



PostgreSQL Cloud Solutions



Проверить, идет ли запись

Меня хорошо видно && слышно?



Тема вебинара

Варианты кластеров высокой доступности для PostgreSQL



Алексей Железной

Data Engineer

- 4 года опыта Инженером данных/аналитиком (Python, Airflow, Clickhouse, Greenplum)
- 2 года опыта преподавания в OTUS (курсы DWH Analyst/DE/PostgreSQL Cloud)

[LinkedIn](#)



Правила вебинара



Активно
участвуем



Задаем вопрос
в чат или голосом



Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения



Индивидуально



Время, необходимое
на активность



Пишем в чат



Говорим голосом

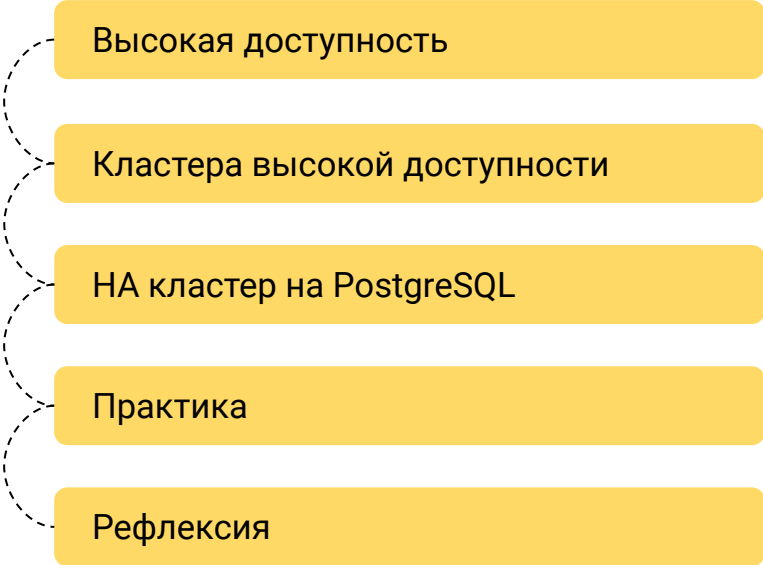


Документ



Ответьте себе или
задайте вопрос

Маршрут вебинара



Высокая доступность

Кластера высокой доступности

HA кластер на PostgreSQL

Практика

Рефлексия

Цели вебинара

К концу занятия вы будете

1. Иметь представление о высокой доступности
2. Знать варианты классических HA кластеров для PostgreSQL
3. Уметь настраивать классический PostgreSQL HA кластер

Смысл

Зачем вам это уметь

1. Выбрать оптимальный вариант высокой доступности для PostgreSQL
2. Уметь настроить высокодоступный кластер PostgreSQL своими руками

Высокая доступность

Высокая доступность

High Availability, или HA - показатель устойчивости системы к сбоям инфраструктуры:

- гарантирует отсутствие длительного эффекта от сбоя сервера или системы, позволяет контролировать и поддерживать работоспособность внутренних серверов
- измеряется в 9-ках
- все хотят 99,999 - 5 минут простоя в год
- все предлагают 99,99 - час простоя в год
- метод измерения и критерий доступности - тот еще вопрос



[Калькулятор SLA: 99.9% аптайм](#)

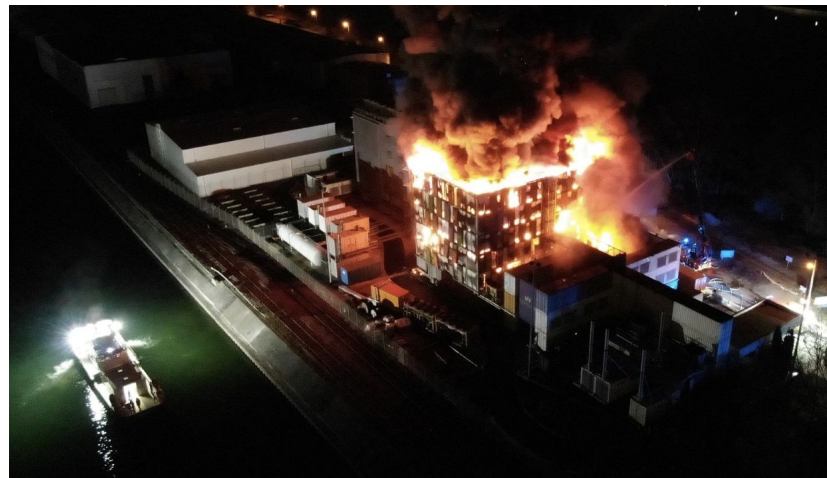
Производительность

- Performance
- Время отклика на запрос
- Как измерить?
- Мониторинг?



Факторы, влияющие на доступность

- Отказы
 - профилактика: запланированные
 - аварии: незапланированные
- Частота
 - по вине человека
 - техники
 - природы
- Метод повышения доступности
 - защита от отказов, или отказоустойчивость



Отказоустойчивость

- Reliability
 - метод повышения доступности (high availability)
- Устойчивость к отказам
- Не путать с DR (disaster recovery)
- Обеспечивается техническими и организационными средствами
- Техническое средство - избыточность
 - Это практика хранения данных в двух или более местах в базе данных или системе хранения данных. Избыточность данных гарантирует, что организация сможет продолжать работу или предоставлять услуги в случае, если с данными что-то случится - например, в случае повреждения или потери данных.

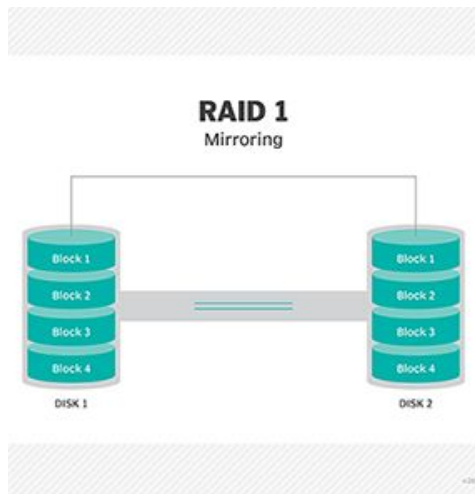
High Availability vs. Disaster Recovery

- HA - высокая доступность
 - пять девяток еще достижимо
 - нет потери данных
 - не дальше 100 км
- DR - восстановление после катастроф (когда HA уже не помог)
 - никаких пяти девяток
 - потеря данных почти всегда
 - только дальше 100 км

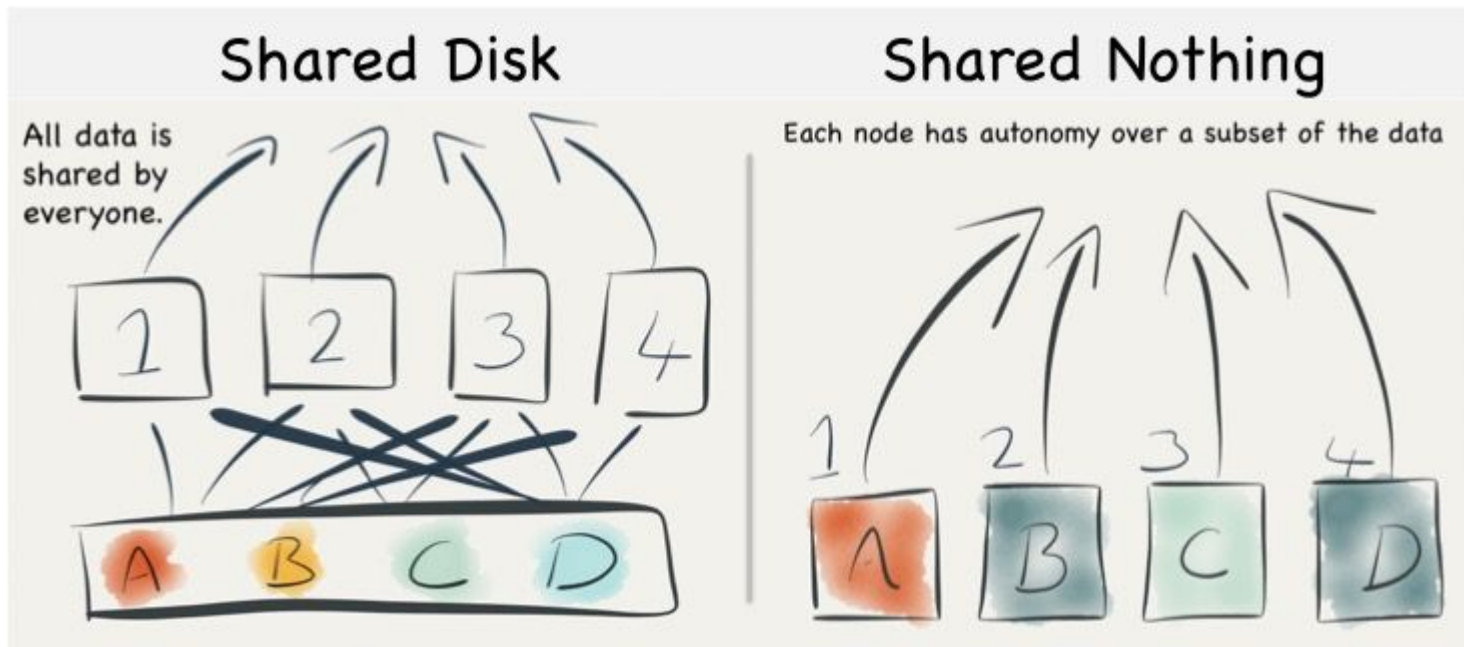
Избыточность - redundancy

Redundancy - чтобы перенести задачу с аварийного сервера на работоспособный требуются свободные вычислительные ресурсы. Избыточность:

- Сервисов (compute)
 - active-passive кластеры - резервный узел включается только при отказе основного
 - параллельные кластеры
- Данных (data)
 - копии (raid - redundancy array of the independent disk)
 - реплики

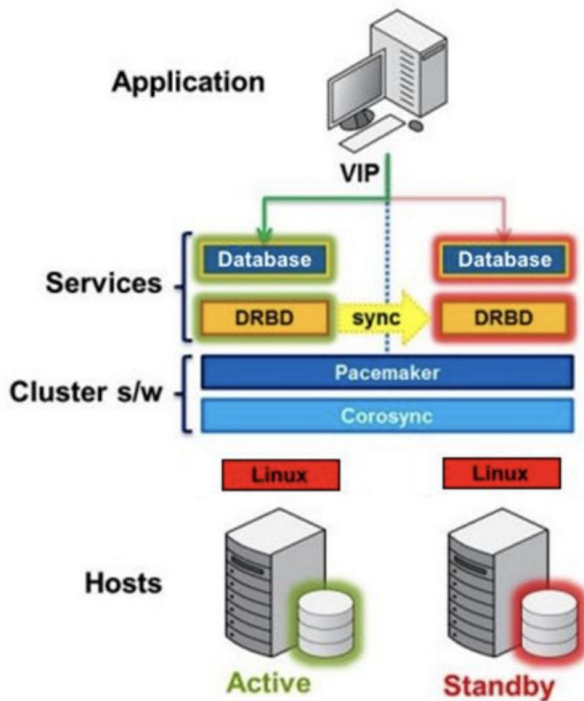


Кластера



Пример shared storage кластера

- Corosync
 - сообщения между серверами
- Pacemaker
 - запускает и останавливает сервисы
 - контролирует что сервисы запущены только на одном из узлов
- DRBD
 - распределенное сетевое устройство хранения



Вопросы?



Ставим "+",
если вопросы есть



Ставим "-",
если вопросов нет

Репликация PostgreSQL

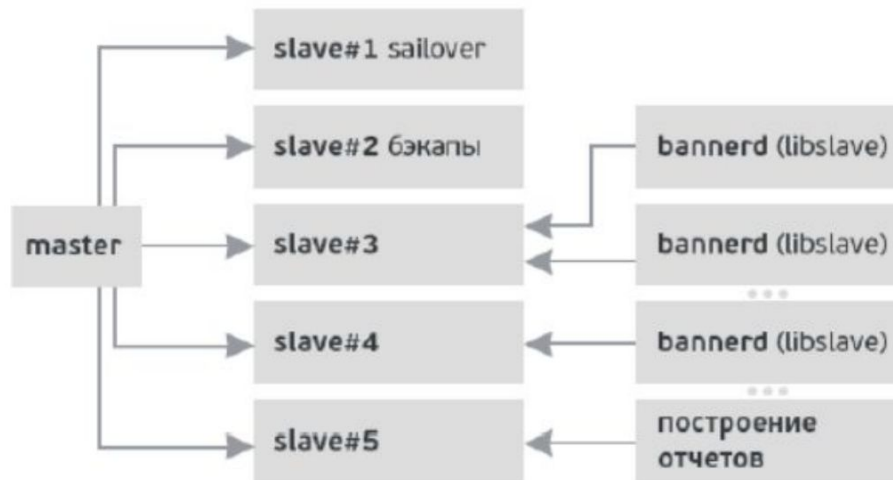
Репликация PostgreSQL

master - slave, primary - standby

- Тип
 - логическая
 - физическая
- Задержка
 - синхронная
 - асинхронная
- Доступность реплики
 - warm
 - hot



Репликация PostgreSQL - чего не хватает?



*Структура проекта Mail.Ru Target

Репликация PostgreSQL - чего не хватает?

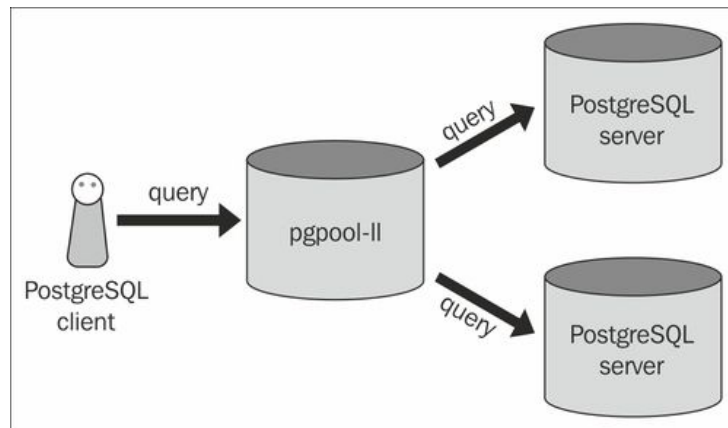
- Управление трафиком
 - запросы должны автоматически отправляться на активный сервер
 - в случае hot standby запросы на чтение могут автоматически отправляться на standby сервер
- Управление сервисами
 - failover - процесс переключения на резервный объект
 - failback - процесс возвращения на исходное место после аварии или запланированного периода технического обслуживания.

HA кластеры PostgreSQL

- Классические
 - [Pgpool II](#)
 - [pg_bouncer + haproxy](#)
 - [pg-auto-failover](#)
 - [repmgr](#)
- [k8s](#)
- Cloud native
 - [patroni](#)
 - [stolon](#)
 - [slony](#)
 - [ClusterControl](#)

PgPool II

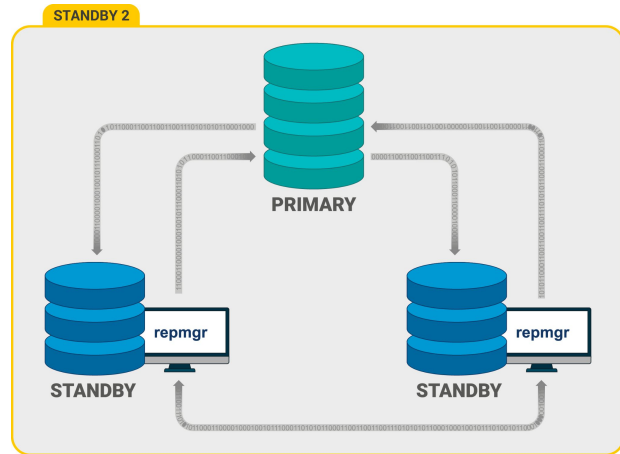
- Перечень наиболее востребованных предлагаемых функций:
 - встроенная репликация
 - объединение соединений
 - балансировка нагрузки для масштабирования чтения
 - высокая доступность (сторожевой таймер с виртуальным IP, онлайн-восстановление и обход отказа)



[Полезная информация](#)

repmgr

- repmgr - для управления репликацией и отказоустойчивостью в кластере PostgreSQL серверов. Он расширяет встроенные возможности репликации PostgreSQL утилитами для настройки резервных серверов, мониторинга репликации и выполнения failover или переключения (switchover operations).
- Последняя версия repmgr (5.2.1) поддерживает все версии PostgreSQL от 9.5 до 15.
- repmgr распространяется по лицензии GNU GPL 3 и поддерживается 2ndQuadrant (EDB)



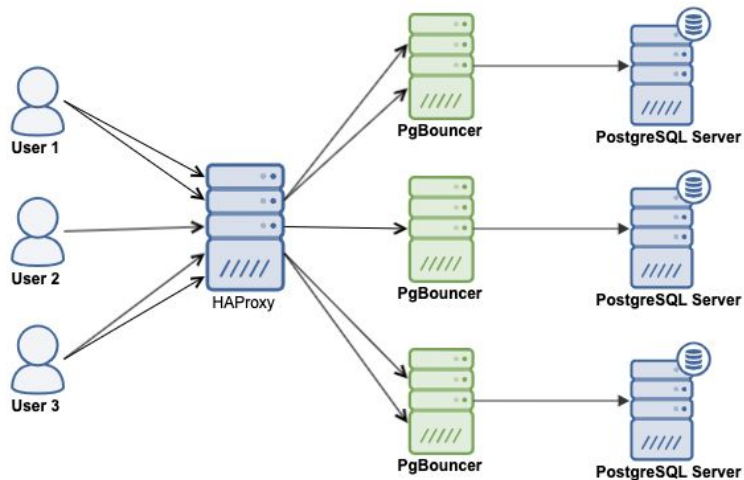
PgBouncer + HAproxy

PgBouncer - Lightweight connection pooler for PostgreSQL.

- PgBouncer действует как сервер PostgreSQL, создает соединение с сервером PostgreSQL или повторно использует его, если оно существует.

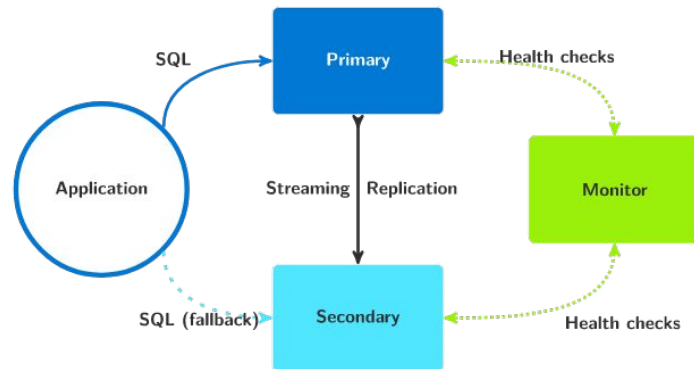
HAProxy - The Reliable, High Performance TCP/HTTP Load Balancer.

[Полезная информация](#)



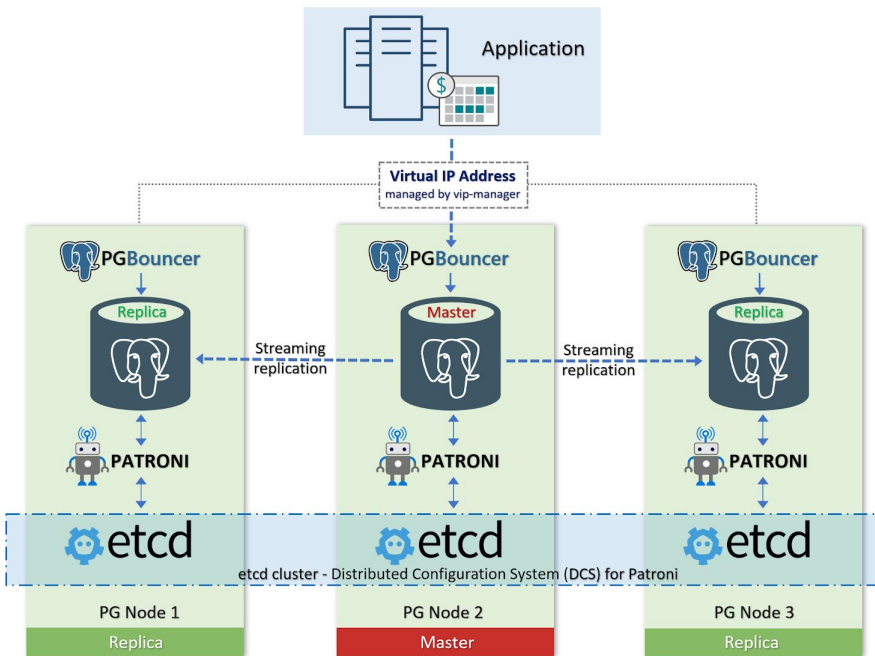
pg_auto_failover

- Простой и надежный способ управления автоматическим отказоустойчивым Postgres.
- Изменение конфигурации в “на горячую” без простоев.
- pg_auto_failover рассчитан на работу с одной службой PostgreSQL с использованием трех узлов. При такой настройке система устойчива к потере любого из трех узлов.
- Высокая доступность
- Отказоустойчивость



Patroni

Patroni - это шаблон для создания собственного специализированного решения высокой доступности с использованием Python и - для максимальной доступности - распределенного хранилища конфигурации, такого как ZooKeeper, etcd, Consul или Kubernetes.

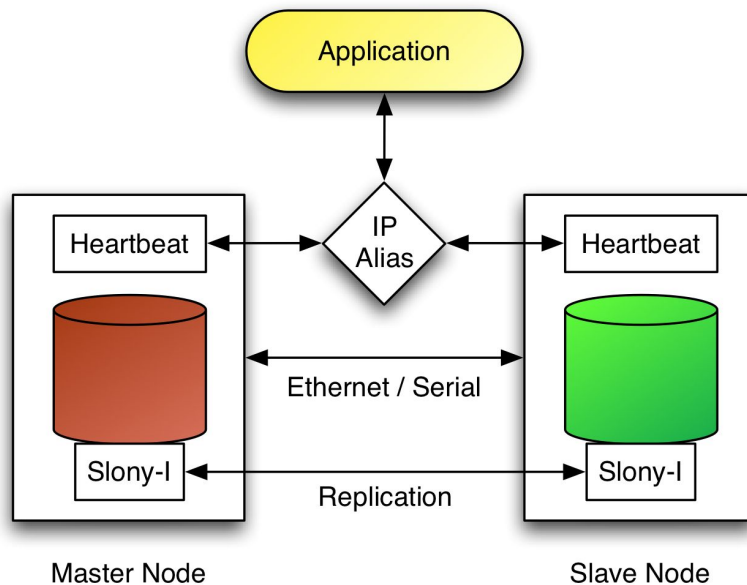


Stolon

- Использует потоковую репликацию PostgreSQL.
- Устойчив к любым видам разбиения.
- Интеграция с kubernetes
- Использует кластерное хранилище, такое как etcd, consul или kubernetes API server
- Асинхронная (по умолчанию) и синхронная репликация.
- Полная настройка кластера за несколько минут.
- Простое администрирование кластера
- Возможность восстановления в момент времени, интегрированная с вашим предпочтительным инструментом резервного копирования/восстановления.
- Резервный кластер (для многосайтовой репликации и миграции с практически нулевым временем простоя).
- Автоматическое обнаружение сервисов и динамическая реконфигурация.
- Возможность использования pg_rewind для быстрой ресинхронизации экземпляров с текущим мастером.

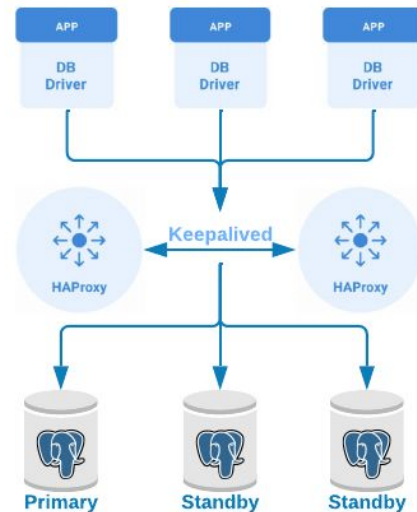
Slony

- Slony-I - это система репликации "от ведущего к нескольким ведомым" для PostgreSQL, поддерживающая каскадирование (например - узел может питать другой узел, который питает другой узел...) и обход отказа (failover)



ClusterControl

- Позволяет гибко управлять базами данных и упростить настройку.
- Управляет failover'ом, автоматическим резервным копированием, параметрами высокой доступности, балансировкой нагрузки и распределенным развертыванием
- Простое и безопасное развертывание
- Расширенный мониторинг
- Полностью интегрированный CLI
- Автоматизированные советники по производительности



Вопросы?



Ставим “+”,
если вопросы есть



Ставим “-”,
если вопросов нет

Рассмотрим задачу

Условия

Потребность иметь хранилище с данными для почти любого приложения — необходимость. А иметь это хранилище устойчивым к невзгодам в сети или на физических серверах — хороший тон грамотного архитектора.

Другой аспект — высокая доступность сервиса даже при больших конкурирующих запросах на обслуживание, что означает легкое масштабирование при необходимости.



Проблемы для решения

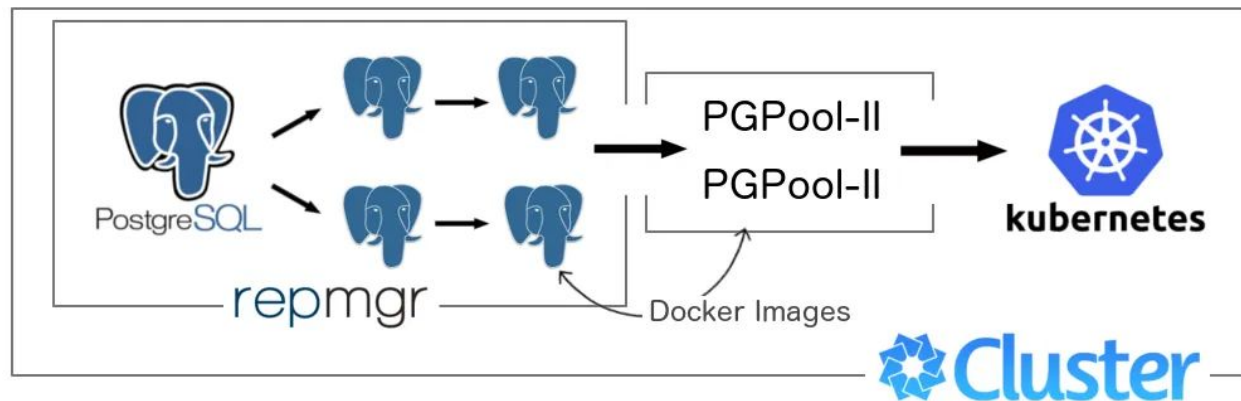
- Физически распределенный сервис
- Балансировка
- Неограниченное масштабирование путем добавление новых узлов
- Автоматическое восстановление при сбоях, уничтожении и потери связи узлами
- Отсутствие единой точки отказа

```
master (primary node1)  --\
|- slave1 (node2)       ---\   / balancer \
|  |- slave2 (node3)    -----|----|         |-----client
|- slave3 (node4)       ---/       \ balancer /
|  |- slave4 (node5)    ---/
```

Примерная схема

При условии входных данных:

- Больше число запросов на чтение (по отношению к записи)
- Линейный рост нагрузки при пиках до $\times 2$ от среднего



Pgpool. Умеет почти все

- Балансировка
- Хранение пачки коннектов для оптимизации соединения и скорости доступа к БД
- Поддержка разных вариантов репликации (stream, slony)
- Авто определение Primary сервера для записи, что важно при реорганизации ролей в кластере
- Поддержка failover/failback
- Собственная репликация master-master
- Согласованная работа нескольких узлов Pgpool-ов для искоренения единой точки отказа



Pgpool. Почему не всегда хорош?

- Восстановление с помощью Pgpool2 не предлагает никакой системы принятия решения о следующем мастере — вся логика должна быть описана в командах failover/failback
- Время записи, при репликации master-master, сводится к удвоенному по отношению варианта без него, вне зависимости от количества узлов
- Сложно построить каскадный кластер (когда один slave читает с предыдущего slave)

Split-brain

- Это ситуация, в которой разные сегменты кластера могут создать/избрать нового Master-а и думать, что проблема решена.

Помогает Repmgr. Он может:

- Клонировать Master и автоматически настроить вновьрожденный Slave
- Реанимировать кластер при смерти Master-а
- Избрать нового Master-а и перенастроить все Slave сервисы следовать за ним.
- Выводить из кластера узлы
- Мониторить здоровье кластера
- Выполнять команды при событиях внутри кластера



Вопросы?



Ставим “+”,
если вопросы есть



Ставим “-”,
если вопросов нет

Практика

Bitnami

Установка PostgreSQL с архитектурой HA в кластере Kubernetes

- Helm chart на основе схемы bitnami/postgresql, включает некоторые изменения для обеспечения высокой доступности, такие как:
 - Добавлено новое развертывание, сервис был добавлен для развертывания Pgpool-II, чтобы действовать как прокси для PostgreSQL бэкенда.
 - Замена bitnami/postgresql на bitnami/postgresql-repmgr, который включает и настраивает repmgr.

Вопросы?



Ставим “+”,
если вопросы есть



Ставим “-”,
если вопросов нет

Список материалов для изучения

1. [Готовим нагруженный Postgres в Yandex.Cloud](#)
2. [Типичные ошибки при построении высокодоступных кластеров и как их избежать. Александр Кукушкин](#)
3. [Управление высокодоступными PostgreSQL кластерами с помощью Patroni. А.Клюкин, А.Кукушкин](#)
4. [Кластер высокой доступности на postgresql 9.6 + repmgr + pgbouncer + haproxy + keepalived + контроль через telegram](#)
5. [Postgres streaming replication cluster for any docker environment](#)
6. [How to Achieve PostgreSQL High Availability with pgBouncer](#)
7. [Использование ClusterControl для аварийного восстановления PostgreSQL в гибридном облаке](#)

Рефлексия

Цели вебинара

К концу занятия вы будете

1. Иметь представление о высокой доступности
2. Знать варианты классических HA кластеров для PostgreSQL
3. Уметь настраивать классический PostgreSQL HA кластер

Рефлексия



С какими впечатлениями уходите с вебинара?



Как будете применять на практике то, что узнали на вебинаре?

**Заполните, пожалуйста,
опрос о занятии
по ссылке в чате**

Спасибо за внимание!

Приходите на следующие вебинары



Алексей Железной

Data Engineer

- 4 года опыта Инженером данных/аналитиком (Python, Airflow, Clickhouse, Greenplum)
- 2 года опыта преподавания в OTUS (курсы DWH Analyst/DE/PostgreSQL Cloud)

[LinkedIn](#)