Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Бэкап PostgreSQL для 1С на pgBackRest: схема для офиса

Физические бэкапы pgBackRest 2.58, архивация WAL, PITR, холодная копия в S3 и проверяемый restore

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Компании переходят с MS SQL на PostgreSQL (1С-сборки Postgres Pro), но оставляют «бэкап» уровня выгрузки .dt: она блокирует базу на десятки минут, рвётся на больших объёмах и не даёт восстановления на момент времени. Отказ диска или шифровальщик — и теряется и база, и единственная копия. Мы закрываем это физическими бэкапами pgBackRest с непрерывной архивацией WAL, PITR и холодной копией в S3.

Почему это важно бизнесу

- Потеря базы 1С = недели ручного ввода из бумаг и остановка учёта и отгрузок
- Без PITR теряется весь рабочий день: RPO 24 ч вместо 5–15 минут
- Шифровальщик забирает базу и локальные копии сразу — спасает только immutable-копия вне офиса
- Простой боевой 1С измеряется прямыми деньгами: часы отгрузок, зарплат, отчётности

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

2.58

версия pgBackRest, которую разворачиваем: block-инкременты (repo-block), zst, поддержка S3

pgBackRest 2.58 (01.2026)

3

compress-level для zst: 3 — дефолт уровня по доке, диапазон −7…22; оптимум скорость/сжатие

по докам pgBackRest

60 с

archive-timeout: предельная задержка отгрузки WAL-сегмента; наш порог RPO

наш стандарт

14 дн

repo-retention-full-type=time, repo-retention-full=14 — глубина хранения копий

наш стандарт

3-2-1

правило копий: 3 копии, 2 носителя, 1 вне офиса (NAS + реплика + S3)

наш стандарт

17.x

PostgreSQL 1C-сборка Postgres Pro, на ней строим wal_level=replica и архивацию

postgrespro 1C

РЕАЛИЗАЦИЯ 1 ИЗ 3

Установка pgBackRest и стэнза на узле 1С/PostgreSQL

Что настраиваем

Узел PostgreSQL (1С-сборка Postgres Pro 17) + бэкап-репозиторий на NAS

Как мы это делаем

1. Ставим pgBackRest 2.58 на сервер БД и на бэкап-узел, единый UID пользователя postgres, обмен SSH-ключами
2. В /etc/pgbackrest/pgbackrest.conf описываем стэнзу main: pg1-path, repo1-path на NAS, compress-type=zst, compress-level=3, process-max по числу ядер
3. В postgresql.conf: wal_level=replica, archive_mode=on, archive_command='pgbackrest --stanza=main archive-push %p', archive-async=y
4. pgbackrest --stanza=main stanza-create, затем check — проверяем сквозную архивацию WAL
5. Первый полный: pgbackrest --stanza=main --type=full backup; включаем repo-block для block-инкрементов

РЕЗУЛЬТАТ

Непрерывная архивация WAL с задержкой ≤60 с, полный/диф/инкр без блокировки пользователей 1С, контроль контрольных сумм на каждом блоке репозитория.

КЛЮЧЕВОЙ НЮАНС

Сверяемся с Приложением К Postgres Pro (настройка под 1С); wal_level ниже replica ломает archive-push и рвёт PITR-цепочку.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Расписание бэкапов, retention 14 дней и холодная копия в S3

Что настраиваем

Узел БД + репозиторий NAS Synology (RAID6) + холодная копия в Яндекс S3

Как мы это делаем

1. cron: full — вс 02:00 (--type=full), diff — пн–сб 02:00 (--type=diff), incr — каждые 6 ч (--type=incr)
2. Retention: repo1-retention-full-type=time, repo1-retention-full=14; expire по расписанию, следим за diff/WAL
3. Второй репозиторий repo2-type=s3 в Яндекс Object Storage (endpoint storage.yandexcloud.net), repo2-cipher-type=aes-256-cbc
4. На версионируемом S3-бакете включаем Object Lock (immutable) с retention — копию нельзя удалить или зашифровать даже с правами админа
5. pgbackrest info — контроль цепочки; pgbackrest verify — сверка контрольных сумм всего репозитория

РЕЗУЛЬТАТ

RPO 5–15 мин, история 14 дней занимает в ~1.3–1.5× меньше самой базы за счёт zst и block-инкрементов; облачная копия недоступна шифровальщику.

КЛЮЧЕВОЙ НЮАНС

Retention по diff/incr настраиваем осознанно: иначе WAL-архив копится в repo и переполняет NAS — классическая ловушка retention в pgBackRest.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Проверяемое восстановление и мониторинг в Zabbix

Что настраиваем

Тестовый сервер (отдельная VM) + Zabbix + Telegram-алерты

Как мы это делаем

1. Еженедельный bash: restore последней копии на тест-VM, старт PostgreSQL, pg_isready, набор SQL-проверок, тест-исправление конфигуратором 1С
2. PITR-репетиция: restore с --type=time --target=... и restore_command='pgbackrest --stanza=main archive-get %f "%p"'
3. Zabbix UserParameter поверх pgbackrest info (JSON) + плагин check_pgbackrest: возраст full/incr, задержка WAL, место в repo
4. Триггеры: full >8 дн — авария; incr >9 ч — предупреждение; WAL-lag >5 мин; место в repo <20% — авария; провал restore — авария с эскалацией
5. Алерты в Telegram-канал клиента и дублируются дежурной смене 24×7

РЕЗУЛЬТАТ

Бэкап подтверждён реальным восстановлением, а не галочкой в логе; сбои цепочки ловим до того, как клиент их заметит.

КЛЮЧЕВОЙ НЮАНС

Без регулярного restore бэкап — самообман: pgbackrest info не гарантирует восстановимость, только verify плюс тестовый restore.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Бэкап на том же диске, что база**
  repo1-path на том же томе/RAID: деградация RAID5 при ребилде уносит и базу, и копию. Выносим repo на отдельный узел/NAS + S3.

- ✕ **WAL не архивируются**
  Настроен только nightly pg_dump, RPO 24 ч. Включаем archive_mode=on и archive-async=y, отгрузка WAL с задержкой ≤60 с.

- ✕ **Бэкап есть, restore не проверяют**
  Год «зелёных» логов, а копия месяцами битая. Ставим еженедельный авто-restore на тест-VM плюс pgbackrest verify.

- ✕ **Одна копия бэкапа**
  Только NAS: сгорел или зашифрован — восстанавливать нечем. Правило 3-2-1: NAS + реплика + S3 с immutable Object Lock.

- ✕ **wal_level ниже replica**
  При minimal archive-push молча не строит цепочку для PITR. Ставим wal_level=replica и проверяем pgbackrest --stanza=main check.

- ✕ **Retention не настроен**
  repo пухнет от старых full и WAL, NAS переполняется. Задаём repo-retention-full-type=time=14 и expire в cron.

- ✕ **gz по умолчанию вместо zst**
  compress-type по умолчанию gz — медленно и грузит CPU в рабочее окно. Ставим zst level 3 и process-max по числу ядер.

- ✕ **S3 без Object Lock**
  Шифровальщик с админ-правами удаляет облачную копию. Включаем immutable Object Lock с retention на версионируемом бакете.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- pgBackRest + стэнза, полный бэкап на отдельный NAS, глубина 7–14 дней
- archive_mode=on, архивация WAL, целевой RPO ≤ 15 минут
- Ручная проверка восстановления на тест-сервере раз в месяц

НОРМАЛЬНО

- full/diff/incr по cron, zst level 3, block-инкременты (repo-block)
- Вторая копия в S3 с включённым Object Lock (immutable)
- Еженедельный авто-restore на тест-VM + pgbackrest verify
- Zabbix-триггеры на возраст бэкапа, WAL-lag и место в repo

ХОРОШО

- Три копии 3-2-1: NAS + горячая реплика + S3 immutable
- Потоковая реплика, авто-failover Patroni 4.1 (DCS, maximum_lag_on_failover)
- Barman 3.19 вторым уровнем на базах 500 ГБ+ для каталога и отчётности
- Эскалация провала restore в дежурную смену 24×7

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Репозиторий бэкапов вынесен на отдельный узел/NAS, а не на том базы?
- [ ] archive_mode=on и WAL отгружаются с задержкой ≤60 с (archive-timeout)?
- [ ] Настроены полный + дифференциальный + инкрементальный по cron?
- [ ] Глубина хранения задана явно (repo-retention-full-type=time)?
- [ ] Есть вторая копия в S3 с включённым Object Lock (immutable)?
- [ ] Еженедельный авто-restore на тест-сервере реально проходит?
- [ ] pgbackrest verify сверяет контрольные суммы репозитория?
- [ ] В Zabbix есть триггеры на возраст full/incr, WAL-lag и место в repo?
- [ ] Алерты уходят в Telegram и дублируются дежурной смене?
- [ ] Есть документация по восстановлению, понятная не только автору?

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).

- Разворачиваем pgBackRest 2.58 под 1С за 4–6 ч: стэнза, cron-расписание, первый full, тест-restore
- Настраиваем холодную копию в Яндекс Object Storage с immutable Object Lock и шифрованием
- Подключаем Zabbix-мониторинг бэкапов и Telegram-алерты, дежурная смена 24×7
- Ставим горячую реплику и авто-failover на Patroni там, где простой критичен
- Бесплатный аудит бэкап-схемы 1С: оценка RPO/RTO и письменный отчёт

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

ДОКУМЕНТАЦИЯ И НАШ СТЕК

Техническая база

01 pgBackRest — Configuration Reference (compress-type, repo-retention, repo-block) (pgbackrest.org — 2.58)

02 pgBackRest — User Guide (stanza, archive-push, restore/PITR) (pgbackrest.org — 2.58)

03 Postgres Pro — Приложение K. Настройка для решений 1С (postgrespro.ru — 17)

04 PostgreSQL — Continuous Archiving and PITR (wal_level, archive_command) (postgresql.org — 17)

05 Barman — Documentation (catalog, retention policy) (pgbarman.org — 3.19)

06 Patroni — Documentation (DCS, maximum_lag_on_failover) (patroni.readthedocs.io — 4.1.3)

07 check_pgbackrest — плагин мониторинга (возраст бэкапа, WAL) (github.com/pgstef — 2026)

08 Наш шаблон Zabbix «PG-Backup» + плейбук restore (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.