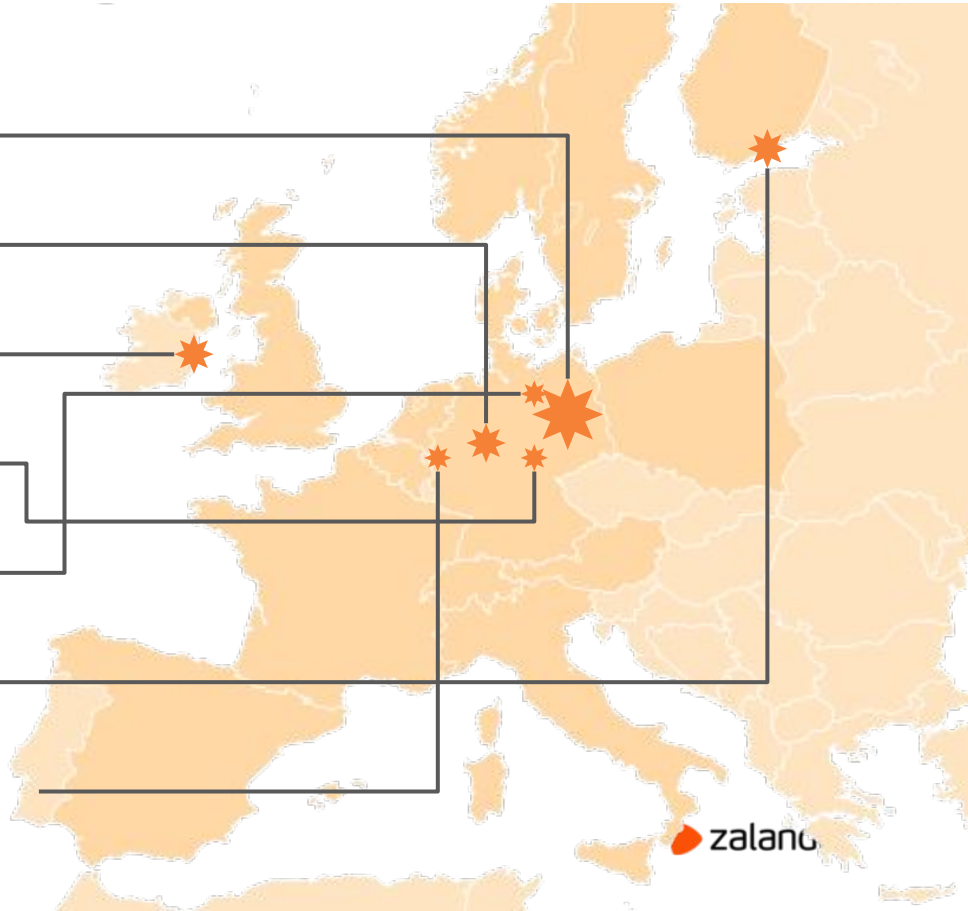
DUBLIN

ERFURT

HAMBURG

HELSINKI

MÖNCHENGLADBACH



ZALANDO TECHNOLOGY

- > 150 databases in DC
- > 120 databases on AWS
- > 1600 tech employees
- We are hiring!

POSTGRESQL

The world's most advanced open-source database

- Rock-solid by default
- Transactional DDL
- Standard-compliant modern SQL
- Blazing performance
- PostgreSQL is a community

zalando

RUNNING DATABASES AT SCALE

RUNNING DATABASES AT SCALE

CLOUD DATABASES

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

zalando

AUTOMATIC FAILOVER

“PostgreSQL does not provide the system software required to identify a failure on the primary and notify the standby database server.”

CC0 Public Domain

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 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

DCS STRUCTURE

- /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\":{\"max_connections\":\"300\"}}}"

KEYS WITH TTL

- leader
  - "key": "/service/testcluster/leader",
    "value": "dbnode2",
    "ttl": 22
- members
  - "key": "/service/testcluster/members/dbnode2",
    “value": "{\"role\":\"master\",\"state\":\"running\",\"xlog_location\":67393608,
    \"conn_url\":\"postgres://172.17.0.3:5432/postgres\",
    \"api_url\":\"http://172.17.0.3:8008/patroni\"}",
    "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

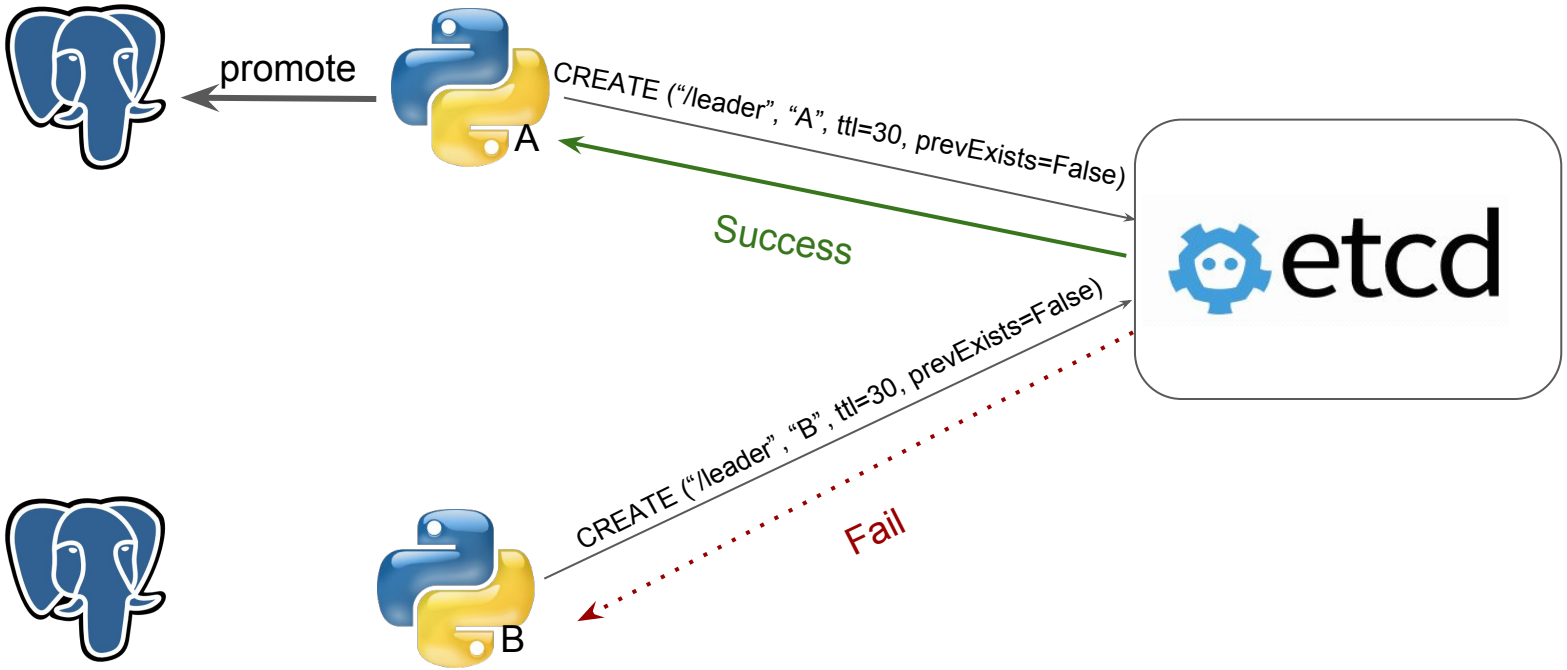
EVENT LOOP OF A RUNNING CLUSTER (REPLICA)

- 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
- Update the member key
- 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

LEADER RACE



zalando

LIVE DEMO

PATRONI FEATURES

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

DYNAMIC CONFIGURATION

- Store Patroni/PostgreSQL parameters in DCS and apply them dynamically
- 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
- Inform the user that PostgreSQL needs to be restarted (pending_restart flag)

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

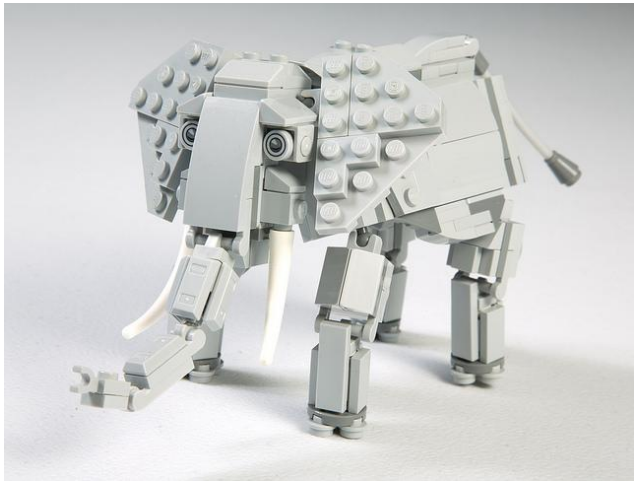
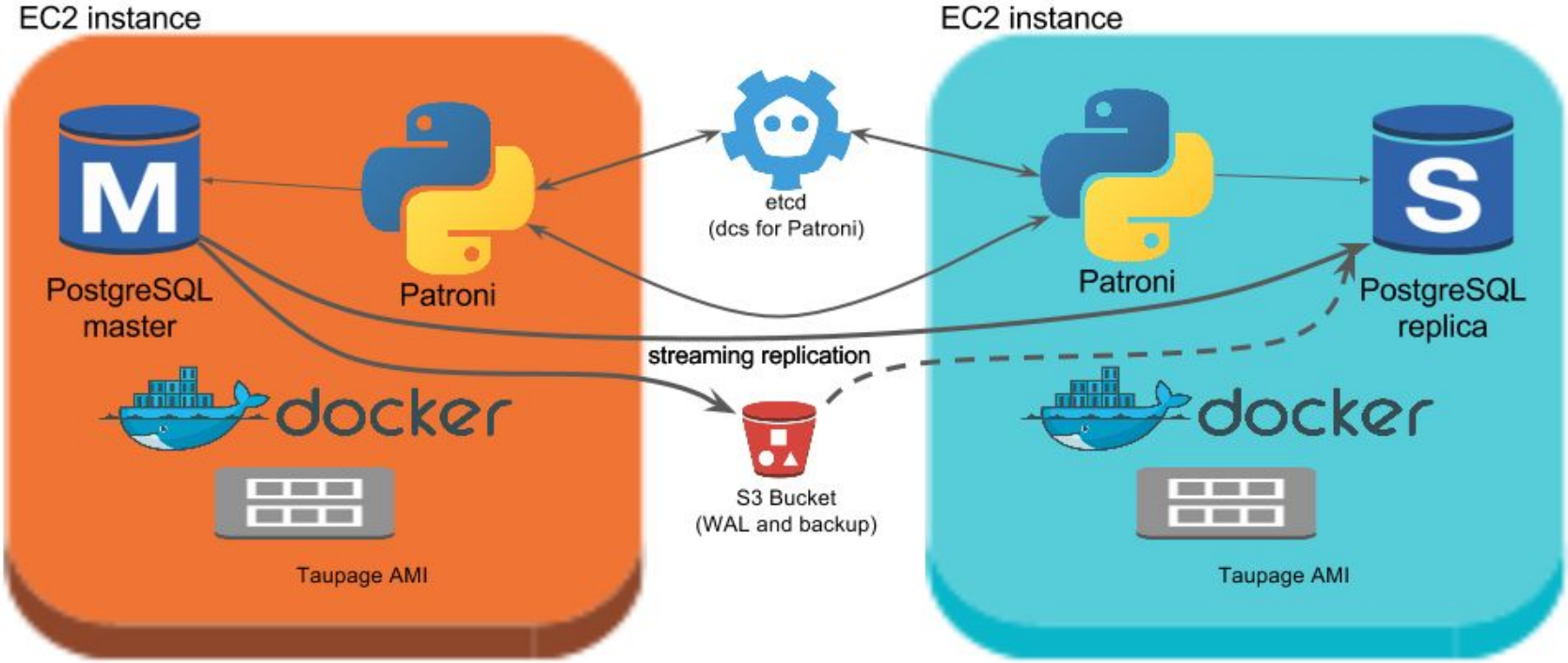


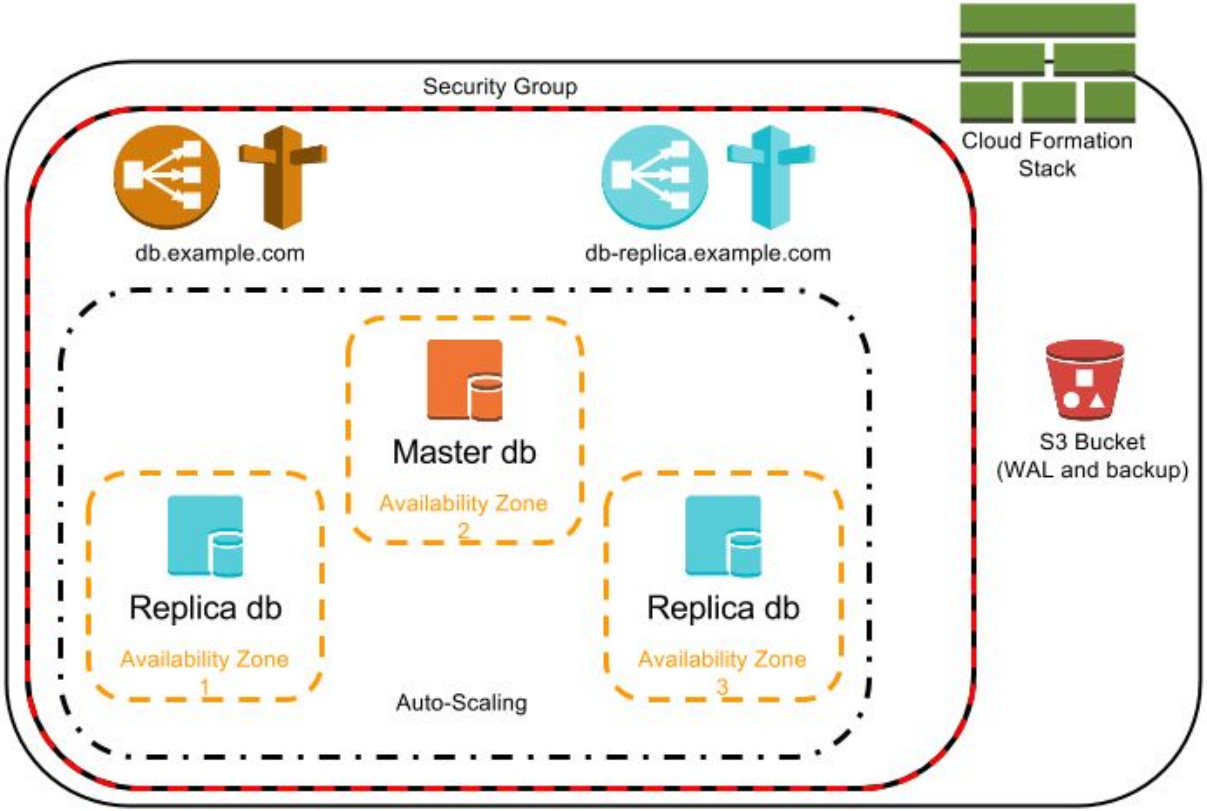
Image by flickr user https://www.flickr.com/photos/brickset/

SPILO: DOCKER + PATRONI + WAL-E + AWS/K8S



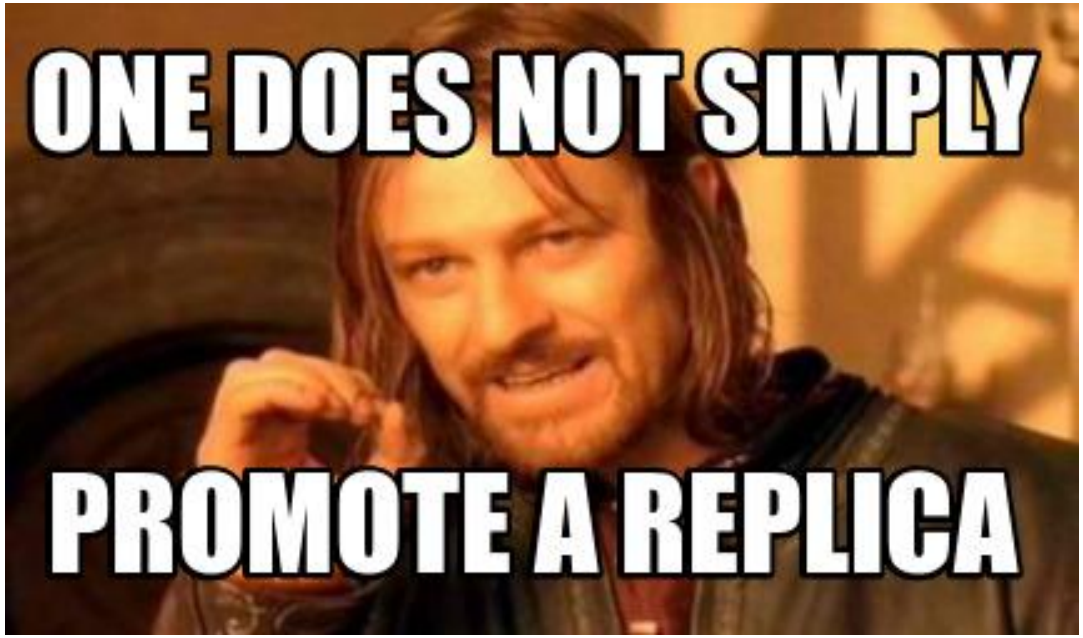
zalando

SPILO DEPLOYMENT



zalando

AUTOMATIC FAILOVER IS HARD



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 REPLICA

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

USEFUL LINKS

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

Thank you!

https://github.com/zalando/patroni