OTUS

Онлайн образование

otus.ru

Тема вебинара

PostgreSQL. Backup + Репликация

Прусов Василий

e-mail: vasiliyqa@gmail.com

Преподаватель

Василий Прусов

Более 8 лет в IT индустрии, из них более 3-х на должности системного инженера

Ранее занимался тестированием разнообразных продуктов, занимался автоматизацией тестирования

Ведущий системный инженер в “Тета Дата Солюшнс”

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в Slack
#webinars-2021-07
или #general

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

Цели вебинара

После занятия вы сможете

1. Настраивать бэкапы
2. Восстанавливать информацию после сбоя
3. Настраивать репликацию

Смысл

Зачем вам это уметь

1. Чтобы предотвратить потерю информации
2. Обеспечить высокую доступность и организовать масштабируемость

Backup

Основные рекомендации

1. Бэкап должен быть всегда
2. Бэкап должен быть автоматическим
3. Восстановление из бэкапа - это крайняя мера
4. Бэкап нужно хранить отдельно от данных
5. Бэкап нужно регулярно проверять
6. Полезно дублировать бэкап на удаленную площадку
7. Бэкап - это нагрузка на работающую систему

12 типичных ошибок

[12 типичных ошибок при бэкапе баз данных / Хабр]

Виды архивирования

Логическое резервное копирование

Физическое резервное копирование

Логическое копирование

Логическая копия

+ можно сделать копию отдельного объекта или базы
+ можно восстановить н акластере другой основной версии
+ можно восстановить на другой архитектуре

- невысокая стоимость относительно физической

Варианты логического архивирования

- команда **COPY**

COPY { имя_таблицы [(имя_столбца [, ...])] | (запрос) } **TO** { 'имя_файла' | PROGRAM 'команда' | STDOUT } [[WITH] (параметр [, ...])]

COPY { имя_таблицы [(имя_столбца [, ...])] **FROM** { 'имя_файла' | PROGRAM 'команда' | STDIN } [[WITH] (параметр [, ...])]

- утилита **PG_DUMP**

Выдает на консоль или в файл или в специальный архив

Параллельное выполнение

Позволяет ограничить набор выгружаемых объектов

- утилита **PG_DUMPALL**

Сохраняет весь кластер

Параллельное выполнение не поддерживается

Физическое копирование

Физическое резервирование

Используется механизм восстановления после сбоя

+ скорость восстановления
+ можно восстановить кластер на определенный момент времени

- нельзя восстановить отдебльную БД, только весь кластер
- восстановление только на той же основной версии и аррхитекутре

Виды физического резервирования

Холодное - когда БД остановлена

- сервер корректно остановлен (необходимы только файлы данных)
- сервер некорректно выключен (файлы данных и wal сегменты)

Горячее - на работающем экземпляре

- необходимы как файлы данных, так и wal сегменты, причем нужно проконтролировать, чтобы сервер сохранил все wal файлы на время копирования основных данных
- снэпшоты

Создание автономной копии

Автономная копия содержит и файлы данных и wal.

Резервное копирование - **pg_basebackup**

- подключается к серверу по протоколу репликации
- выполняет контрольную точку
- переключается на следующий сегмент wal
- копирует указанный каталог в указанную директорию
- переключается на следующий сегмент wal
- сохраняет все сегменты wal, сгенерированные за время копирования

Восстановление

- разворачиваем созданную автономную копию
- запускаем сервер

Создание автономной копии

Создадим 2 кластер

$ pg_createcluster -d /var/lib/postgresql/10/main2 10 main2

Удалим оттуда файлы

$ rm -rf /var/lib/postgresql/10/main2

Сделаем бэкап нашей БД (запуск на вторичном сервере, если другой хост то -h и настройка в pg_hba доступа по слоту репликации)

$ pg_basebackup -p 5432 -D /var/lib/postgresql/10/main2

-- Зададим другой порт в версии до 10

-- $ echo 'port = 5433' >> /var/lib/postgresql/10/main2/postgresql.auto.conf

Стартуем кластер

$ pg_ctlcluster 10 main2 start

Смотрим как стартовал

$ pg_lsclusters

Архив журналов

Файловый архив

- сегменты WAL копируются в архив по мере заполнения
- механизм работает под управлением сервера
- неизбежны задержки попадания данных в архив
- select pg_switch_wal(); - переключимся на следующий файл

Потоковый архив

- в архив постоянно записывается поток журнальных записей
- требуются внешние средства - pg_receivewal
- задержки минимальны
- второй сервер в режиме stand by

Файловый архив журналов

Процесс archiver

Параметры

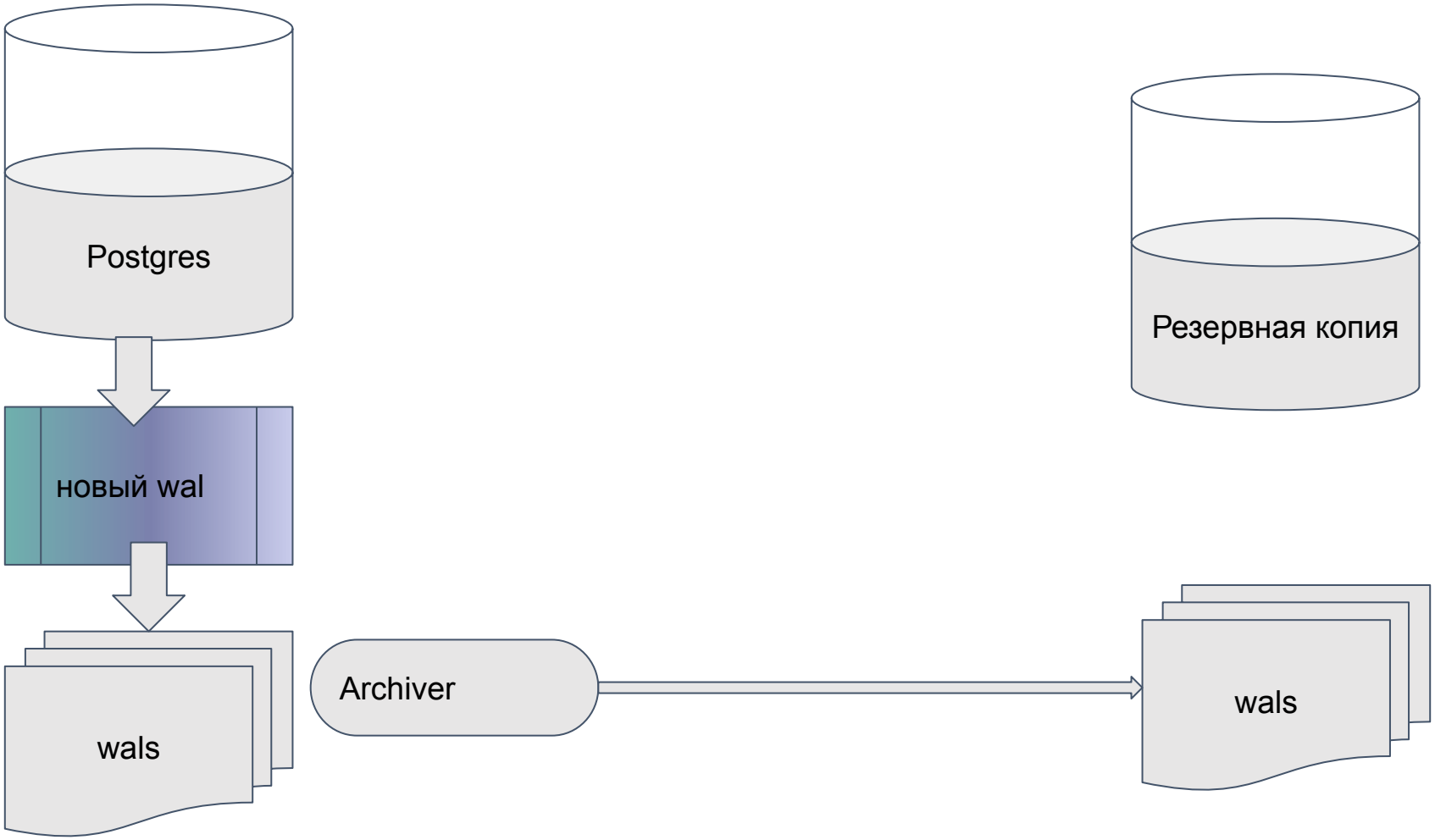
SELECT name, setting FROM pg_settings WHERE name IN ('archive_mode','archive_command','archive_timeout');

- ALTER SYSTEM SET archive_mode = on
- ALTER SYSTEM SET archive_command - команда shell для копирования сегмента WAL в отдельное хранилище, если результат отличен от 0 будет ретраить
- ALTER SYSTEM SET archive_timeout - максимальное время для переключения на новый сегмент WAL
- требуется рестарт сервера

Алгоритм

- при заполнении сегмента WAL вызывается команда archive_command
- если команда завершается со статусом 0, сегмент удаляется
- если команда возвращает не 0 (в частности, если команда не задана),сегмент остается до тех пор, пока попытка не будет успешной

Файловый архив журналов

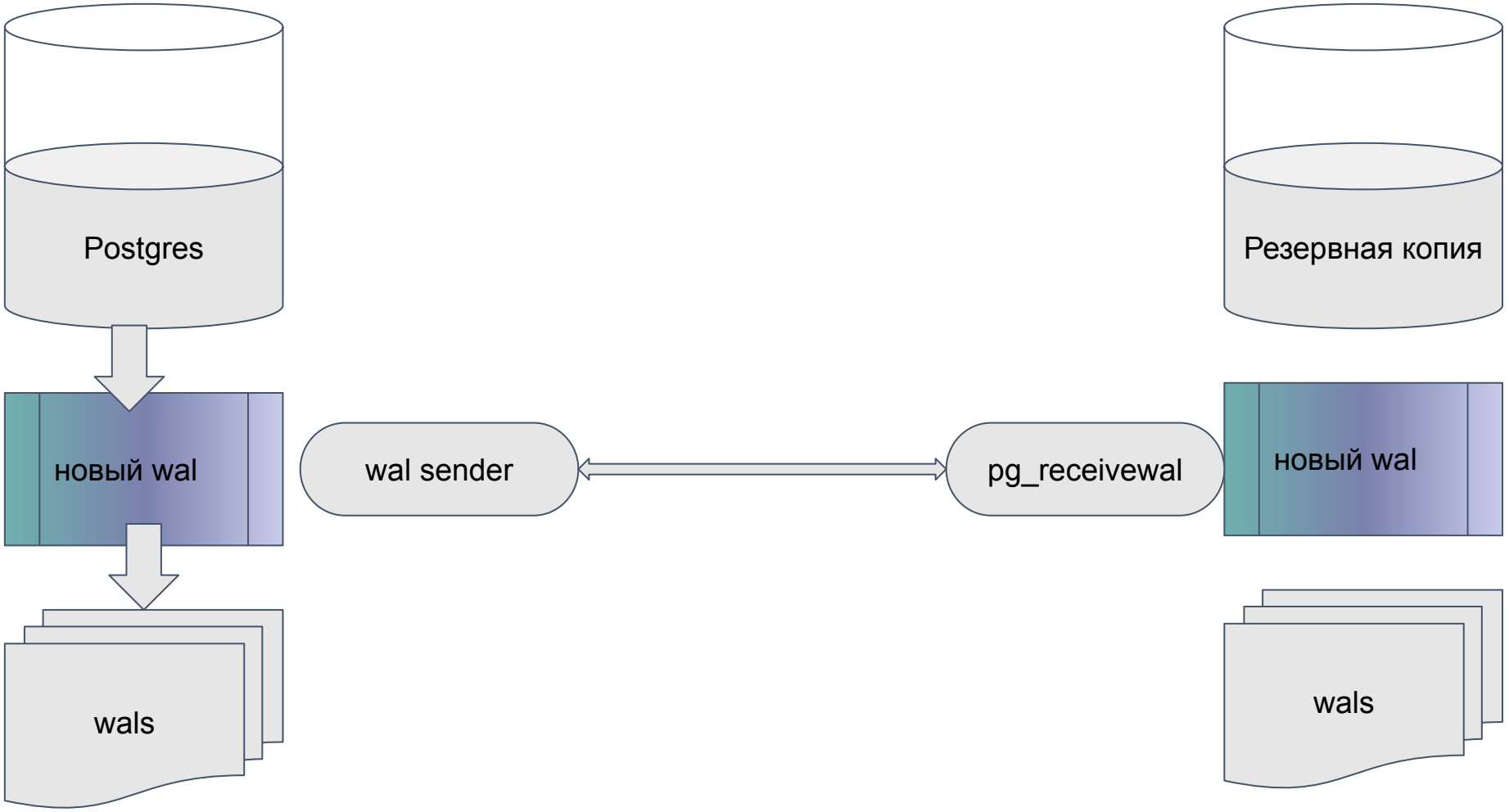


Потоковый архив журналов

Утилита pg_receivewal

- подключается по протоколу репликации (можно использовать слот)
- и направляет поток записей WAL в файлы-сегменты
- стартовая позиция — начало сегмента, следующего за последним заполненным сегментом, найденным в каталоге,
- или начало текущего сегмента сервера, если каталог пустой
- в отличие от файлового архива, записи пишутся постоянно
- при переходе на новый сервер надо перенастраивать параметры

Потоковый архив



Еще способы бэкапирования

pgBackRest

- параллельный бэкап и восстановление
- локальные бэкапы или используя ssh
- можно одновременно использовать несколько удаленных площадок для бэкапа
- полный, инкрементный и дифференциальный бэкапы
- поддержка ротации бэкапов
- поддержка истечения сроков у бэкапов
- потоковая компрессия и чексуммы
- поддержка GCS, S3, Azure

pg_probackup

- инкрементальный бэкап и восстановление
- проверка бэкапа на правильность и возможность восстановления
- параллельная работа
- компрессия
- дедупликация
- частичное восстановление

WAL-G

https://github.com/wal-g/wal-g

Позволяет хранить wal как в файлах, так и в облачных бакетах

- Amazon S3
- Google Cloud Storage
- Azure Storage

Данные хранятся в облаке, поэтому мониторить ничего не нужно, это уже проблема облачного сервиса, как обеспечить доступность ваших данных, когда они вам нужны.

Валидация из бэкапа

Валидация из бэкапа

Постгрес допускает ошибки при формирования бэкапов, поэтому:

- Докер образ для минимального запуска
- pg_hba.conf → trust
- скрипт с восстановлением из бэкапа
- прогоняем тест (например статистика по таблицам на момент бэкапа и после восстановления)

Репликация

Зачем нужна репликация

1. Высокая доступность. Бэкап это хорошо, но нужно время на его развертывание.
2. Что делать, когда закончились физические ядра и память у сервера? горизонтально масштабировать
3. Бэкап лучше делать с реплики, а не мастера.
4. Геораспределение нагрузки.
5. Нагрузку по чтению и отчетам можно переложить на реплику

Виды репликации

Виды репликации

Физическая репликация

Логическая репликация

Физическая репликация

- мастер-слейв: поток данных только в одну сторону
- трансляция потока журнальных записей или файлов журнала
- требуется двоичная совместимость серверов
- возможна репликация только всего кластера

Физическая репликация

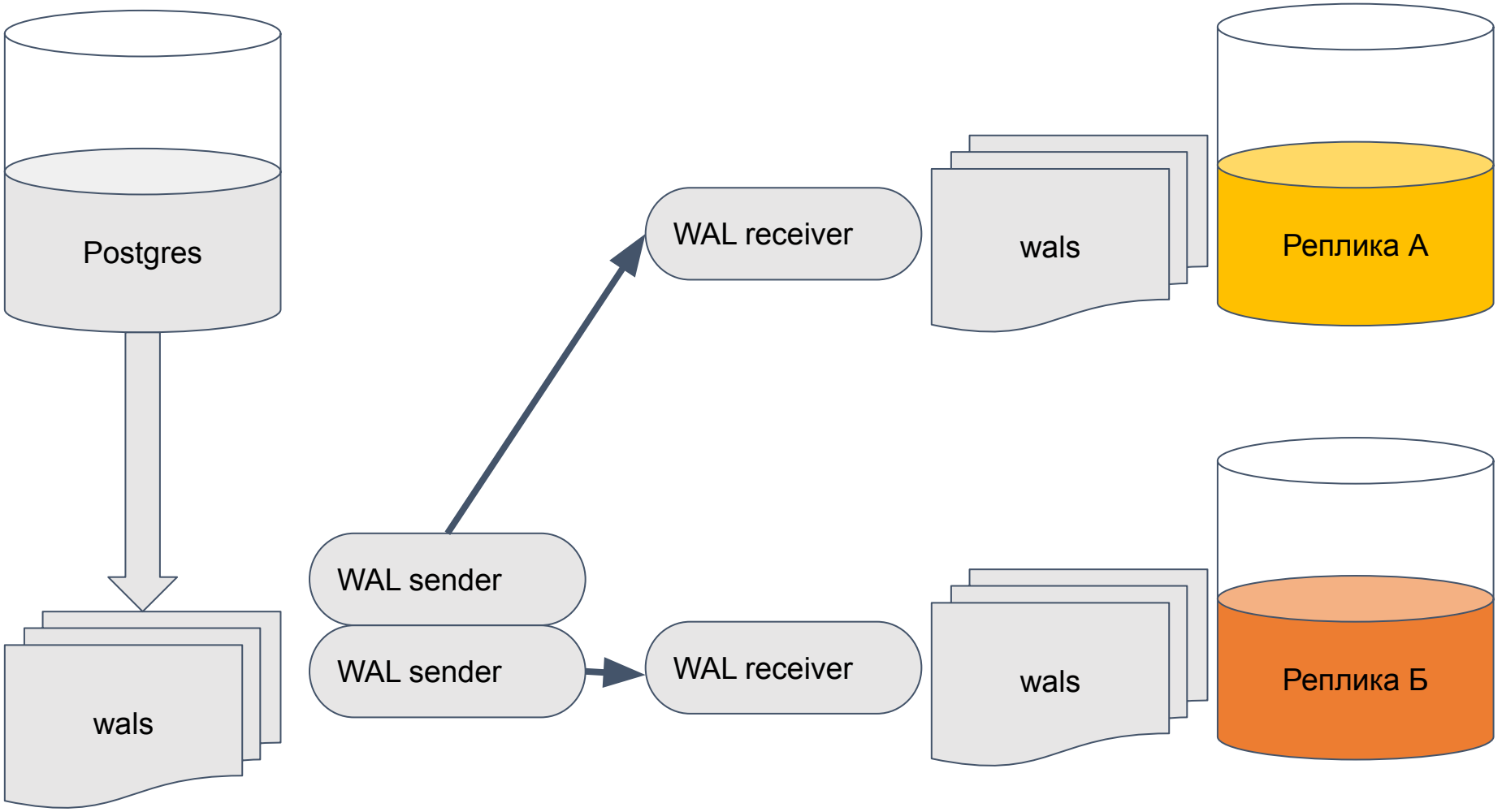
базовая резервная копия — **pg_basebackup**

- разворачиваем резервную копию,
- создаем управляющий файл recovery.conf

  (standby_mode = on)

- и запускаем сервер
- сервер восстанавливает согласованность
- и продолжает применять поступающие журналы
- доставка — поток по протоколу репликации или архив WAL
- подключения (только для чтения) разрешаются
- сразу после восстановления согласованности

Физическая репликация



Физическая репликация

Допускаются на реплике

- запросы на чтение данных (select, copy to, курсоры)
- установка параметров сервера (set, reset)
- управление транзакциями (begin, commit, rollback...)
- создание резервной копии (pg_basebackup)

Не допускаются

- любые изменения (insert, update, delete, truncate, nextval...)
- блокировки, предполагающие изменение (select for update...)
- команды DDL (create, drop...), в том числе создание временных таблиц
- команды сопровождения (vacuum, analyze, reindex...)
- управление доступом (grant, revoke...)
- не срабатывают триггеры и пользовательские (advisory) блокировки

Физическая репликация

Перевод реплики в состояние мастера

- **$ pg_ctl -w -D /var/lib/postgresql/10/main2 promote**

- waiting for server to promote.... done
- server promoted

Что произойдет?

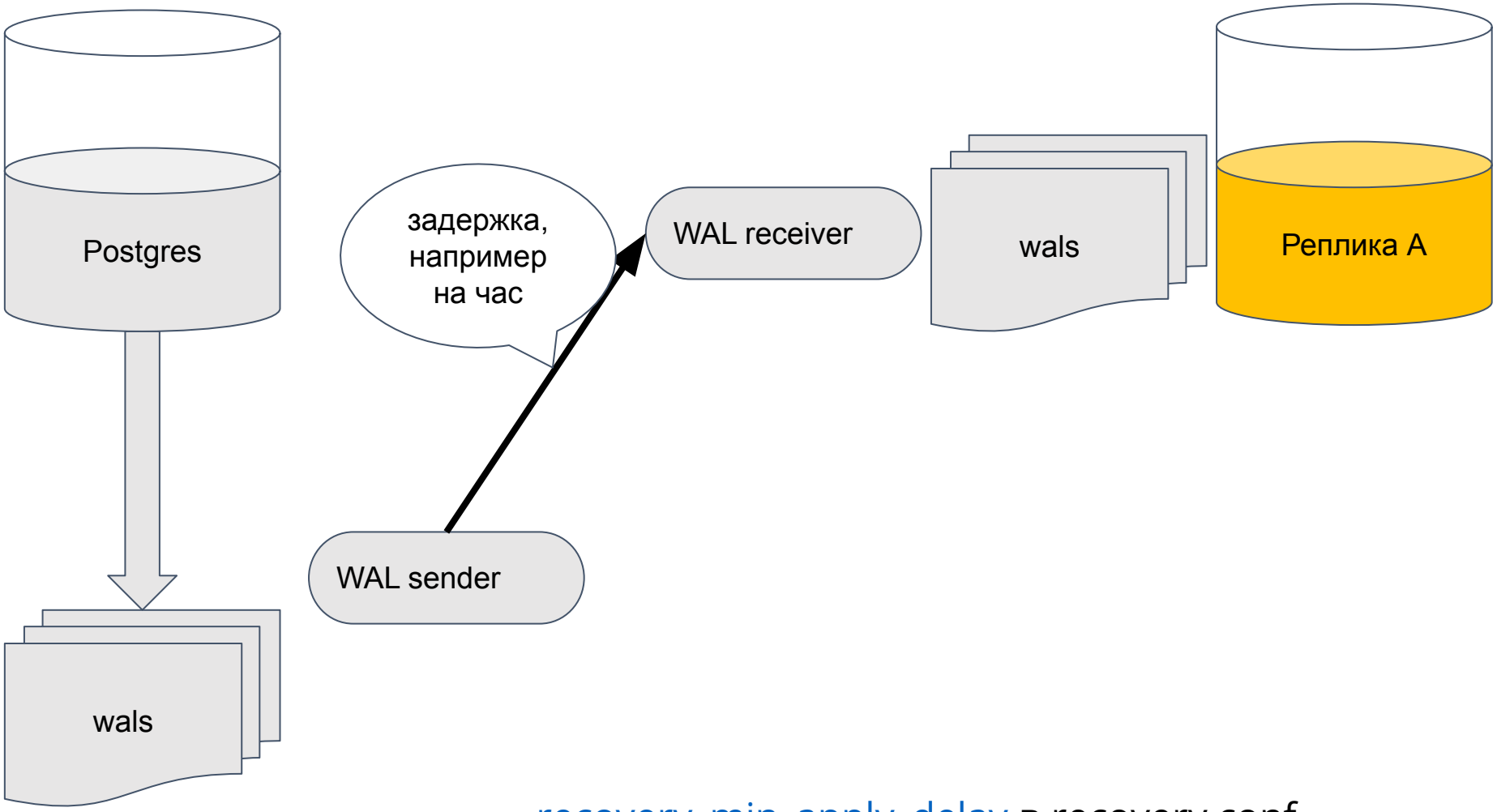
Физическая репликация

Задача: в случае сбоя основного сервера, не потерять никакие данные при переходе на реплику.

- Решение состоит в использовании синхронной репликации (**synchronous_commit = on**)
- Фиксация изменений на мастере не завершается до тех пор, пока не получает подтверждение от реплики. При необходимости синхронностью можно управлять на уровне транзакций.
- Синхронная репликация не обеспечивает идеальной согласованности данных между серверами: изменения могут становятся видимыми на мастере и на реплике в разные моменты времени.
- Начиная с версии 9.6 синхронизация может происходить с несколькими репликами.
- В 10 версии доступна синхронизация с учетом кворума.

Потоковая репликация в PostgreSQL и пример фейловера

Time machine



recovery_min_apply_delay в recovery.conf

Физическая репликация. Практика

Начиная с версии 10, все необходимые настройки уже присутствуют по умолчанию:

- wal_level = replica;
- max_wal_senders
- разрешение на подключение в pg_hba.conf по протоколу репликации.

Создадим 2 кластер
pg_createcluster -d /var/lib/postgresql/13/main2 13 main2
Удалим оттуда файлы
rm -rf /var/lib/postgresql/13/main2
Сделаем бэкап нашей БД. Ключ -R создаст заготовку управляющего файла recovery.conf
(запуск на вторичном сервере, если другой хост то -h
pg_basebackup -p 5432 -R -D /var/lib/postgresql/13/main2
Добавим параметр горячего резерва, чтобы реплика принимала запросы на чтение
echo 'hot_standby = on' >> /var/lib/postgresql/13/main2/postgresql.auto.conf
Стартуем кластер
pg_ctlcluster 13 main2 start
Смотрим как стартовал
pg_lsclusters

Физическая репликация. Практика

Посмотрим на процессы реплики:
$ ps -o pid,command --ppid `head -n 1 /var/lib/postgresql/10/main2/postmaster.pid`

Процесс wal receiver принимает поток журнальных записей, процесс startup применяет изменения.

Сравним с процессами мастера:
$ ps -o pid,command --ppid `head -n 1 /var/lib/postgresql/10/main/postmaster.pid`

Физическая репликация. Практика

Теперь переведем реплику из режима восстановления в обычный режим. Таким образом, получим два самостоятельных, никак не связанных друг с другом сервера.

pg_ctlcluster 13 main2 promote

Логическая репликация

Логическая репликация

- поставщик-подписчик: поток данных возможен в обе стороны
- информация о строках (уровень журнала logical)
- требуется совместимость на уровне протокола
- репликация между разными основными версиями Postgres
- возможна выборочная репликация отдельных таблиц

Логическая репликация

- Встроенная логическая репликация доступна в версиях PostgreSQL, начиная с 10. Для более ранних версий аналогичный функционал доступен в расширении pg_logical.
- Для передачи логических изменений (на уровне строк) используется протокол репликации. Для работы такой репликации требуется установка уровня журнала **logical**.
- Другой способ организации логической репликации состоит в использовании триггеров для перехвата изменений, помещения этой информации в очередь событий и передача ее на другой сервер. Такой способ, однако, менее эффективен, и уходит в прошлое (**Slony-I**).
- При логической репликации у сервера нет выделенной роли мастера или реплики, что позволяет организовать в том числе и двунаправленную репликацию.

Логическая репликация

Публикующий сервер

- выдает изменения данных построчно в порядке их фиксации
- (реплицируются команды INSERT, UPDATE, DELETE), в 11 версии добавили TRUNCATE
- возможна начальная синхронизация
- всегда используется слот логической репликации
- DDL не передаются, то есть таблицы-приемники на стороне подписчика надо создавать вручную.
- Данные последовательностей не реплицируются.
- Реплицировать данные возможно только из базовых таблиц в базовые таблицы. То есть таблицы на стороне публикации и на стороне подписки должны быть обычными, а не представлениями, мат. представлениями, секционированными или сторонними таблицами.
- применение изменений происходит без выполнения команд SQL и связанных с этим накладных расходов на разбор и планирование, что уменьшает нагрузку на подписчика.
- параметр **wal_level = logical**

Логическая репликация

Подписчики

- получают и применяют изменения
- без разбора, трансформаций и планирования — сразу выполнение
- возможны конфликты с локальными данными
- триггеры срабатывают для каждого подписчика отдельно

Логическая репликация

Режимы идентификации для изменения и удаления

- столбцы первичного ключа (по умолчанию)
- столбцы указанного уникального индекса с ограничением NOT NULL
- все столбцы
- без идентификации (по умолчанию для системного каталога)

Конфликты — нарушение ограничений целостности

- репликация приостанавливается до устранения конфликта вручную
- либо исправление данных,
- либо пропуск конфликтующей транзакции

Логическая репликация. Практика

Используем два сервера, полученные на предыдущей практике и настроим логическую репликацию. Для этого нам понадобится дополнительная информация в журнале.

ALTER SYSTEM SET wal_level = logical;

Рестартуем кластер
$ sudo pg_ctlcluster 13 main restart

На первом сервере создаем публикацию:
\c replica
CREATE TABLE test(i int);
CREATE PUBLICATION test_pub FOR TABLE test;
\dRp+

Логическая репликация. Практика

создадим подписку на втором экземпляре

```
\c replica
CREATE TABLE test(i int);
CREATE SUBSCRIPTION test_sub
CONNECTION 'host=localhost port=5432 user=postgres password=test
dbname=replica'
PUBLICATION test_pub WITH (copy_data = false);
\dRs
```

состояние подписки

```
SELECT * FROM pg_stat_subscription \gx
```

Итог кратко

Механизм репликации основан на передаче журнальных записей на реплику и их применении

- трансляция потока записей или файлов WAL

Физическая репликация создает точную копию всего кластера

- однонаправленная
- требует двоичной совместимости

Логическая репликация передает изменения строк отдельных таблиц

- разнонаправленная
- совместимость на уровне протокола

Спасибо за внимание!

Приходите на следующие вебинары

Прусов Василий Валерьевич

Ведущий системный инженер

e-mail: vasiliyqa@gmail.com