PostgreSQL Cluster

Цели

Какие у нас есть варианты

Как делать/не делать Failover

Архитектура Patroni

Создание кластера

Как менять конфигурацию кластера

Немного про DCS

Перенаправление клиентов на Master

Создание реплик и их реинициализация

Высокая доступность

- Распределенное хранилище
  - NFS NAS/SAN
  - DRBD
  - ISCSI (+ LVM)
- Мульти-мастер
  - BDR, Bucardo
- Логическая репликация
  - pglogical, slony, встроенная фича в postgresql 10
- Физическая репликация
  - В postgresql начиная с 9.0
- Облака: Azure, Amazon: Aurora/RDS

Варианты

- Встроенные решения
- Patroni
- Stolon:
  - Проксирует все запросы в мастер ноду. Нельзя давать нагрузку на реплики
  - Мастер выбирается самостоятельно при switchover-е
- repmgr:
  - Нет фэнсинга из коробки (защита от двойного мастера)
  - Нет нужды в DCS - на мой взгляд это минус

Slony

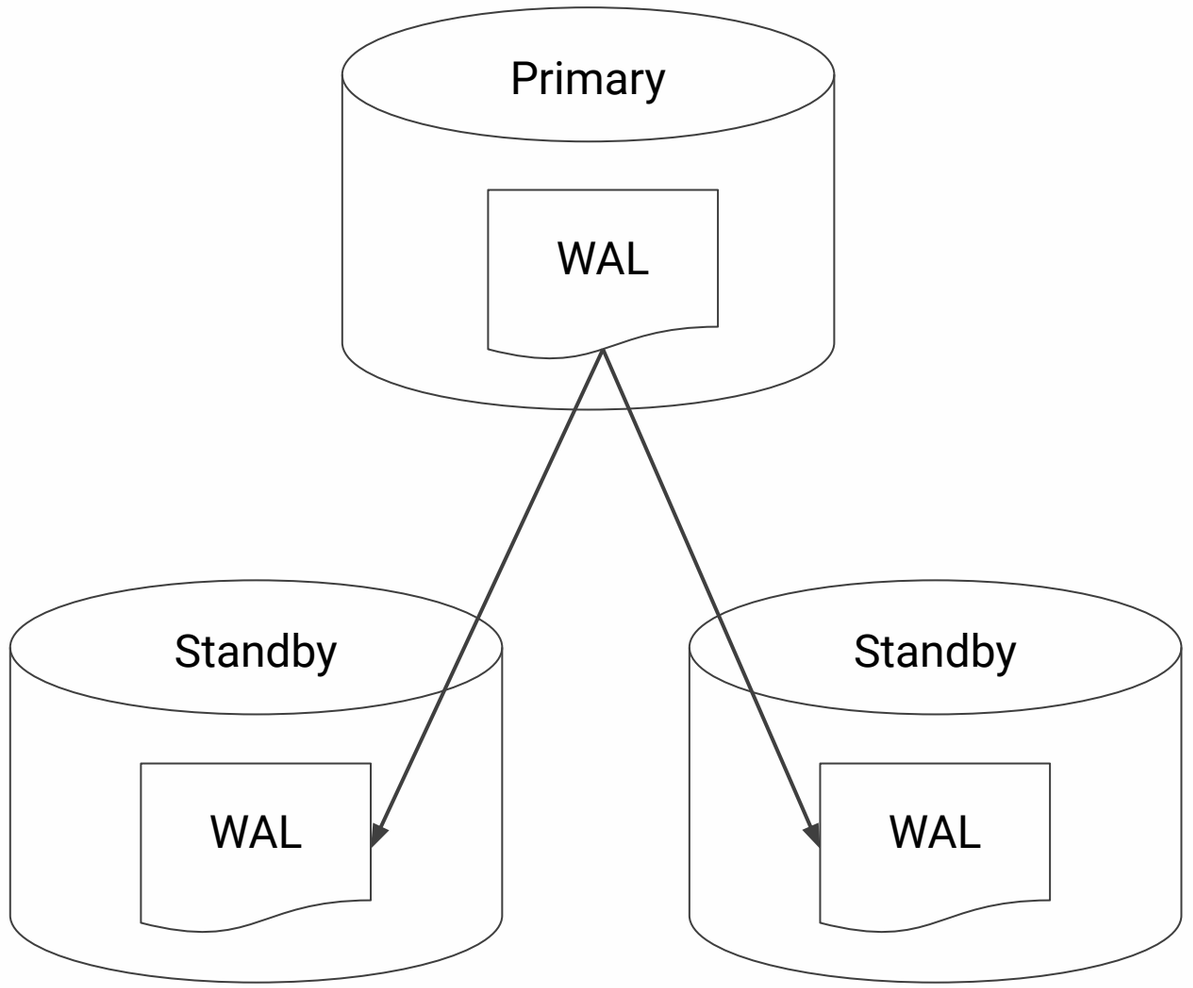
Физическая репликация

Плюсы:

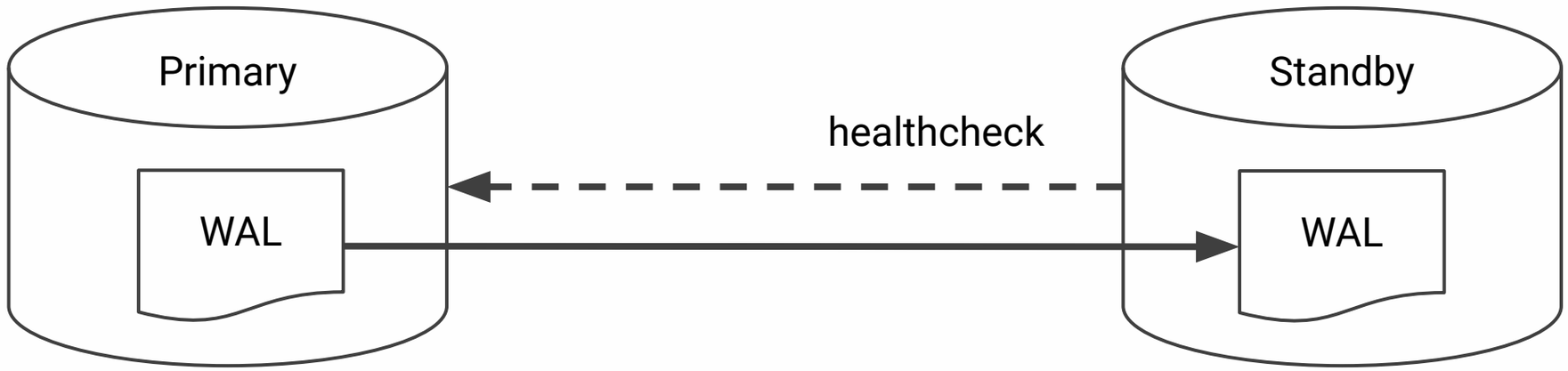
- Встроенная фича
- Минимальная задержка
- Идентичные копии

Минусы:

- Нужны одинаковый мажорные версии
- Нет автоматического failover

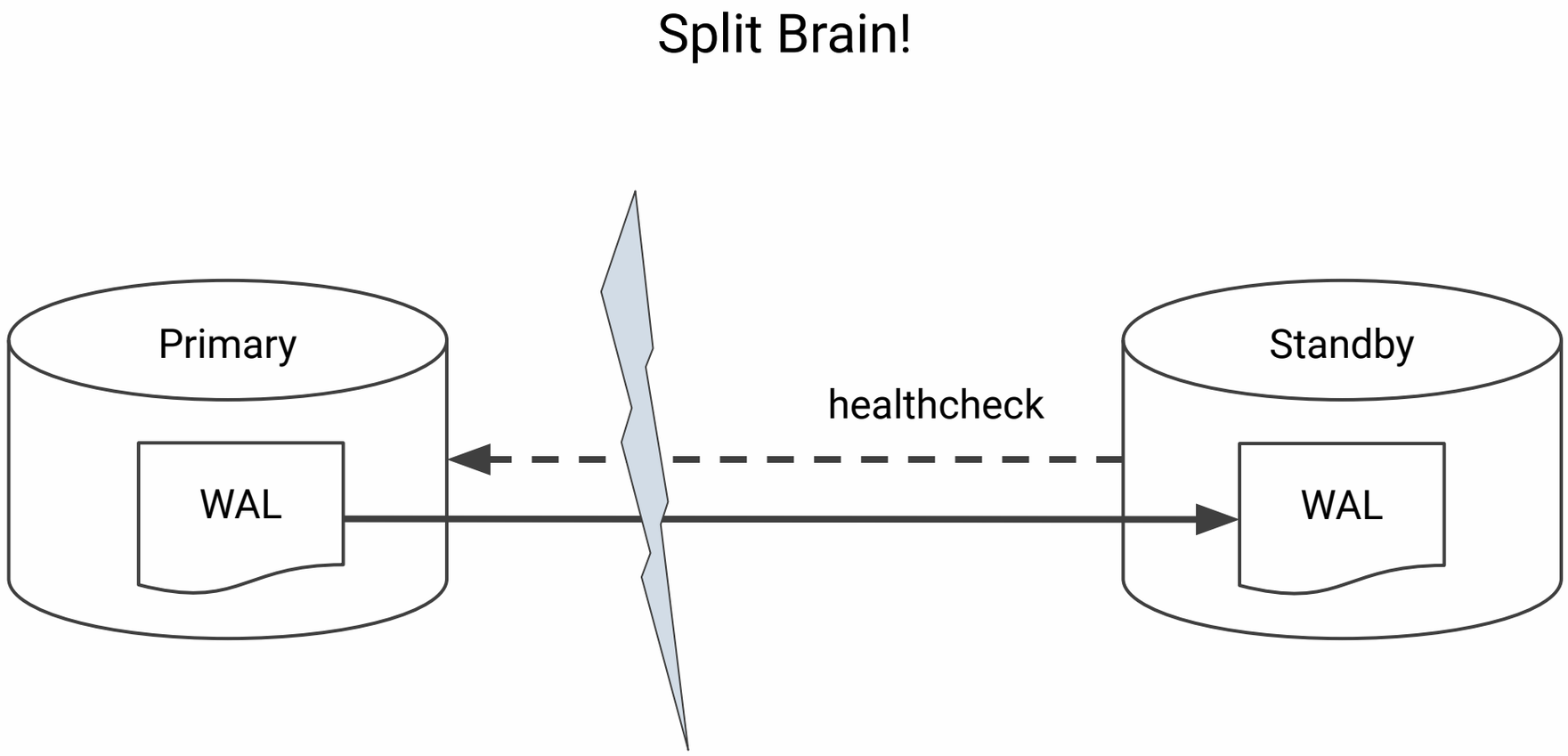


Автоматический Failover

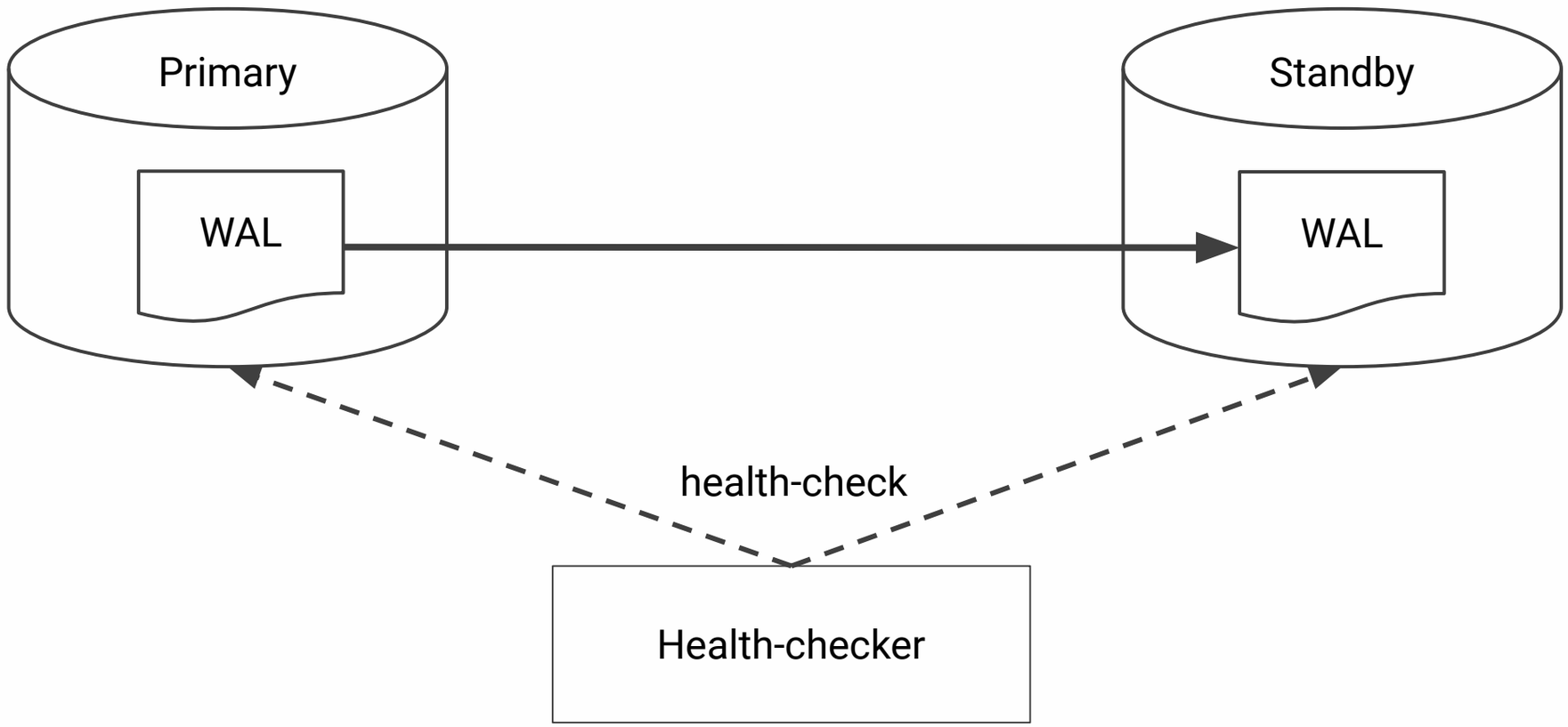


Запускаем healthcheck со стендбая и при отрицательном ответе продвигаем (promote) его до Мастера

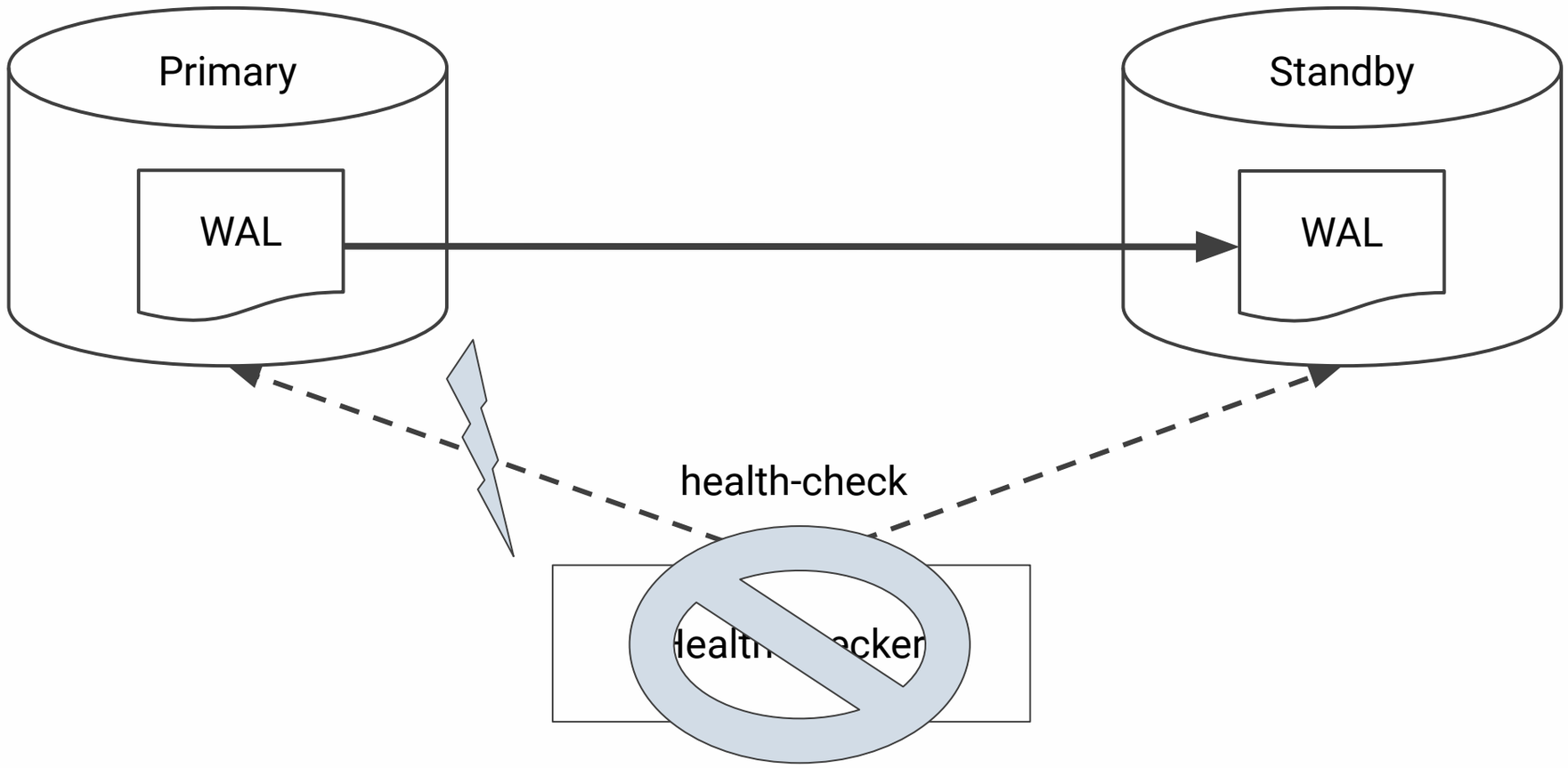
Автоматический Failover



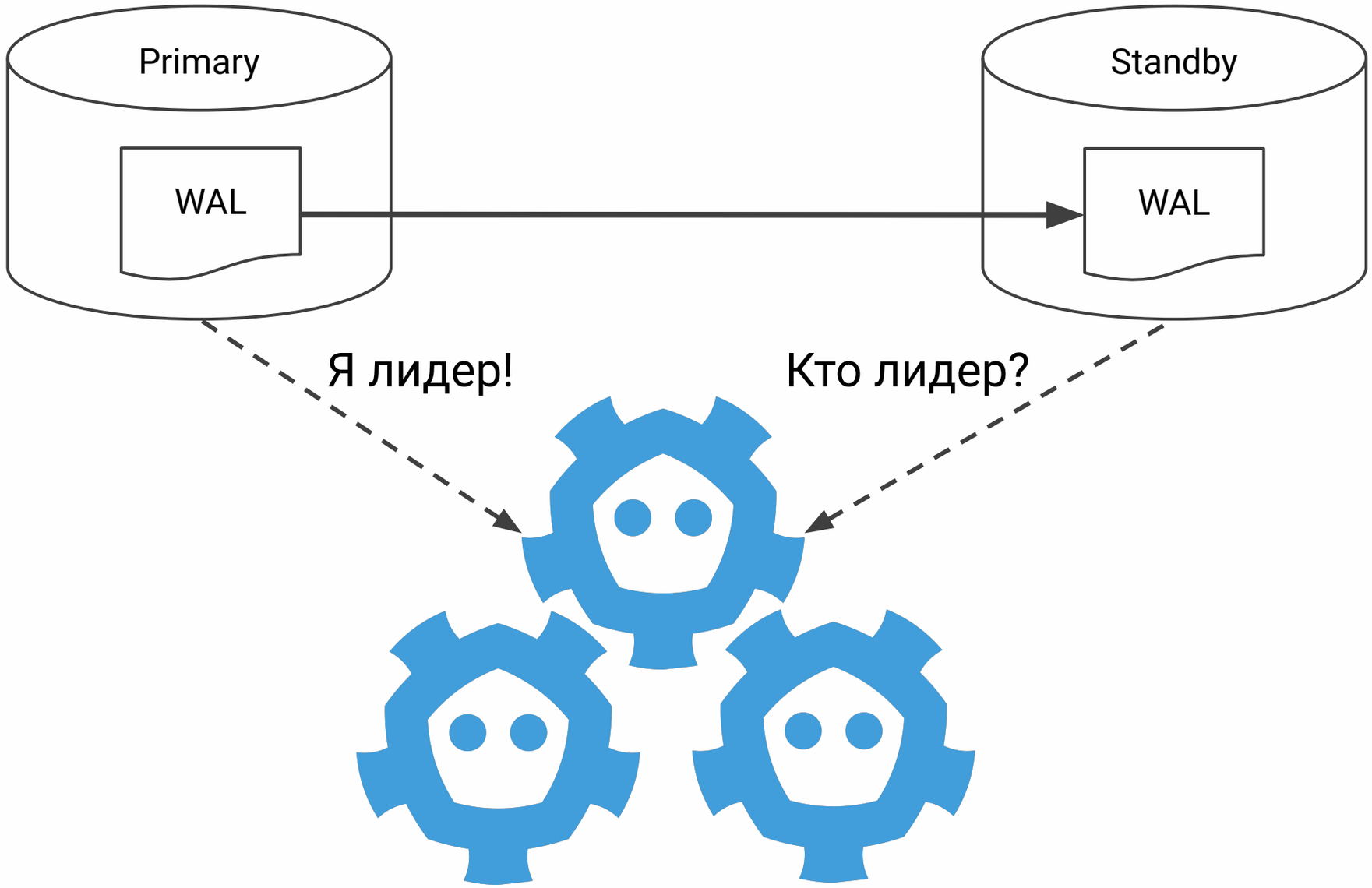
Автоматический Failover



Автоматический Failover



Автоматический Failover



Автоматический Failover

У постгреса нет
какого либо решения
по автоматическому
фейловеру из коробки

Функции DCS

- etcd (или Consul, Zookeeper) хранят информацию о том, кто сейчас лидер
- DCS хранит конфигурацию кластера
- помогает решить проблему с партиционированием сети
- убивает старые клиентские коннекты
- STONITH
- Неплохо бы иметь watchdog (Например, Nomad)

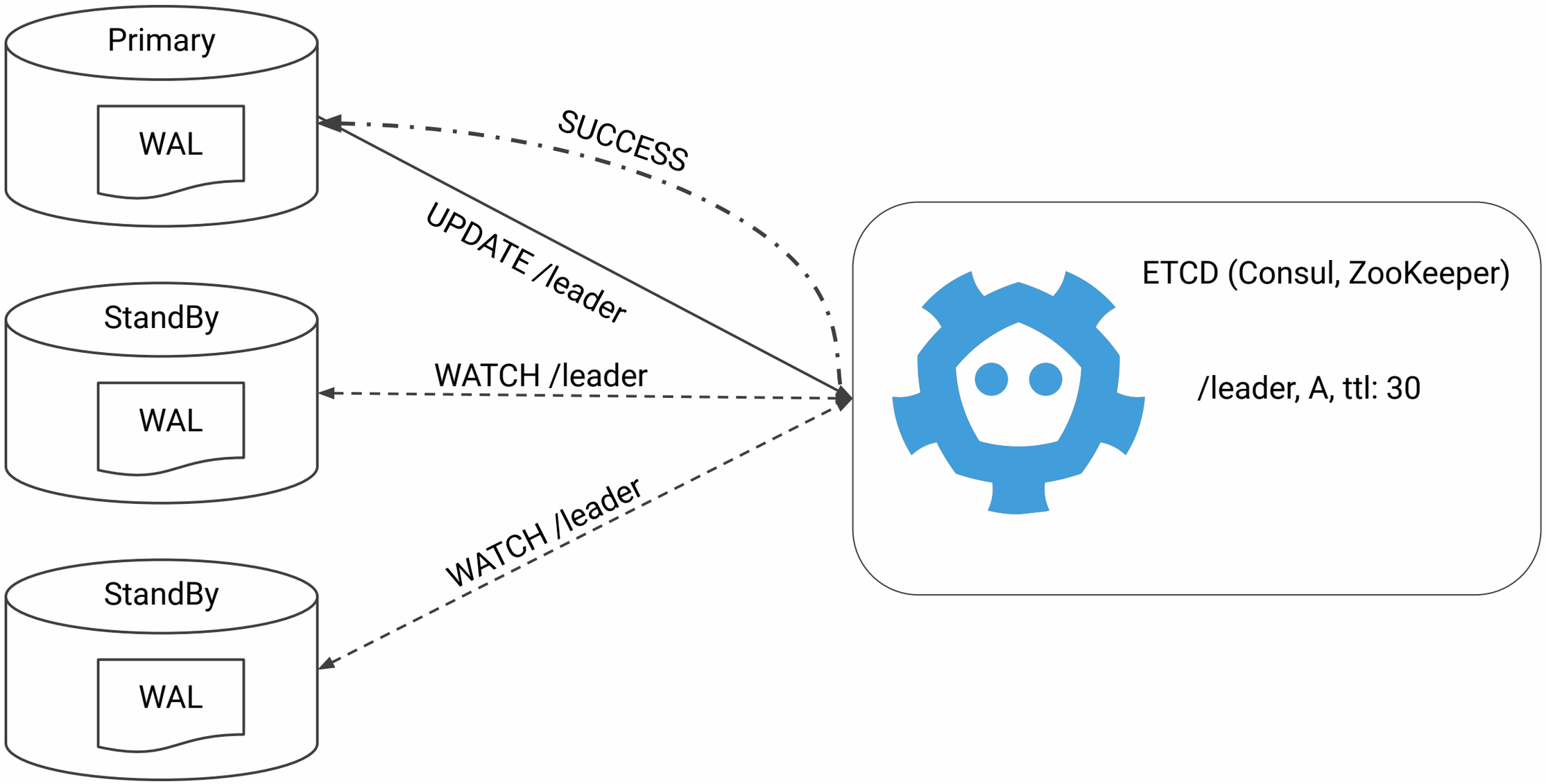
Почему Consul

- Service check
- + Consul templates
- Есть GUI =)
- Есть свой DNS
- Patroni может анонсировать master/replica
- ETCD при большой загрузке замечен в высокой нагрузке на дисковую подсистему

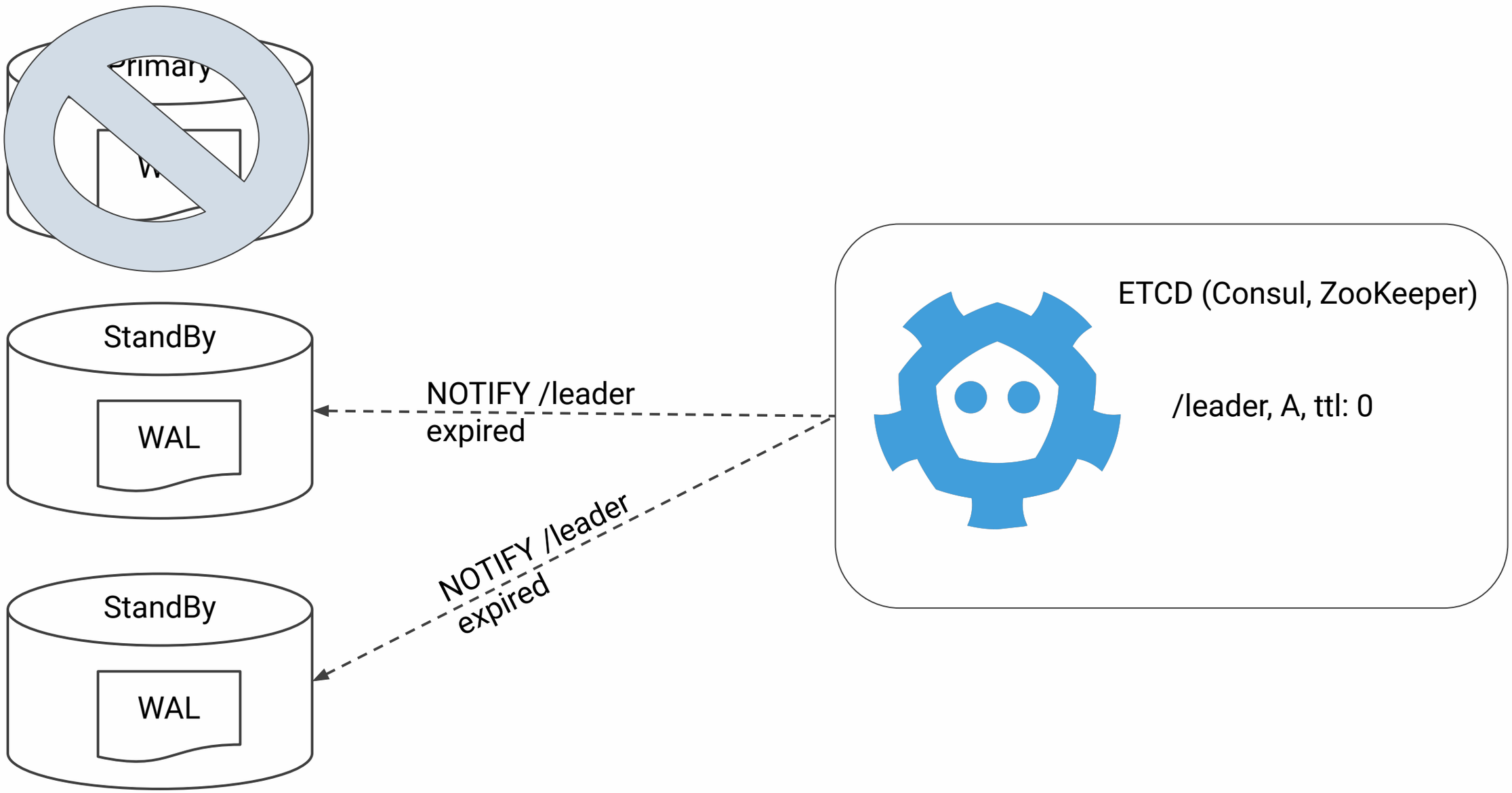
Patroni

- PostgreSQL не умеет взаимодействовать с etcd
- Демон будет запущен рядом с PostgreSQL
- Демон умеет взаимодействовать с etcd
- Демон принимает решение promotion/demotion

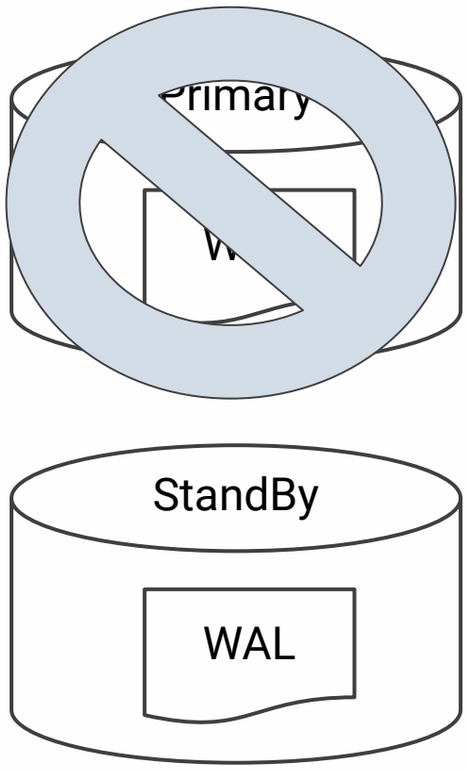
Автоматическая репликация



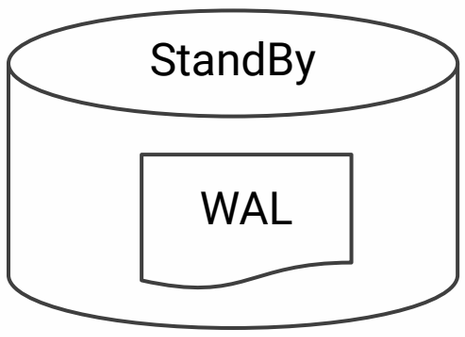
Автоматическая репликация



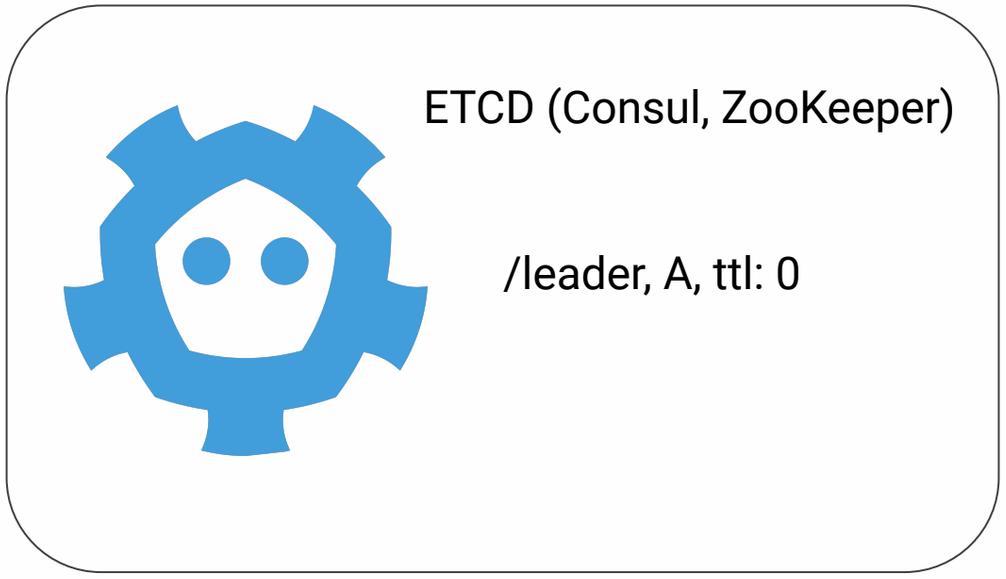
Автоматическая репликация



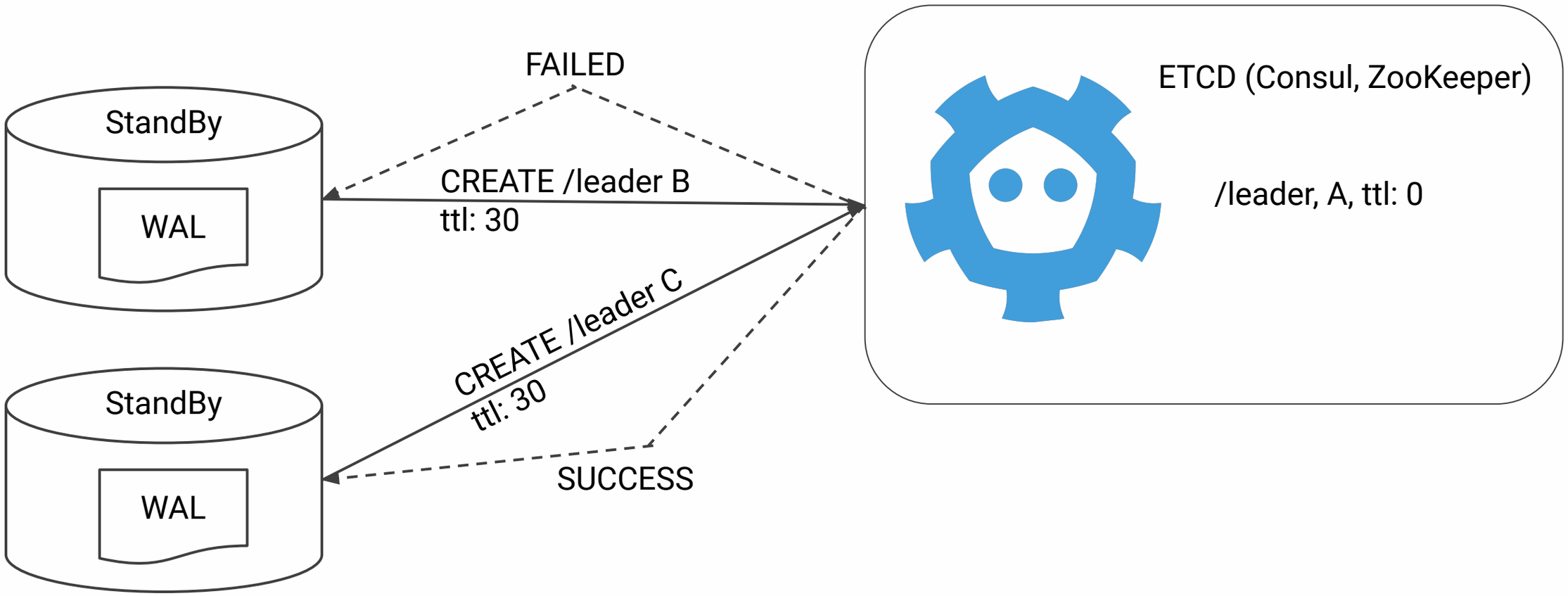
Node B:
GET hostA:patroni -> Timeout
GET hostB:patroni -> wal_position: 200
GET hostC:patroni -> wal_position: 100



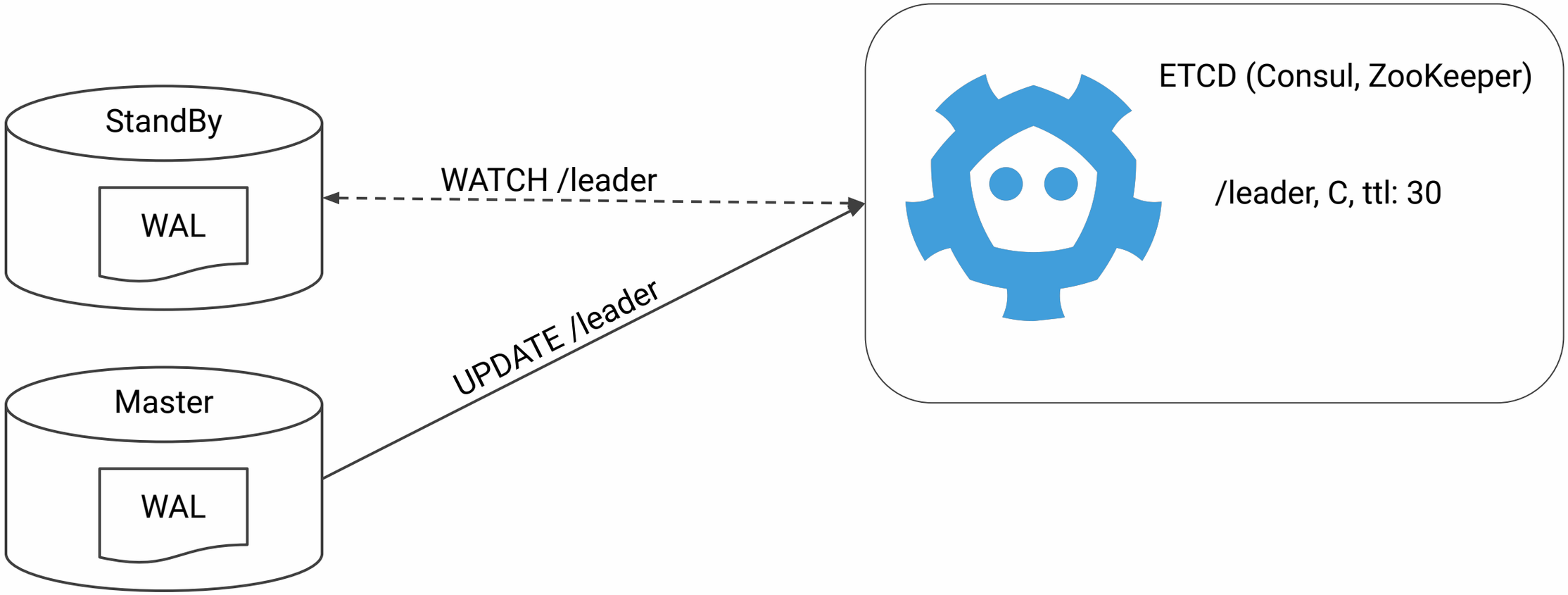
Node C:
GET hostA:patroni -> Timeout
GET hostB:patroni -> wal_position: 200
GET hostC:patroni -> wal_position: 100



Автоматическая репликация



Автоматическая репликация



Состояние кластера

- patronictl - утилита для управления кластером
- patronictl -c /opt/app/patroni/etc/postgresql.yml list

```
[root@pg01 ~]# patronictl -c /etc/patroni.yml list
+----------+--------+-------------+--------+---------+----+-----------+
| Cluster  | Member |     Host    |  Role  |  State  | TL | Lag in MB |
+----------+--------+-------------+--------+---------+----+-----------+
| postgres |  pg01  | 10.128.0.47 |        | running |  7 |       0.0 |
| postgres |  pg02  | 10.128.0.46 | Leader | running |  7 |       0.0 |
| postgres |  pg03  | 10.128.0.45 |        | running |  7 |       0.0 |
+----------+--------+-------------+--------+---------+----+-----------+
```

Переменные окружения

- **PATRONI_CONFIG_FILE -** путь до конфигурационного файла
- **PATRONI_NAME -**
  имя текущей ноды. Должно быть уникально в контексте кластера
- **PATRONI_SCOPE** - имя кластера
- **PATRONI_LOG_*** - все что связано с логами

export PATRONI_CONSUL_HOST=’192.168.11.100:8500’
export PATRONI_CONSUL_TOKEN=aabbccddeeff

Автоматический Failover

systemctl stop patroni - любой другой способ протестировать failover =)

- 30 секунд по умолчанию на истечение ключа в DCS
- После чего Patroni стучится на каждую ноду в кластере и спрашивает, не мастер ли ты, проверяет WAL логи, насколько близки они к мастеру. В итоге если WAL логи у всех одинаковые то, промоутится следующий по порядку
- Опрос нод идёт параллельно

Важные параметры

Обновление данных в DCS идет циклично:

- loop_wait - минимальный промежуток в секундах между попытками обновить ключ лидера.
- ttl - время жизни ключа лидера. Рекомендация: как минимум loop_wait + retry_timeout, но вообще таким комфортным, чтобы избежать нескольких медленных/неудавшихся вызовов к DCS
- retry-timeout - общее время всех попыток внутри одной операции

- maximun_lag_on_failover - максимальное отставание ноды от лидера для того, чтобы учавствовать в выборах

- synchronous_mode: - вкл/выкл синхронной реплики
- synchronous_mode_strict: - вкл/выко строго синхронного режима

Редактирование конфигурации

```
[root@pg02]# patronictl -c /opt/app/patroni/etc/postgresql.yml edit-config
---
+++
@@ -2,5 +2,6 @@
 maximum_lag_on_failover: 1048576
 postgresql:
   use_pg_rewind: true
+  parameters:
+    maintenance_work_mem: 256MB
 retry_timeout: 10
 ttl: 30
Apply these changes? [y/N]:
```

Mar 21 09:59:50 pg03 patroni: 2019-03-21 09:59:50,666 INFO: Changed maintenance_work_mem from 65536 to 256MB

Mar 21 09:59:50 pg03 patroni: 2019-03-21 09:59:50,667 INFO: PostgreSQL configuration items changed, reloading configuration.

Локальная конфигурация

Что делать если нужно поменять конфигурацию PostgreSQL только локально.

- etcd
- patroni.yml
- postgresql.base.conf
- ALTER SYSTEM SET - имеет наивысший приоритет

Некоторые параметры, такие как: max_connections, max_locks_per_transaction, wal_level, max_wal_senders, max_prepared_transactions, max_replication_slots, max_worker_processes не могу быть переопределены локально - Patroni их перезаписывает.

Monitoring

Проверка запущен ли PostgreSQL мастер:

- GET /master - должно возвращать 200 ТОЛЬКО для одной ноды

Проверка работают ли реплики

- GET /patroni с мастера должно возвращать replication:[{state: streaming}] для всех реплик

Запущен ли сам PostgreSQL:

- GET /patroni должен возвращать state:running для каждой ноды

Отставание реплики:

- GET /patroni - xlog: location с реплик не должен быть далеко от этого же параметра на мастере

Направление клиентов

- HAProxy
- KeepaliveD
- TCP Proxy (NGINX)

Пользовательские скрипты. ХУКИ!

```
postgresql:
  callbacks:
    on_start: /opt/pgsql/pg_start.sh
    on_stop: /opt/pgsql/pg_stop.sh
    on_role_change: /opt/pgsql/pg_role_change.sh
```

Tags

- nofailover (true/false) - в положении true нода никогда не станет мастером
- noloadbalance (true/false) - /replica всегда возвращает код 503
- clonefrom (true/false) - patronictl выберет предпочтительную ноду для pgbasebackup
- nosync (true/false) - нода никогда не станет синхронной репликой
- replicatefrom (node name) - указать реплику с которой снимать реплику

Switchover vs failover

- Switchover
  - Переключение роли Мастера на новую ноду. Делается вручную, по сути плановые работы
- Failover
  - Экстренное переключение Мастера на новую ноду
  - Происходит автоматически
  - Ручной вариант - manual failover - только когда не система не может решить на кого переключать, или не настроен автомат

switchover

- patronictl switchover cluster_name
- Отложенный switchover
- Смена мастера для работы с ним

Перезагрузка

- patronictl -c /opt/app/patroni/etc/postgresql.yml restart postgres pg2
- Применение новых параметров требующих обязательной перезагрузки

Реинициализация

- patronictl -c /opt/app/patroni/etc/postgresql.yml reinit postgres pg03
  - Реинициализирует ноду в кластере. Т.е. по сути удаляет дата директорию и делает pg_basebackup, если это поведение не изменено параметром create_replica_method

Режим паузы

- Отключается автоматический failover
- Ставиться глобальная пауза на все ноды
- Проведение плановых работ, например с etcd или обновление PostgreSQL

Тем не менее:

- Можно создавать реплики
- Ручной switchover возможен

- patronictl -c /opt/app/patroni/etc/postgresql.yml pause|resume

Синхронная репликация

- **synchronous_mode**: true/false - не делает failover ни на какую реплику кроме синхронной
- **synchronous_mode_strict:** true/false - если синхронная реплика пропала, то мастер не принимает новые записи пока она не вернется

Бэкап кластера

Полные и инкрементные бэкапы создаются кастомными скриптами

- Роль узла в кластере можно узнать запросом к DCS
- Архивные транзакционные логи (WAL):
  - сегментами в 16 Мб с мастер узла (**archive_command=on**)
  - потоком по протоколу физической репликации

Ваши вопросы?