Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Скрытые сбои WAL-архива PostgreSQL: мимо failed_count

Как мы ловим падения archive_command, которые pg_stat_archiver не считает ошибкой

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Разворачиваем WAL-архивирование PostgreSQL 15-18 на 1С и CRM и видим слепую зону: archive_command падает (диск полон, оборвалась сеть, протух ключ S3), а failed_count не растёт. По доке сбой при сигнале, exit-коде шелла больше 125 и ERROR/FATAL из архив-модуля в pg_stat_archiver не отражается: архивер молча перезапускается. Дашборд зелёный, pg_wal раздувается до PANIC.

Почему это важно бизнесу

- Точка невозврата по WAL наступает молча: восстановление возможно только на дату последнего валидного сегмента в архиве
- Диск с pg_wal переполняется без предупреждения и приводит к PANIC-остановке базы 1С/CRM в рабочее время
- Восстановление после инцидента занимает часы вместо минут: цепочка WAL уже разорвана, а обнаружено это поздно
- Дежурный инженер верит зелёному дашборду и не эскалирует проблему, пока клиент не заметит простой
- Повторный инцидент бьёт по SLA и репутации ИТ-подрядчика перед собственником бизнеса

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

125

exit-код шелла, выше которого сбой archive_command не попадает в pg_stat_archiver

PostgreSQL 18, разд. 25.3.1

0

archive_timeout по умолчанию: принудительное переключение сегмента отключено

PostgreSQL 18, разд. 19.5.3

60 с

archive_timeout «порядка минуты» — рекомендация доки и наш базовый профиль

PostgreSQL 18, разд. 19.5.3

5 мин

порог возраста старейшего .ready-файла в archive_status, после которого алертим

наш стандарт мониторинга

85%

порог заполнения раздела pg_wal, после которого эскалируем инцидент

наш стандарт мониторинга

15

версия PostgreSQL, с которой доступен archive_library вместо shell-команды

PostgreSQL 15, E.20 Release 15

РЕАЛИЗАЦИЯ 1 ИЗ 3

Мониторинг архивации мимо pg_stat_archiver

Что настраиваем

Кластер PostgreSQL 15-18 с архивированием через archive_command на выделенном хранилище логов клиента

Как мы это делаем

1. Снимаем pg_stat_archiver раз в 60 с (archived_count, failed_count, last_archived_time, last_failed_time) как базовый, но не единственный сигнал
2. Парсим postgresql.log на паттерны archive command failed with exit code и was terminated by signal — это ловит случаи, не попадающие в failed_count
3. Считаем .ready-файлы в pg_wal/archive_status через pg_ls_archive_statusdir() и алертим по возрасту старейшего файла
4. Архивную команду строим строго по шаблону доки: test ! -f <архив>/%f && cp %p <архив>/%f — без опоры на cp -i
5. Отдельный таймер считает age(now(), last_archived_time) и алертит, если разрыв превышает 2xarchive_timeout

РЕЗУЛЬТАТ

Ловим сбои архивации за минуты вместо суток слепого мониторинга и снимаем ложное «всё зелено», когда archive_command убита сигналом или OOM-killer'ом, а счётчики стоят на месте

КЛЮЧЕВОЙ НЮАНС

По разд. 25.3.1 сбой не отражается в pg_stat_archiver, если команда убита сигналом (кроме SIGTERM при shutdown), шелл вернул код больше 125 или архив-модуль выбросил ERROR/FATAL

РЕАЛИЗАЦИЯ 2 ИЗ 3

Переход на archive_library и модуль basic_archive

Что настраиваем

Базы 1С и учётных систем на PostgreSQL 15-18 с высокой WAL-нагрузкой (от 50 ГБ WAL в сутки)

Как мы это делаем

1. Включаем archive_mode = on, archive_library = 'basic_archive' и basic_archive.archive_directory — каталог должен существовать заранее
2. Проверяем набор callback-функций модуля: обязателен только archive_file_cb, startup_cb/check_configured_cb/shutdown_cb опциональны
3. Перед стартом после краша чистим временные файлы с префиксом archtemp в архивной директории (разд. F.5.2)
4. Требуем идемпотентности: при уже существующем файле модуль возвращает true только если содержимое идентично, иначе false и повтор
5. Мигрируем с archive_command на archive_library поэтапно на реплике, сверяя archived_count/failed_count до и после

РЕЗУЛЬТАТ

Снимаем накладные расходы на fork shell-процесса для каждого WAL-сегмента и получаем доступ к серверным ресурсам напрямую из C-модуля, сохранив те же метрики контроля

КЛЮЧЕВОЙ НЮАНС

archive_command и archive_library взаимоисключающие: при обоих заданных параметрах сервер выдаёт ошибку. Шаблон test ! -f обязателен — GNU cp -i вернёт 0 при уже существующем файле

РЕАЛИЗАЦИЯ 3 ИЗ 3

pgBackRest и WAL-G как страховочный контур

Что настраиваем

Инсталляции с требованием RPO до 15 минут — бухгалтерские и торговые базы клиентов на 1С

Как мы это делаем

1. archive_command = pgbackrest --stanza=<name> archive-push %p, дополнительно pgbackrest info --output=json в проверке Zabbix
2. Разбираем коды выхода pgBackRest отдельно от кодов PostgreSQL: 082 archive-timeout, 087 archive-disabled, 104 command (команда завершилась с ошибками)
3. Для WAL-G ставим wal-g wal-push как archive_command, а разрыв цепочки ловим командой wal-g wal-verify integrity timeline по расписанию
4. Раз в сутки проверяем конфигурацию и доставку архива командой pgbackrest check независимо от показаний архивера

РЕЗУЛЬТАТ

Двойной контур контроля: даже если счётчики архивера молчат, инструмент бэкапа сам детектирует и логирует разрыв цепочки WAL и таймаут доставки сегмента

КЛЮЧЕВОЙ НЮАНС

Статистика pg_stat_archiver и лог инструмента бэкапа — разные источники правды, сверяем оба, иначе видим только половину картины сбоя

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Верить только failed_count**
  Счётчик не растёт при убийстве архивера сигналом и коде выхода больше 125 — нужен парсинг postgresql.log на archive command failed и terminated by
- ✕ **archive_timeout = 0**
  Значение по умолчанию отключает принудительное переключение сегмента — при низком трафике о разрыве архивации узнаём через часы, а не минуты
- ✕ **cp без test ! -f**
  GNU cp с ключом -i возвращает нулевой статус, если целевой файл уже существует, — архив тихо расходится с содержимым pg_wal
- ✕ **Не чистим archtemp после краша**
  Временные файлы basic_archive с префиксом archtemp остаются в архивной директории после падения сервера — доку требует удалять их до старта
- ✕ **archive_command и archive_library вместе**
  Оба параметра заданы одновременно — сервер поднимает ошибку конфигурации, архивирование не работает
- ✕ **Считать last_archived_wal гарантией**
  После краша сервер может заново архивировать уже отданный сегмент: фактическую очередь показывают .ready-файлы, а не последнее имя в статистике
- ✕ **Не смотреть на .ready-файлы**
  Число .ready в pg_wal/archive_status растёт независимо от статистики и первым сигнализирует о заторе архивации
- ✕ **Не считать рост pg_wal**
  Пока архивация не работает, PostgreSQL не удаляет и не переиспользует WAL-сегменты — раздел заполняется до PANIC-остановки базы

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- archive_mode=on, рабочий archive_command или archive_library, archive_timeout > 0
- Алерт на возраст last_archived_time из pg_stat_archiver, опрос каждые 60 с
- Мониторинг заполнения раздела pg_wal с порогом 85%

НОРМАЛЬНО

- Парсинг postgresql.log на archive command failed и terminated by signal
- Алерт по возрасту .ready-файлов в archive_status через pg_ls_archive_statusdir()
- Идемпотентная команда архивации: test ! -f перед cp, без опоры на cp -i
- Регламент удаления archtemp-файлов архивной директории после краша сервера

ХОРОШО

- Двойной контур: pg_stat_archiver плюс check/verify инструмента бэкапа
- Переход на archive_library (basic_archive или свой C-модуль) при высокой WAL-нагрузке
- Регулярный тест восстановления из архива на стенде с фиксацией фактического RPO
- Отдельные алерты по кодам pgBackRest 082/087/104 и по wal-g wal-verify

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] archive_mode=on и задан рабочий archive_command или archive_library, а не пустая строка?
- [ ] archive_timeout задан ненулевым (порядка 60 с) и согласован с требованием RPO?
- [ ] Есть алерт на возраст last_archived_time, а не только на рост failed_count?
- [ ] Логи PostgreSQL парсятся на archive command failed и terminated by signal?
- [ ] Есть контроль количества и возраста .ready-файлов в pg_wal/archive_status?
- [ ] Команда архивации идемпотентна: test ! -f перед cp, а не cp -i?
- [ ] Настроен алерт на заполнение раздела pg_wal по проценту диска (порог 85%)?
- [ ] pgBackRest/WAL-G проверяется отдельно от pg_stat_archiver своим check или wal-verify?
- [ ] После краша сервера регламентно удаляются archtemp-файлы архивной директории?
- [ ] Проведено тестовое восстановление из архива за последние 90 дней?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Настраиваем WAL-архивирование PostgreSQL: archive_command, archive_library, archive_timeout под RPO
- Строим двойной контур мониторинга: pg_stat_archiver, парсинг логов и контроль .ready-файлов
- Внедряем pgBackRest или WAL-G с раздельной обработкой их кодов ошибок и wal-verify
- Проводим тесты восстановления из архива и фиксируем фактические RPO и RTO
- Берём на сопровождение алертинг по pg_wal и по остановке архивации WAL

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

Техническая база

01 25.3 Continuous Archiving and Point-in-Time Recovery (PITR) (postgresql.org — 18)

02 25.3.1 Setting Up WAL Archiving (postgresql.org — 18)

03 19.5.3 Write Ahead Log / Archiving (postgresql.org — 18)

04 27.2.12 pg_stat_archiver (Cumulative Statistics System) (postgresql.org — 18)

05 Chapter 49. Archive Modules, 49.2 Archive Module Callbacks (postgresql.org — 18)

06 F.5 basic_archive — an example WAL archive module (postgresql.org — 18)

07 User Guide: archive-push, check; коды ошибок 082/087/104 (pgbackrest.org — 2026)

08 PostgreSQL: wal-push и wal-verify (integrity, timeline) (wal-g.readthedocs.io — 2026)

09 Наш шаблон мониторинга WAL-архивации (Zabbix + лог-парсер) (стандарт ITfresh — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.