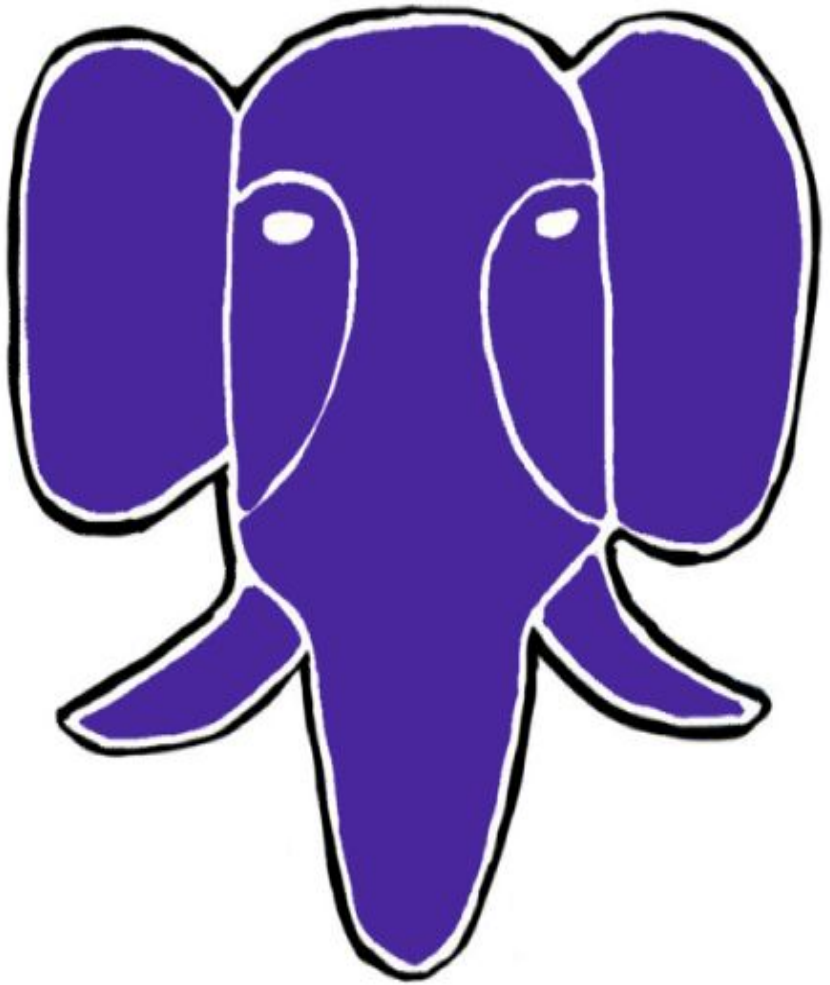
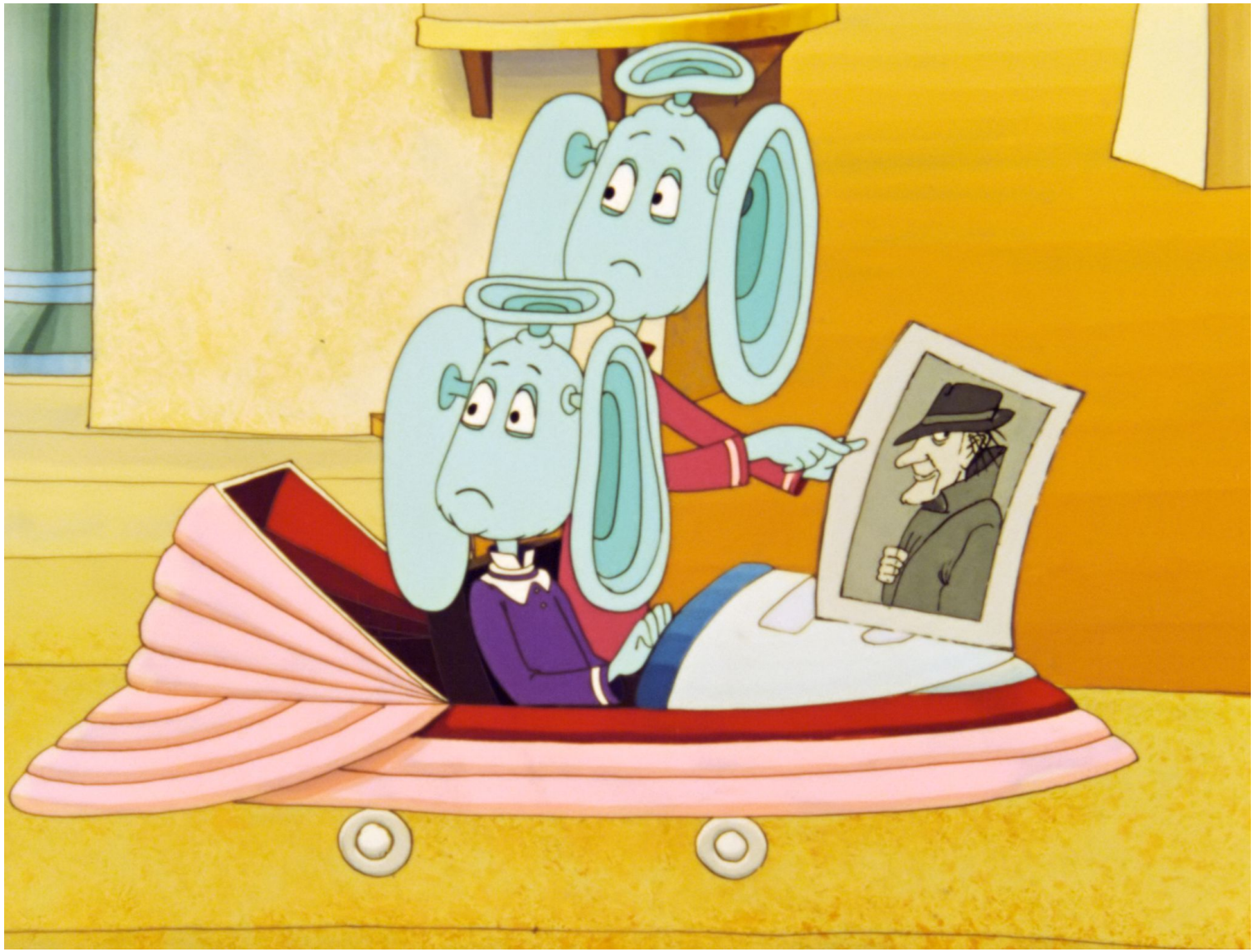
Евгений Аристов

Открытый урок
Patroni in k8s





Аристов Евгений Николаевич

Founder & CEO aristov.tech
25 лет занимаюсь разработкой БД и ПО

Архитектор высоконагруженных баз данных и инфраструктуры

Спроектировал и разработал более ста проектов для финансового сектора, сетевых магазинов, фитнес-центров, отелей.

Сейчас решаю актуальные для бизнеса задачи: аудит и оптимизация БД и инфраструктуры, миграция на PostgreSQL, обучение сотрудников.

Автор более 10 практических курсов по PostgreSQL, MySQL, Mongo и др..

Автор книг по PostgreSQL. Новинка **PostgreSQL 16: лучшие практики оптимизации**

Правила вебинара

Задаем вопрос в чат

Вопросы вижу, отвечу в момент логической паузы

Если есть вопрос голосом - поставьте знак ? в чат

Если остались вопросы, можно написать мне через сайт aristov.tech

Маршрут вебинара

1. Вспоминаем проблему
2. Etcd
3. Patroni
4. K8s
5. Практика
6. Коротко о проекте aristov.tech
7. Розыгрыш книг и скидки 30% на курс
8. Ответы на вопросы

https://aristov.tech

Проблема

Репликация PostgreSQL

master slave

тип

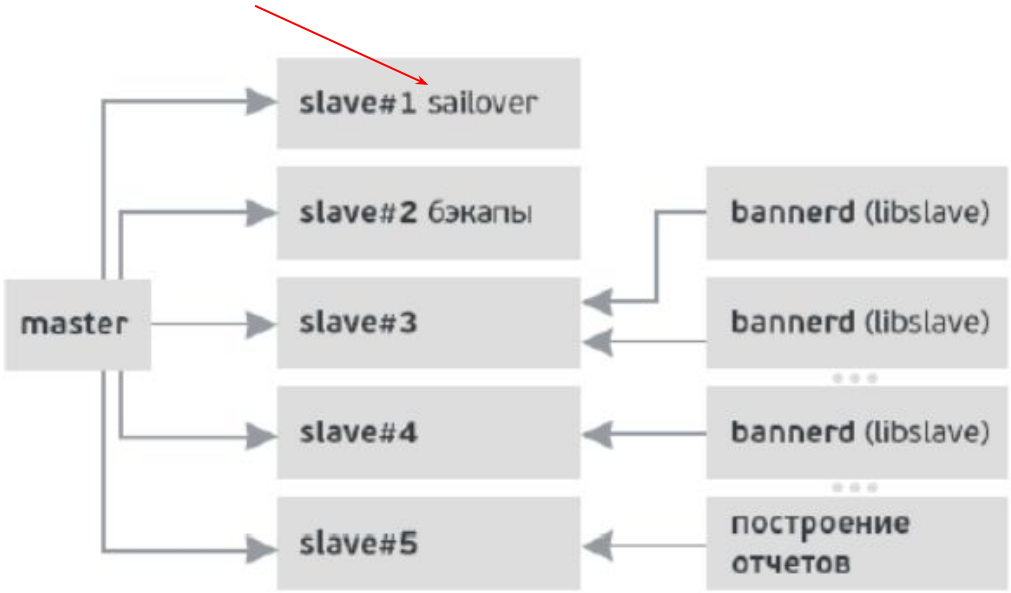
- ❖ логическая
- ❖ физическая

задержка

- ❖ синхронная
- ❖ асинхронная

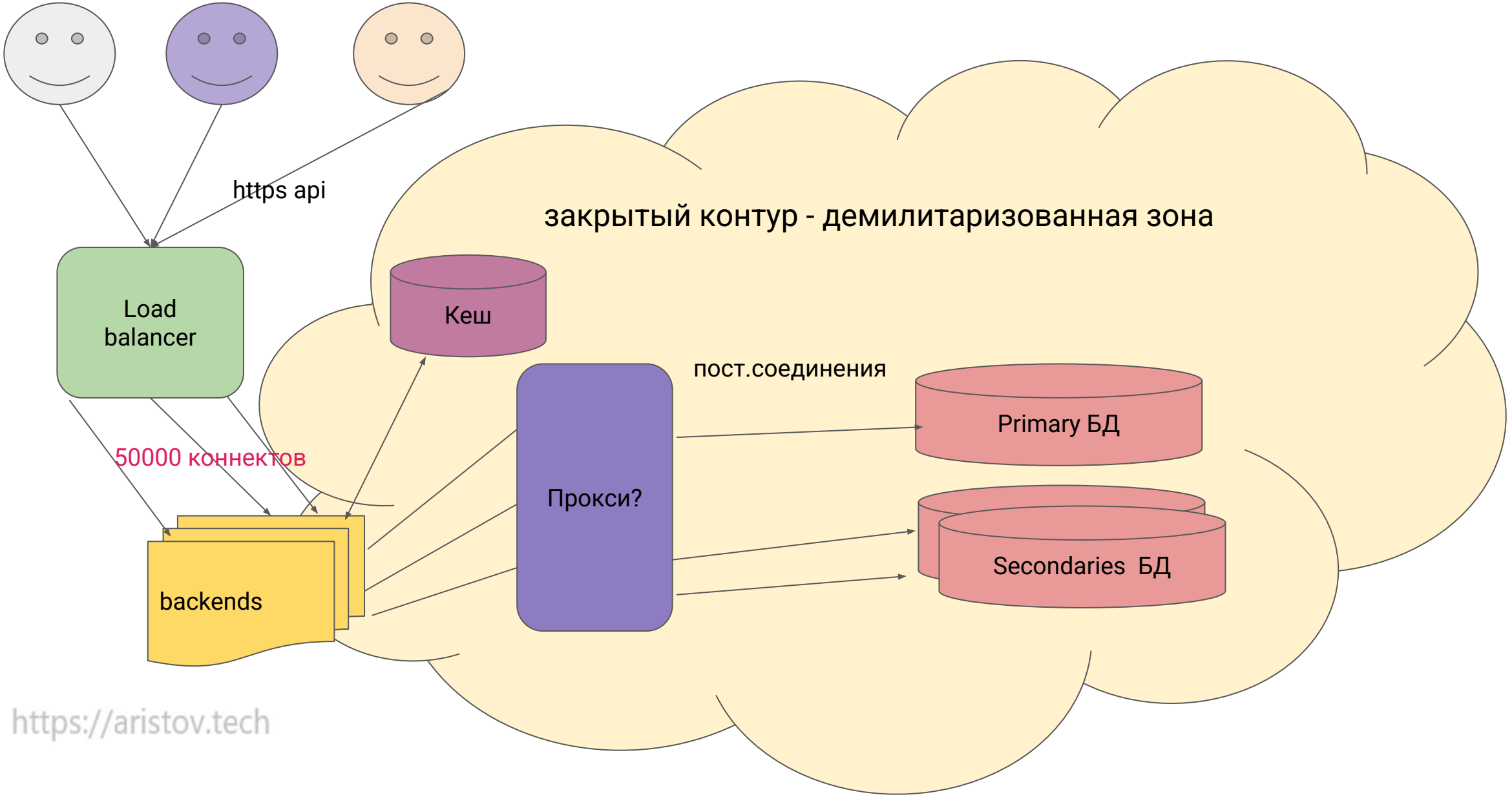
доступность реплики

- ❖ warm
- ❖ hot

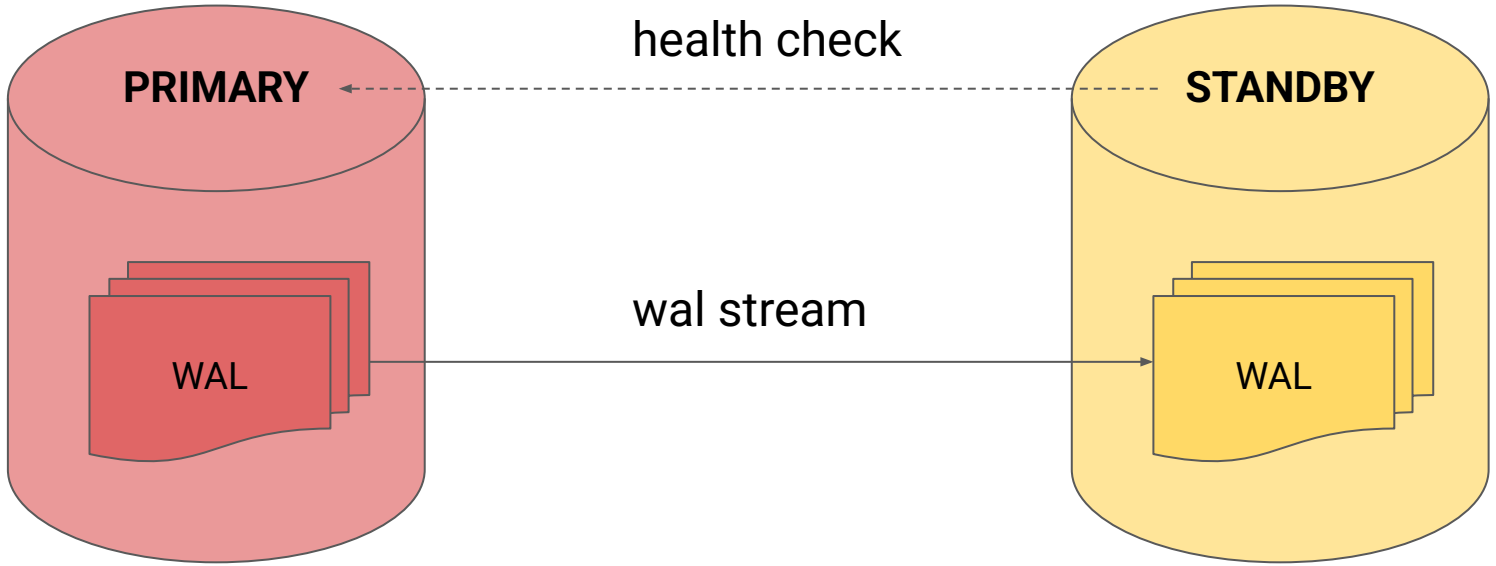


*Структура проекта Mail.Ru Target

Добавим кэш. Какие проблемы остались?



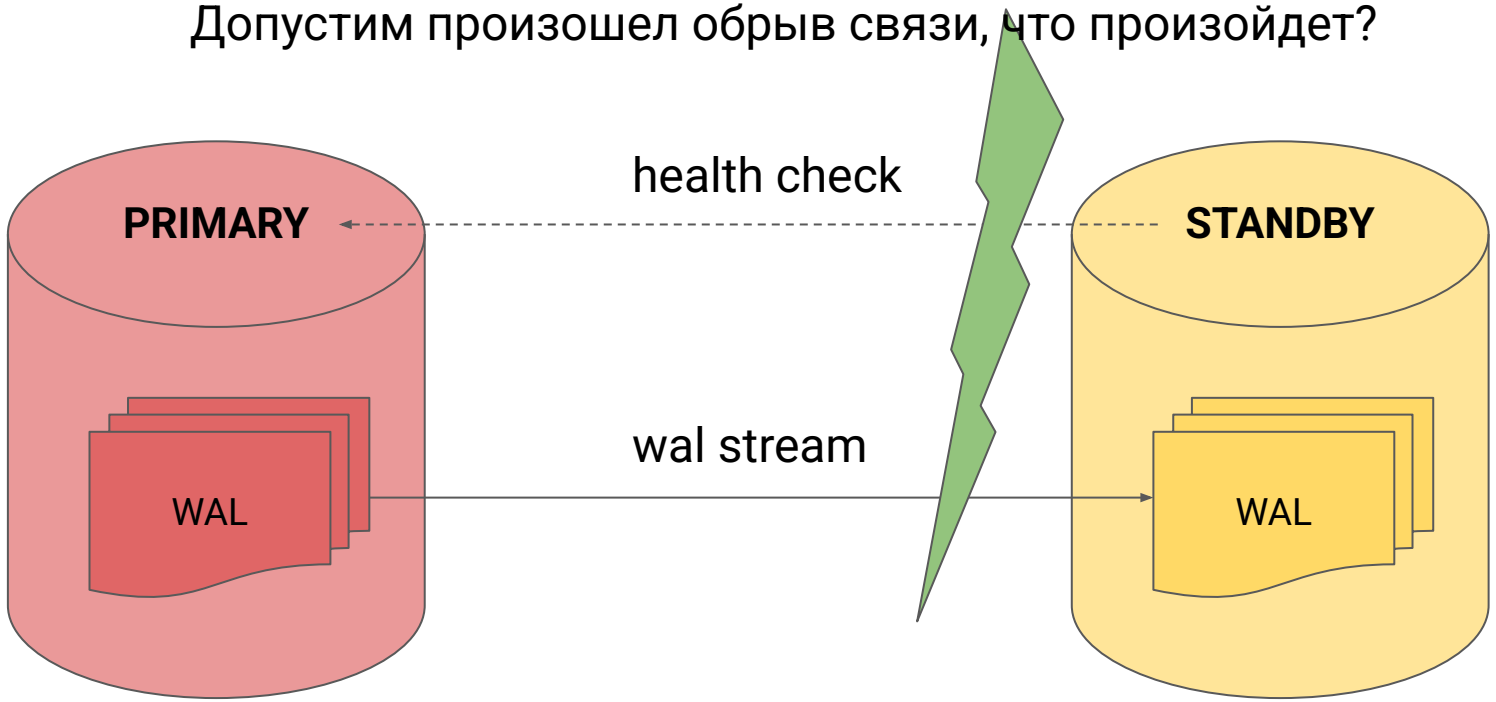
HA из коробки нет. Кластер из 2 нод



https://aristov.tech

Кластер из 2 нод

Допустим произошел обрыв связи, что произойдет?



https://aristov.tech

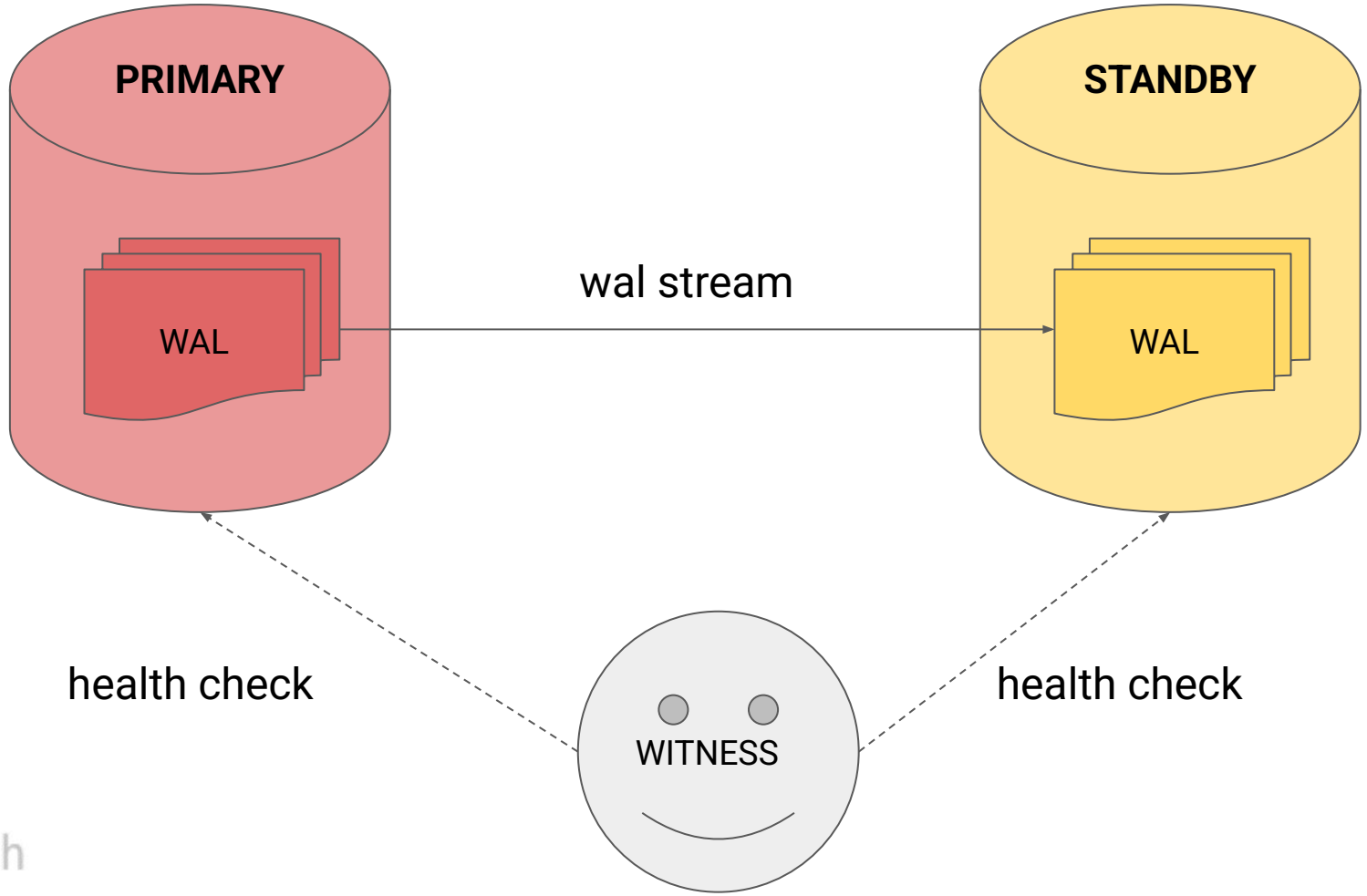
Кластер из 2 нод

Допустим произошел обрыв связи, что произойдет?

Сплитбрейн !!!

Кластер из 2 нод with Witness

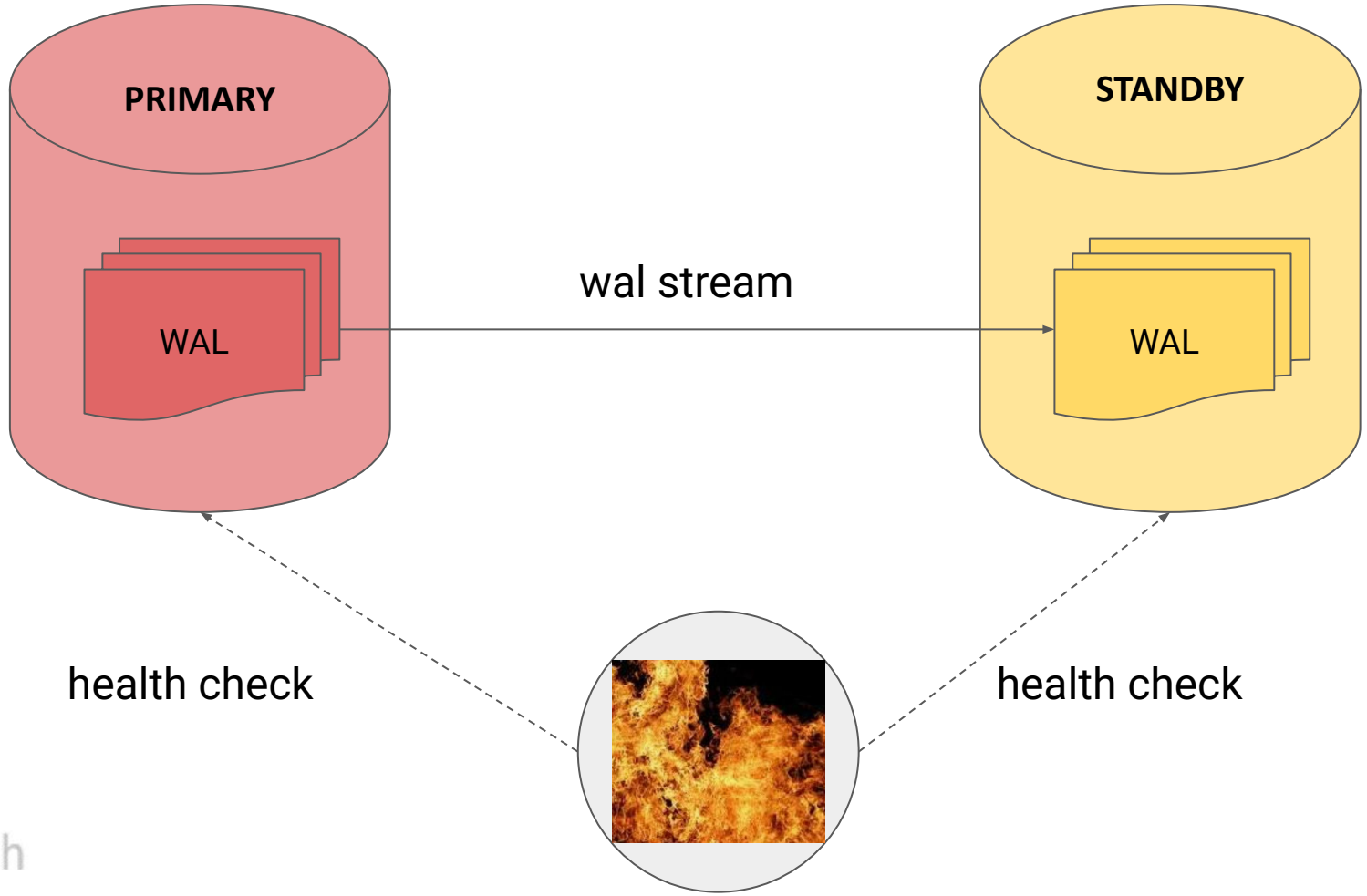
Что может пойти не так??



https://aristov.tech

Кластер из 2 нод with Witness

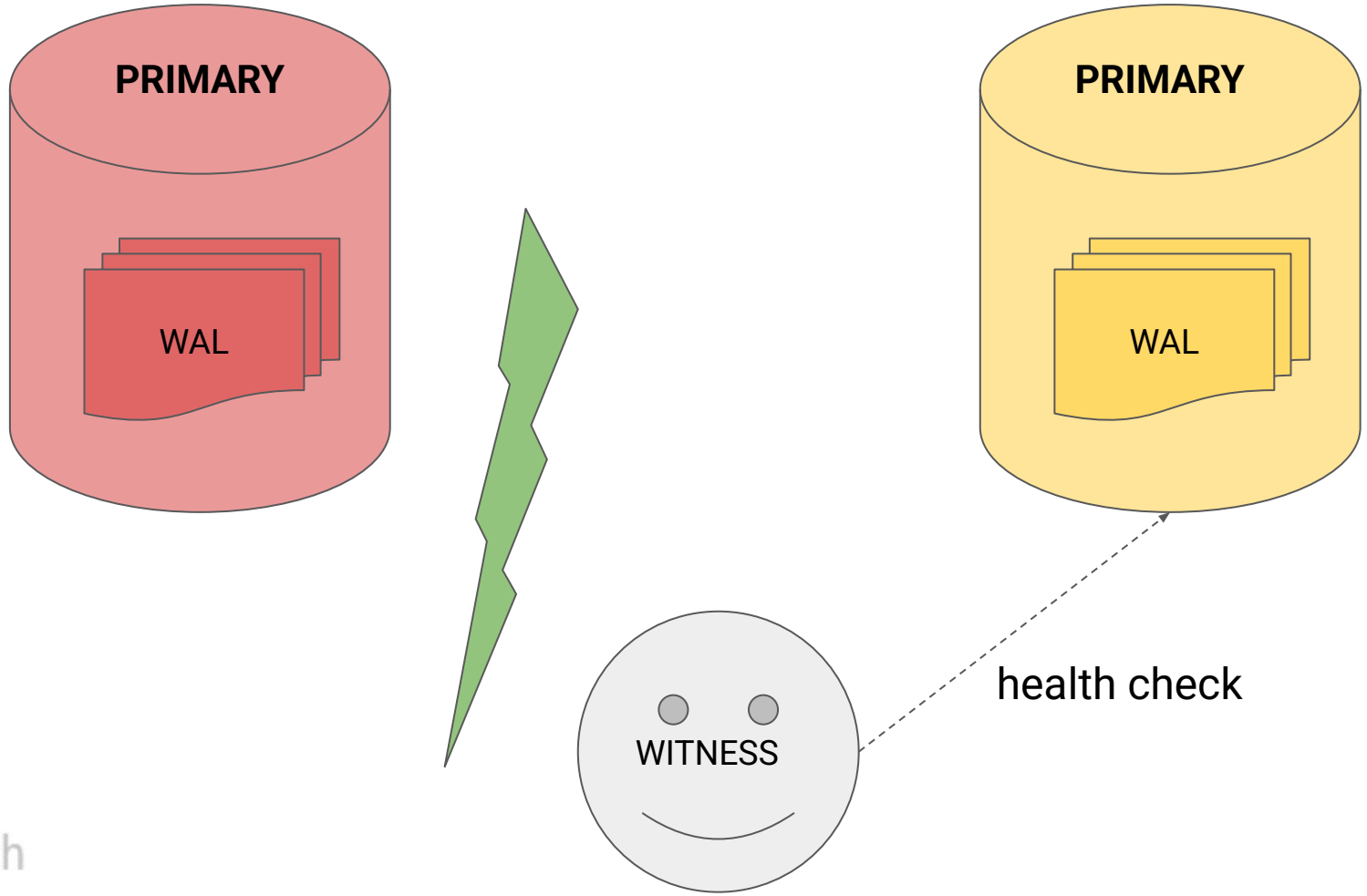
Умер наблюдатель



https://aristov.tech

Кластер из 2 нод with Witness

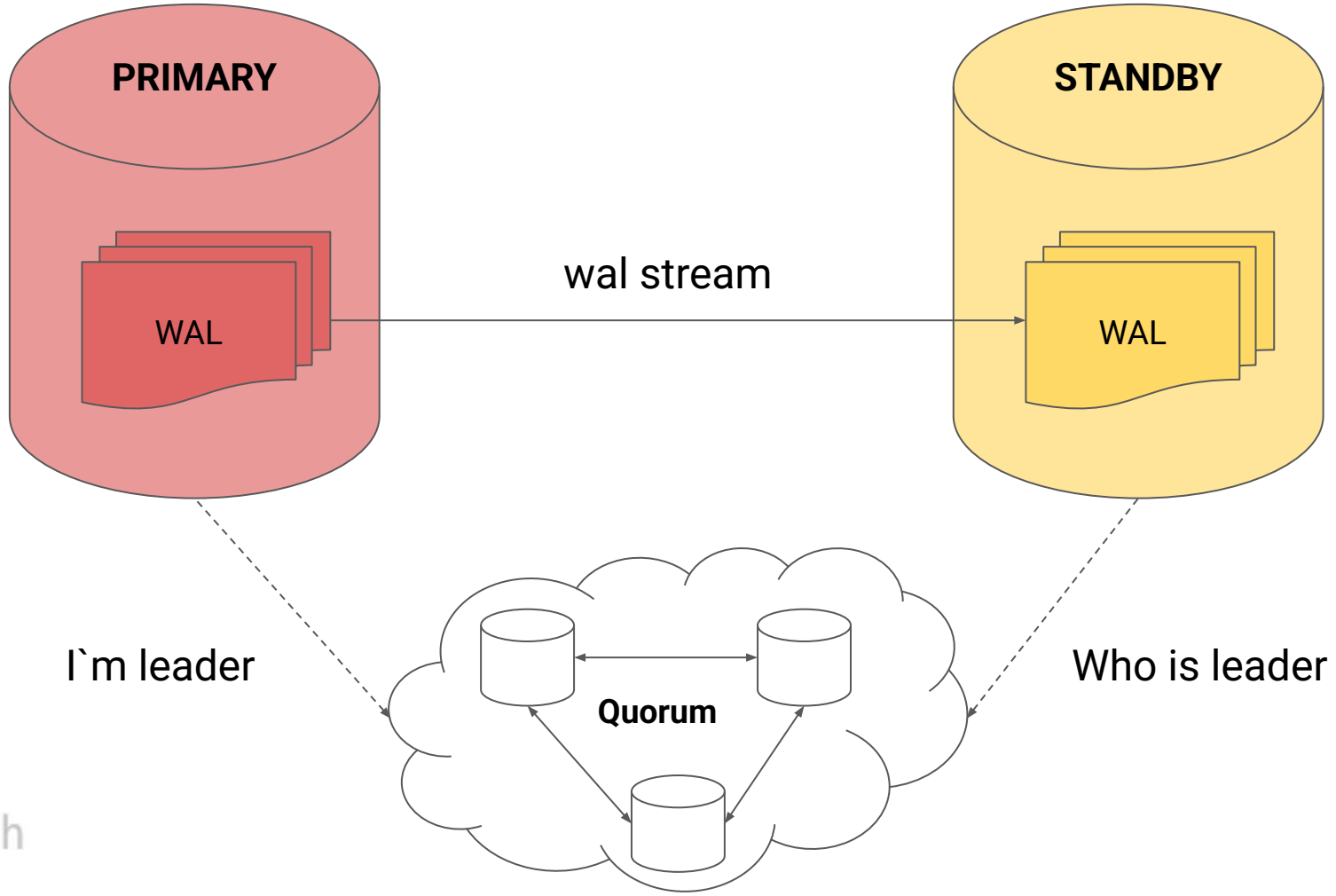
Обрыв соединения с основной нодой. Казалось бы и что? А снова сплитбрейн(



https://aristov.tech

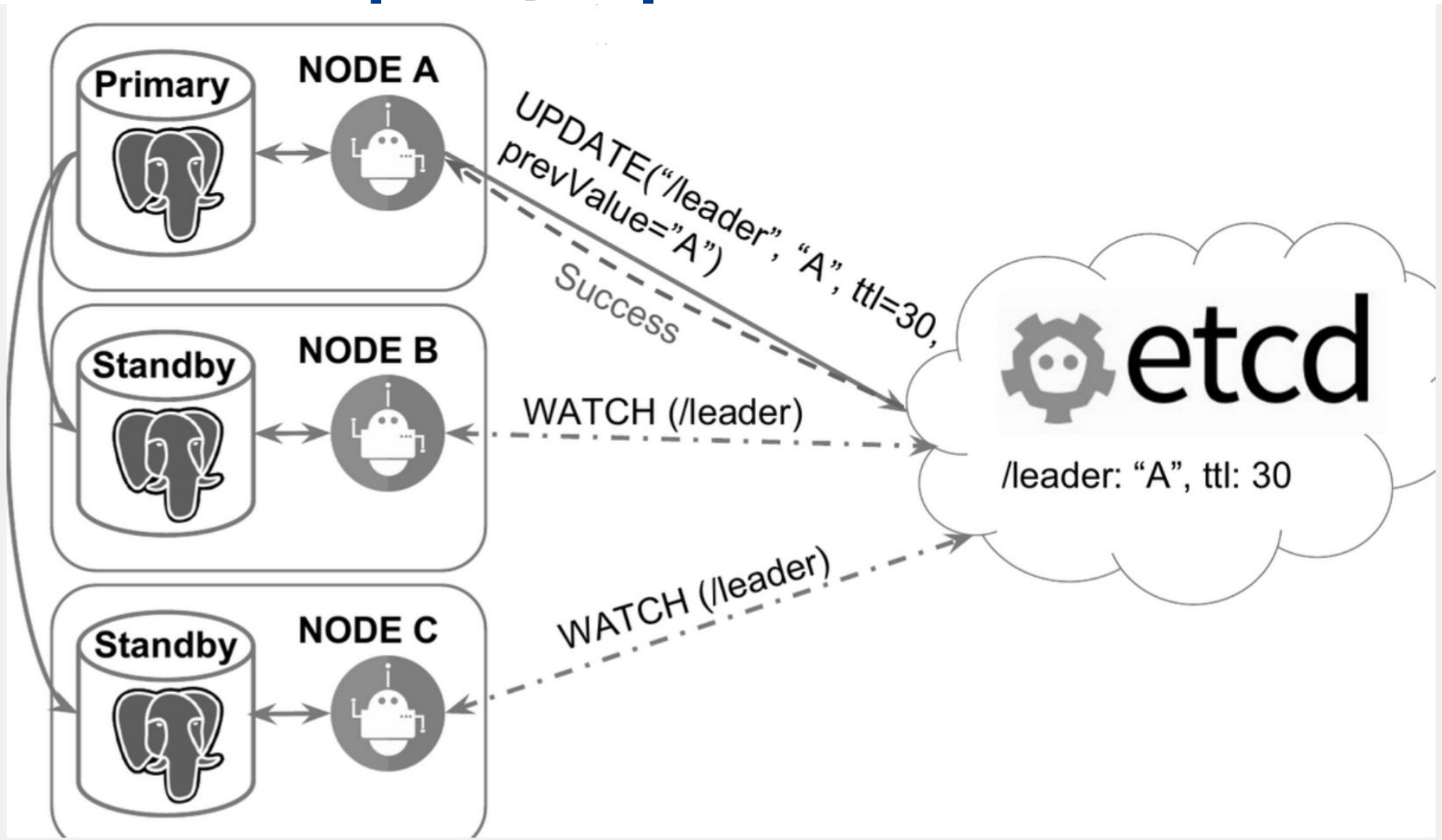
Кластер из 2 нод с ETCD/CONSUL/ZOOKEEPER

Одно из решений (Patroni)



https://aristov.tech

Принцип работы Patroni



https://aristov.tech

Готовые кластера. Большинство Open Source

HA:

- Patroni
- Stolon
- Slony
- ClusterControl
- Kubegres

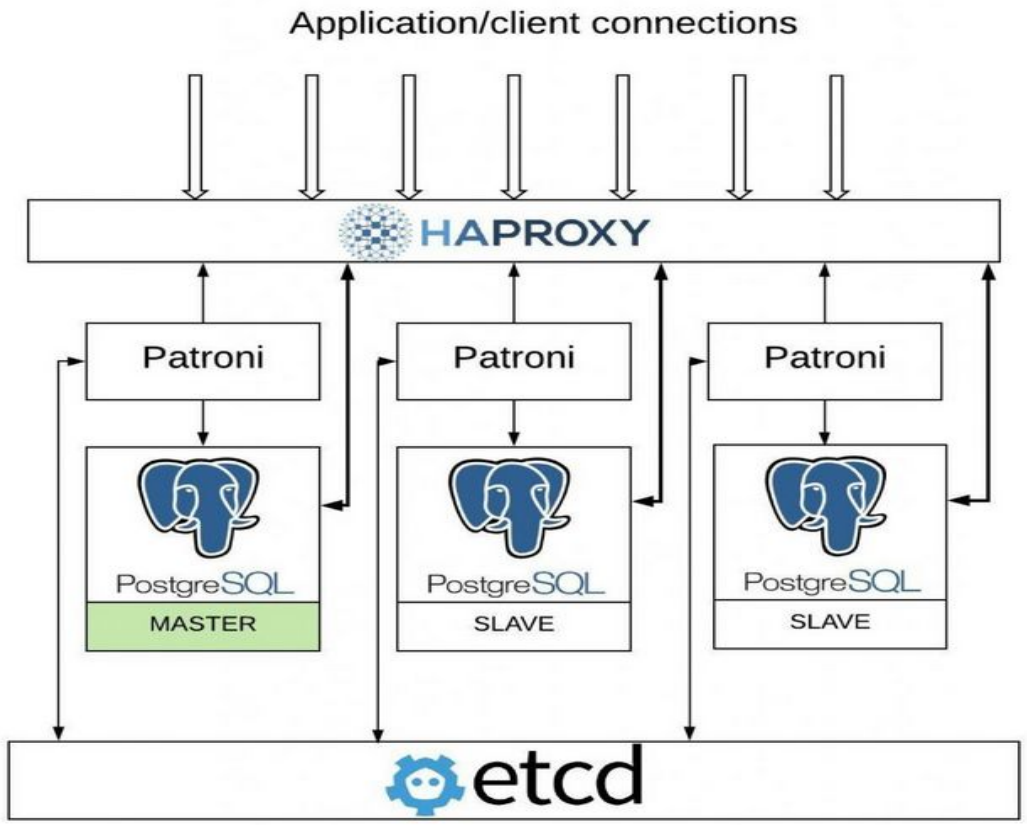
Параллельные кластера:

- Postgres-BDR
- CitusData
- Bucardo
- CockroachDB
- Yogabyte
- Greenplum
- Arenadata

Опять же как развернуть.. on-premise, docker, k8s…

А уж сколько облачных решений…

Patroni



можно еще добавить pgbouncer, keepalived+2 HAPROXY для HA

https://aristov.tech

Patroni

Функции DCS (*distributed* control system)

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

https://aristov.tech

Patroni

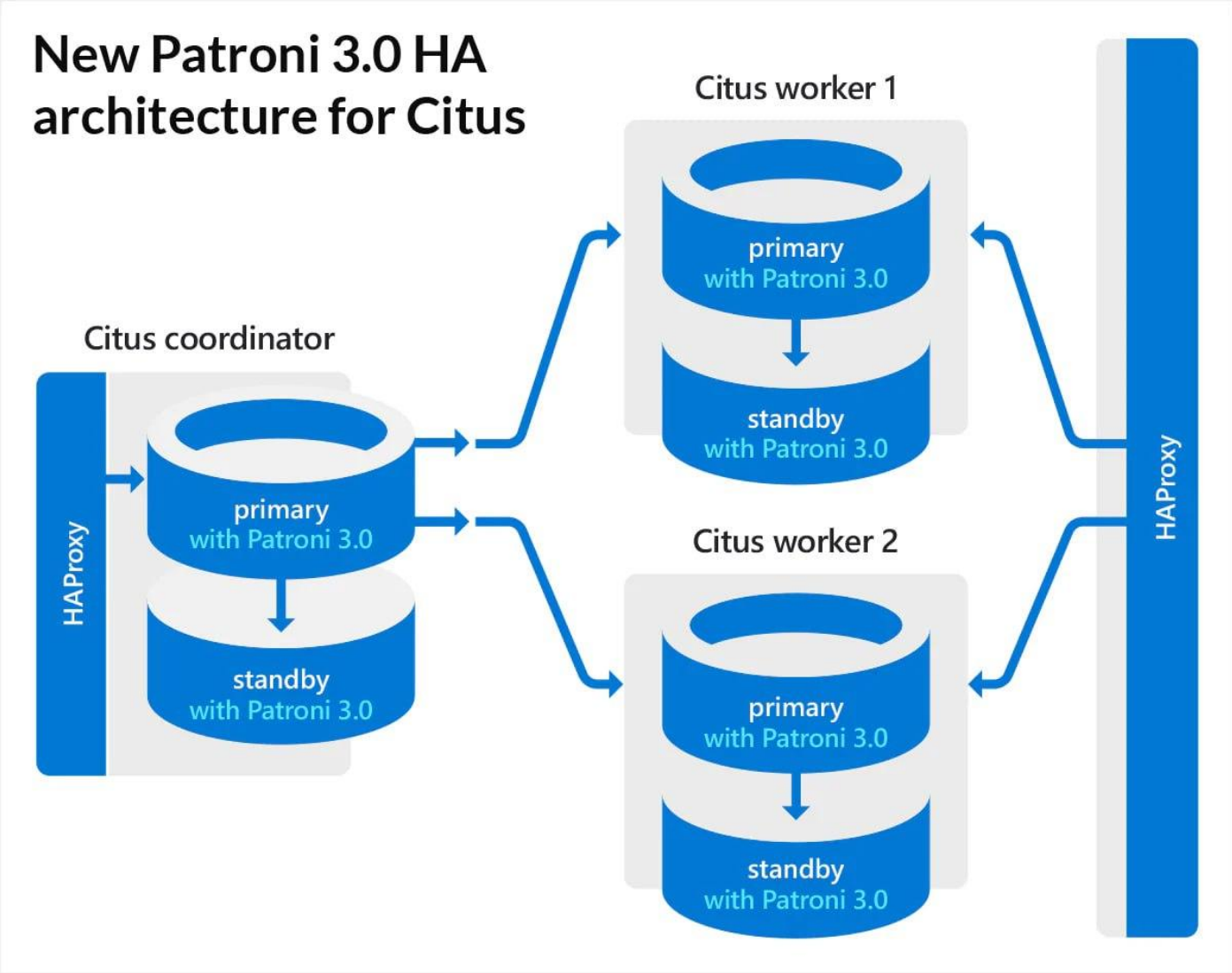
Consul

- Service check
- Есть GUI =)
- Есть свой DNS
- Patroni может анонсировать master/replica
- *ETCD при большой загрузке замечен в высокой нагрузке на дисковую подсистему (обычно когда кладут на те же ноды с Потсгресом)*

с версии 2.0 нативно поддерживает выборы нового мастера по рафт протоколу без использования etcd/consul
https://raft.github.io/

https://aristov.tech

Пример CITUS + HA Patroni



https://aristov.tech

P.S. вообще в 3 Patroni протокол raft derpracated

Patroni

Зачем нужен Patroni

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

https://aristov.tech

DCS на основе ETCD

Etcd

etcd - это распределенное хранилище данных вида "**ключ-значение**"

Особенности:

- хранение небольших объемов метаданных в виде ключей относительно небольшого размера
- полная репликация между нодами и высокая степень доступности
- **все данные пишутся на диск**, in-memory отсутствует
- использует для работы алгоритм консесуса RAFT
- написан на Go, кроссплатформенный, имеет небольшой размер и большое сообщество

Etcd

Преимущества:

- простой API интерфейс http + json
- иерархическая структура хранения данных по аналогии с файловой системой
- возможность отслеживание изменений (watch) и реакция на них (в этом качестве используется в Kubernetes)
- распределенные блокировки
- транзакции
- B-tree индексы для ключей
- полностью ACID

Кластер Etcd

Особенности:

- минимально отказоустойчивый кластер можно собрать из 3 нод
- мультимастер, *но не совсем)*
- допустимое количество вышедших из строя нод можно посмотреть в таблице:

  https://etcd.io/docs/v2/admin_guide/

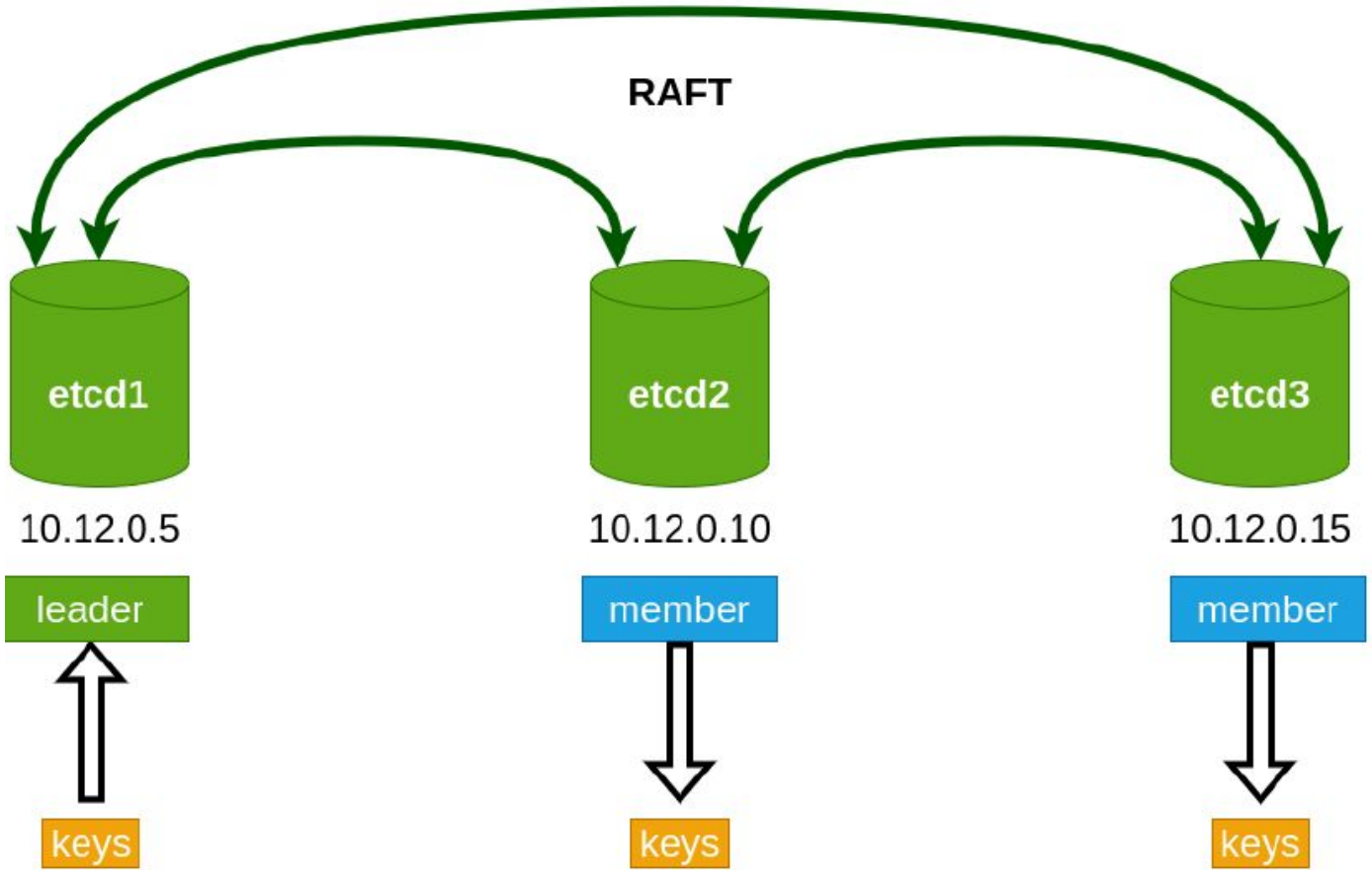
  обратите внимание, что сейчас версия API 3+

- теоретически количество нод не ограничено, но надо помнить о том, что любое изменение данных согласуют все ноды кластера
- задержка записи-чтения (так как используется запись на диск) в свою очередь приводит к нестабильной работе кластера и постоянным переизбраниям мастера
- так как плохо переживает нагрузку на дисковую подсистему - не рекомендуется размещать ноды кластера на используемых уже в продакшне ВМ
- *рекомендация размера кластера - 5 нод*

  https://www.crunchydata.com/blog/patroni-etcd-in-high-availability-environments

Кластер Etcd

Схема тестового стенда:



https://aristov.tech

Patroni

Patroni

Состояние кластера

- **patronictl** - утилита для управления кластером
- patronictl -c /etc/patroni.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 |
+----------+--------+-------------+--------+---------+----+-----------+
```

Patroni

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

Patroni

Редактирование конфигурации

patronictl -c /etc/patroni.yml edit-config

Ручной Switchover:

patronictl -c /etc/patroni.yml switchover

Перезагрузка

patronictl -c /etc/patroni.yml restart postgres pg02

Применение новых параметров требующих обязательной перезагрузки

Реинициализация

patronictl -c /etc/patroni.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

https://aristov.tech

Patroni

[Истории аварий с Patroni, или Как уронить PostgreSQL-кластер]

https://aristov.tech

Современный Patroni

Patroni, REST API 4.0.3

Можно получить доступ к текущим параметрам кластера:

https://patroni.readthedocs.io/en/latest/rest_api.html

GET /read-only: как и вышеупомянутая конечная точка, но включает также основную.

GET /synchronous или **GET /sync**: возвращает HTTP-код состояния 200, только если узел Patroni работает как синхронный резервный.

GET /read-only-sync: аналогично вышеуказанной конечной точке, но также включает основную.

GET /quorum: возвращает HTTP-код состояния 200, только если этот узел Patroni указан в качестве узла кворума в synchronous_standby_names на основном узле.

GET /read-only-quorum: аналогично вышеуказанной конечной точке, но также включает первичный узел.

GET /asynchronous или **GET /async**: возвращает HTTP-код состояния 200, только если узел Patroni работает как асинхронный резервный.

Patroni, REST API 4.0.3. GET /patroni


```
$ curl -s http://localhost:8008/patroni | jq .
{
  "state": "running",
  "postmaster_start_time": "2024-08-28 19:39:26.352526+00:00",
  "role": "primary",
  "server_version": 160004,
  "xlog": {
    "location": 67395656
  },
  "timeline": 1,
  "replication": [
    {
      "usename": "replicator",
      "application_name": "patroni2",
      "client_addr": "10.89.0.6",
      "state": "streaming",
      "sync_state": "async",
      "sync_priority": 0
    },
    {
      "usename": "replicator",
      "application_name": "patroni3",
      "client_addr": "10.89.0.2",
      "state": "streaming",
      "sync_state": "async",
      "sync_priority": 0
    }
  ],
  "dcs_last_seen": 1692356718,
  "tags": {
    "clonefrom": true
  },
  "database_system_identifier": "7268616322854375442",
  "patroni": {
    "version": "4.0.0",
    "scope": "demo",
    "name": "patroni1"
  }
}
```

https://aristov.tech

Patroni, REST API 4.0.3

GET /health: возвращает HTTP-код состояния 200, только если PostgreSQL работает.

GET /liveness: возвращает HTTP-код состояния 200, если цикл сердцебиения Patroni запущен правильно, и 503, если последний запуск был более ttl секунд назад на основной или 2*ttl на реплике. Может использоваться для livenessProbe.

GET /readiness: возвращает HTTP-код состояния 200, когда узел Patroni работает как лидер или когда PostgreSQL запущен и работает. Конечная точка может быть использована для readinessProbe, когда невозможно использовать конечные точки Kubernetes для выборов лидера (OpenShift).

```
readinessProbe:
  httpGet:
    scheme: HTTP
    path: /readiness
    port: 8008
  initialDelaySeconds: 3
  periodSeconds: 10
  timeoutSeconds: 5
  successThreshold: 1
  failureThreshold: 3
```

```
livenessProbe:
  httpGet:
    scheme: HTTP
    path: /liveness
    port: 8008
  initialDelaySeconds: 3
  periodSeconds: 10
  timeoutSeconds: 5
  successThreshold: 1
  failureThreshold: 3
```

Patroni, REST API 4.0.3. GET /metrics

Маленький кусок JSON - пример передаваемых параметров в системы мониторинга

```
$ curl http://localhost:8008/metrics

# HELP patroni_version Patroni semver without periods. \
# TYPE patroni_version gauge
patroni_version{scope="batman",name="patroni1"} 040000
# HELP patroni_postgres_running Value is 1 if Postgres is running, 0 otherwise.
# TYPE patroni_postgres_running gauge
patroni_postgres_running{scope="batman",name="patroni1"} 1
# HELP patroni_postmaster_start_time Epoch seconds since Postgres started.
# TYPE patroni_postmaster_start_time gauge
patroni_postmaster_start_time{scope="batman",name="patroni1"} 1724873966.352526
# HELP patroni_primary Value is 1 if this node is the leader, 0 otherwise.
# TYPE patroni_primary gauge
patroni_primary{scope="batman",name="patroni1"} 1
# HELP patroni_xlog_location Current location of the Postgres transaction log, 0 if this node is not th
# TYPE patroni_xlog_location counter
patroni_xlog_location{scope="batman",name="patroni1"} 22320573386952
# HELP patroni_standby_leader Value is 1 if this node is the standby_leader, 0 otherwise.
# TYPE patroni_standby_leader gauge
patroni_standby_leader{scope="batman",name="patroni1"} 0
# HELP patroni_replica Value is 1 if this node is a replica, 0 otherwise.
# TYPE patroni_replica gauge
patroni_replica{scope="batman",name="patroni1"} 0
# HELP patroni_sync_standby Value is 1 if this node is a sync standby replica, 0 otherwise.
# TYPE patroni_sync_standby gauge
patroni_sync_standby{scope="batman",name="patroni1"} 0
# HELP patroni_quorum_standby Value is 1 if this node is a quorum standby replica, 0 otherwise.
# TYPE patroni_quorum_standby gauge
patroni_quorum_standby{scope="batman",name="patroni1"} 0
# HELP patroni_xlog_received_location Current location of the received Postgres transaction log, 0 if t
# TYPE patroni_xlog_received_location counter
patroni_xlog_received_location{scope="batman",name="patroni1"} 0
# HELP patroni_xlog_replayed_location Current location of the replayed Postgres transaction log, 0 if t
```

Patroni, REST API 4.0.3. GET /history

История смены таймлайнов (смены основной ноды)

```
$ curl -s http://localhost:8008/history | jq .
[
  [
    1,
    25623960,
    "no recovery target specified",
    "2019-09-23T16:57:57+02:00"
  ],
  [
    2,
    25624344,
    "no recovery target specified",
    "2019-09-24T09:22:33+02:00"
  ],
  [
    3,
    25624752,
    "no recovery target specified",
    "2019-09-24T09:26:15+02:00"
  ],
  [
    4,
    50331856,
    "no recovery target specified",
    "2019-09-24T09:35:52+02:00"
  ]
```

https://aristov.tech

Patroni, REST API 4.0.3. GET /config

Текущая динамическая конфигурация

```
$ curl -s http://localhost:8008/config | jq .
{
  "ttl": 30,
  "loop_wait": 10,
  "retry_timeout": 10,
  "maximum_lag_on_failover": 1048576,
  "postgresql": {
    "use_slots": true,
    "use_pg_rewind": true,
    "parameters": {
      "hot_standby": "on",
      "wal_level": "hot_standby",
      "max_wal_senders": 5,
      "max_replication_slots": 5,
      "max_connections": "100"
    }
  }
}
```

https://aristov.tech

Patroni, REST API 4.0.3. PATCH /config

Изменить онлайн параметр

```
$ curl -s -XPATCH -d \
       '{"loop_wait":5,"ttl":20,"postgresql":{"parameters":{"max_connections":"101"}}}' \
       http://localhost:8008/config | jq .
{
  "ttl": 20,
  "loop_wait": 5,
  "maximum_lag_on_failover": 1048576,
  "retry_timeout": 10,
  "postgresql": {
    "use_slots": true,
    "use_pg_rewind": true,
    "parameters": {
      "hot_standby": "on",
      "wal_level": "hot_standby",
      "max_wal_senders": 5,
      "max_replication_slots": 5,
      "max_connections": "101"
    }
  }
}
```

https://aristov.tech

Patroni, REST API 4.0.3. Switchover

/switchover

/failover

DELETE /switchover - можно удалить планируемое ручное переключение с мастера

```
$ curl -s http://localhost:8008/switchover -XPOST -d '{"leader":"postgresql1"}'
Successfully switched over to "postgresql2"
```

```
$ curl -s http://localhost:8008/switchover -XPOST -d \
        '{"leader":"postgresql1","candidate":"postgresql2"}'
Successfully switched over to "postgresql2"
```

```
$ curl -s http://localhost:8008/switchover -XPOST -d \
        '{"leader":"postgresql0","scheduled_at":"2019-09-24T12:00+00"}'
Switchover scheduled
```

https://aristov.tech

Patroni, REST API 4.0.3. Доп возможности

POST /reload

POST /reinitialize - переинициализация через pg_basebackup или стороннюю утилиту, указанную replica creation method

Patroni. Безопасность REST API

Ранее доступ был без шифрования и доступ мог получить кто угодно.

Можно зашифровать трафик и требовать сертификаты для доступа и управления patroni

https://www.percona.com/blog/securing-patroni-rest-api-end-points-part-1/

```
1 …
2 restapi:
3   listen: 0.0.0.0:8008
4   connect_address: pg1:8008
5   certfile: /etc/patroni/pg1.crt
6   keyfile: /etc/patroni/pg1.key
7   cafile: /etc/patroni/ca.crt
8   verify_client: required
9 …
```

https://aristov.tech

PGBouncer

https://aristov.tech

PGbouncer

PgBouncer

PgBouncer - пул соединений:

- легковесный - 2 Кб на соединение
- можно выбрать тип соединения: на сессию, транзакцию или каждую операцию
- онлайн-реконфигурация без сброса подключений

Сравнение с PGpool2

https://scalegrid.io/blog/postgresql-connection-pooling-part-4-pgbouncer-vs-pgpool/

Конфигурация

http://www.pgbouncer.org/usage.html

https://aristov.tech

HAproxy

https://aristov.tech

Балансируем нагрузку

Неплохо бы еще добавить HAPROXY для балансинга нагрузки

https://www.percona.com/blog/2018/10/02/scaling-postgresql-using-connection-poolers-and-load-balancers-for-an-enterprise-grade-environment/

Еще варианты балансеров:

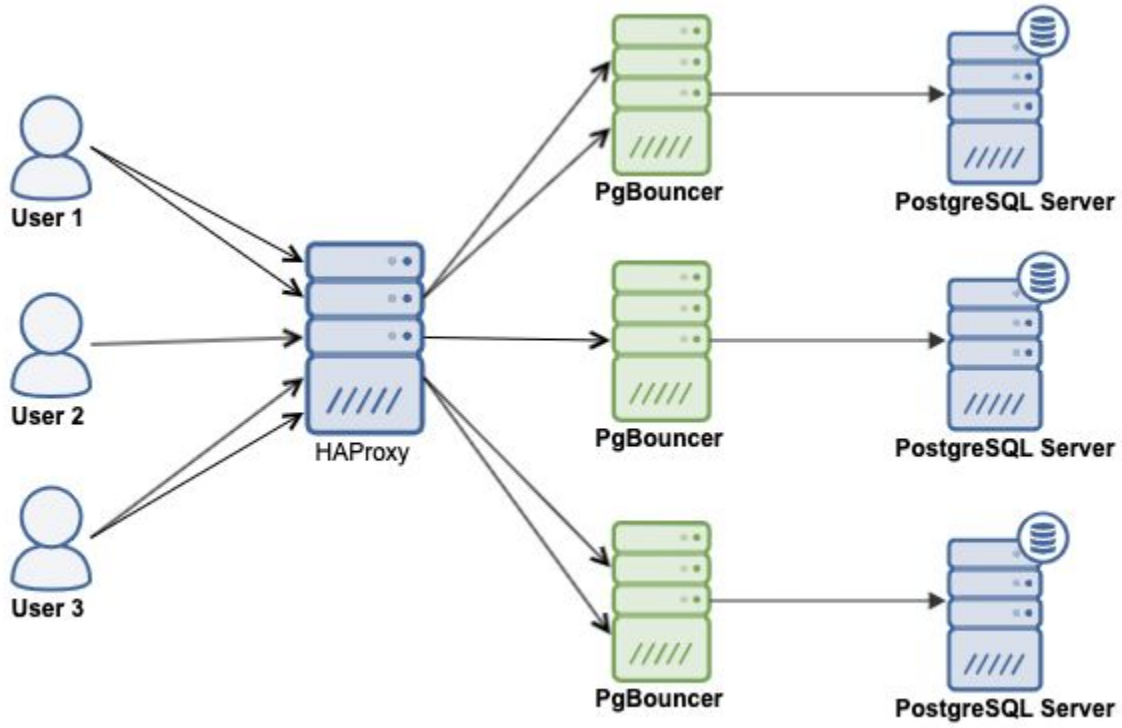
- используем или встроенный GLB в облако или варианты:
- NGINX https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/
- *Множество хостов в строке подключения*
  - *jdbc:postgresql://node1,node2,node3/postgres?targetServerType=master*
  - *postgresql://host1:port2,host2:port2/?target_session_attrs=read-write*

4 классических варианта со своими + и -:

Итого варианты архитектуры

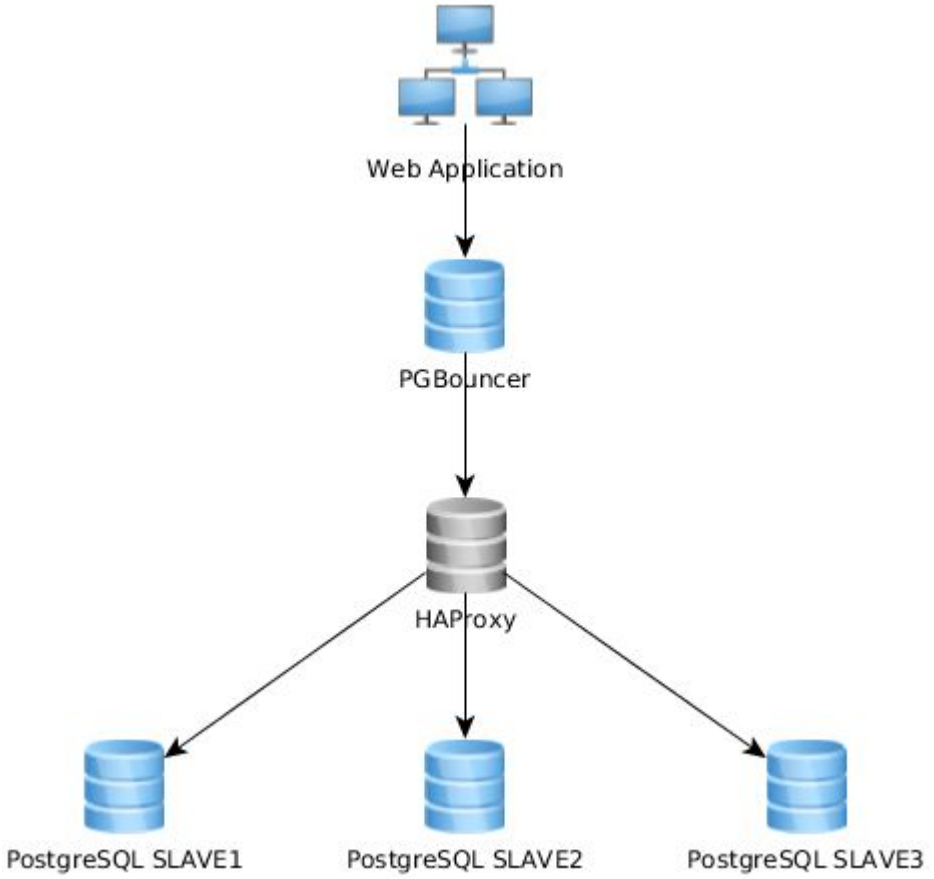
https://aristov.tech

Балансируем нагрузку



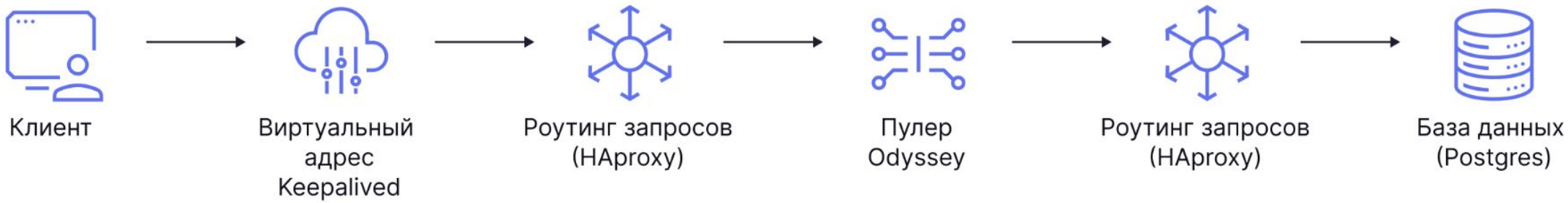
https://aristov.tech

Балансируем нагрузку



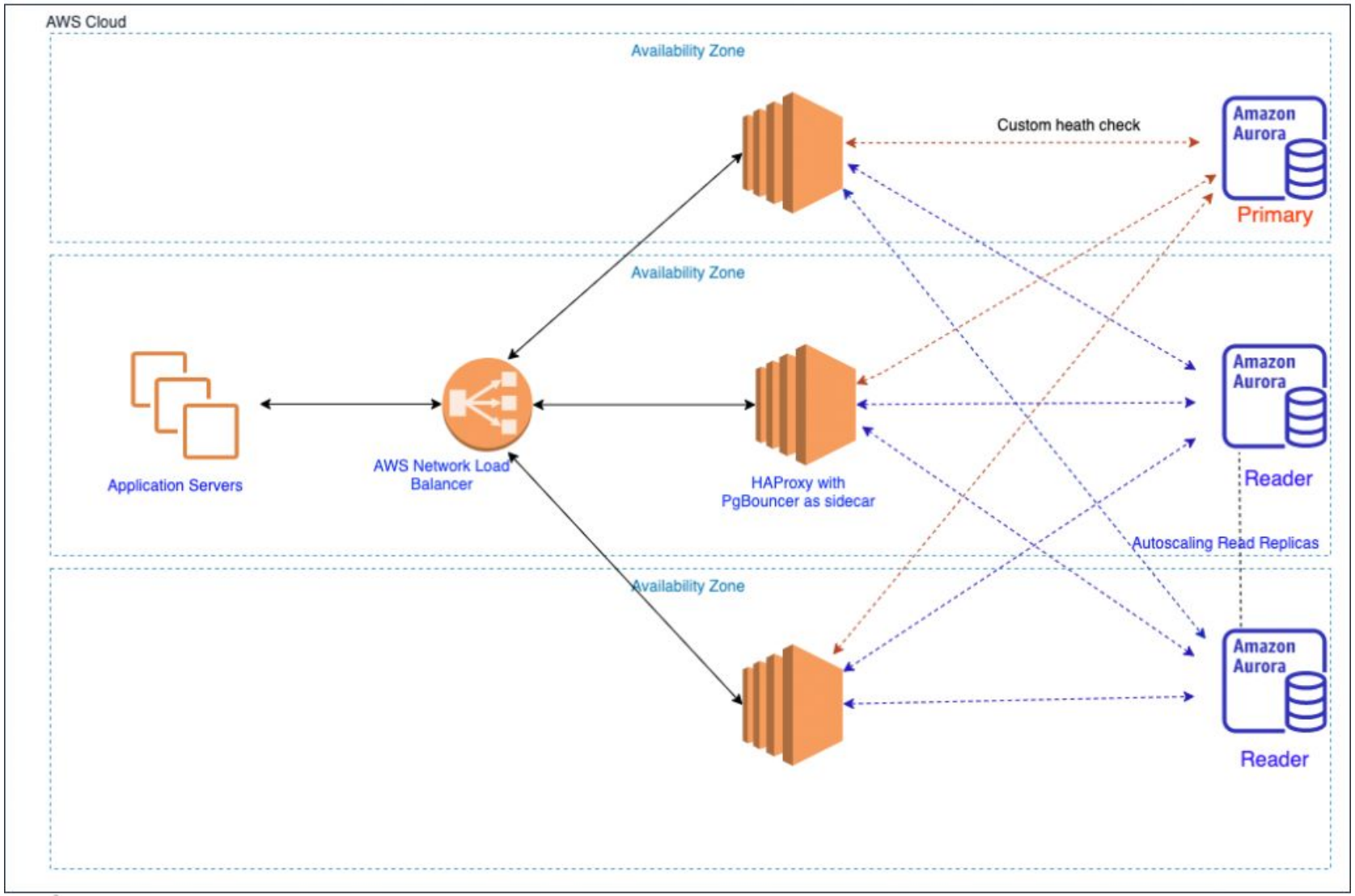
https://aristov.tech

Балансируем нагрузку. Решение Orion Soft



https://wiki.orionsoft.ru/proximadb/latest/admin-guide/
в новой версии выпилили лишний ХАпрокси, в совсем новой откажутся от него)
рассказать почему

Балансируем нагрузку

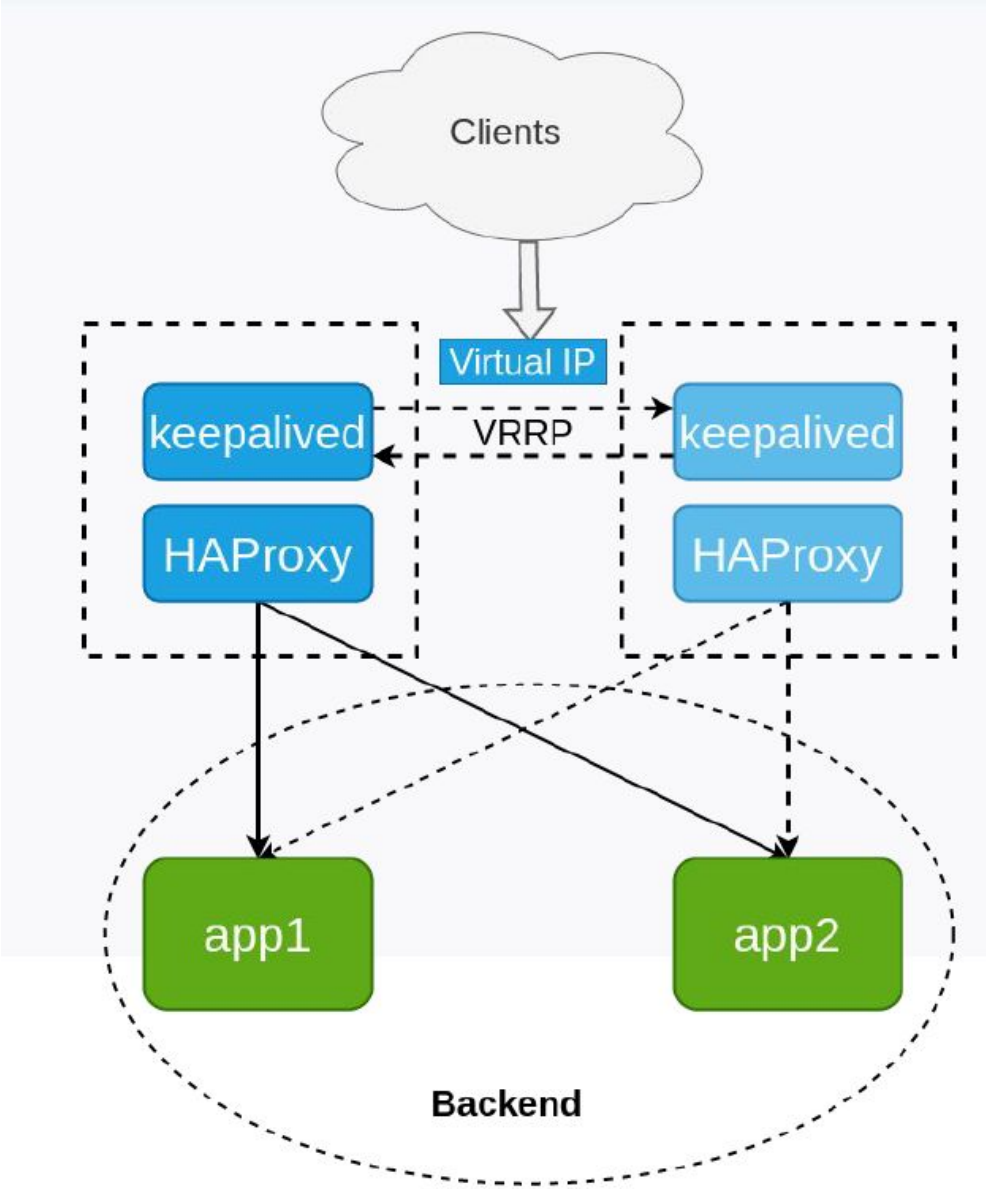


https://aristov.tech

keepalived

https://aristov.tech

keepalived



https://aristov.tech

keepalived

https://dasunhegoda.com/how-to-setup-haproxy-with-keepalived/833/

Keepalived is a routing software written in C. The main goal of this project is to provide simple and robust facilities for loadbalancing and high-availability to Linux system and Linux based infrastructures. Loadbalancing framework relies on well-known and widely used Linux Virtual Server (IPVS) kernel module providing Layer4 loadbalancing

https://keepalived.readthedocs.io/en/latest/software_design.html

В гугле печаль(в ЯО работает

https://aristov.tech

Облачные и кубер возможности

Облачные варианты

DBaaS у любого провайдера.

Плюсы:

- кластер за пару кликов в GUI
- бэкапы, репликация за 1 нажатие

Минусы:

- повышенная стоимость - только платные лицензии
- нет контроля над инстансом
- минимум настроек
- дебаг запросов превращается в многоуровневый квест
- обычно HA реплику не получится использовать как read реплику
- ооочень огромная сложность построения мултиклауд конфигураций и даже реплики у другого провайдера
- время развертывания инстанса значительно превышает вариант on premise

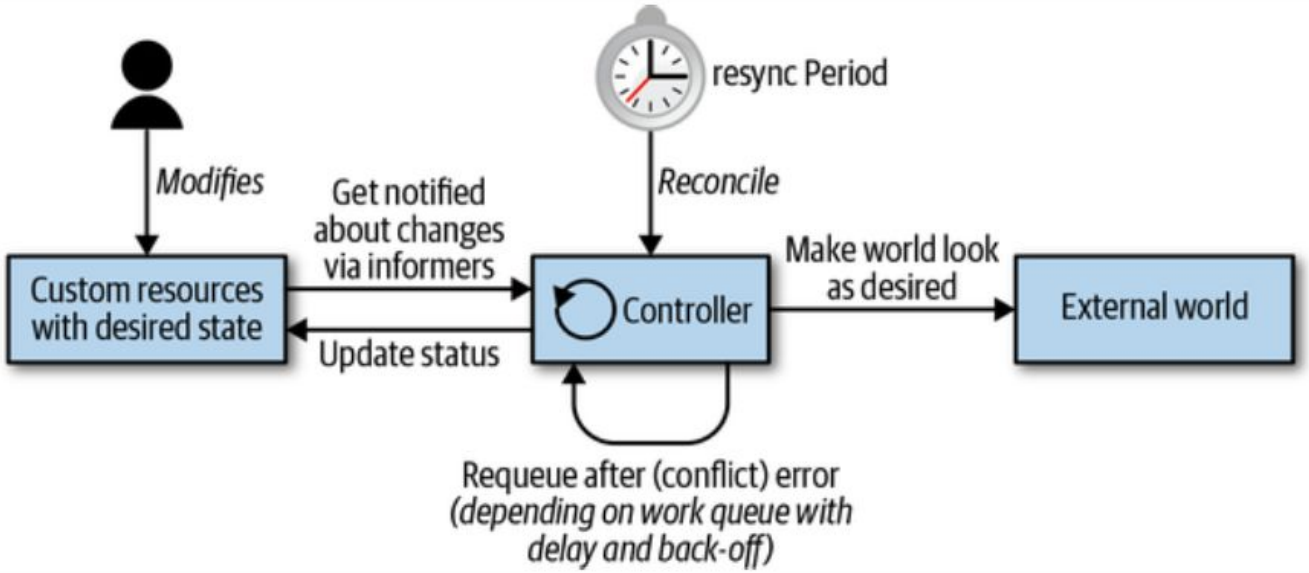
https://aristov.tech

Kubernetes

K8s хранит декларативное описание желаемого состояния в виде набора ресурсов разных типов, а набор контроллеров (и операторов):

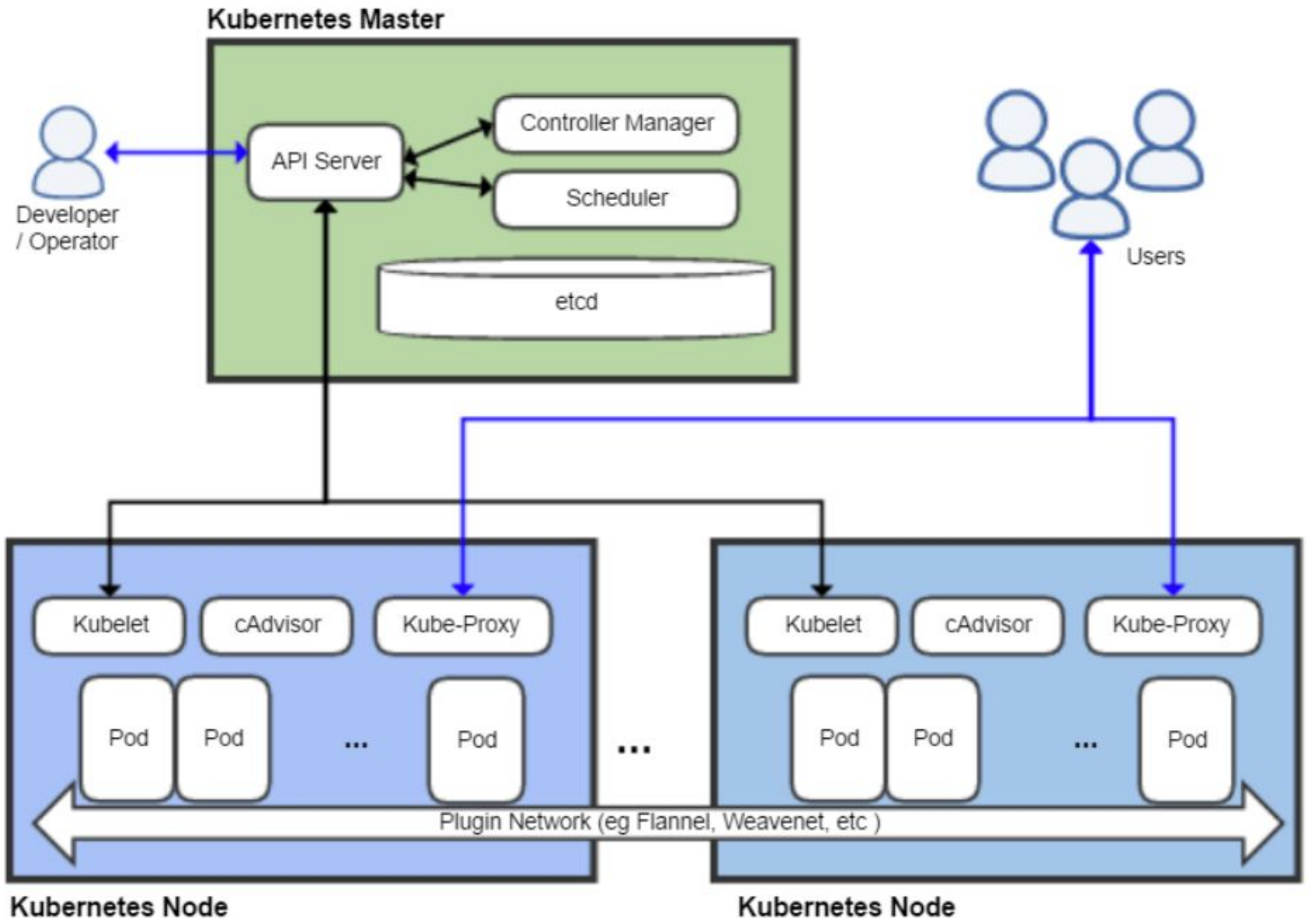
1. Обнаруживает несоответствия и делает все возможное, чтобы их устранить
2. Слушает изменения состояния и обновляет статус объектов

K8s – это конструктор, который позволяет хранить и работать с разными типами ресурсов и выбирать подходящий контроллер для достижения того или иного состояния.



https://aristov.tech

Kubernetes



https://aristov.tech

Kubernetes

Как в облачном так и on premise кластере.

Плюсы - гибкость и простота развертывания HA конфигураций со встроенными пулерами и т. д., обычно есть GUI для управления.

Варианты:

- свой релиз обернуть в чарт
- **cloud native**
  - **patroni**
  - **stolon**
  - **ClusterControl**
- операторы (для YAML разработчиков новый тип ресурса - postgresql)
  - от авторов Патрони https://github.com/zalando/postgres-operator
  - от дистрибьютора Crunchy https://github.com/CrunchyData/postgres-operator
  - от контрибьютора EnterpriseDB https://cloudnative-pg.io/
  - на хабе операторов 15 вариантов

https://aristov.tech

Kubernetes operator

Посмотрим архитектуру на примере: https://github.com/zalando/postgres-operator

```yaml
apiVersion: "acid.zalan.do/v1"
kind: postgresql
metadata:
  name: acid-minimal-cluster
spec:
  teamId: "acid"
  volume:
    size: 1Gi
  numberOfInstances: 2
  users:
    zalando:  # database owner
    - superuser
    - createdb
    foo_user: []  # role for application foo
  databases:
    foo: zalando  # dbname: owner
  preparedDatabases:
    bar: {}
  postgresql:
    version: "16"
```

Kubernetes

В статье на Хабре собраны и протестированы основные операторы:

	Stolon	Crunchy Data	Zalando	KubeDB	StackGres	CloudNativePG
Текущая версия	0.17.0	5.1.2	1.8.0	0.17	1.2.0	1.16.0
Версии PostgreSQL	9.6-14	10-14	9.6-14	9.6-14	12, 13	10-14
Общие возможности						
Кластеры PgSQL	✓	✓	✓	✓	✓	✓
Теплый и горячий резерв	✓	✓	✓	✓	✓	✓
Синхронная репликация	✓	✓	✓	✓	✓	✓
Потоковая репликация	✓	✓	✓	✓	✓	✓

Kubernetes. За и против

- Расплачиваемся за дополнительный уровень абстракции ресурсами
- Огромная гибкость, масштабируемость и отказоустойчивость
- Миграция управляющих pod между нодами
- Выше порог входа для k8s
- Ниже для создания HA PostgreSQL, но выше для решения возникших проблем
- Наоборот для On-premise

- On-premise позволяет утилизировать 100% ресурсов
- Производительность локальных NVMe значительно выше CEPH и аналогов

https://aristov.tech

Проект aristov.tech

aristov.tech

Книги по 14 Постгресу (Архитектуре/16 Оптимизации) https://aristov.tech/#orderbook
Эксклюзивный **курс** по Оптимизации Постгреса - 6 поток - постоянно дорабатывается исходя из бизнес-кейсов https://aristov.tech/blog/kurs-po-optimizaczii-postgresql/ - **старт 24 мая**
Со всеми **отзывами** без цензуры можно ознакомится по ссылке https://aristov.tech/blog/otzivi-kurs/
Блог с популярными темами https://aristov.tech/blog
ТГ **канал** с новостями блога и проектаhttps://t.me/aristov_tech
Ютуб канал с интересными видео https://www.youtube.com/@aristovtech
Рутуб - https://rutube.ru/u/aristovtech/
ВКВидео - https://vkvideo.ru/@aristov.tech
Моя **группа** ДБА https://t.me/dbaristov - уже ~370 экспертов
Открытие вебинары (6 штук + этот) - https://aristov.tech/blog/otkrytye-lekczii-ot-aristov-tech/
Воркшопы (пока 5) - https://aristov.tech/blog/vorkshopy-ot-aristov-tech/

https://aristov.tech

aristov.tech

Также за прошедшее время был запущен проект **менторинга** в котором уже участвует 17 лучших экспертов в отрасли ДБА/DevOps и разработки.
Ознакомится с преподавателями и проектом https://aristov.tech/mentorship

В направлении крутых **вакансий** уже 5 предложений от партнёров
https://aristov.tech/blog/vakansii-ot-partnerov-aristov-tech/

Всегда можно со мной связаться через сайт для консультация, аудита вашего проекта, менторинга, обучения и многого другого

aristov.tech

Розыгрыш книг и скидки 30% на курс (только очные варианты)

регистрируемся по форме:

https://forms.gle/9i8eRcSHGLnMXi3B7

Сам розыгрыш в db-fiddle

https://www.db-fiddle.com/f/43wbuUzv7YUAcS3aypkvvC/1

https://aristov.tech

ДЗ

ДЗ

1. Подписаться на канал
2. Подписаться на Ютуб
3. Вступить в мою группу ДБА
4. Посещать открытые уроки
5. Посещать воркшопы или посмотреть их в записи
6. Прийти на курс %)
7. Прокачаться с помощью менторинга
8. Устроиться на одну из вакансий 450+

П.С. Книги просто бомбические

https://aristov.tech

Спасибо за внимание!

Есть уже 2 воркшопа по Патрони на 5 и 4 часа соответственно

Следующий ОУ в августе 2025

Аристов Евгений