PostgresPro

PGMeetup.PERM 2025

PostgreSQL High Availability:
от теории к практике без ошибок

Павел Конотопов
p.konotopov@postgrespro.ru

We're ready to rock and roll...

Комментарий перед стартом БД

Postgres: src/include/access/tableam.h; TableAmRoutine
https://pgconf.ru/talk/2032287

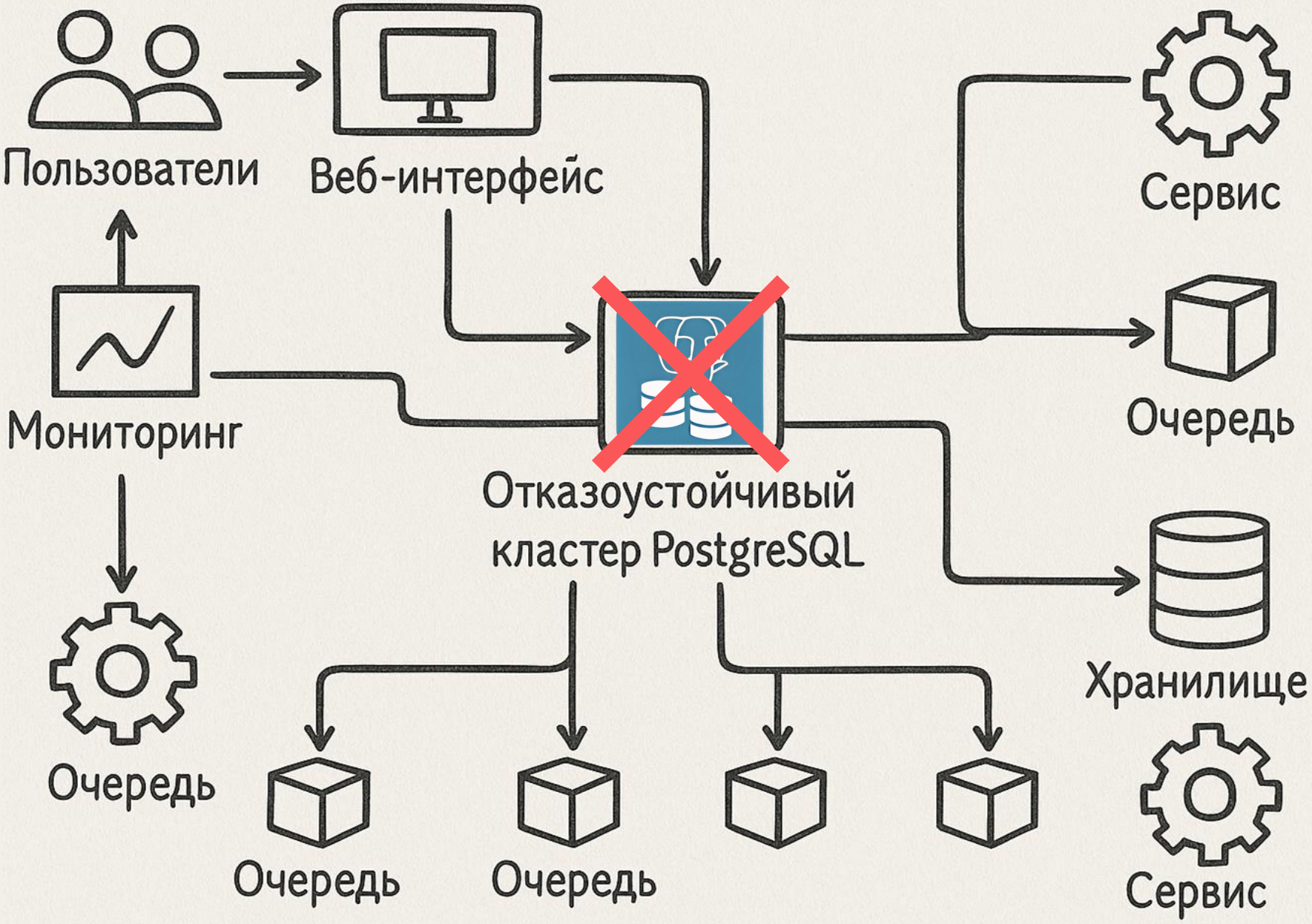

КАЖДЫЙ ПРОГРАММИСТ – НЕМНОГО МУЗЫКАНТ

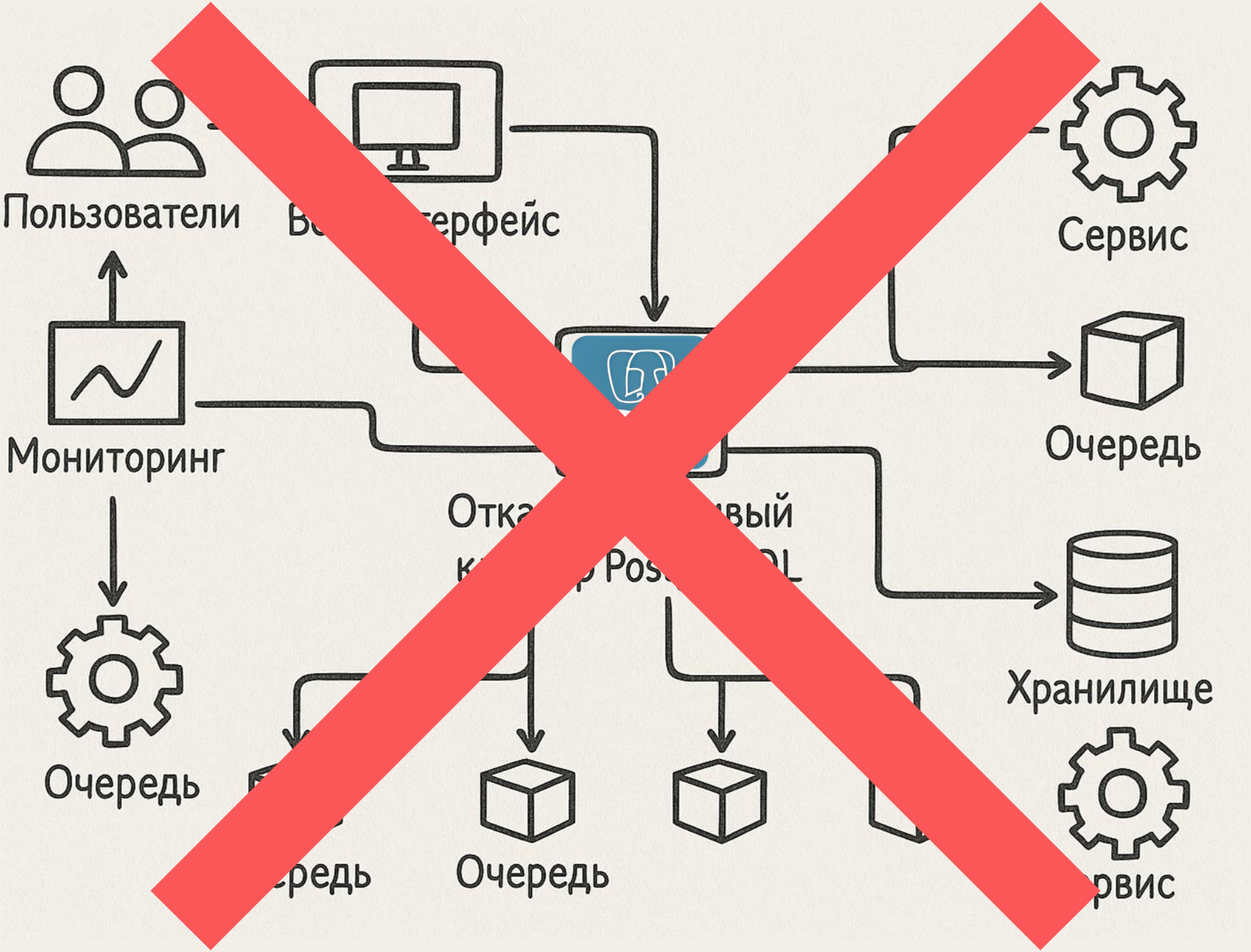

Пользователи
Веб-интерфейс
Сервис
Мониторинг
Отказоустойчивый
кластер PostgreSQL
Очередь
Очередь
Хранилище
Очередь
Очередь
Очередь
Сервис

— Те, двое, называют его «босс»!

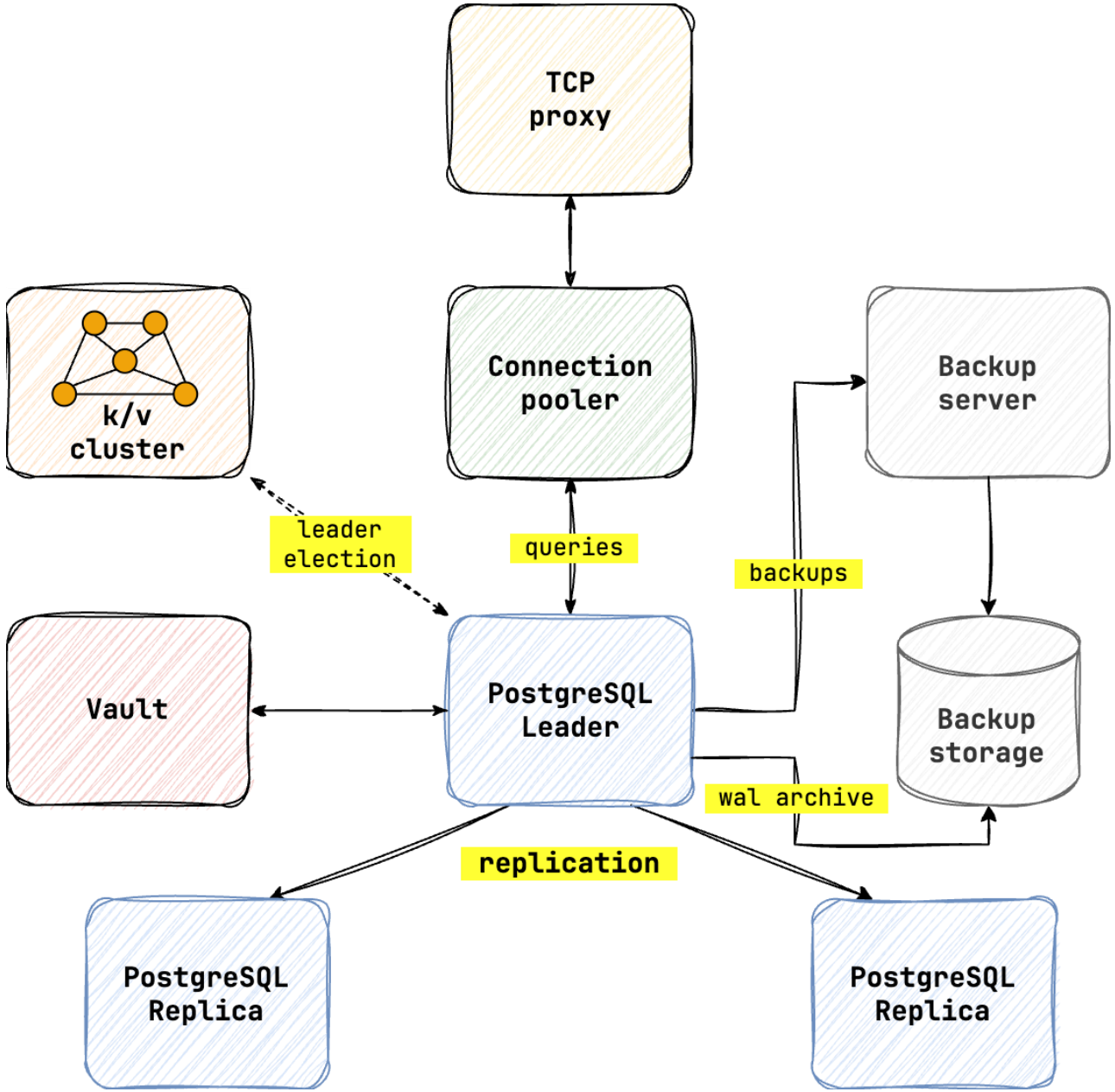
Анекдот про попугаев

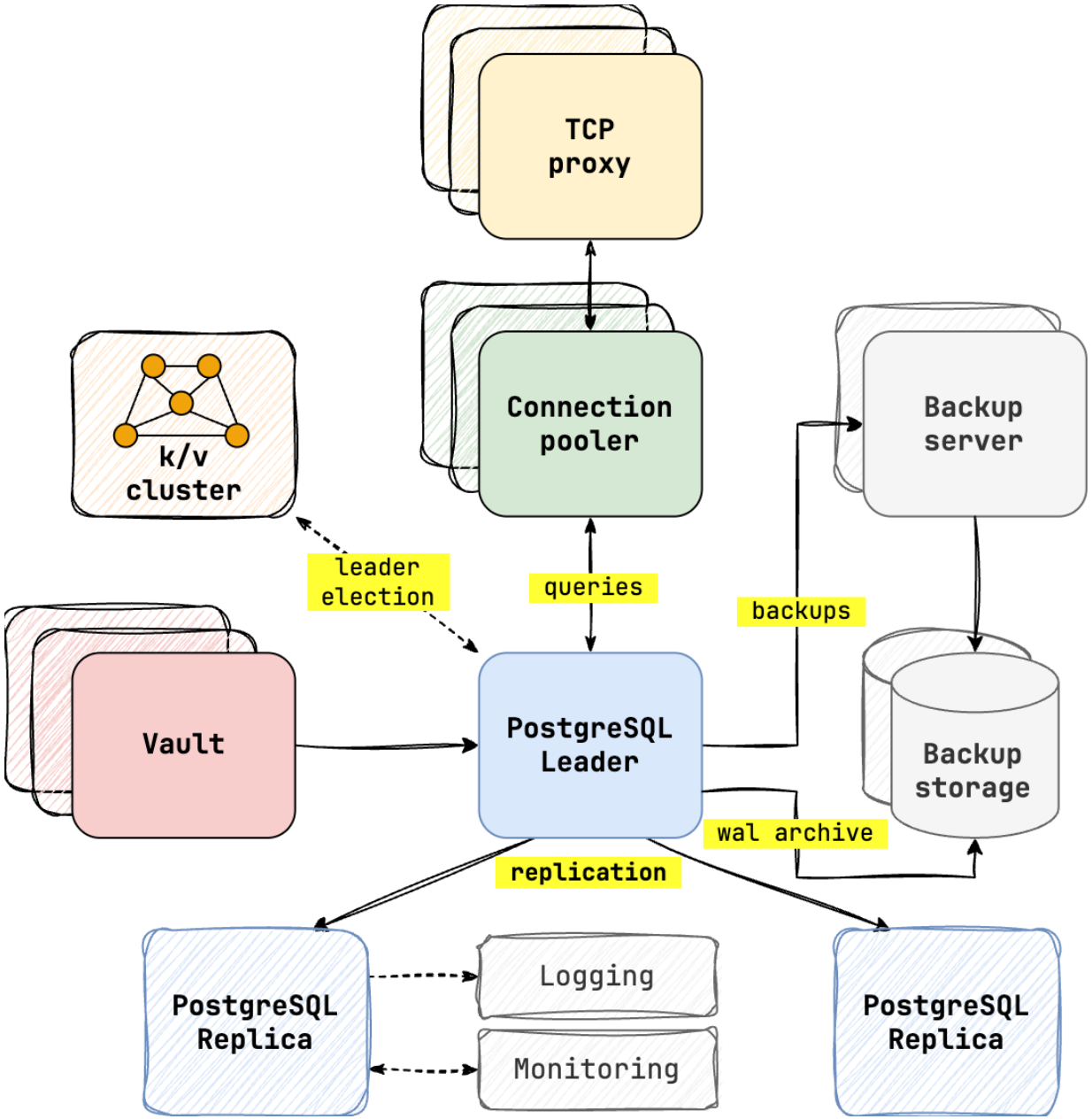
PostgresPro


Пользователи
Веб-интерфейс
Сервис
Мониторинг
Очередь
Отказоустойчивый кластер PostgreSQL
Хранилище
Очередь
Очередь
Очередь
Сервис


Пользователи
Сервис
Мониторинг
Очередь
Очередь
Хранилище
Очередь
Очередь

PostgresPro


TCP
proxy
Connection
pooler
k/v
cluster
Backup
server
leader
election
queries
backups
Vault
PostgreSQL
Leader
Backup
storage
wal archive
replication
PostgreSQL
Replica
PostgreSQL
Replica


TCP
proxy
Connection
pooler
k/v
cluster
Backup
server
leader
election
queries
backups
Vault
PostgreSQL
Leader
Backup
storage
wal archive
replication
PostgreSQL
Replica
Logging
Monitoring
PostgreSQL
Replica

Что нужно СУБД?

PostgresPro

- Общее хранилище
  - Etcd/Consul
- Кластерное ПО
  - Patroni/Stolon
  - Corosync/Pacemaker
  - pg_auto_failover
- Точка входа
  - TCP Proxy
  - DNS
  - VIP
  - Multi DSN

Что нужно СУБД?

PostgresPro

- Пулер соединений
  - Pgbouncer
  - Odyssey
- Резервное копирование
  - pgBackRest
  - WAL-G
  - pg_probackup
- Мониторинг
  - exporters
  - sql exporter
  - otel collector
- Логгинг
  - Vector/Fluentd

Хранение конфигурации

PostgresPro

Etcd и Consul

- распределенное хранилище k/v
- хранение небольших объемов данных
- лучше, если данные могут полностью поместиться в памяти

Consul

- сервисное решение
- управление соединением между сервисами
- обнаружение сервисов
- объединение сервисов
- управление трафиком
- автоматическое обновление инфраструктуры

Хранение конфигурации

PostgresPro

Etcd

- Простой
- RBAC
- Нет полноценного multi DC

Consul

- Сложный
- Если нужно что-то большее, чем хранение ключей
- Полноценное service discovery
- Multi DC setup
- Можно использовать в качестве DNS
- ACL

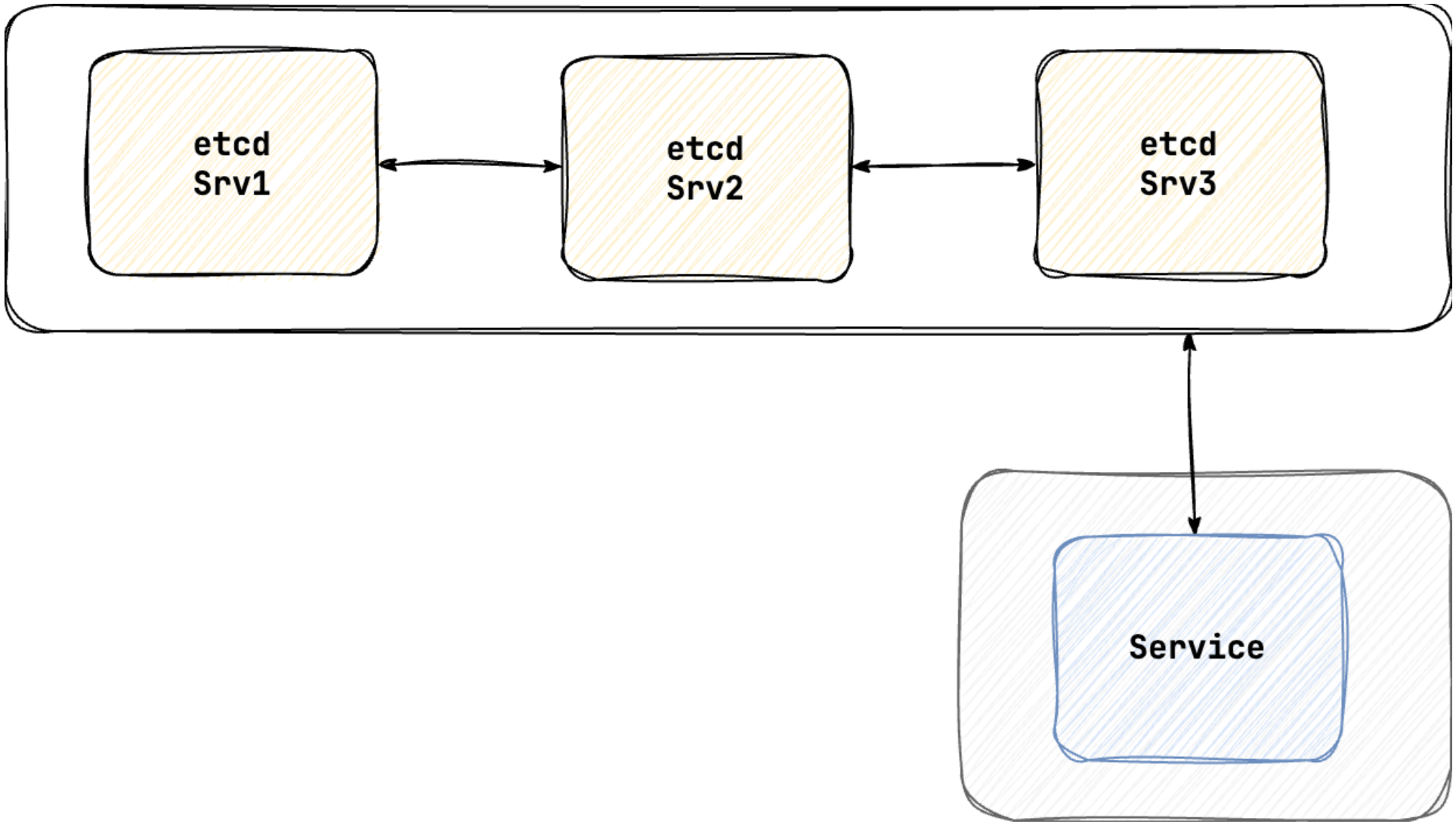
Хранение конфигурации

PostgresPro

- Могут обслуживать много кластеров СУБД
- Лучше размещать отдельно
  - требовательные к I/O
  - если есть ресурсы
- etcd versus other key-value stores
  - https://etcd.io/docs/v3.5/learning/why/

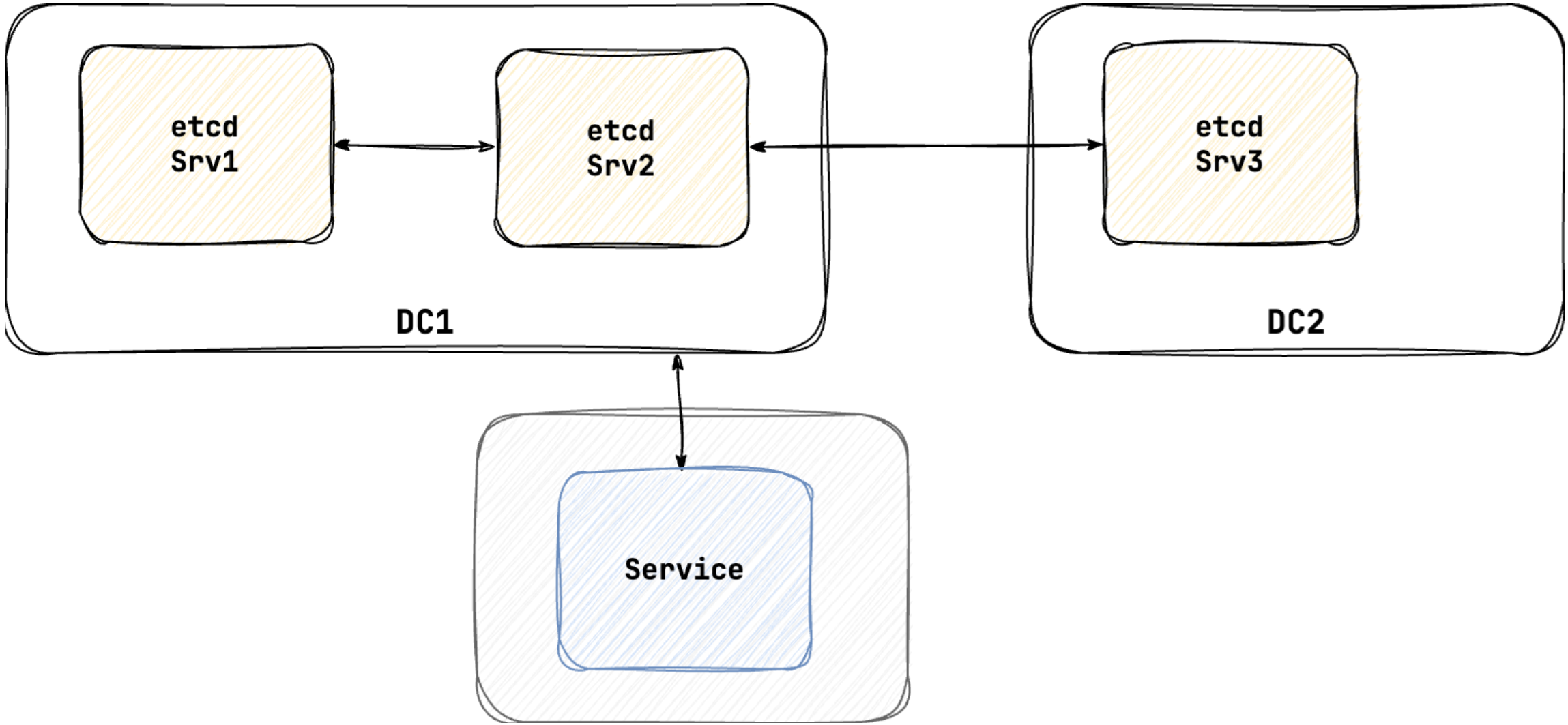
Etcd

PostgresPro



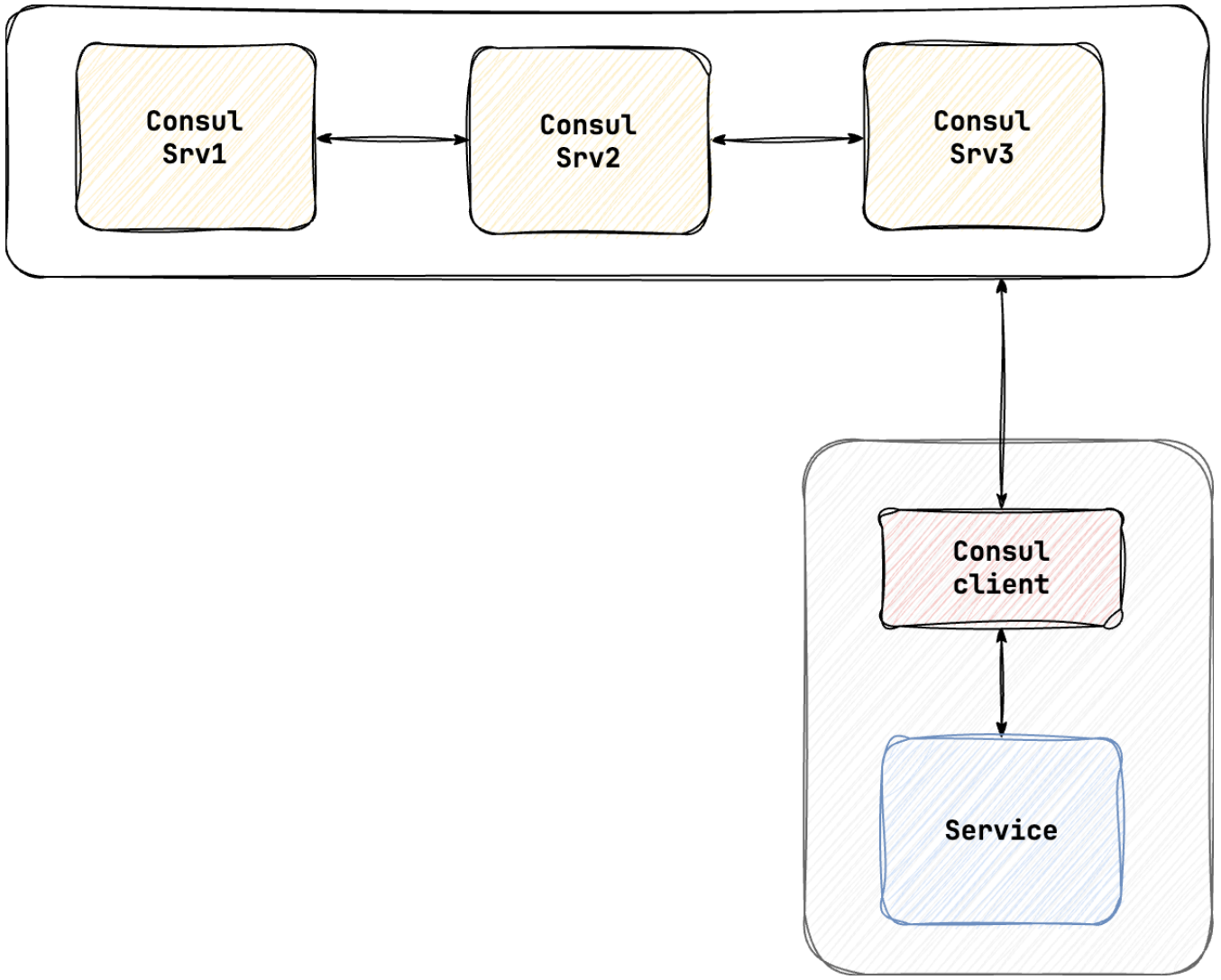
Etcd

PostgresPro



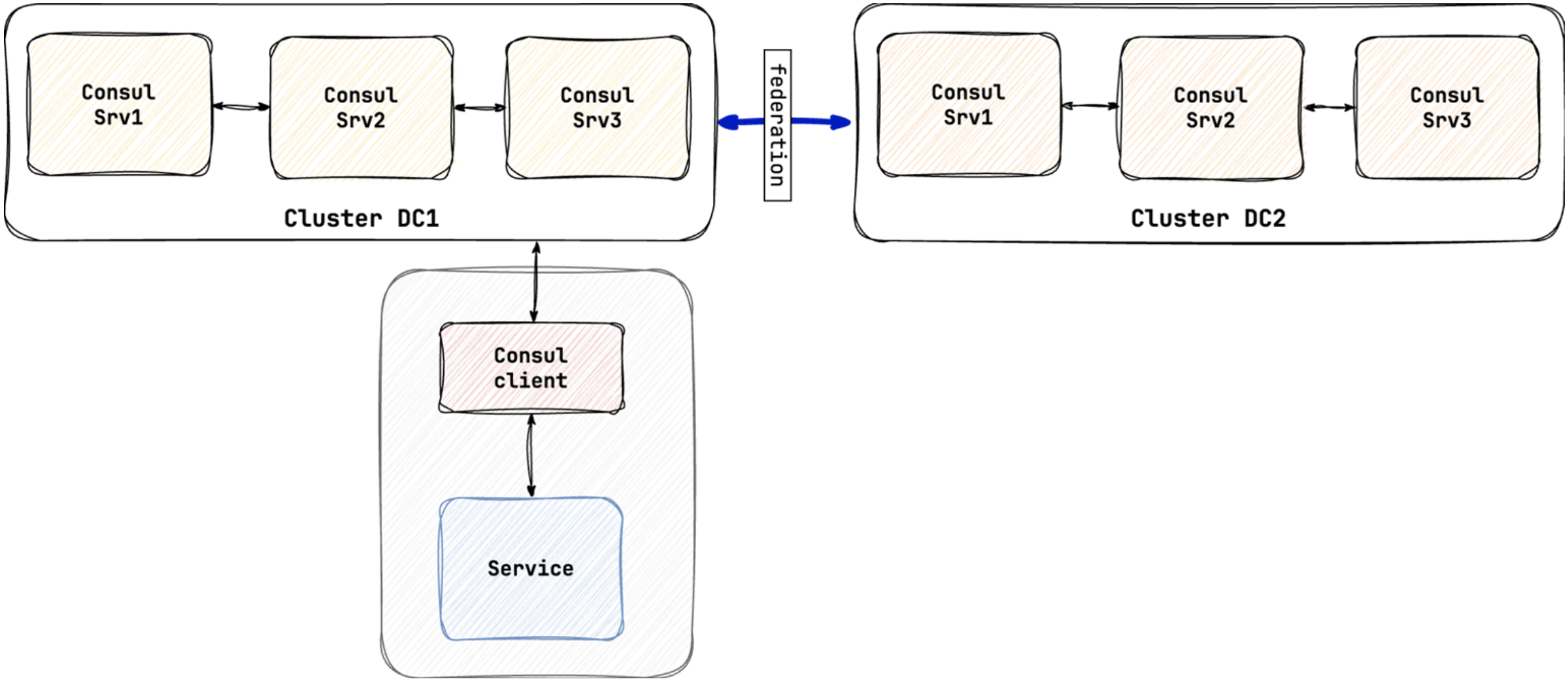
Consul

PostgresPro



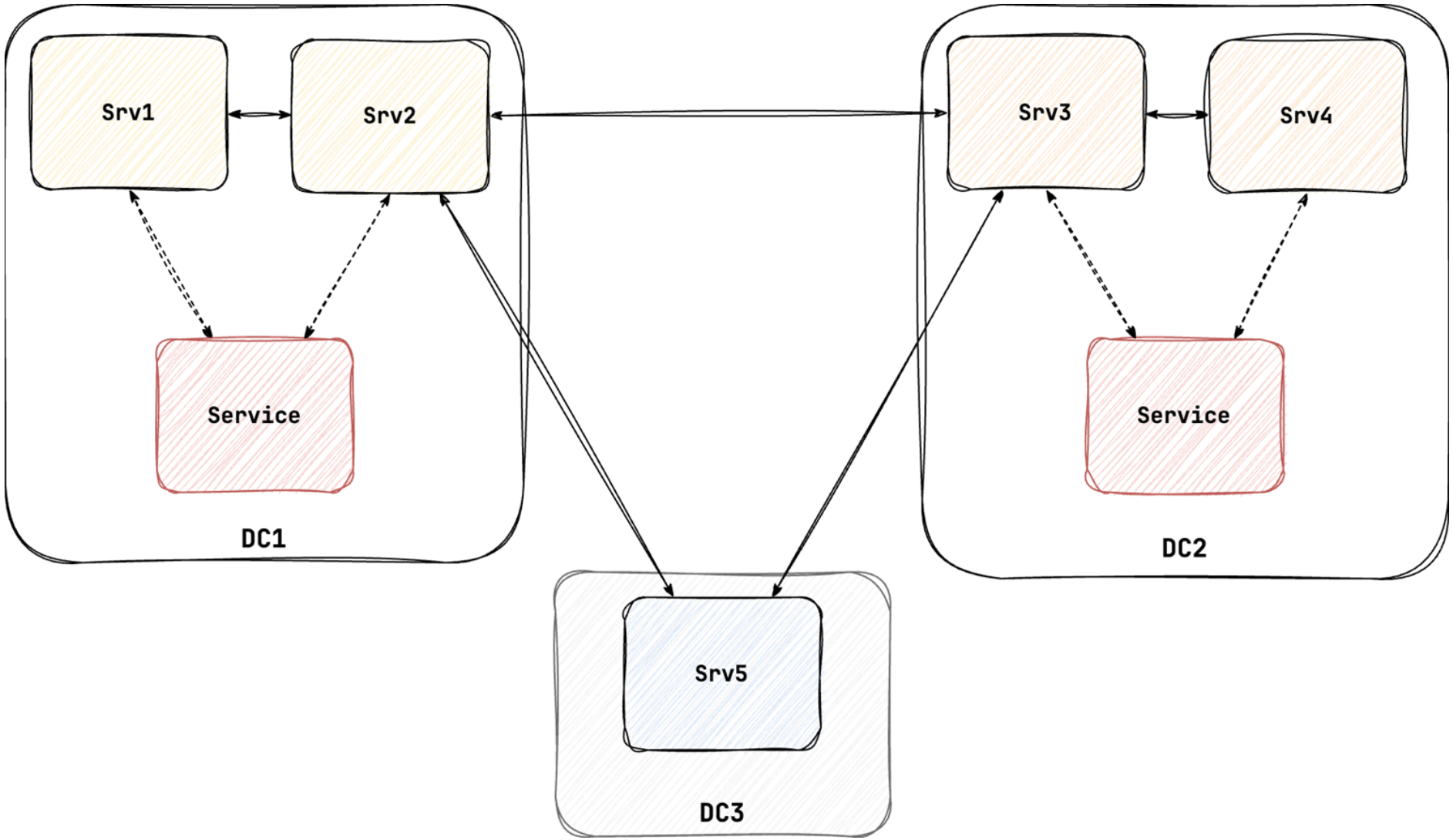
Consul

PostgresPro



Растянутый Multi DC setup

PostgresPro



Patroni или Stolon

PostgresPro

Patroni

PostgreSQL

- Поддержка форков
- CitusDB
- Кворумная репликация

Настройки и управление

- Колбеки
- Методы инициализации кластера
- Методы создания реплик
- Валидация конфигурации
- Управление через ctl/web api
- Фенсинг – Watchdogs, коллбек
- Теги и приоритеты

Установка

- Пакеты Python
- virtual environments
- K8s оператор
- Контейнеры

Stolon

PostgreSQL

- Только ванилла
- Использует свой прокси
- Игнорирует параметры

Настройки и управление

- Фенсинг – Stolon Proxy
- Поддержка k8s
- Управление через ctl

Установка

- Бинарные файлы

Почему не Stolon?

PostgresPro

- **Использует свой прокси**
- **Нет приоритетов при выборах лидера**
- **Можно при инициализации кластера указать, кто мастер**
- **Все запросы идут только на мастер**
- **На реплики отправить запрос через прокси невозможно**
- **Игнорирует некоторые параметры**
  - listen_addresses
  - port
  - unix_socket_directories
  - wal_keep_segments
  - wal_log_hints
  - hot_standby
  - max_replication_slots
  - max_wal_senders
  - synchronous_standby_names
  - pg_hba.conf
- **Проблемы с рестартом кластера**

Почему Patroni?

PostgresPro

API для управления

Большой опыт в продакшене

Есть коллбеки, если кластер меняет своё состояние (об этом можно куда-то сообщить)

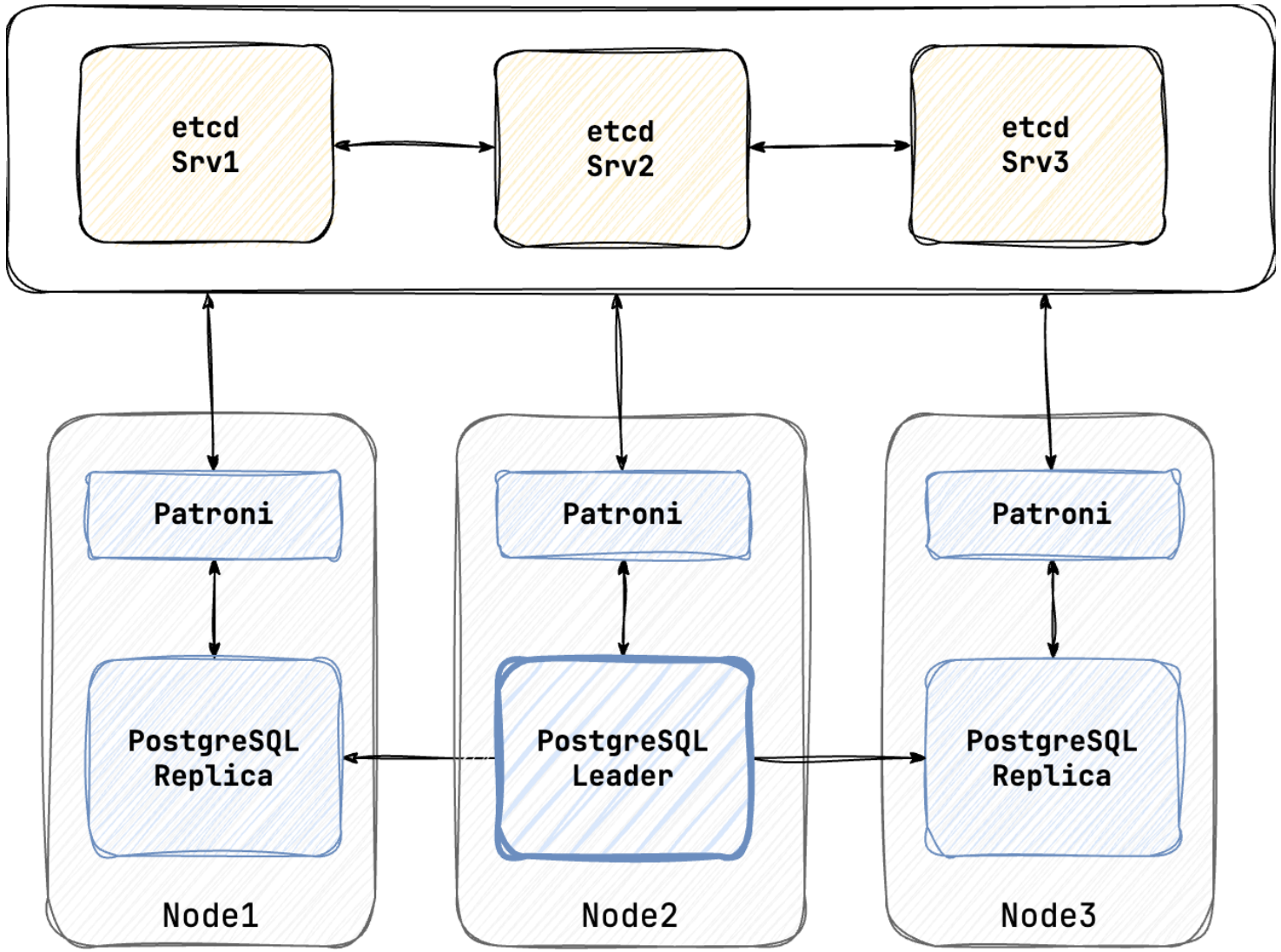
С большой вероятностью найдем решения проблем

Гибкий в настройке

Больше экспертов

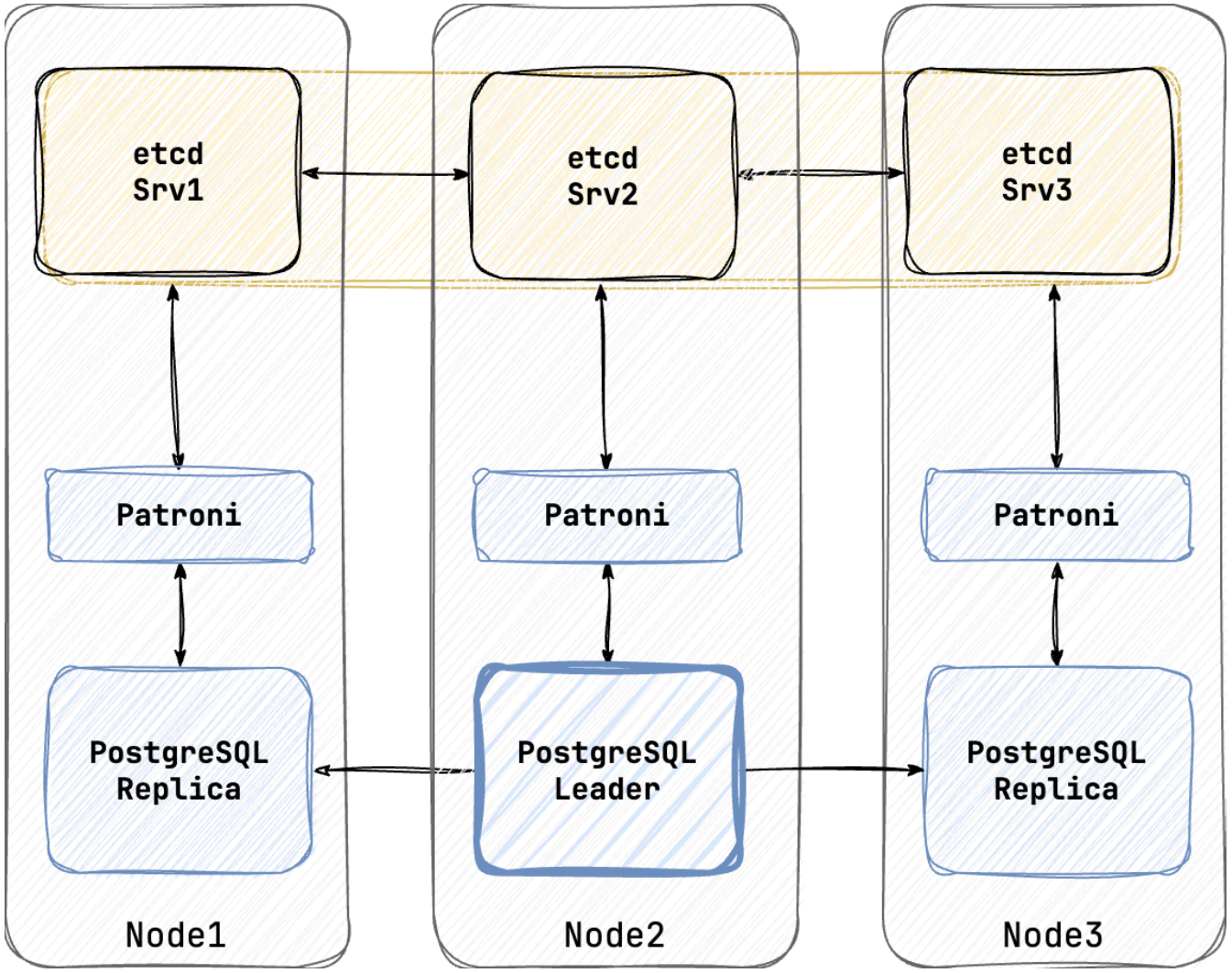
etcd + patroni + pg

PostgresPro



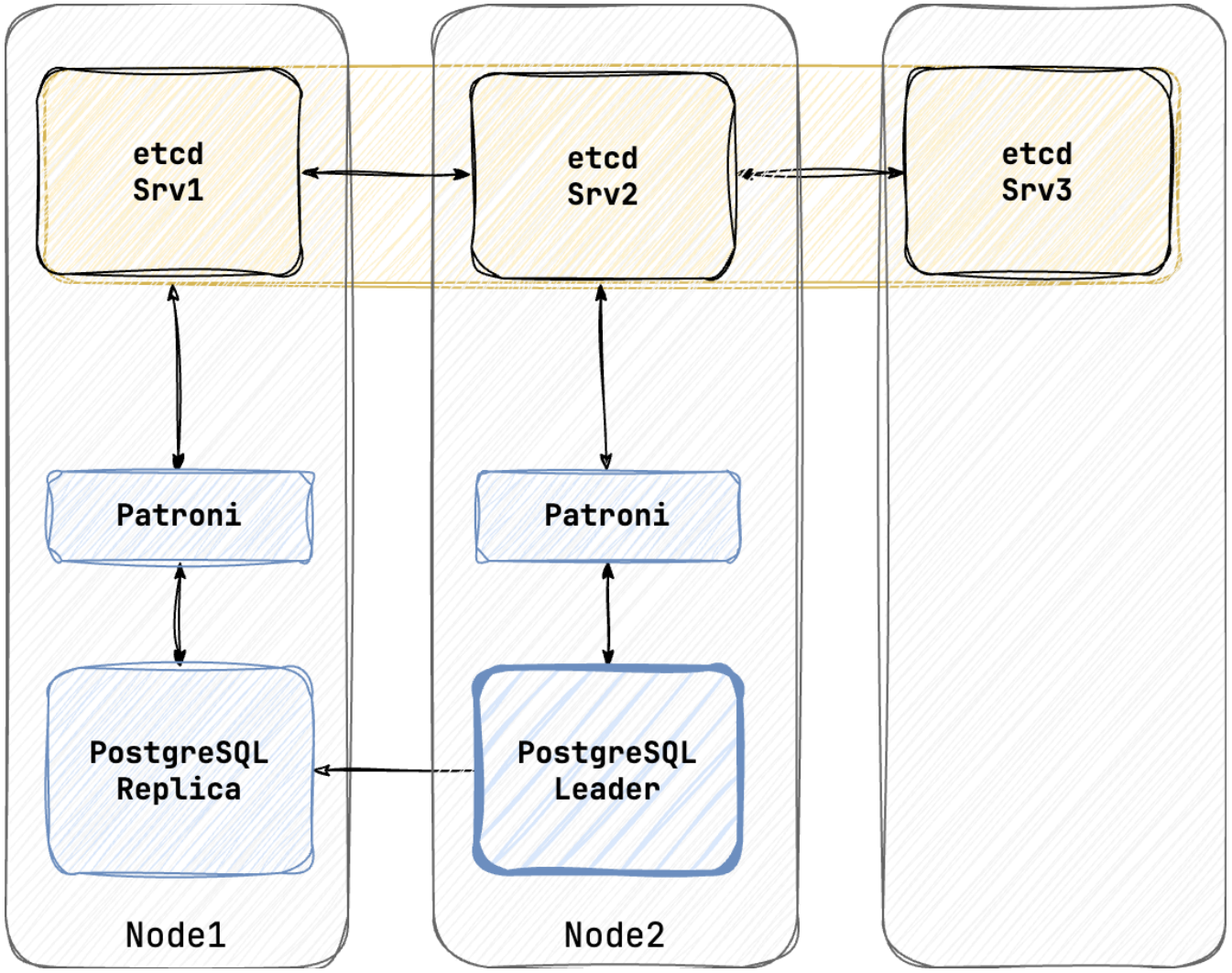
etcd + patroni + pg

PostgresPro



etcd + patroni + pg

PostgresPro



Для обозначения понятия грех, используется еврейское חטא – хаттá, или греческое ἁμαρτία — амартúа.

Словарь Стронга

https://bible.by/strong-hebrew
https://bible.by/strong-greek

В обоих языках глагольные формы этих слов означают «промахнуться; не попасть в цель» — в смысле не достичь цели, не дойти до нужной отметки.

HAProxy

PostgresPro

(01)

External-check

Можно написать какие угодно проверки
на самом уютном языке

(02)

Http-check

Используется в связке с Patroni,
там есть встроенный http-сервер

(03)

Pgsql-check

Когда нет Patroni

(04)

Tcp-check

Когда хочется упороться

HAProxy

PostgresPro

```
backend leader

    option tcp-check
    tcp-check connect

# Send Auth-req
    tcp-check send-binary 00030000                  # protocol version           (  4 bytes )
    tcp-check send-binary 7573657200                # "user"                     (  5 bytes )
    tcp-check send-binary 7573657200                # "user"                     (  5 bytes )
    tcp-check send-binary 6461746162617365500       # "database"                 (  9 bytes )
    tcp-check send-binary 74656d706c6174653100      # "template1"                ( 10 bytes )
    tcp-check send-binary 00                        # terminator                 (  1 byte  )

# expect: Auth
    tcp-check expect binary 52                      # Auth request
    tcp-check expect binary 00000008                # packet length : 8 bytes in (  4 bytes )
    tcp-check expect binary 00000000                # auth response ok           (  4 bytes )
```

Согрешать надо умеючи – 1!

PostgresPro

(01)

В кластере произошел фейловер

(02)

**Но только на тех запросах,
которые возвращали много данных**

(03)

Приложение начало «тупить»

(04)

**EXPLAIN ANALYZE показывает,
что ничего не поменялось**

Согрешать надо умеючи – 1!

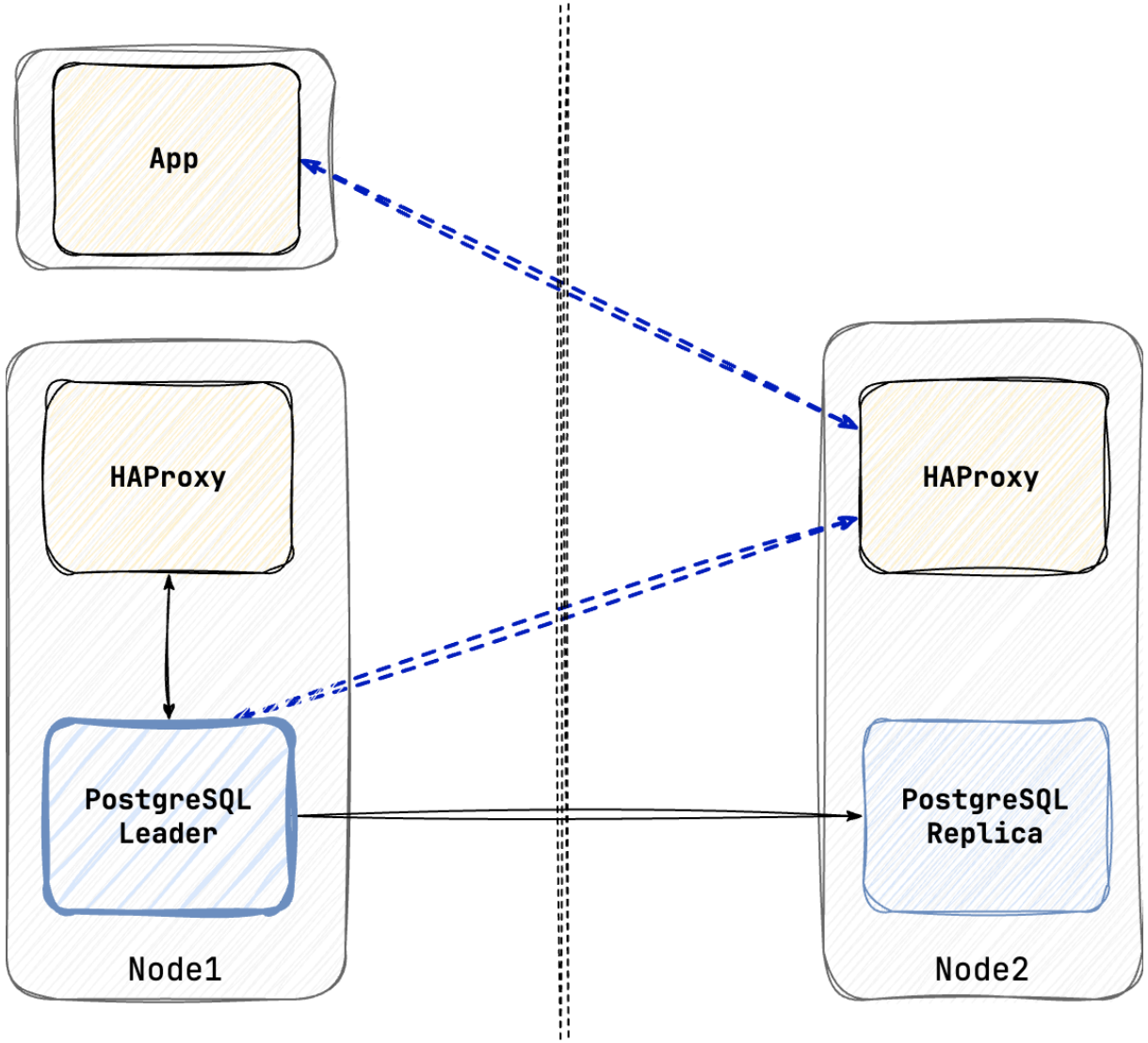
PostgresPro

```
postgres=# select mean_exec_time from pg_stat_statements where
query like '%test%' \gx

-[ RECORD 1 ]---+---------
mean_exec_time  | 1.680021
```

```
postgres=# select mean_exec_time from pg_stat_statements where
query like '%test%' \gx

-[ RECORD 1 ]---+------------
mean_exec_time  | 4004.916112
```

Согрешать надо умеючи – 1!

PostgresPro



Сильно упрощенный механизм работы сбора статистики планов pg_stat_statements (pgss)

PostgresPro

Для каждого SQL-запроса выбирается план

Перед занесением информации в pgss рассчитывается время выполнения запроса

Верхнеуровневый узел плана отдаёт строки клиенту

Информация о запросе заносится pgss

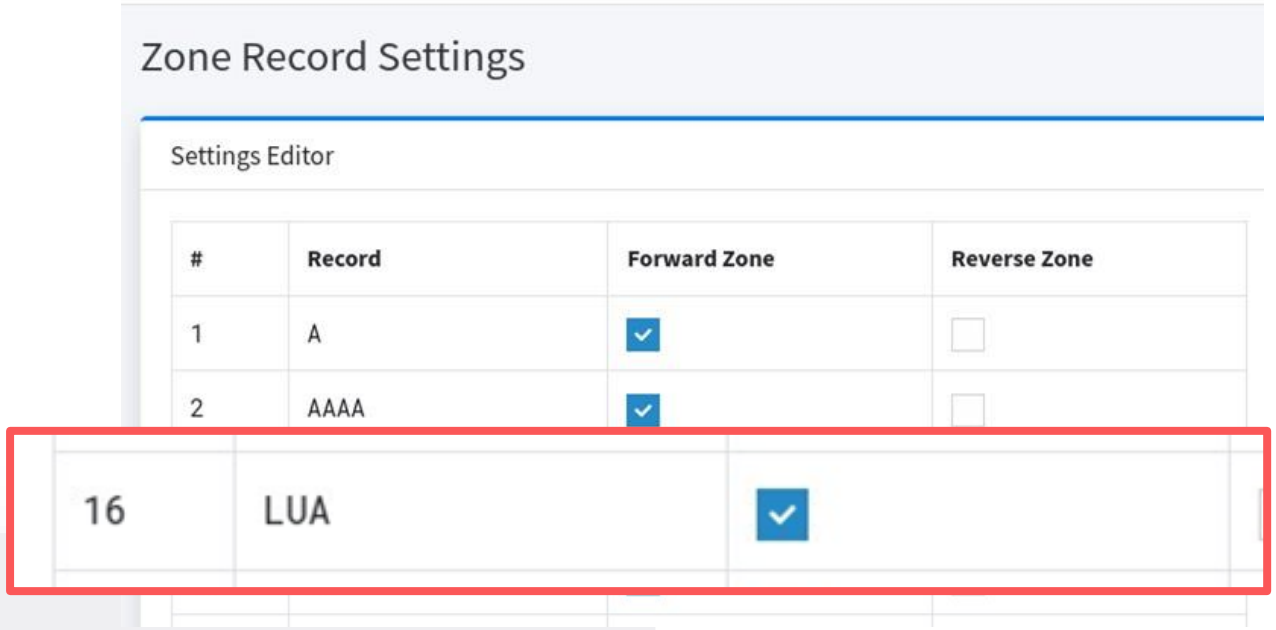
Узел завершает свою работу только после отдачи всех строк

Иными словами, pgss считает временем окончания запроса, момент, когда закончилась отдача всех строк клиенту.

DNS

PostgresPro

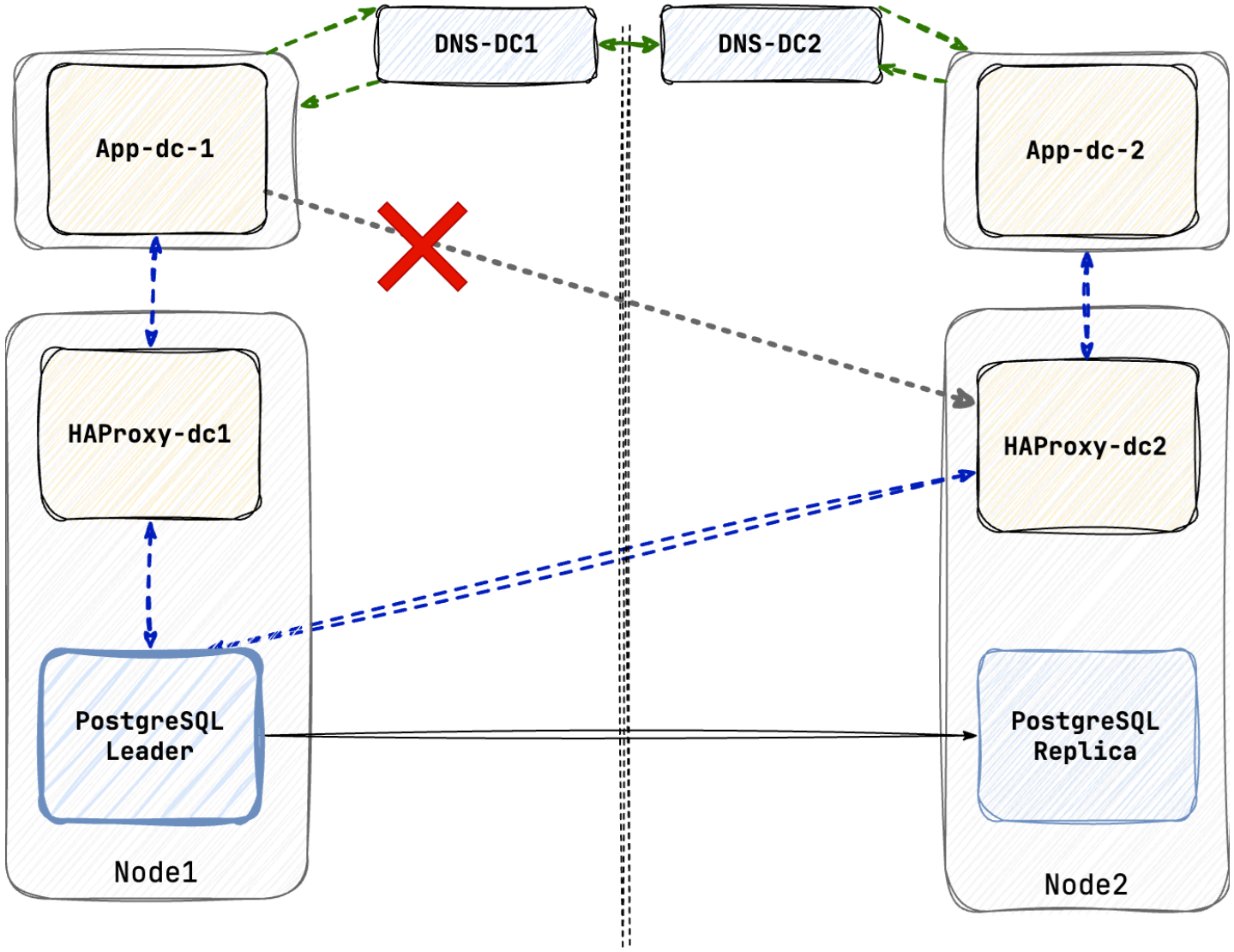
- Должен быть сервер, поддерживающий тип записи LUA
- Например, PowerDNS
- curl -I http_endpoint




```
leader LUA 60 A "ifurlextup(
{{
 ['192.168.21.113']='http://192.168.21.113:8008/leader',
 ['192.168.21.114']='http://192.168.21.114:8008/leader',
 ['192.168.21.115']='http://192.168.21.115:8008/leader'
}}
)"
```

DNS

PostgresPro



Virtual IP

PostgresPro

- **Скрипт на коллбек**
  - трудно сделать атомарным
  - on_start, on_stop, on_role_change, pre_promote
- **VIP-manager**
  - базируется на проверке ключа лидера
- **Keepalived**
  - старый добрый
  - скрипты для проверки ключа лидера
  - скрипты для обхода API ендпоинтов

Multi Host DSN

PostgresPro

- Строка подключение к БД содержит все узлы кластера
- Параметры
  - connect_timeout
  - connect_failover_timeout
    - нет в ванилле
    - при фейловере надо учесть ретрай
  - target_session_attrs=read-write

Согрешать надо умеючи – 2!

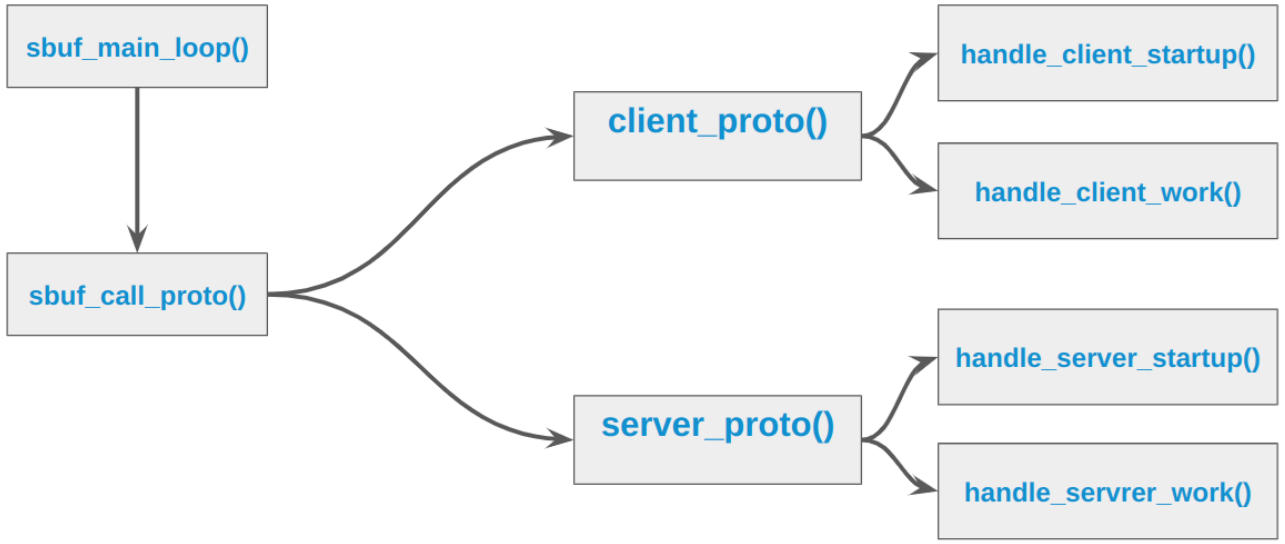
PostgresPro

- Если первый узел в строке не лидер, а асинхронная реплика
- Приходят две сессии
  - первая – `create database`
  - вторая сессия пытается подключится к этой БД
  - попадает на асинхронную реплику
  - проверка, что узел не реплика идет после подключения к БД
  - все сломается, так как этой базы нет
    - репликация не доехала

Pgbouncer

PostgresPro

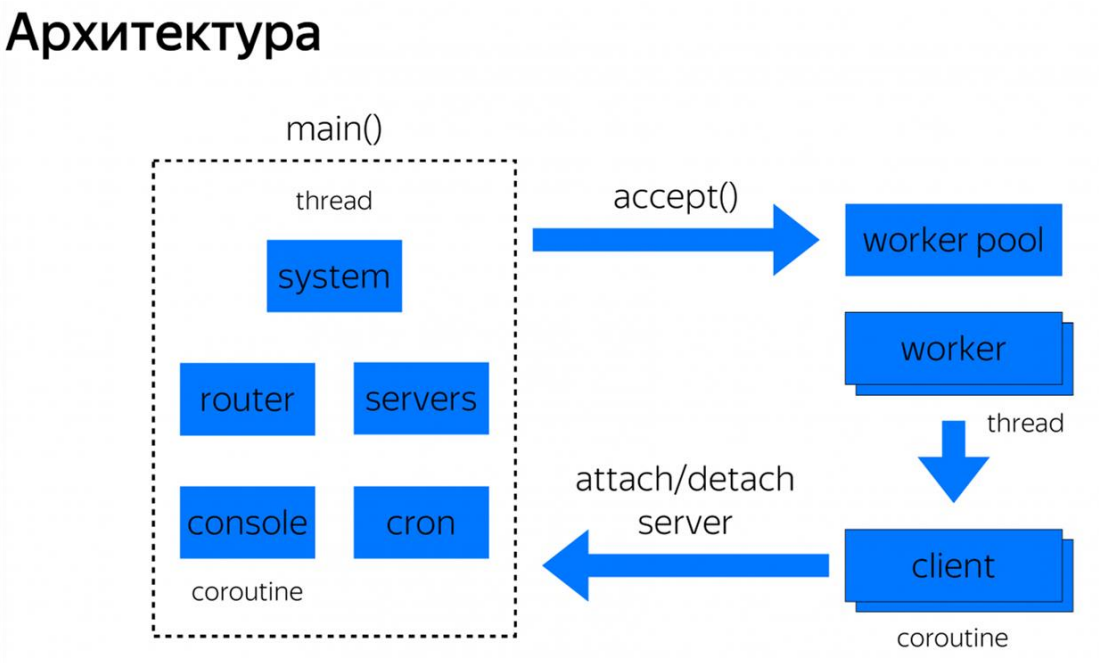
- **Событийно-ориентированная программа**
- **Каждое соединение описывается машиной состояний**
  - CL_LOGIN, CL_ACTIVE, SV_LOGIN_SV_IDLE, SV_ACTIVE
  - можно наблюдать в консоли управления
- **Проблемы масштабируемости**
  - производительность работа с TLS
  - требует максимального участия CPU
  - однопоточная программа



Odyssey

PostgresPro

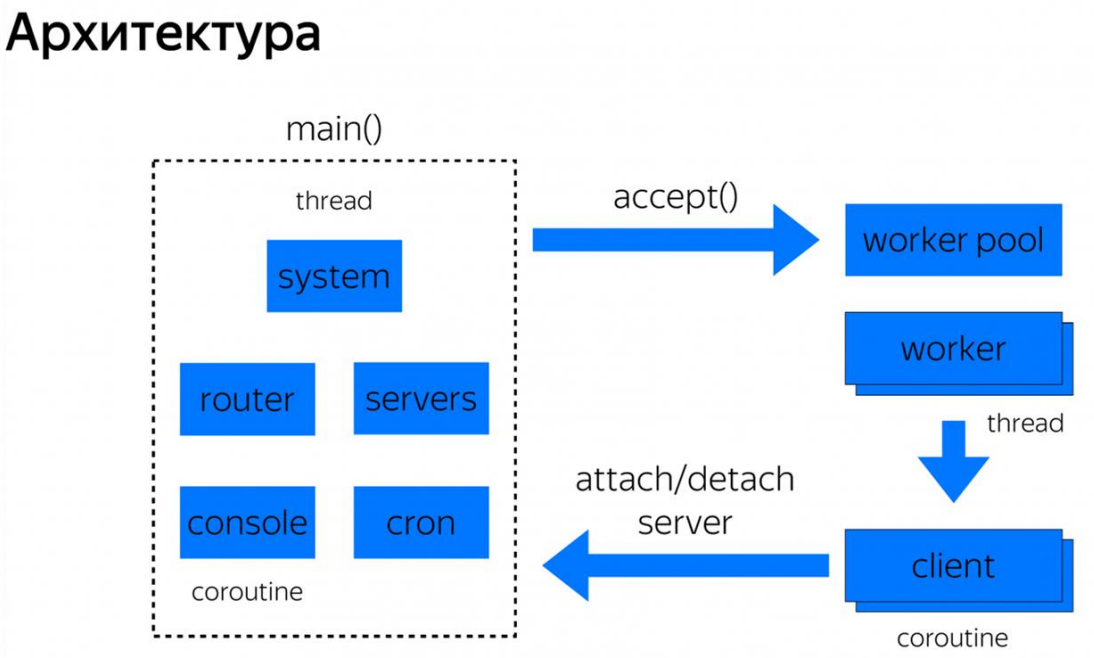
- Реализован с помощью двух библиотек
  - Kiwi, разбора протокола PostgreSQL
  - Machinarium, кооперативный многозадачный движок на C++
- Используется идея кооперативной многозадачности
- Каждое соединение обслуживает отдельная корутина
- Исполняется на какой-то из машин



Odyssey

PostgresPro

- Машина – поток, прикрепленный к определённому процессорному ядру
- На нём регистрируются корутины
- Заменяет машину состояний pgbouncer
- Цепочки вызовов функций во время обработки соединений



	Pgbouncer	Odyssey
Многопоточность	Да, с оговорками	Да, из коробки
Поддежка cancel	Да (с 1.19)	Да
Последний релиз	Апрель 2025	Июнь 2022
Производительность	Лидер	30-50% ниже лидера

Pgbouncer

- Развивается линейно и регулярно
- Позволяет использовать один порт несколькими процессами

Odyssey

- Не обновлялся уже больше года
- Часть фич, которые обещали реализовать в 2019 так и не случились
  (например, принудительное направление RO-запросов в реплики)
- Стабильность latest-сборки вызывает вопросы
- Поддерживает многопоточность

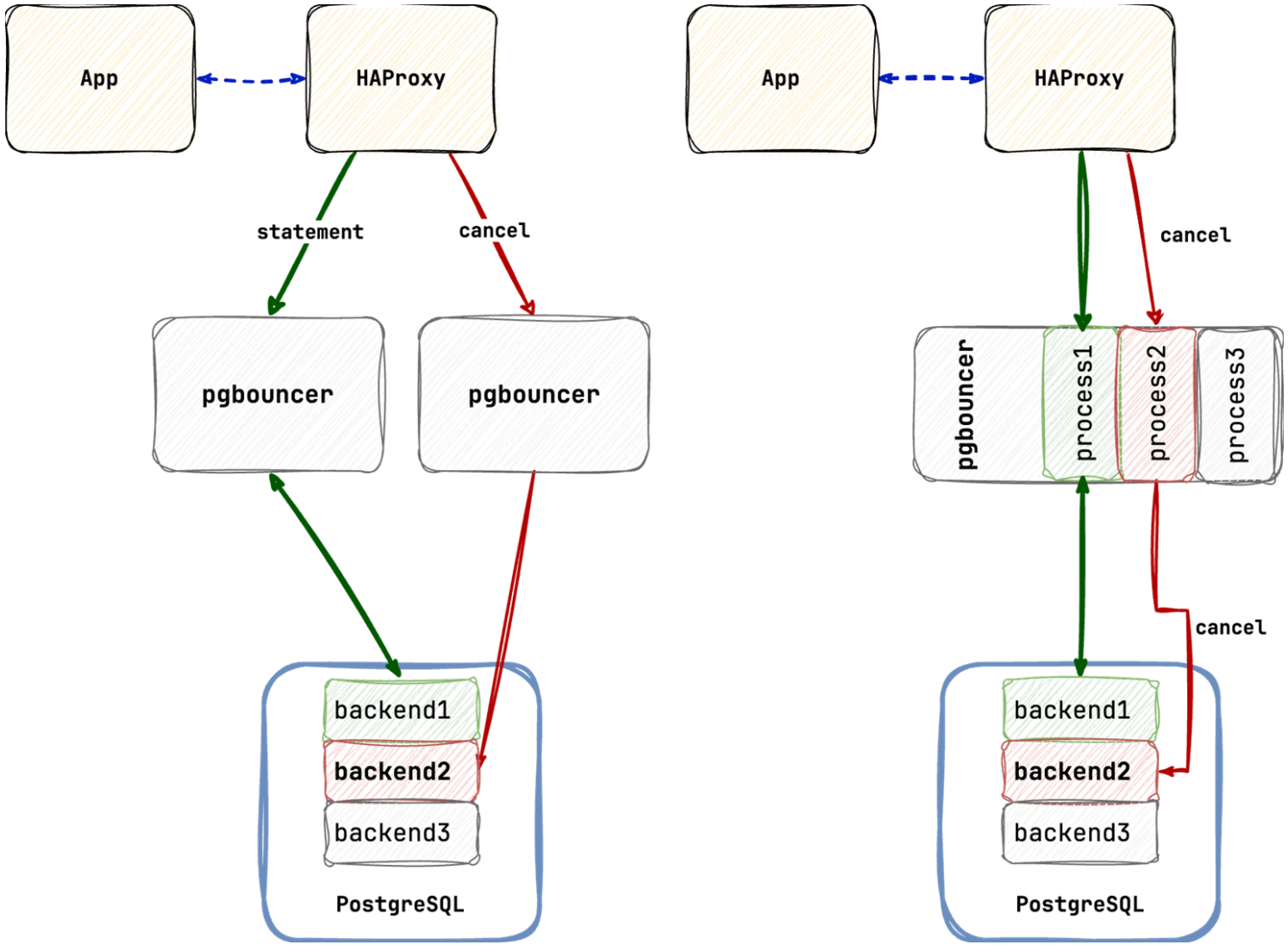
Pgbouncer

PostgresPro

- Если pgbouncer утилизирует ядро на 100%,
  то можно использовать блок peers в конфигурации
- Можно распределить нагрузку на несколько ядер,
  добавляя рабочие процессы, которые работают на одном порту
- Если приложение может работать в режиме transaction
  или statement, то это позволит увеличить производительность
  и снизить утилизацию ресурсов
- TLS offload
- Альтернатива:
  pg_doorman – многопоточный pg_bouncer на Rust
  https://github.com/ozontech/pg_doorman

Согрешать надо умеючи – 3!

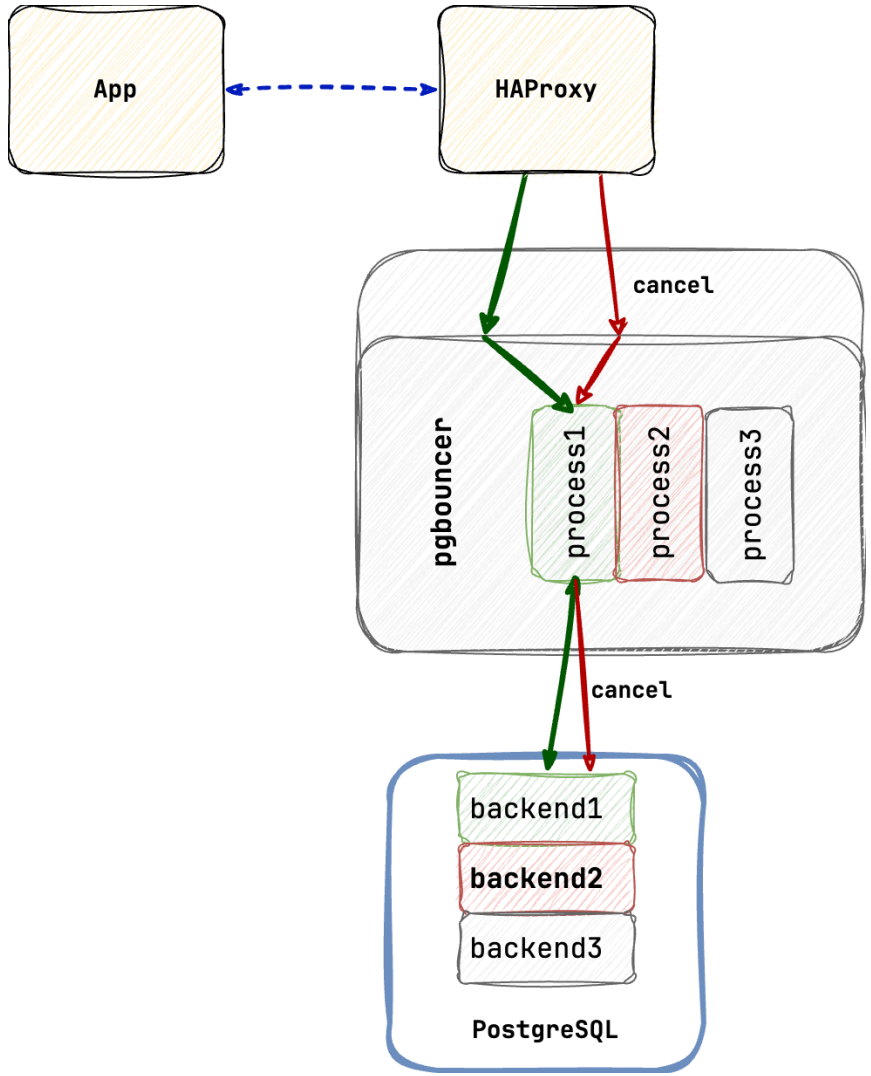
PostgresPro



Согрешать надо умеючи – 3!

PostgresPro

- при получении Cancel проверяется идентификатор сессии
- Cancel будет перенаправлен на пир, которому принадлежит сессия
- SO_REUSEPORT
  - несколько слушающих сокетов на одном порту
  - ядро распределяет входящие соединения



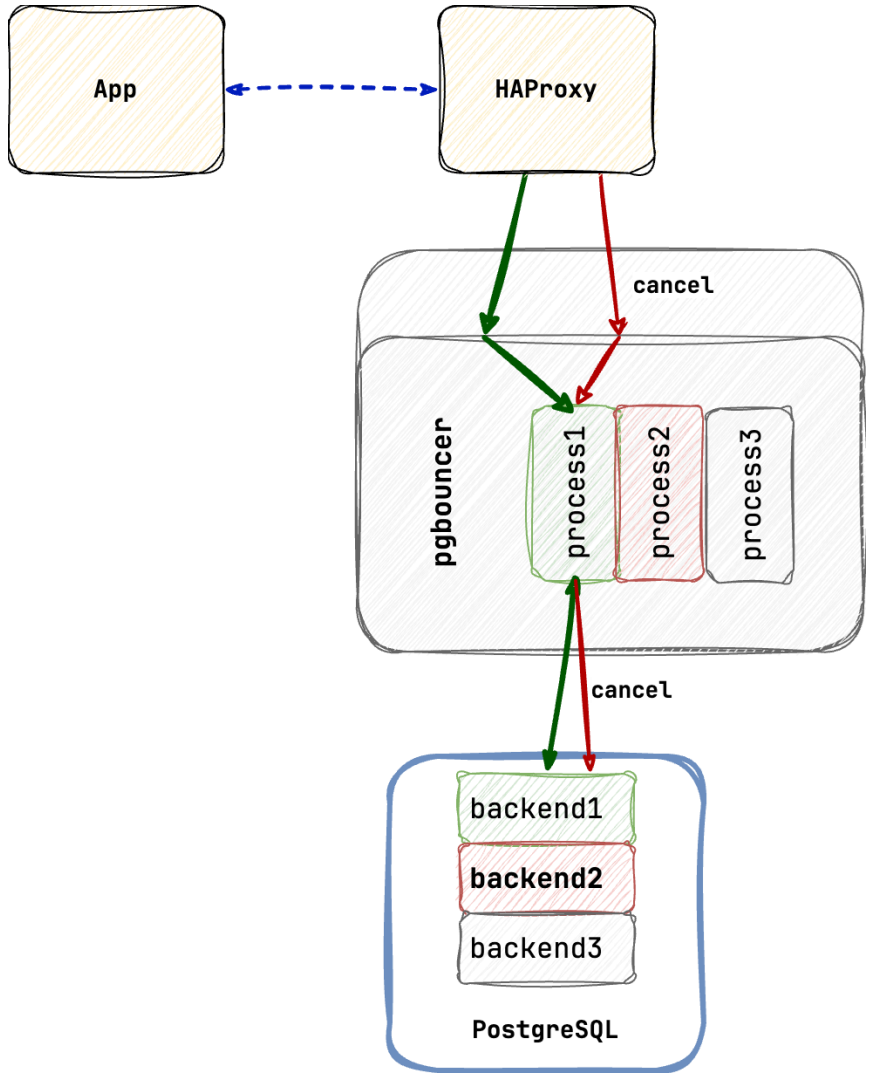
Согрешать надо умеючи – 3!

PostgresPro

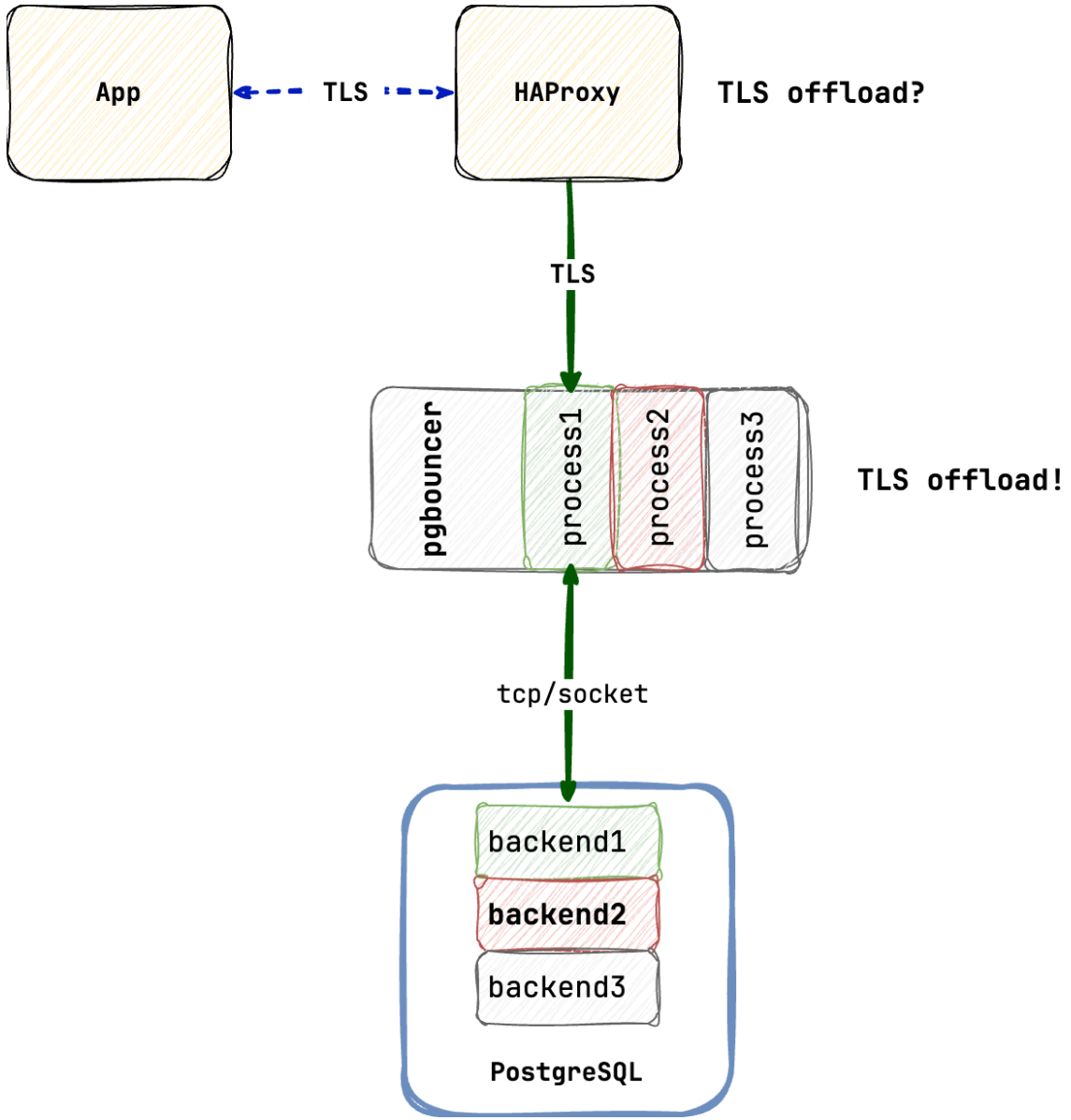
- Systemd unit
  - pgbouncer@.socket
  - pgbouncer@.service

- pgbouncer.ini

```
peer_id=<num>
so_reuseport=1
[peers]
1 = host=/var/run/pgbouncer/pgbouncer1
2 = host=/var/run/pgbouncer/pgbouncer2
3 = host=/var/run/pgbouncer/pgbouncer3
```



TLS offload


App
TLS
HAProxy
TLS offload?
TLS
pgbouncer
process1
process2
process3
TLS offload!
tcp/socket
backend1
backend2
backend3
PostgreSQL

Load Balancer

PostgresPro

(01)

Один endpoint на кластер (или группа endpoint'ов в случае использования нескольких балансировщиков)

(02)

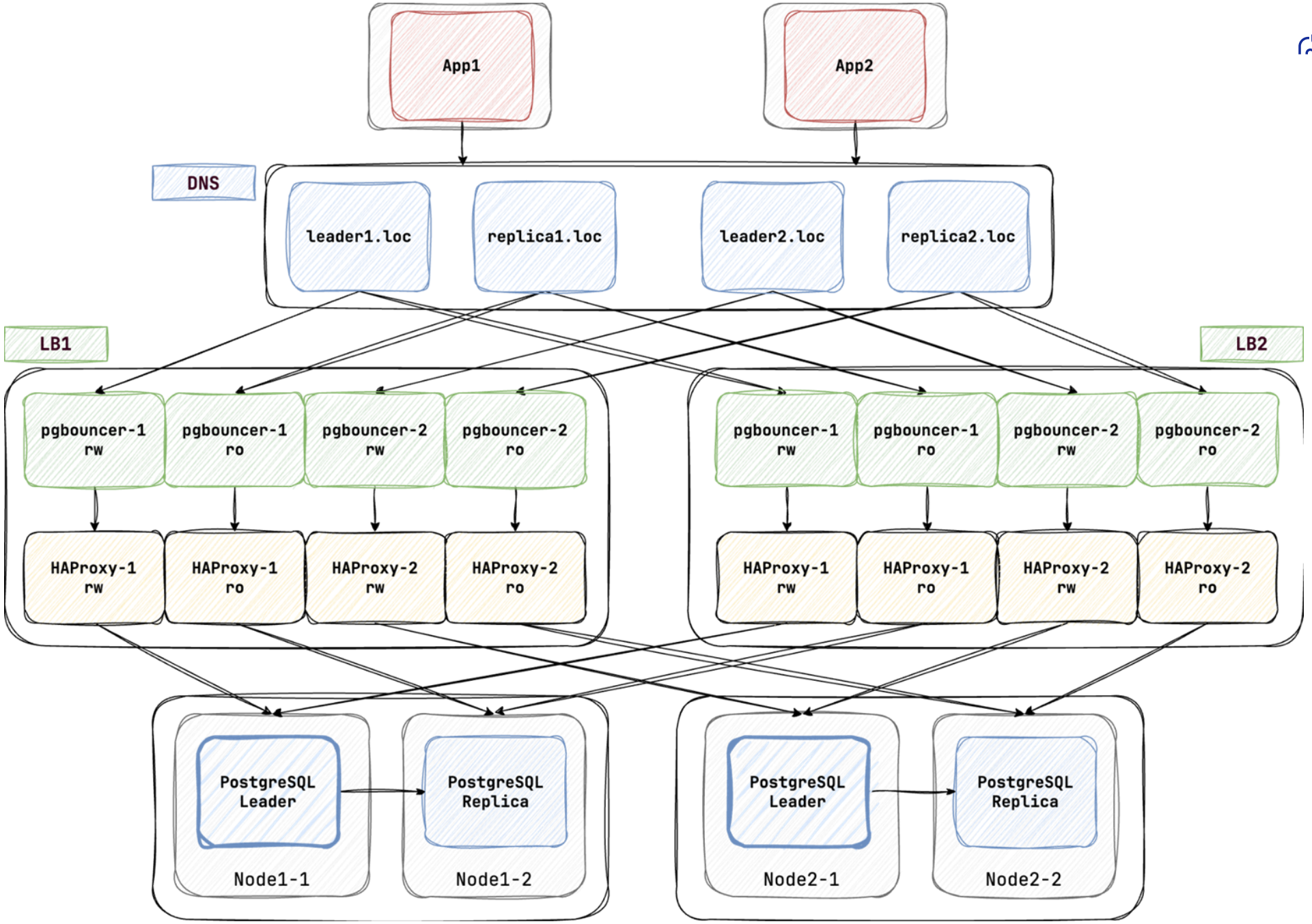
Сохранение коннекта (но не сессии) при переключении мастера

(03)

Равномерное распределение соединений между репликами

(04)

Гарантии разделения пулов между мастером и репликами


App1
App2
DNS
leader1.loc
replica1.loc
leader2.loc
replica2.loc
LB1
LB2
pgbouncer-1 rw
pgbouncer-1 ro
pgbouncer-2 rw
pgbouncer-2 ro
pgbouncer-1 rw
pgbouncer-1 ro
pgbouncer-2 rw
pgbouncer-2 ro
HAProxy-1 rw
HAProxy-1 ro
HAProxy-2 rw
HAProxy-2 ro
HAProxy-1 rw
HAProxy-1 ro
HAProxy-2 rw
HAProxy-2 ro
PostgreSQL Leader
PostgreSQL Replica
Node1-1
Node1-2
PostgreSQL Leader
PostgreSQL Replica
Node2-1
Node2-2

Vault

PostgresPro

Hashicorp Vault

Хранение сертификатов

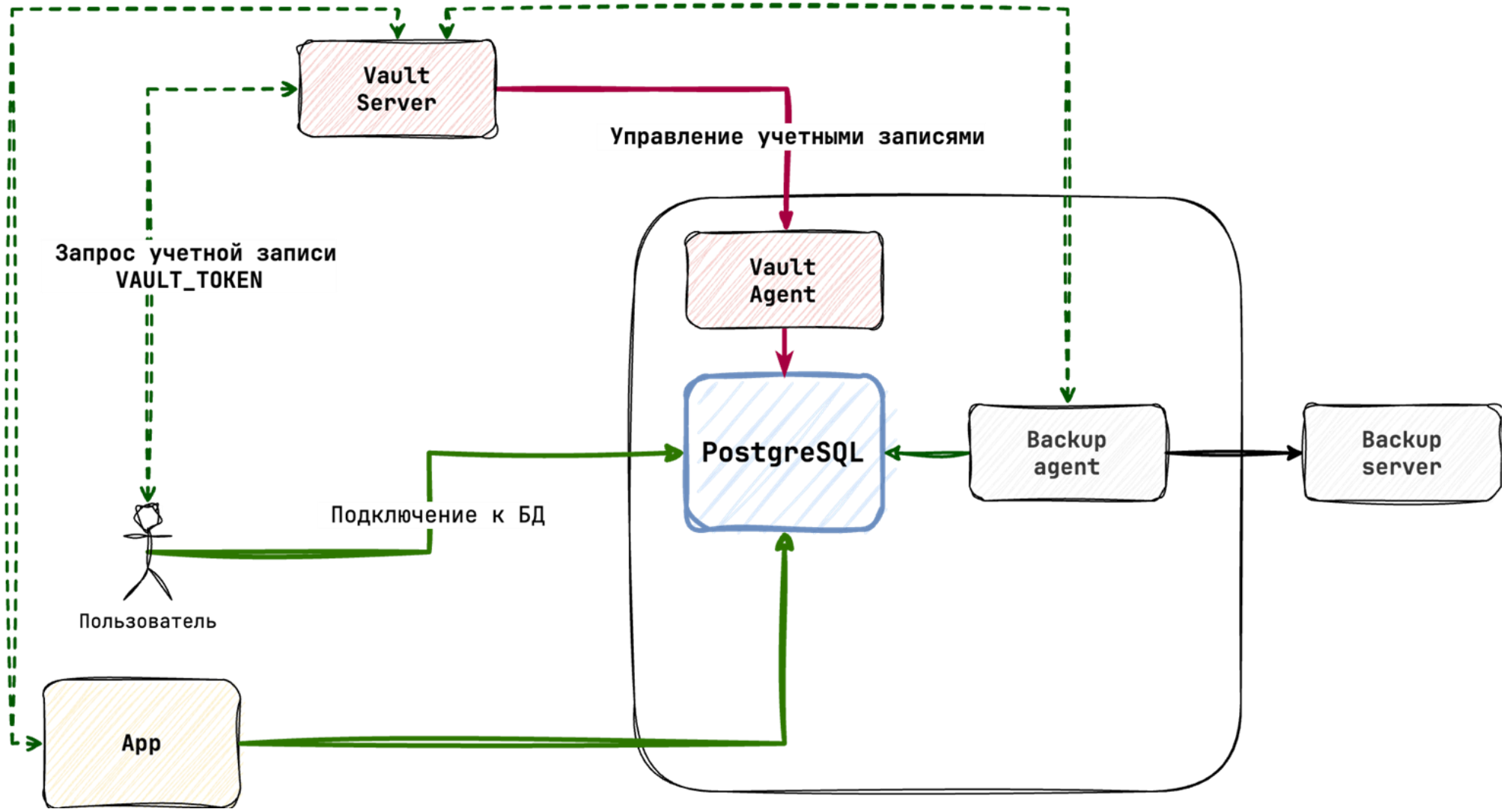
Плагин для PostgreSQL

Ротация паролей и сертификатов

Дает возможность получать короткоживущие пары логин-пароль для доступа к данным

Vault

PostgresPro



Vault

PostgresPro

```
vault write database/roles/readonly \
    db_name=postgres \
    creation_statements=@readonly.sql \
    default_ttl=1h \
    max_ttl=24h
```

```
CREATE ROLE "{{name}}"
WITH LOGIN PASSWORD '{{password}}'
VALID UNTIL '{{expiration}}' INHERIT;
GRANT ro TO "{{name}}";
```

Backup

PostgresPro

pgBackRest

S3 в инфраструктуре

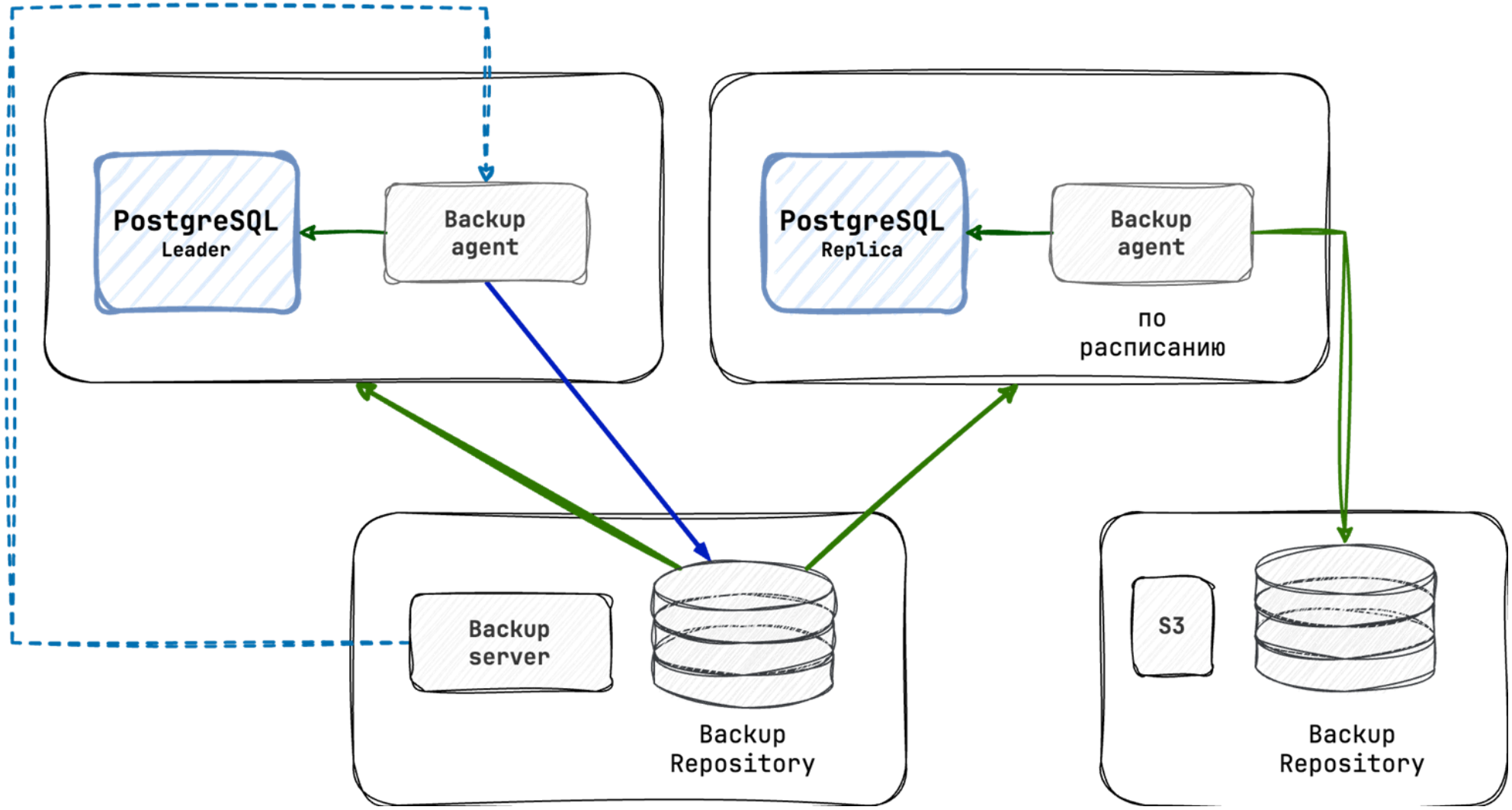
WAL-G

Общий репозиторий РК
на несколько ЦОД

pg_probackup

Backup

PostgresPro



Мониторинг и логгинг

PostgresPro

Otel-collector

Разные экспортёры

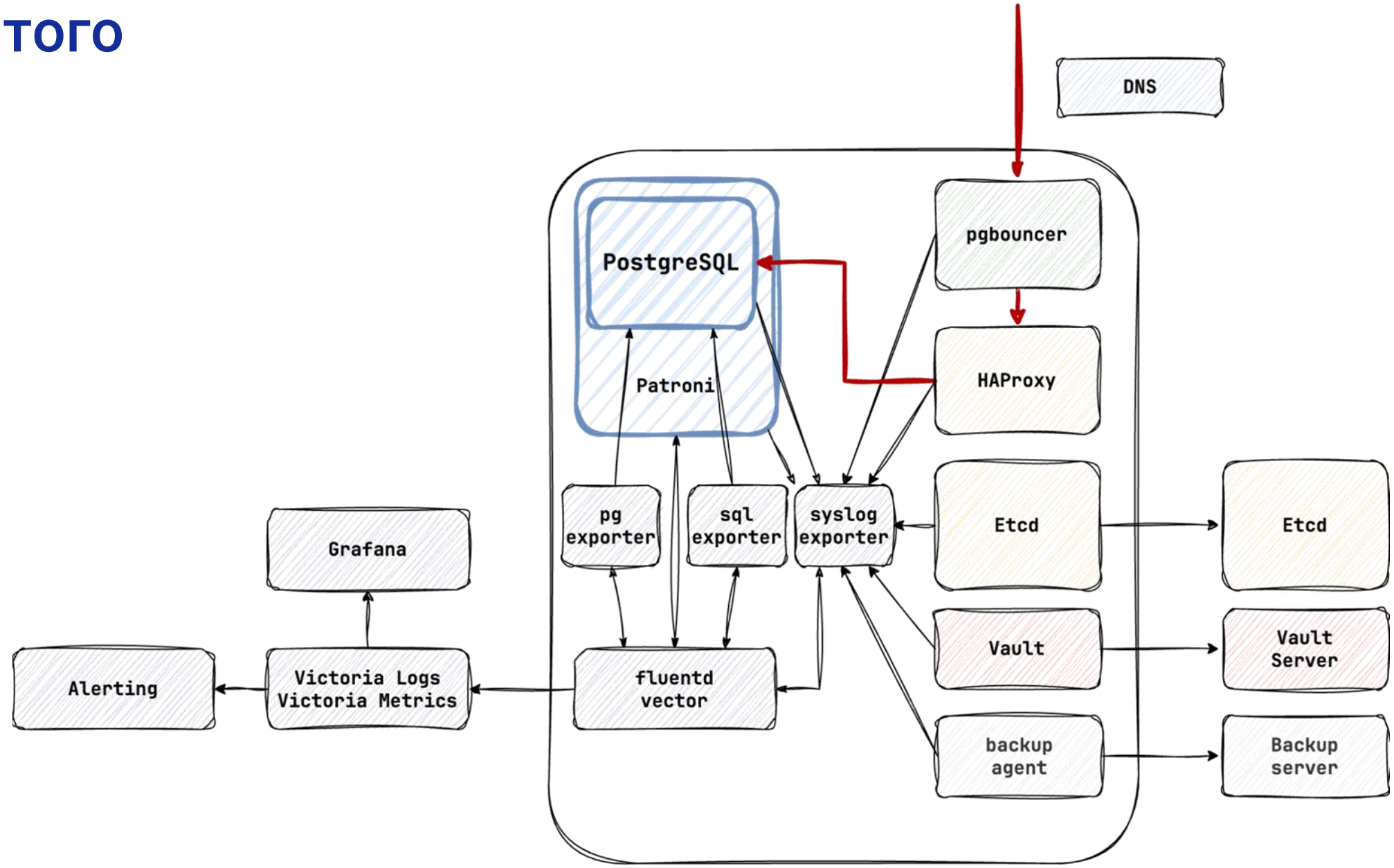
- postgres-exporter
- sql-exporter

Vector/fluentd

- локальный скреппинг
- отсылка набора метрик

Итого

PostgresPro



Косты

PostgresPro

Нужны девопсы

Нужно выбирать самые популярные решения

Нужны ДБА

Избегать самописных вещей

В идеале нужны два в одном (это редкость)

Если можно написать shell-скрипт,
то лучше его

Затраты на поддержание инфры

PostgresPro

Отвечу
на ваши вопросы!