Документация pg_probackup 3.2.0

Документация pg_probackup 3.2.0

Глава 1. О pg_probackup3

pg_probackup3 — это решение для управления локальным и удалённым резервным копированием и восстановлением кластеров баз данных Postgres Pro. Оно предназначено для регулярного создания резервных копий экземпляра Postgres Pro, позволяющих восстанавливать сервер в случае необходимости. pg_probackup3 поддерживает Postgres Pro и PostgreSQL версии 14 и выше.

В pg_probackup3 входит вся ключевая функциональность предыдущих версий утилиты pg_probackup. Некоторые менее популярные возможности могут отсутствовать, но будут добавлены в будущем.

1.1. Основные функциональные возможности

pg_probackup3 позволяет создавать полные или инкрементальные резервные копии и предоставляет все необходимые возможности для управления ими:

- Полные резервные копии. Содержат все файлы данных, необходимые для восстановления кластера баз данных с нуля.
- Инкрементальные копии. Создаются на уровне страниц и включают только те данные, которые изменились со времени последнего копирования. Это позволяет сэкономить место на диске и создавать копии быстрее, чем при полном копировании. Восстановление инкрементальных копий также осуществляется быстрее, чем воспроизведение файлов WAL. pg_probackup3 поддерживает следующие режимы инкрементального копирования:
  - Разностное копирование. В режиме `DELTA` pg_probackup3 считывает все файлы данных в каталоге данных и копирует только те страницы, которые изменились со времени предыдущего копирования. В этом режиме объём ввода-вывода может равняться объёму при полном резервном копировании.
  - Копирование изменений. В режиме `PTRACK` Postgres Pro отслеживает изменения страниц на лету. Чтобы он работал, не требуется производить непрерывное архивирование. При каждом изменении страницы отношения она помечается в специальной карте `PTRACK`. Это отслеживание привносит небольшие издержки в работу сервера, но значительно ускоряет инкрементальное резервное копирование.
- Обеспечение целостности данных при резервном копировании. pg_probackup3 может производить копирование только работающего экземпляра, а для обеспечения целостности таких копий требуется копировать WAL. Поэтому вне зависимости от выбранного режима копирования (FULL или DELTA), чтобы pg_probackup3 мог получить полноценную копию, нужно выбрать один из следующих *режимов доставки WAL*:
  - STREAM. Такие резервные копии включают все файлы, необходимые для восстановления целостного состояния кластера на момент создания копии. Вне зависимости от того, осуществляется ли непрерывное архивирование, необходимые для восстановления сегменты WAL считываются по протоколу потоковой репликации во время резервного копирования и включаются в состав резервной копии. Поэтому такие резервные копии называются *автономными* или *самодостаточными*.
  - ARCHIVE. В таком режиме целостность копий обеспечивается посредством непрерывного архивирования. Это режим доставки WAL по умолчанию.
- Для управления резервными копиями pg_probackup3 создаёт *каталог резервных копий*. В этом каталоге сохраняются все файлы резервных копий с дополнительной метаинформацией, а также архивы WAL, необходимые для восстановления на момент времени. Список резервных копий в каталоге, а также список всех линий времени WAL и соответствующая метаинформация выводятся в текстовом формате или в формате JSON.
- Вы можете хранить резервные копии разных экземпляров в отдельных подкаталогах одного каталога копий. pg_probackup3 позволяет хранить резервные копии как на локальных дисках, так и в сетевых файловых системах. Поддерживаются следующие протоколы сетевого хранения:
  - NFS версий 3 и 4

    - S3 на базе объектного хранилища MinIO, Amazon S3 и VK Cloud. Данные резервного копирования передаются в S3 и обратно без сохранения в промежуточных хранилищах, что устраняет необходимость в большом временном хранилище.
    - WebDav
    - SAMBA
    - FTP/SFTP
- Политика хранения. Управление архивами WAL и резервными копиями в соответствии с установленными правилами их хранения. Вы можете ограничить хранение резервных копий по времени или их количеству, а также переопределить время жизни (TTL) для избранных копий. Потерявшие актуальность резервные копии могут объединяться или удаляться.
- Многопоточность. Выполнение внутренних процессов команд `backup`, `restore`, `merge`, `delete`, `catchup` и `validate` возможно в несколько параллельных потоков.
- Удалённый режим работы. Выполнение резервного копирования экземпляра Postgres Pro, находящегося в удалённой системе, и удалённое восстановление.
- Архивирование внешних каталогов. Резервное копирование файлов и каталогов, расположенных вне каталога данных Postgres Pro (`PGDATA`), например скриптов, файлов конфигурации, журналов или SQL-дампов.
- Частичное восстановление:
    - Восстановление только избранных баз данных
    - Восстановление на определённый момент времени (PITR, point-in-time recovery)

1.2. Новые возможности pg_probackup3

В сравнении с *pg_probackup*, pg_probackup3 содержит следующие новые функции и улучшения:

- Версионная независимость. Одна и та же версия pg_probackup3 теперь может использоваться с различными версиями Postgres Pro или PostgreSQL, обеспечивая совместимость и адаптивность.
- Интеграция с API. pg_probackup3 предоставляет программные интерфейсы для интеграции, позволяя внешним СРК использовать функциональность pg_probackup3 для создания резервных копий Postgres Pro и централизованного управления резервным копированием.
- Работа без SSH. pg_probackup3 может работать без SSH-соединения, упрощая передачу данных.
- FUSE. В pg_probackup3 реализована команда `fuse`, которая с помощью механизма FUSE (Filesystem in User Space, Файловая система в пользовательском пространстве) обеспечивает работу с базой данных прямо из резервной копии, минуя этап полного восстановления.
- Запуск от имени непривилегированных пользователей. pg_probackup3 может запускаться пользователями, не имеющими прав доступа к PGDATA. Это повышает уровень безопасности и снижает риск возможных ошибок.
- Новый формат резервных копий. Каждая резервная копия теперь хранится в виде единого файла, что облегчает управление и хранение резервных копий, а также предоставляет дополнительные возможности использования.
- pg_probackup3 поддерживает несколько протоколов резервного копирования, доступных в следующих режимах источника данных:
    - BASE. Используется репликационный протокол pg_basebackup.
    - PRO. Используется собственный репликационный протокол pg_probackup3, обеспечивающий безопасную и быструю передачу данных от сервера Postgres Pro.
    - DIRECT. Репликационные протоколы не используются, применяется прямой доступ к файлам данных.

  Подробная информация о режимах источника данных представлена в следующем разделе.
- Объединение цепочек инкрементальных копий. Теперь можно объединять цепочки инкрементальных резервных копий, таким образом сохраняя место в хранилище и организовывая более гибкие регламенты резервного копирования.
- Удобная работа с ленточными хранилищами. Новый формат резервного копирования pg_probackup3 исключает фрагментацию файлов и позволяет настраивать размер резервных копий, обеспечивая оптимальную работу с ленточными системами.

- Полностью переработанное ядро
- Новая архитектура
- Улучшенная производительность

1.3. Режимы источника данных резервного копирования pg_probackup3

Утилита pg_probackup3 поддерживает следующие режимы источника данных: BASE, DIRECT и PRO. Выбранный режим определяет, каким образом будет выполнено подключение к серверу баз данных и как будут получены файлы для создания резервной копии.

В таблице ниже представлены ключевые различия между этими режимами.

Режим источника данных	**Способ передачи данных**	**Требуется библиотека libpgprobackup**	**Проверка**	**Поддержка CFS**	**Требуются дополнительные расширения**
BASE	Через репликационный протокол pg_basebackup	Нет	Да	Нет	Нет
DIRECT	Прямой доступ через файловую систему	Нет	Да	Да	PTRACK (для инкрементального резервного копирования)
PRO	Через собственный репликационный протокол	Да	Да	Да	pgpro_bindump (репликационный протокол), PTRACK (для инкрементального резервного копирования)

Подробное описание каждого режима представлено в разделах ниже.

1.3.1. BASE

В режиме BASE используется стандартный репликационный протокол pg_basebackup. Данные копируются без использования расширенных механизмов отслеживания изменений и сегментации.

Возможности:

- Копирует данные по протоколу репликации с использованием библиотеки `libpq`.
- Совместим с PostgreSQL и не требует дополнительных расширений.

Ограничения:

- Требует доступ к системным функциям.
- Не поддерживает ускоренные инкрементальные механизмы (PTRACK).
- Как правило, медленнее по сравнению с другими режимами при работе с большими объёмами данных.

1.3.2. DIRECT

В режиме DIRECT утилита pg_probackup3 получает доступ к каталогу `PGDATA` напрямую через файловую систему. Репликационный протокол для передачи файлов не используется.

Возможности:

- Использует стандартное подключение к базе данных.
- Обеспечивает проверку целостности данных при передаче.

- Подходит для случаев, когда отсутствует или нежелателен доступ по репликационному протоколу.

Ограничения:

- Требует прямой доступ к файловой системе сервера.
- Требует доступ к системным функциям.
- Для работы с удалёнными серверами необходим доступ к файловой системе через SSH. Для доступа к `PGDATA` SSH не требуется.

1.3.3. PRO

В режиме PRO используется библиотека, включающая в себя логику обработки данных, и собственный протокол резервного копирования. Этот протокол разработан специально для безопасной и быстрой передачи данных от сервера.

Возможности:

- Использует подключение к базе данных с помощью системной библиотеки `libpq`.
- Передаёт данные по собственному репликационному протоколу.
- Поддерживает все типы резервных копий: FULL, DELTA и PTRACK.
- Оптимизирован для производительности и снижения нагрузки на сервер.
- Обеспечивает проверку целостности данных при передаче.

Ограничения:

- Требует установленного плагина pgpro_bindump на стороне сервера.
- Требует наличия библиотеки `libpb3_encoder.so` на сервере. Для pg_probackup3 отдельная библиотека не требуется.
- Может использоваться только с Postgres Pro.
- Для настройки непрерывного архивирования WAL на удалённый сервер требуется подключение по SSH (не требуется для режима STREAM).

1.4. Ограничения pg_probackup3

В настоящее время pg_probackup3 имеет следующие ограничения:

- Любая резервная копия (FULL или DELTA), созданная с помощью pg_probackup3, должна использовать один из следующих режимов доставки WAL: ARCHIVE или STREAM.
- Режим удалённого сервера на платформе Windows не поддерживается.
- На сервере Postgres Pro, где была сделана копия, и на сервере, где она будет восстанавливаться, должны быть одинаковые значения параметров *block_size* и *wal_block_size* и одинаковая основная версия. В зависимости от конфигурации кластера, Postgres Pro может накладывать дополнительные ограничения, например, по архитектуре процессора и версии libc/icu.
- pg_probackup3 поддерживает только Postgres Pro и PostgreSQL версии 14 и новее.
- В Postgres Pro и PostgreSQL 14: если между полной резервной копией и копией в режиме DELTA будет создана новая база данных, инкрементальное резервное копирование будет завершаться ошибкой до создания новой полной резервной копии.
- Создание резервных копий с использованием разных режимов источника данных (`--backup-source`) в рамках одной цепочки запрещено.
- Возможность запуска pg_probackup3 в многопоточном режиме (с использованием параметра `-j`) в текущей версии реализована только для следующих команд: `backup`, `restore`, `merge`, `catchup` и `validate`.
- Путь к каталогу WAL при резервном копировании и восстановлении по умолчанию установлен на `PGDATA/pg_wal`.
- Параметр `--dry-run` поддерживается только для команды `delete`.

- Проверка доступна только для резервных копий целиком.
- Инкрементальное резервное копирование не поддерживается при изменении статуса TDE (включение или отключение). После любого изменения статуса TDE необходимо выполнить полное резервное копирование.
- Операции `merge` запрещены, если цепочки инкрементальных резервных копий содержат резервную копию, фиксирующую изменение статуса TDE. В таких случаях требуется полная резервная копия всего кластера.
- Кластеры с включённым TDE не поддерживают операции `catchup` и `fuse`.

Глава 2. Установка и настройка

2.1. Установка pg_probackup3

Чтобы установить pg_probackup3, выполните следующие действия:

1. Подключите репозиторий пакетов pg_probackup3, предназначенный для вашей операционной системы. Конкретные адреса репозиториев и команды для их подключения в поддерживаемых дистрибутивах Linux вы можете узнать у специалистов поддержки Postgres Pro.
2. Установите пакеты.

   В системах семейства Debian:

   ```
   sudo apt update
   sudo apt install pg-probackup3
   ```

   В системах семейства Red Hat:

   ```
   sudo dnf install pg-probackup3
   ```

   В более ранних версиях систем может потребоваться `yum` вместо `dnf`.
3. Чтобы убедиться в успешной установке, выполните запрос версии pg_probackup3:

   ```
   pg_probackup3 --version
   ```

4. Для использования pg_probackup3 в режиме PRO настройте Postgres Pro следующим образом:

   1. Установите плагин *pgpro_bindump*.
   2. Добавьте следующие параметры в файл `postgresql.conf`:

      ```
      shared_preload_libraries = 'pgpro_bindump'
      wal_level = 'replica'  # or 'logical'
      walsender_plugin_libraries = 'pgpro_bindump'
      ```

   3. Перезапустите экземпляр Postgres Pro.

2.2. Инициализация каталога резервных копий

pg_probackup3 сохраняет все файлы копируемых данных и WAL в соответствующих подкаталогах каталога резервных копий.

Для инициализации каталога резервных копий выполните команду:

```
pg_probackup3 init -B каталог_копий
```

здесь *каталог_копий* — это каталог, предназначенный для резервных копий. Если *каталог_копий* уже существует, он должен быть пустым. В противном случае pg_probackup3 выдаст ошибку.

pg_probackup3 создаёт *каталог_копий* со следующими подкаталогами:

- `wal/` — каталог для файлов WAL.
- `backups/` — каталог для файлов резервных копий.

Проинициализировав каталог резервных копий, вы можете добавить определение копируемого экземпляра.

2.3. Определение копируемого экземпляра

pg_probackup3 может сохранять резервные копии разных кластеров баз данных в одном каталоге резервных копий. Для создания необходимых подкаталогов необходимо добавить представление

резервной копии (экземпляр) каждого кластера баз данных, копию которого вы будете делать, в каталог копий.

Перед созданием нового копируемого экземпляра убедитесь, что pg_probackup3 может подключиться к целевому кластеру баз данных. Чтобы добавить экземпляр, выполните команду:

```
pg_probackup3 add-instance -B каталог_копий -D каталог_данных --instance=имя_экземпляра
 [параметры_ssh]
```

Здесь:

- *каталог_данных* — каталог, содержащий данные кластера, копию которого вы хотите сделать. Для подготовки и использования pg_probackup3 необходимо иметь право записи в этот каталог.
- *имя_экземпляра* — это имя подкаталогов, в которых будут храниться файлы копируемых данных и WAL для этого кластера.
- *параметры_ssh* должны задаваться дополнительно, если *каталог_данных* располагается удалённо.

pg_probackup3 создаёт подкаталоги *имя_экземпляра* в каталогах `backups/` и `wal/` внутри каталога резервных копий. Каталог `backups/`*имя_экземпляра* содержит файл конфигурации `pg_probackup3.conf` с параметрами pg_probackup3, относящимися к данному экземпляру копии. Если этой команде передать параметры SSH, они будут добавлены в `pg_probackup3.conf`.

Более подробно тонкая настройка pg_probackup3 описывается в Разделе 2.4.

Пользователь, запускающий pg_probackup3, должен иметь полный доступ к *каталогу_копий* и как минимум доступ на чтение всего содержимого *каталога_данных*. Если вы зададите путь к каталогу копий в переменной окружения `BACKUP_PATH`, соответствующий параметр в командах pg_probackup3 можно не указывать.

> **Примечание**
>
> Рекомендуется использовать параметр *--allow-group-access*, чтобы выполнить копирование мог любой пользователь ОС, включённый в группу владельца кластера. В этом случае пользователь должен иметь права на чтение каталога кластера.

2.4. Настройка pg_probackup3

Проинициализировав каталог резервных копий и добавив определение копируемого экземпляра, вы можете использовать файл `pg_probackup3.conf` в каталоге *каталог_копий*`/backups/`*имя_экземпляра* для тонкой настройки конфигурации pg_probackup3 этого экземпляра.

Например, команда backup работает через обычное подключение Postgres Pro. Чтобы каждый раз не задавать параметры подключения в командной строке, вы можете определить их в файле конфигурации `pg_probackup3.conf` с помощью команды set-config.

Изначально `pg_probackup3.conf` содержит следующие параметры:

- `PGDATA` — путь к каталогу данных кластера, который будет копироваться.
- `system-identifier` — уникальный идентификатор экземпляра Postgres Pro.

Вы можете дополнительно установить параметры хранения копий, ведения журнала и сжатия, используя команду `set-config`:

```
pg_probackup3 set-config -B каталог_копий --instance=имя_экземпляра
[--external-dirs=путь_внешнего_каталога] [параметры_подключения] [параметры_сжатия]
 [параметры_хранения] [параметры_журнала]
```

Чтобы просмотреть текущие параметры, выполните эту команду:

```
pg_probackup3 show-config -B каталог_копий --instance=имя_экземпляра
```

Параметры, заданные в `pg_probackup3.conf`, можно переопределить, задавая соответствующие переменные окружения или параметры командной строки при вызове команд pg_probackup3.

> **Примечание**
>
> Редактировать `pg_probackup3.conf` вручную **не рекомендуется**. Изменения файла конфигурации с помощью команды set-config исключает возможность случайных опечаток.

2.5. Указание параметров подключения

Если вы определите параметры подключения в файле конфигурации `pg_probackup3.conf`, вы можете не указывать их во всех последующих командах pg_probackup3. Однако если установлены соответствующие переменные окружения, они имеют больший приоритет. Параметры, заданные в командной строке, переопределяют как переменные окружения, так и параметры в файле конфигурации.

Если не задано ничего, используются значения по умолчанию. В частности, pg_probackup3 пытается подключиться к Unix-сокету локального сервера (или к `localhost` в Windows), а в качестве имени базы данных и имени пользователя выбирает значение переменной окружения `PGUSER` либо имя текущего пользователя ОС.

2.6. Настройка кластера баз данных

Хотя pg_probackup3 можно использовать от имени суперпользователя, рекомендуется создать отдельную роль с минимальными правами, необходимыми для выбранной стратегии копирования. В этих инструкциях по настройке такой ролью служит роль `backup`.

Из соображений безопасности для выполнения следующих конфигурационных SQL-запросов рекомендуется использовать отдельную базу данных.

```
postgres=# CREATE DATABASE backupdb;
postgres=# \c backupdb
```

Для выполнения backup роль `backup` должна иметь следующие разрешения на сервере Postgres Pro (только в базе данных, к которой **производится подключение**).

```
BEGIN;
CREATE ROLE backup WITH LOGIN REPLICATION;
GRANT USAGE ON SCHEMA pg_catalog TO backup;
GRANT SELECT ON TABLE pg_catalog.pg_tablespace TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.current_setting(text) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.set_config(text, text, boolean) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_is_in_recovery() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_backup_start(text, boolean) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_backup_stop(boolean) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_create_restore_point(text) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_switch_wal() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_last_wal_replay_lsn() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current_snapshot() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_snapshot_xmax(txid_snapshot) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_control_checkpoint() TO backup;
COMMIT;
```

В файле *pg_hba.conf* разрешите подключение к кластеру баз данных пользователю с ролью `backup`.

> **Примечание**
>
> Доступ к каталогу данных `PGDATA` не требуется для резервного копирования в режимах PRO и BASE, но обязателен для режима DIRECT. По умолчанию используется режим PRO.

В зависимости от того, будете ли вы делать самодостаточные или архивные копии, конфигурация кластера Postgres Pro будет разной (об особенностях рассказывается ниже). Чтобы запустить pg_probackup3 в удалённом режиме или создать резервную копию PTRACK, требуется дополнительная настройка.

Подробнее об этом рассказывается в подразделах Настройка потокового резервного копирования, Настройка непрерывного архивирования WAL, Настройка подключения по SSH и Настройка копирования PTRACK.

2.7. Настройка потокового резервного копирования

Чтобы настроить кластер для потокового резервного копирования, выполните следующие действия:

- Если роль `backup` не существует, создайте её с правом `REPLICATION` при Настройке кластера базы данных:

  ```
  CREATE ROLE backup WITH LOGIN REPLICATION;
  ```

- Если роль `backup` уже существует, дайте ей право `REPLICATION`:

  ```
  ALTER ROLE backup WITH REPLICATION;
  ```

- В файле *pg_hba.conf* разрешите выполнение репликации для роли `backup`.
- Установите для параметра *max_wal_senders* достаточно большое значение, предусматривающее минимум одно подключение для процесса резервного копирования.
- Задайте для параметра *wal_level* значение выше `minimal`.

Если вы намерены выполнять восстановление на момент времени с потоковыми копиями, вам тем не менее надо будет настроить архивирование WAL, как описано в подразделе Настройка непрерывного архивирования WAL.

После этих подготовительных действий вы можете делать резервные копии в режимах FULL, DELTA и PTRACK, используя потоковую доставку WAL.

> **Примечание**
>
> Если вы намерены использовать *.pgpass* для прохождения аутентификации при выполнении копирования в потоковом режиме, файл .pgpass должен содержать учётные данные для подключения к базе данных `replication`. Например: pghost:5432:replication:backup_user:my_strong_password

2.8. Настройка непрерывного архивирования WAL

Для восстановления на момент времени и создания резервных копий с использованием режима доставки WAL ARCHIVE должно осуществляться *непрерывное архивирование WAL*. Чтобы настроить непрерывное архивирование, выполните следующие действия:

- Задайте для параметра *wal_level* значение выше `minimal`.
- Если вы настраиваете резервное копирование на ведущем сервере, параметр *archive_mode* должен иметь значение `on` или `always`.
- Установите параметр *archive_command*:

```
archive_command = '"путь_инсталляции/pg_probackup3" archive-push -B "каталог_копий"
 --instance=имя_экземпляра --wal-file-name=%f [параметры_ssh]'
```

Здесь *путь_инсталляции* — путь к каталогу установленной версии pg_probackup3, которую вы хотите использовать, *каталог_копий* и *имя_экземпляра* должны указывать на уже проинициализированный для данного кластера БД копируемый экземпляр, а параметры SSH должны задаваться только в случае расположения архива WAL в удалённой системе. Подробнее все возможные параметры `archive-push` рассматриваются в archive-push.

После этих подготовительных действий вы сможете использовать режим доставки WAL ARCHIVE, а также выполнять восстановление на момент времени.

Вы можете просмотреть текущее состояние архива WAL, воспользовавшись командой show. За подробностями обратитесь к Подразделу 3.3.2.

> **Примечание**
>
> Вместо использования команды pg_probackup3 archive-push вы можете воспользоваться любым другим средством, при условии, что в процессе непрерывного архивирования сегменты WAL будут попадать в каталог *каталог_копий*/wal/*имя_экземпляра*. Для сжатия сегментов, если в нём есть потребность, должен использоваться алгоритм `gzip`, а сжатые файлы сегментов должны иметь расширение `.gz`.

> **Примечание**
>
> Организовать непрерывное архивирование можно не только с помощью параметров `archive_mode` и `archive_command`, но и применяя утилиту *pg_receivewal*. В этом случае аргумент pg_receivewal `-D` *каталог* должен указывать на каталог *каталог_копий*/wal/*имя_экземпляра*. Программа pg_probackup3 принимает сжатые WAL, которые может сохранять pg_receivewal. Стратегию архивирования «без потерь» можно реализовать только с использованием pg_receivewal.

2.9. Настройка частичного восстановления

Если вы планируете производить частичное восстановление с помощью параметров `--db-exclude-name` и `--db-include-name`, выполните следующее дополнительное действие:

- Дайте право на чтение `pg_catalog.pg_database` роли `backup` в той базе данных, через которую **выполняется подключение** к серверу Postgres Pro:

```
GRANT SELECT ON TABLE pg_catalog.pg_database TO backup;
```

2.10. Настройка копирования PTRACK

Режим копирования PTRACK может использоваться только в инсталляциях Postgres Pro Standard и Postgres Pro Enterprise или в ванильных патчах PostgreSQL.

Если вы намерены использовать режим копирования PTRACK, выполните описанные далее действия.

> **Примечание**
>
> Для роли, которая будет выполнять резервное копирование в режиме PTRACK (роль `backup` в примерах ниже), требуемые права доступа указаны в Разделе 2.6. Роль должна иметь права только в той базе данных, которая используется для подключения к серверу Postgres Pro.

1. Добавьте `ptrack` в переменную `shared_preload_libraries` в файле `postgresql.conf`:

   ```
   shared_preload_libraries = 'ptrack'
   ```

2. Чтобы включить отслеживание изменений страниц, задайте для параметра `ptrack.map_size` положительное целое значение и перезапустите сервер.

   Для оптимальной производительности рекомендуется задавать `ptrack.map_size` равным *`N / 1024`*, где *`N`* — объём кластера Postgres Pro в мегабайтах. Если этот параметр будет иметь меньшее значение, это увеличит вероятность наложения информации разных блоков в карте PTRACK, что повлечёт ложные положительные результаты при определении изменённых блоков и, как следствие, увеличение размера инкрементальной копии, так как в копию будут попадать и фактически неизменённые блоки. Использовать значения `ptrack.map_size`, превышающие 1024, не рекомендуется, хотя PTRACK поддерживает большие карты.

   > **Примечание**
   >
   > В случае изменения значения `ptrack.map_size` ранее созданный файл карты PTRACK очищается, и отслеживание новых блоков начинается с начала. Таким образом, прежде чем создавать новые инкрементальные копии в режиме PTRACK после изменения `ptrack.map_size` необходимо сделать новую полную копию кластера.

3. Создайте расширение PTRACK:

   ```
   CREATE EXTENSION ptrack;
   ```

2.11. Настройка подключения по SSH

pg_probackup3 поддерживает удалённый режим, в котором резервное копирование и архивирование WAL могут выполняться удалённо. Для работы в удалённом режиме pg_probackup3 использует SSH-соединение между сервером Postgres Pro (где запущен pg_probackup3) и сервером хранилища резервных копий. Это позволяет управлять резервными копиями на удалённом сервере так, как если бы они хранились локально.

pg_probackup3 может хранить и читать файлы резервных копий и метаданные на SSH-сервере по SFTP-протоколу. Эта схема работы похожа на S3.

Если вы хотите использовать pg_probackup3 в удалённом режиме через SSH, настройте подключение по SSH к серверу без указания пароля с помощью открытого и закрытого ключей: открытый ключ необходимо разместить на сервере, закрытый — на клиенте.

За подробностями о параметрах подключения обратитесь к разделу Параметры SSH.

pg_probackup3 работает в удалённом режиме по протоколу SSH следующим образом:

- В удалённом режиме поддерживаются все команды.
- Для работы в удалённом режиме не требуется, чтобы программа pg_probackup3 была установлена и в удалённой системе.

Можно указать параметры SSH в файле конфигурации с помощью `--config-file`. За подробностями обратитесь к разделу Общие параметры.

2.12. Настройка подключения к хранилищу S3

pg_probackup3 поддерживает интерфейс S3 для хранения резервных копий. Данные резервного копирования передаются в S3 и обратно без сохранения в промежуточных хранилищах, что устраняет необходимость в большом временном хранилище.

> **Примечание**
>
> Функциональность S3 доступна только при использовании pg_probackup3 с Postgres Pro Enterprise.

Если вы хотите использовать pg_probackup3 с интерфейсом S3, выполните следующие действия:

- Создайте для будущих копий в хранилище S3 бакет с уникальным и осмысленным именем.
- Создайте ключи ACCESS_KEY и SECRET_ACCESS_KEY, которые будут использоваться для безопасного подключения вместо имени и пароля пользователя.
- Для взаимодействия pg_probackup3 с сервером S3 задайте переменные окружения, требуемые для подключения к вашему серверу S3. Например:

```
export PG_PROBACKUP_S3_HOST=127.0.0.1
export PG_PROBACKUP_S3_PORT=9000
export PG_PROBACKUP_S3_REGION=ru-msk
export PG_PROBACKUP_S3_BUCKET_NAME=test1
export PG_PROBACKUP_S3_ACCESS_KEY=admin
export PG_PROBACKUP_S3_SECRET_ACCESS_KEY=password
export PG_PROBACKUP_S3_HTTPS=ON
```

В качестве альтернативы можно указать параметры сервера S3 в файле конфигурации или с помощью параметров командной строки. За подробностями обратитесь к описанию параметра `--config-file` в разделе Общие параметры, а также к разделу Параметры S3.

Если `--s3=minio`, стоит указывать параметры сервера S3, как описано в разделе Параметры S3.

Можно указать следующие переменные окружения:

`PG_PROBACKUP_S3_HOST`

Адрес сервера S3. Можно также указать номер порта через двоеточие. Если номер порта не указан в строке адреса, используется значение `PG_PROBACKUP_S3_PORT`. Добавляйте двоеточие только при указании номера порта.

Например:

```
export PG_PROBACKUP_S3_PORT=80
export PG_PROBACKUP_S3_HOST="127.0.0.1:9000"
```

В этом примере для адреса «127.0.0.1» явно указан порт 9000 и он будет использоваться вместо значения 80, указанного в `PG_PROBACKUP_S3_PORT`.

`PG_PROBACKUP_S3_PORT`

Порт сервера S3.

`PG_PROBACKUP_S3_REGION`

Регион сервера S3. По умолчанию — `us-east-1`.

`PG_PROBACKUP_S3_BUCKET_NAME`

Имя бакета на сервере S3.

`PG_PROBACKUP_S3_ACCESS_KEY`
`PG_PROBACKUP_S3_SECRET_ACCESS_KEY`

Ключи безопасности для доступа к серверу S3.

`PG_PROBACKUP_S3_HTTPS`

Какой протокол использовать. Поддерживаются следующие значения:

- `ON` или `HTTPS` — использовать HTTPS
- Иное — использовать HTTP

`PG_PROBACKUP_S3_BUFFER_SIZE`

Размер буфера чтения/записи для организации связи с S3, в МиБ. По умолчанию — `16`.

`PG_PROBACKUP_S3_RETRIES`

Максимальное количество попыток выполнения запроса к S3 в случае сбоя. По умолчанию — `3`.

`PG_PROBACKUP_S3_TIMEOUT`

Максимальное время выполнения HTTP-запроса к серверу S3 в секундах. По умолчанию — `300`.

`PG_PROBACKUP_S3_IGNORE_CERT_VER`

Не проверять сертификат узла и узла-партнёра. По умолчанию: `OFF`.

`PG_PROBACKUP_S3_CA_CERTIFICATE`

Указать путь к каталогу файла с сертификатом от доверенного центра сертификации (ЦА).

`PG_PROBACKUP_S3_CA_PATH`

Указать каталог, в котором должны храниться сертификаты доверенного ЦС.

`PG_PROBACKUP_S3_CLIENT_CERT`

Установить клиентский сертификат SSL.

`PG_PROBACKUP_S3_CLIENT_KEY`

Установить файл закрытого ключа для клиентских сертификатов TLS и SSL.

2.12.1. Необходимые разрешения для S3

Для ключа доступа, используемого pg_probackup3, при отключённом версионировании должны быть предоставлены следующие минимальные разрешения для целевого бакета S3:

- для команды init:

```
s3:GetBucketVersioning
s3:ListBucket
```

- для команд add-instance, set-config и set-backup:

```
s3:GetBucketVersioning
s3:ListBucket
s3:PutObject
```

- для команды del-instance:

```
s3:GetBucketVersioning
s3:ListBucket
```

```
s3:DeleteObject
```

- для команд backup, archive-push и file-map:

```
s3:ListBucket
s3:PutObject
s3:GetBucketVersioning
s3:AbortMultipartUpload
s3:GetObject
```

- для команд restore, fuse, show, show-config и send-backup:

```
s3:GetBucketVersioning
s3:GetObject
s3:ListBucket
```

- для команды validate:

```
s3:GetBucketVersioning
s3:GetObject
s3:ListBucket
s3:PutObject
```

- для команд merge и retention:

```
s3:ListBucket
s3:PutObject
s3:GetBucketVersioning
s3:AbortMultipartUpload
s3:GetObject
s3:DeleteObject
```

- для команды delete:

```
s3:GetBucketVersioning
s3:GetObject
s3:ListBucket
s3:DeleteObject
```

При включённом версионировании требуются следующие дополнительные разрешения:

- `s3:ListBucketVersions` для всех команд, которым требуется `s3:ListBucket`
- `s3:DeleteObjectVersion` для всех команд, которым требуется `s3:DeleteObject`

Глава 3. Сценарии использования

3.1. Создание резервной копии

Чтобы создать резервную копию, выполните следующую команду:

```
pg_probackup3 backup -B каталог_копий --instance=имя_экземпляра -b режим_копирования -s источник_копирования -i ид_резервной_копии
```

Здесь *режим_копирования* может принимать следующие значения: FULL, DELTA и PTRACK.

А *источник_копирования* может принимать одно из следующих значений: DIRECT, BASE и PRO.

> **Предупреждение**
>
> В режимах источника данных BASE и DIRECT не поддерживается CFS.

> **Примечание**
>
> В режиме источника данных BASE поддерживается резервное копирование только в режимах FULL и DELTA.

Некоторые параметры можно не указывать, в зависимости от цели пользователя:

- Если *режим_копирования* не указан, по умолчанию используется режим FULL.
- PRO — значение по умолчанию для *источник_копирования*.
- Если идентификатор резервной копии не указан явно в теле запроса, *ид_резервной_копии* примет значение даты и времени, когда резервная копия была создана.
- Если идентификатор резервной копии указан и включает в себя путь к каталогу, *каталог_копий* и *имя_экземпляра* можно не указывать. Например: -i /mnt/ramdisk/backups/2.backup.
- Если путь к каталогу данных не указан ни через *каталог_копий*, ни через параметр --backup-id, текущий каталог будет использоваться в качестве каталога по умолчанию.
- Если опустить идентификатор **родительской** копии при выполнении инкрементального копирования, pg_probackup3 использует последнюю подходящую копию из цепочки копий. Если родительскую резервную копию подобрать не удастся, процесс копирования завершится ошибкой.
- Если указан параметр --from-full, инкрементальная резервная копия будет создана из последней родительской полной копии.

3.1.1. Режим ARCHIVE

Режим ARCHIVE используется в качестве режима доставки WAL по умолчанию.

Чтобы создать полную копию в режиме ARCHIVE доставки WAL, выполните:

```
pg_probackup3 backup -B каталог_копий --instance=имя_экземпляра -b FULL
```

Резервное копирование ARCHIVE требует организации непрерывного архивирования, посредством которого считываются сегменты WAL, требующиеся для восстановления согласованного состояния кластера на момент создания копии.

3.1.2. Режим STREAM

Чтобы сделать полную резервную копию в потоковом (STREAM) режиме, добавьте флаг --stream к команде, показанной выше:

```
pg_probackup3 backup -B каталог_копий --instance=имя_экземпляра -b FULL --stream
```

В отличие от копий ARCHIVE, копии типа STREAM включают все сегменты WAL, необходимые для восстановления согласованного состояния кластера на момент создания копии.

В процессе выполнения команды backup pg_probackup3 передаёт файлы WAL, содержащие записи от `Start LSN` до `Stop LSN`, в файл с резервной копией.

Даже если вы используете непрерывное архивирование, копирование в режиме STREAM может быть полезно в следующих случаях:

- Копии типа STREAM могут быть восстановлены на сервере, не имеющем файлового доступа к архиву WAL.
- Копии типа STREAM позволяют восстановить состояние кластера на тот момент времени, для которого уже нет файлов WAL.

3.1.3. Внешние каталоги

Чтобы заархивировать каталог, размещённый вне каталога данных, воспользуйтесь необязательным параметром `--external-dirs`, в котором можно задать путь к нужному каталогу. Если вы хотите заархивировать несколько внешних каталогов, их пути нужно разделить двоеточиями в Linux.

Например, чтобы в системе Linux включить каталоги `/etc/dir1` и `/etc/dir2` в полную копию экземпляра *`имя_экземпляра`*, которая будет размещаться в *`каталоге_копий`*, выполните:

```
pg_probackup3 backup -B каталог_копий --instance=имя_экземпляра -b FULL --external-dirs=/etc/dir1:/etc/dir2
```

Например, чтобы включить каталоги `C:\dir1` и `C:\dir2` в полную копию, в Windows нужно выполнить:

```
pg_probackup3 backup -B каталог_копий --instance=имя_экземпляра -b FULL --external-dirs=C:\dir1;C:\dir2
```

Для каждого внешнего каталога pg_probackup3 создаёт отдельный подкаталог в каталоге резервной копии и рекурсивно копирует в него всё содержимое внешнего каталога. Так как внешние каталоги, попадающие в разные резервные копии, не обязательно должны быть одинаковыми, при восстановлении кластера из инкрементальной копии будут восстановлены только те каталоги, которые относятся именно к ней. Внешние каталоги, сохранённые в предыдущих копиях, восстановлены не будут.

Чтобы нужные каталоги включались в каждую резервную копию вашего кластера, их список можно сохранить в файле конфигурации `pg_probackup3.conf`, воспользовавшись командой set-config с ключом `--external-dirs`.

> **Примечание**
>
> Внешние каталоги не поддерживаются в режиме BASE.

3.2. Восстановление кластера

3.2.1. Полное восстановление

Чтобы восстановить кластер баз данных из резервной копии, выполните команду restore как минимум со следующими параметрами:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра -i ид_резервной_копии
```

Здесь:

- *`каталог_копий`* — каталог, в котором хранятся все файлы резервных копий и метаданные.
- *`имя_экземпляра`* — имя экземпляра резервной копии кластера, который будет восстановлен.
- *`ид_резервной_копии`* определяет, из какой резервной копии будет восстановлен кластер.

Если вы восстанавливаете копию ARCHIVE или выполняете восстановление PITR, pg_probackup3 создаёт файл конфигурации восстановления после копирования всех файлов данных в целевой каталог. Этот файл включает необходимые для восстановления параметры, за исключением пароля, заданного в *primary_conninfo*; если он требуется, его нужно дополнительно задать вручную или воспользоваться параметром `--primary-conninfo`. pg_probackup3 сохраняет эти параметры в файле `probackup_recovery.conf` в каталоге данных и подключает его в `postgresql.auto.conf`.

Если восстанавливалась копия типа STREAM, восстановление завершается сразу, и кластер возвращается в согласованное состояние на момент времени, в который была сделана резервная копия. Для копий типа ARCHIVE Postgres Pro воспроизводит все имеющиеся в архиве сегменты WAL, в результате чего восстанавливается самое последнее состояние кластера на текущей линии времени. Это поведение можно изменить, определив параметры точки восстановления для команды `restore` как описывается в Подразделе 3.2.4.

Если кластер, подлежащий восстановлению, содержит табличные пространства, pg_probackup3 по умолчанию восстанавливает их в исходные расположения. Чтобы сменить расположения табличных пространств, воспользуйтесь параметром `--tablespace-mapping`/`-T`. В противном случае при восстановлении кластера на том же сервере произойдёт ошибка, если эти табличные пространства будут использоваться, так как восстанавливаемые данные нужно будет записать в те же каталоги.

Используя параметр `--tablespace-mapping`/`-T`, вы должны задать абсолютные пути к старому и новому каталогу табличного пространства. Если путь содержит знак равно (=), экранируйте его обратной косой чертой. Данный параметр может указываться неоднократно для перемещения нескольких табличных пространств. Например:

```
pg_probackup3 restore -B каталог_копий --instance имя_экземпляра -D каталог_данных -j 4
 -i ид_резервной_копии -T каталог_табл_пространства1=новый_каталог_табл_пространства1 -
T каталог_табл_пространства2=новый_каталог_табл_пространства2
```

3.2.2. Инкрементальное восстановление

Скорость восстановления резервной копии можно значительно увеличить, заменяя в существующем каталоге данных Postgres Pro только некорректные или изменённые страницы. Это можно реализовать, используя параметры инкрементального восстановления с командой restore.

> **Примечание**
>
> В текущей версии pg_probackup3 инкрементальное восстановление доступно только в режиме PRO. Поддержка для режимов BASE и DIRECT будет реализована в следующих выпусках.

Чтобы восстановить кластер баз данных из резервной копии в инкрементальном режиме, выполните команду restore со следующими параметрами:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра -D каталог_данных -
I инкрементальный_режим
```

Здесь *инкрементальный_режим* может быть следующим:

- `CHECKSUM` — прочитать все файлы данных в целевом каталоге, проверить заголовок и контрольную сумму каждой страницы и заменить только некорректные страницы, а также страницы, в которых контрольная сумма и LSN отличаются от значений в соответствующей странице в копии. Это самый простой и надёжный инкрементальный режим. Его рекомендуется использовать по умолчанию.
- `LSN` — прочитать файл *pg_control* в каталоге данных, получить из него значения REDO LSN и REDO TLI, позволяющие определить точку в истории (точку сдвига), в которой состояние каталога данных сдвинулись с цепочки резервных копий. Если точка сдвига находится за пределами истории резервных копий, восстановление прерывается. Если же точка сдвига достижима, прочитываются все файлы данных в каталоге данных, проверяется заголовок и контрольная сумма на каждой странице, а затем заменяются только страницы с неверной контрольной суммой или с LSN, превышающим позицию точки сдвига. В этом режиме обеспечивается уве-

личенная скорость по сравнению с режимом CHECKSUM, но требуется выполнение двух условий. Во-первых, в целевом каталоге должны быть включены *контрольные суммы*, чтобы не произошло повреждения данных из-за изменения вспомогательных битов. Это условие будет проверяться перед инкрементальным восстановлением — в случае отсутствия контрольных сумм операция прерывается. Во-вторых, необходима синхронность файла `pg_control` с состоянием каталога данных. Это условие нельзя проверить в начале восстановления, так что гарантировать правильность информации в `pg_control` должен пользователь. Поэтому использовать режим LSN рекомендуется не во всех ситуациях, например, он противопоказан, когда файл `pg_control` повреждён или его содержимому нельзя доверять: после выполнения `pg_resetxlog`, после восстановления из копии без процедуры воспроизведения журнала и т. д.

- `NONE` — обычное восстановление без инкрементальных оптимизаций.

Вне зависимости от выбранного инкрементального режима, pg_probackup3 будет проверять, что с заданным целевым каталогом не работает процесс `postmaster` и значения `system-identifier` в целевом экземпляре и копии совпадают.

Пример использования инкрементального восстановления с табличным пространством в режиме CHECKSUM:

```
========================================================================================================
Instance Version ID      End time                  Mode  WAL Mode TLI Duration Data WAL
 Zalg Zratio Start LSN  Stop LSN   Status
========================================================================================================
inst     17      1-full  2025-08-05 16:14:10+0700 FULL  STREAM   1   1s       47MB -
 none 1.46   0/A3000028 0/A3000160 OK
inst     17      1-delta 2025-08-05 16:14:33+0700 DELTA STREAM   1   1s       21MB -
 none 3.18   0/A5000028 0/A5000160 OK
inst     17      2-delta 2025-08-05 16:14:37+0700 DELTA STREAM   1   0ms      21MB -
 none 3.18   0/A6000028 0/A6000160 OK

backup_user@backup_host:~$ ./pg_probackup3 restore -D /mnt/node -B /mnt/b3b --
instance=inst -I checksum --backup-id 2-delta -T /mnt/ts1=/mnt/ts2 --threads=1
[2025-08-05 16:20:38.061117] [264669] [0x79693fc1a4c0] [info] command: ./pg_probackup3
 restore -D /mnt/node -B /mnt/backups --instance=inst -I checksum --backup-id 2-delta -
T /mnt/ts1=/mnt/ts2 --threads=1
[2025-08-05 16:20:38.061244] [264669] [0x79693fc1a4c0] [info] execute command:
 'restore', instance 'inst', backup_id '2-delta'
[2025-08-05 16:20:38.061855] [264669] [0x79693fc1a4c0] [info] Start validate 2-
delta ...
[2025-08-05 16:20:38.065368] [264669] [0x79693d5a66c0] [info] Validating backup 1-full
 chunk #0 out of 8
[2025-08-05 16:20:38.088840] [264669] [0x79693d5a66c0] [info] Validating backup 1-full
 chunk #1 out of 8
[2025-08-05 16:20:38.112253] [264669] [0x79693d5a66c0] [info] Validating backup 1-full
 chunk #2 out of 8
[2025-08-05 16:20:38.193395] [264669] [0x79693d5a66c0] [info] Validating backup 1-full
 chunk #3 out of 8
[2025-08-05 16:20:38.213982] [264669] [0x79693d5a66c0] [info] Validating backup 1-full
 chunk #4 out of 8
[2025-08-05 16:20:38.235532] [264669] [0x79693d5a66c0] [info] Validating backup 1-full
 chunk #5 out of 8
[2025-08-05 16:20:38.256768] [264669] [0x79693d5a66c0] [info] Validating backup 1-full
 chunk #6 out of 8
[2025-08-05 16:20:38.278902] [264669] [0x79693d5a66c0] [info] Validating backup 1-full
 chunk #7 out of 8
[2025-08-05 16:20:38.300380] [264669] [0x79693fc1a4c0] [info] Validate time 235ms
[2025-08-05 16:20:38.301628] [264669] [0x79693d5a66c0] [info] Validating backup 1-delta
 chunk #0 out of 8
```

```
[2025-08-05 16:20:38.305281] [264669] [0x79693d5a66c0] [info] Validating backup 1-delta
 chunk #1 out of 8
[2025-08-05 16:20:38.368129] [264669] [0x79693d5a66c0] [info] Validating backup 1-delta
 chunk #2 out of 8
[2025-08-05 16:20:38.371719] [264669] [0x79693d5a66c0] [info] Validating backup 1-delta
 chunk #3 out of 8
[2025-08-05 16:20:38.375494] [264669] [0x79693d5a66c0] [info] Validating backup 1-delta
 chunk #4 out of 8
[2025-08-05 16:20:38.379185] [264669] [0x79693d5a66c0] [info] Validating backup 1-delta
 chunk #5 out of 8
[2025-08-05 16:20:38.383215] [264669] [0x79693d5a66c0] [info] Validating backup 1-delta
 chunk #6 out of 8
[2025-08-05 16:20:38.386733] [264669] [0x79693d5a66c0] [info] Validating backup 1-delta
 chunk #7 out of 8
[2025-08-05 16:20:38.390220] [264669] [0x79693fc1a4c0] [info] Validate time 89ms
[2025-08-05 16:20:38.391428] [264669] [0x79693d5a66c0] [info] Validating backup 2-delta
 chunk #0 out of 8
[2025-08-05 16:20:38.395259] [264669] [0x79693d5a66c0] [info] Validating backup 2-delta
 chunk #1 out of 8
[2025-08-05 16:20:38.399093] [264669] [0x79693d5a66c0] [info] Validating backup 2-delta
 chunk #2 out of 8
[2025-08-05 16:20:38.402872] [264669] [0x79693d5a66c0] [info] Validating backup 2-delta
 chunk #3 out of 8
[2025-08-05 16:20:38.405935] [264669] [0x79693d5a66c0] [info] Validating backup 2-delta
 chunk #4 out of 8
[2025-08-05 16:20:38.468891] [264669] [0x79693d5a66c0] [info] Validating backup 2-delta
 chunk #5 out of 8
[2025-08-05 16:20:38.472336] [264669] [0x79693d5a66c0] [info] Validating backup 2-delta
 chunk #6 out of 8
[2025-08-05 16:20:38.475977] [264669] [0x79693d5a66c0] [info] Validating backup 2-delta
 chunk #7 out of 8
[2025-08-05 16:20:38.479477] [264669] [0x79693fc1a4c0] [info] Validate time 88ms
[2025-08-05 16:20:38.479859] [264669] [0x79693fc1a4c0] [info] INFO: Backup 2-delta is
 valid
[2025-08-05 16:20:38.479992] [264669] [0x79693fc1a4c0] [info] Start restore of backup
 2-delta into /mnt/node
[2025-08-05 16:20:38.481239] [264669] [0x79693fc1a4c0] [info] Backup 1-delta is chosen
 as shiftpoint, its Stop LSN will be used as shift LSN
[2025-08-05 16:20:38.483006] [264669] [0x79693d5a66c0] [info] Restoring 2-delta chunk
 #0 out of 8
[2025-08-05 16:20:38.532219] [264669] [0x79693d5a66c0] [info] Restoring 2-delta chunk
 #1 out of 8
[2025-08-05 16:20:38.569631] [264669] [0x79693d5a66c0] [info] Restoring 2-delta chunk
 #2 out of 8
[2025-08-05 16:20:38.608792] [264669] [0x79693d5a66c0] [info] Restoring 2-delta chunk
 #3 out of 8
[2025-08-05 16:20:38.633202] [264669] [0x79693d5a66c0] [info] Restoring 2-delta chunk
 #4 out of 8
[2025-08-05 16:20:38.733030] [264669] [0x79693d5a66c0] [info] Restoring 2-delta chunk
 #5 out of 8
[2025-08-05 16:20:38.764890] [264669] [0x79693d5a66c0] [info] Restoring 2-delta chunk
 #6 out of 8
[2025-08-05 16:20:38.804717] [264669] [0x79693d5a66c0] [info] Restoring 2-delta chunk
 #7 out of 8
[2025-08-05 16:20:38.839108] [264669] [0x79693fc1a4c0] [info] Restore time 356ms
[2025-08-05 16:20:38.839256] [264669] [0x79693fc1a4c0] [info] Removing redundant files
 in destination directory
```

```
[2025-08-05 16:20:38.848171] [264669] [0x79693d5a66c0] [info] Restoring 1-delta chunk
 #0 out of 8
[2025-08-05 16:20:38.860342] [264669] [0x79693d5a66c0] [info] Restoring 1-delta chunk
 #1 out of 8
[2025-08-05 16:20:38.918134] [264669] [0x79693d5a66c0] [info] Restoring 1-delta chunk
 #2 out of 8
[2025-08-05 16:20:38.929649] [264669] [0x79693d5a66c0] [info] Restoring 1-delta chunk
 #3 out of 8
[2025-08-05 16:20:38.942811] [264669] [0x79693d5a66c0] [info] Restoring 1-delta chunk
 #4 out of 8
[2025-08-05 16:20:38.954785] [264669] [0x79693d5a66c0] [info] Restoring 1-delta chunk
 #5 out of 8
[2025-08-05 16:20:38.970253] [264669] [0x79693d5a66c0] [info] Restoring 1-delta chunk
 #6 out of 8
[2025-08-05 16:20:38.983111] [264669] [0x79693d5a66c0] [info] Restoring 1-delta chunk
 #7 out of 8
[2025-08-05 16:20:38.995516] [264669] [0x79693fc1a4c0] [info] Restore time 148ms
[2025-08-05 16:20:38.997560] [264669] [0x79693d5a66c0] [info] Restoring 1-full chunk #0
 out of 8
[2025-08-05 16:20:39.032484] [264669] [0x79693d5a66c0] [info] Restoring 1-full chunk #1
 out of 8
[2025-08-05 16:20:39.062668] [264669] [0x79693d5a66c0] [info] Restoring 1-full chunk #2
 out of 8
[2025-08-05 16:20:39.134805] [264669] [0x79693d5a66c0] [info] Restoring 1-full chunk #3
 out of 8
[2025-08-05 16:20:39.160183] [264669] [0x79693d5a66c0] [info] Restoring 1-full chunk #4
 out of 8
[2025-08-05 16:20:39.189998] [264669] [0x79693d5a66c0] [info] Restoring 1-full chunk #5
 out of 8
[2025-08-05 16:20:39.219022] [264669] [0x79693d5a66c0] [info] Restoring 1-full chunk #6
 out of 8
[2025-08-05 16:20:39.249562] [264669] [0x79693d5a66c0] [info] Restoring 1-full chunk #7
 out of 8
[2025-08-05 16:20:39.279275] [264669] [0x79693fc1a4c0] [info] Restore time 282ms
[2025-08-05 16:20:39.280742] [264669] [0x79693fc1a4c0] [info] INFO: Restore of backup
 2-delta completed.
```

3.2.3. Частичное восстановление

Можно восстанавливать определённые базы данных с помощью параметров частичного восстановления с командой restore. Ниже представлены все возможные варианты частичного восстановления.

3.2.3.1. Частичное восстановление по имени

Если вы настроили частичное восстановление прежде, чем создавать резервные копии, вы можете восстанавливать отдельные базы данных, используя параметры `--db-include-name` и `--db-exclude-name`.

Чтобы восстановить только определённые базы данных, выполните команду restore со следующими параметрами:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра --db-include-
name=имя_бд
```

Параметр `--db-include-name` можно указывать многократно. Например, чтобы восстановить только базы `db1` и `db2`, выполните следующую команду:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра --db-include-
name=db1 --db-include-name=db2
```

Чтобы исключить одну или несколько баз из числа восстанавливаемых, используйте параметр `--db-exclude-name`:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра --db-exclude-name=имя_бд
```

Параметр `--db-exclude-name` можно указывать многократно. Например, чтобы исключить из числа восстанавливаемых только базы `db1` и `db2`, выполните следующую команду:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра --db-exclude-name=db1 --db-exclude-name=db2
```

> **Примечание**
>
> После успешного запуска кластера Postgres Pro восстановленные определения исключённых баз данных можно удалить с помощью команды `DROP DATABASE`.

Чтобы максимально быстро разделить один кластер, содержащий несколько баз данных, на разные кластеры, можно выполнить частичное восстановление исходного кластера в виде ведомого, передав ключ `--restore-as-replica` для определённых баз данных.

> **Примечание**
>
> Базы `template0` и `template1` восстанавливаются всегда.

> **Предупреждение**
>
> Параметры `--db-exclude-name` и `--db-include-name` использовать вместе нельзя.

3.2.3.2. Частичное резервное копирование и восстановление по OID

Можно копировать и восстанавливать определённые базы данных *без дополнительной подготовки* с помощью параметров `--db-include-oid` и `--db-exclude-oid`.

> **Примечание**
>
> Частичное резервное копирование по OID в настоящее время поддерживается только в режиме DIRECT.

Чтобы восстановить только определённые базы данных, выполните команду restore со следующими параметрами:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра --db-include-oid=dboid
```

> **Примечание**
>
> После частичного резервного копирования можно выполнить нечастичное восстановление, в результате чего будет создан экземпляр только с включёнными базами данных.

При восстановлении обрабатываются только WAL-записи для включённых баз данных, остальные игнорируются, что ускоряет операцию.

Параметр `--db-include-oid` можно указывать многократно. Например, чтобы восстановить только базы `db1` и `db2` с OID `dboid1` и `dboid2`, соответственно, выполните следующую команду:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра --db-include-
oid=dboid1 --db-include-oid=dboid2
```

Чтобы исключить одну или несколько баз из числа восстанавливаемых, используйте параметр `--db-exclude-oid`:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра --db-exclude-oid=dboid
```

Параметр `--db-exclude-oid` можно указывать многократно. Например, чтобы исключить из числа восстанавливаемых только базы `db1` и `db2` с OID `dboid1` и `dboid2`, соответственно, выполните следующую команду:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра --db-exclude-
oid=dboid1 --db-exclude-oid=dboid2
```

> **Примечание**
>
> После успешного запуска кластера Postgres Pro восстановленные определения исключённых баз данных можно удалить с помощью команды `DROP DATABASE`.

Чтобы максимально быстро разделить один кластер, содержащий несколько баз данных, на разные кластеры, можно выполнить частичное восстановление исходного кластера в виде ведомого, передав ключ `--restore-as-replica` для определённых баз данных.

> **Примечание**
>
> Базы `template0` и `template1` восстанавливаются всегда.

> **Предупреждение**
>
> Параметры `--db-exclude-oid` и `--db-include-oid` использовать вместе нельзя.

3.2.4. Выполнение восстановления на момент времени (PITR)

Вы можете восстановить состояние кластера на любой момент времени (до заданной точки восстановления), используя с командой restore параметры точки восстановления.

Перед началом восстановления на момент времени убедитесь, что выполнены следующие условия:

- Вы настроили непрерывную архивацию WAL с параметром `wal_level`, установленным в значение `replica`, до создания резервных копий.
- Осуществляется регулярное создание полных и инкрементальных резервных копий с помощью pg_probackup3.

Для восстановления на момент времени может использоваться копия типа STREAM или ARCHIVE при условии, что WAL-архив содержит непрерывную последовательность сегментов от момента создания резервной копии до целевой точки восстановления.

Для восстановления состояния кластера на определённый момент времени выполните приведённые ниже действия, подставляя свои значения.

1. Остановите сервер:

```
pg_ctl stop -D /path/to/database/data
```

2. Удалите каталог данных (PGDATA):

```
rm -rf /path/to/database/data
mv -f /path/to/database/logs/pg_logfile.log /tmp/demo/logs/pg_logfile.log.old
```

   Это необязательный шаг, так как для восстановления может использоваться другой каталог.

   Рекомендуется сохранять файл с журналами в целях диагностики.

3. Выполните команду `restore` со следующими параметрами:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра --recovery-target-
time="2024-04-10 18:18:26+03"
--recovery-target-action=promote
```

4. Запустите сервер:

```
pg_ctl start -D /path/to/database/data
```

По завершении восстановления новый экземпляр будет автоматически повышен до ведущего, и будет создана новая линия времени (timeline).

Другие поддерживаемые методы PITR описаны в разделе «Параметры точки восстановления».

3.3. Управление каталогом резервных копий

С помощью pg_probackup3 вы можете управлять резервными копиями в командной строке:

- Используйте команду `show` для просмотра информации о резервной копии. За подробностями обратитесь к разделу Просмотр информации о резервных копиях.
- Для просмотра информации об архиве WAL выполните команду `show` с параметром `--archive`. Подробнее об этом рассказывается в разделе Просмотр оглавления архива WAL.
- В зависимости от того, требуется ли объединить инкрементальную резервную копию с её родительской полной копией или цепочку инкрементальных копий, можно использовать команду `merge` с параметром `--backup-id`(`-i`) либо `--backup-id` вместе с `--merge-from-id`/`--merge-interval` соответственно. За подробностями обратитесь к разделу Объединение резервных копий.
- Для удаления ненужных резервных копий выполните команду `delete`. Подробнее о процессе рассказывается в разделе Удаление резервных копий.

3.3.1. Просмотр информации о резервных копиях

Чтобы просмотреть список существующих копий для каждого экземпляра, выполните команду:

```
pg_probackup3 show -B каталог_копий
```

pg_probackup3 выводит список всех имеющихся резервных копий. Например:

```
BACKUP INSTANCE 'dev', version 3
=============================================================================================
Instance Version ID       End time                 Mode  WAL Mode TLI Duration Data WAL
 Zalg Zratio Start LSN  Stop LSN   Status
=============================================================================================
dev      17      1-full   2024-12-10 14:51:34+0000 FULL  STREAM   1   1s       38MB -
 none 1.00   0/A000028  0/A000138  OK
dev      17      1-delta  2024-12-10 14:52:02+0000 DELTA STREAM   1            11MB -
 none 1.00   0/D000028  0/D000180  OK
dev      17      2-delta  2024-12-10 14:52:28+0000 DELTA STREAM   1            22MB -
 none 1.00   0/10000028 0/10000138 OK
```

```
dev      17      1a-full  2024-12-10 14:54:10+0000 FULL  ARCHIVE  1   1s       75MB -
 none 1.00   0/12000028 0/12000138 OK
dev      17      1a-delta 2024-12-10 14:54:32+0000 DELTA ARCHIVE  1            17MB -
 none 1.00   0/14000028 0/14000138 OK
```

Для каждой копии выдаются следующие сведения:

- `Instance` — имя экземпляра.
- `Version` — базовая версия Postgres Pro.
- `ID` — идентификатор резервной копии.
- `End time` — время окончания резервного копирования.
- `Mode` — режим, в котором была сделана копия. Возможные значения: `FULL` (полная), `DELTA` (инкрементальная), `PTRACK` (копирование изменений).
- `WAL Mode` — режим доставки WAL. Возможные значения: `STREAM` (потоковый) и `ARCHIVE` (архивный).
- `TLI` — идентификаторы линии времени текущей копии и её родителя.
- `Duration` — время, за которое была выполнена данная копия.
- `Data` — объём файлов данных в этой копии. Это значение не включает в себя размер файлов WAL. Для копий, сделанных в режиме STREAM, общий размер можно рассчитать, сложив значения `Data` и `WAL`.
- `WAL` — размер несжатых файлов WAL, которые должны быть применены в процессе восстановления копии для достижения согласованного состояния.
- `compress-alg` — алгоритм сжатия, используемый при получении резервной копии. Возможные значения: `zlib`, `lz4`, `zstd` и `none` (сжатие не производилось).
- `Zratio` — коэффициент сжатия, вычисленный как отношение «uncompressed-bytes» (объём несжатых данных в байтах) к «data-bytes» (итоговый объём данных).
- `Start LSN` — последовательный номер в журнале WAL, соответствующий началу процесса копирования. С этой позиции накатываются изменения (REDO) в процессе восстановления Postgres Pro.
- `Stop LSN` — последовательный номер в журнале WAL, соответствующий окончанию процесса копирования. Это позиция точки согласованности при восстановлении Postgres Pro.
- `Status` — состояние резервной копии. Возможные варианты:
  - `OK` — резервная копия сделана и пригодна к использованию.
  - `DONE` — резервная копия сделана, но не проверена.
  - `RUNNING` — резервное копирование выполняется.
  - `MERGING` — резервная копия объединяется.
  - `MERGED` — файлы резервной копии были успешно обработаны в процессе объединения копий, но её метаданные ещё изменяются. Это состояние могут иметь только полные резервные копии.
  - `DELETING` — файлы резервной копии удаляются.
  - `CORRUPT` — некоторые файлы резервной копии повреждены.
  - `ERROR` — резервное копирование было прервано из-за неожиданной ошибки.
  - `ORPHAN` — резервная копия непригодна к использованию, так как её родительская копия испорчена или отсутствует.
  - `HIDDEN_FOR_TEST` — скрипт теста пометил копию как несуществующую. (Собственно pg_probackup3 никогда не устанавливает это состояние.)

Восстановить кластер из копии можно только для копий с состоянием `OK` или `DONE`.

Чтобы получить более подробную информацию о копии, укажите в команде `show` её идентификатор:

```
pg_probackup show -B каталог_копий --instance=имя_экземпляра -i ид_резервной_копии
```

Пример вывода:

```
# Backup 2-delta information.
backup_id=2-delta
parent_backup_id=1-delta
backup_mode=delta
tli=1
start_lsn=268435496
stop_lsn=268435768
# start-time 2024-12-10 14:52:28+0000
start_time=1733842348
# end-time 2024-12-10 14:52:28+0000
end_time=1733842348
recovery-time=0
data-bytes=22986632
uncompressed-bytes=22986632
compress-alg=none
compress-level=1
server-version=170001
min_xid=0
min_multixact=0
backup_source=pro
primary_conninfo=user=garbuz reusepass=1 channel_binding=prefer host=localhost
 port=5432 sslmode=prefer sslcompression=0 sslcertmode=allow sslsni=1
 ssl_min_protocol_version=TLSv1.2 gssencmode=disable krbsrvname=postgres
 gssdelegation=0 target_session_attrs=any target_server_type=any hostorder=sequential
 load_balance_hosts=disable
stream=true
program-version=3.0.0
block-size=8192
xlog-block-size=8192
status = OK
```

В расширенном выводе представлены дополнительные атрибуты:

- `compress-alg` — алгоритм сжатия, используемый при получении резервной копии. Возможные значения: `zlib`, `lz4`, `zstd` и `none` (сжатие не производилось).
- `compress-level` — уровень сжатия, применяемый в процессе резервного копирования.
- `block-size` — значение параметра *block_size*, установленное в кластере Postgres Pro в начале копирования.
- `checksum-version` — признак включения параметра *data_checksums* в исходном кластере Postgres Pro. Возможные значения: `1`, `0`.
- `program-version` — полная версия программы pg_probackup3, которая создала эту копию.
- `start-time` — время начала резервного копирования.
- `end-time` — время окончания резервного копирования.
- `end-validation-time` — время окончания проверки резервной копии.
- `expire-time` — время, когда закреплённая копия может быть ликвидирована в соответствии с политикой хранения. Этот атрибут имеется только у закреплённых копий.
- `uncompressed-bytes` — размер файлов данных до добавления заголовков страниц и сжатия. В случае использования сжатия оценить его эффективность можно, сопоставив объём `uncompressed-bytes` (байтов несжатых данных) с `data-bytes` (байтов данных).
- `data-bytes` — размер файлов данных Postgres Pro на момент копирования. Эффективность инкрементального метода копирования можно оценить, сопоставив объём `data-bytes` (байтов в PGDATA) с `uncompressed-bytes` (байтов несжатых данных).
- `recovery-xid` — идентификатор транзакции, соответствующей моменту окончания резервного копирования.

- `parent-backup-id` — идентификатор родительской копии. Определён только для инкрементальных копий.
- `primary_conninfo` — параметры подключения libpq, с использованием которых производилось подключение к кластеру Postgres Pro для получения этой резервной копии. Пароль в эти параметры не включается.
- `note` — текстовое примечание, связанное с копией.
- `content-crc` — контрольная сумма файла `backup_content.control`, рассчитанная по алгоритму CRC32. Она позволяет выявить повреждение метаданных копии.

Можно использовать параметр `--format=tree`, чтобы посмотреть список резервных копий в виде дерева:

```
pg_probackup3 show -B каталог_копий --format=tree
```

Вывод будет выглядеть вот так:

```
BACKUP INSTANCE 'dev', version 3

├── 1-full

│   └── 1-delta

│       └── 2-delta

└── 1a-full

    └── 1a-delta
```

Вы также можете получить подробную информацию о резервной копии в формате JSON:

```
pg_probackup3 show -B каталог_копий --instance=имя_экземпляра --format=json -i
 backup_id
```

Пример вывода:

```
[
    {
        "instance": "dev",
        "backups": [
            {
                "id": "2-delta",
                "parent-backup-id": "1-delta",
                "status": "OK",
                "start-time": "2024-12-10 14:52:28+0000",
                "end-time": "2024-12-10 14:52:28+0000",
                "backup-mode": "DELTA",
                "wal": "STREAM",
                "block-size": 8192,
                "xlog-block-size": 8192,
                "program-version": "3.0.0",
                "server-version": 17,
                "current-tli": 1,
                "start-lsn": "0/10000028",
                "stop-lsn": "0/10000138",
                "data-bytes": 22986632,
                "uncompressed-bytes": 22986632,
                "wal-bytes": 0,
                "compress-alg": "none",
                "compress-level": 1,
                "min-xid": 0,
                "min-multixact": 0,
```

```
                "backup-source": "pro"
            }
        ]
    }
]
```

3.3.2. Просмотр оглавления архива WAL

Чтобы получить информацию об архиве WAL для каждого экземпляра, выполните команду:

```
pg_probackup3 show -B каталог_копий [--instance=имя_экземпляра] --archive
```

pg_probackup3 выводит список всех имеющихся файлов WAL, сгруппированных по линиям времени. Например:

```
BACKUP INSTANCE 'dev', version 3
=============================================================================================
TLI Parent TLI Switchpoint Min Segno                Max Segno                N segments
 Size Zratio N backups Status
=============================================================================================
1              0/0         000000010000000000000001 000000010000000000000006 6
 96MB 1.17   1         OK
```

Для каждой линии времени выдаются следующие сведения:

- `TLI` — идентификатор линии времени.
- `Parent TLI` — идентификатор линии времени, от которой была ответвлена данная.
- `Switchpoint` — LSN момента, когда эта линия времени ответвилась от родительской.
- `Min Segno` — первый сегмент WAL, относящийся к этой линии времени.
- `Max Segno` — последний сегмент WAL, относящийся к этой линии времени.
- `N segments` — количество сегментов WAL, относящихся к этой линии времени.
- `Size` — объём, который занимают файлы на диске.
- `Zalg` — алгоритм сжатия, используемый при получении резервной копии. Возможные значения: `zlib`, `lz4`, `zstd`, `none`(сжатие не производилось).
- `Zratio` — коэффициент сжатия, вычисляемый по формуле `N segments` * *`wal_segment_size`* * *`wal_block_size`* / `Size`.
- `N backups` — число копий, относящихся к этой линии времени. Для получения подробных сведений об этих копиях воспользуйтесь форматом JSON.
- `Status` — состояние архива WAL для этой линии времени. Возможные значения:
  - `OK` — в архиве присутствуют все сегменты WAL между `Min Segno` и `Max Segno`.
  - `DEGRADED` — отсутствуют некоторые сегменты WAL между `Min Segno` и `Max Segno`. Понять, какие файлы утрачены, можно, посмотрев этот отчёт в формате JSON. Такой статус может возникать, если несколько файлов WAL (в середине последовательности) были удалены командой delete с параметром `--delete-wal` в соответствии с политикой хранения. Данный статус не влияет на корректность восстановления, но восстановить кластер до некоторых точек восстановления может быть невозможно.

Чтобы получить более подробную информацию об архиве WAL в формате JSON, выполните команду:

```
pg_probackup3 show -B каталог_копий [--instance=имя_экземпляра] --archive --format=json
```

Пример вывода:

```
[
    {
        "instance": "dev",
        "version": "3",
        "timelines": [
            {
```

```
                "tli": 1,
                "parent-tli": 0,
                "switchpoint": "0/0",
                "min-segno": "000000010000000000000001",
                "max-segno": "000000010000000000000006",
                "n-segments": 6,
                "size": 100663615,
                "zratio": 1.17,
                "status": "OK",
                "backups": [
                    {
                        "id": "1-full",
                        "status": "OK",
                        "start-time": "2025-02-11 14:22:16+0000",
                        "end-time": "2025-02-11 14:22:16+0000",
                        "backup-mode": "FULL",
                        "wal": "STREAM",
                        "block-size": 8192,
                        "xlog-block-size": 8192,
                        "program-version": "3.0.0",
                        "server-version": 17,
                        "current-tli": 1,
                        "start-lsn": "0/5000028",
                        "stop-lsn": "0/5000128",
                        "data-bytes": 60748163,
                        "uncompressed-bytes": 60748163,
                        "wal-bytes": 0,
                        "compress-alg": "none",
                        "compress-level": 1,
                        "min-xid": 0,
                        "min-multixact": 0,
                        "backup-source": "pro"
                    }
                ]
            }
        ]
    }
]
```


В основном в этом формате представлены те же поля, что и в текстовом формате, с некоторыми исключениями:

- Размер выражается в байтах.
- Атрибут `closest-backup-id` содержит идентификатор самой последней доступной копии, принадлежащей к одной из предыдущих линий времени. Вы можете использовать эту копию для восстановления на момент, относящийся к этой линии времени. Если такой копии не существует, данный атрибут будет пустым.
- В массиве `lost-segments` представлены интервалы отсутствующих сегментов на линиях времени в непригодном состоянии (`DEGRADED`). На линиях времени в целостном состоянии `OK` массив `lost-segments` пуст.
- В массиве `backups` перечисляются все резервные копии, относящиеся к данной линии времени. Если к линии времени не относятся резервные копии, этот массив пуст.

3.3.3. Объединение резервных копий

По мере того как вы будете делать новые и новые инкрементальные копии, общий размер каталога резервных копий может существенно увеличиться. Для экономии места на диске вы можете объединить инкрементальные копии с родительскими полными копиями или объединить цепочки инкрементальных копий.

Во время объединения создаётся новая резервная копия, в которую затем войдут все объединяемые копии. Все лишние копии будут удалены *только после* того, как объединение успешно завершено. Такой процесс требует дополнительного места на диске, но помогает избежать потери данных в случае ошибок или системного сбоя.

> **Примечание**
>
> Если несколько резервных копий относятся к одной и той же родительской, такие копии не удаляются после объединения, и место на диске не освобождается.

Чтобы объединить инкрементальную копию с полной родительской, выполните команду `merge`, передав ей идентификатор копии самой последней резервной копии, подлежащей объединению:

```
pg_probackup3 merge -B каталог_копий --instance=имя_экземпляра -i ид_резервной_копии
```

Эта команда объединяет копии, относящиеся к одной цепочке инкрементальных копий. Если выбирается полная копия, она будет объединена с первой инкрементальной копией после неё. Если выбрана инкрементальная копия, она будет объединена с родительской полной копией, включая все инкрементальные копии между ними. После выполнения команды полная копия содержит все объединённые данные, а инкрементальные копии удаляются как ненужные. Таким образом, операция объединения по сути равнозначна созданию новой полной копии с удалением всех устаревших копий, но выполняется она быстрее, особенно с большими объёмами данных, и не нагружает подсистему ввода-вывода и сеть (если pg_probackup3 работает в удалённом режиме).

Чтобы объединить цепочку инкрементальных копий, не включающую полную резервную копию, укажите идентификаторы первой и последней копий в цепочке:

```
pg_probackup3 merge -B каталог_копий --instance=имя_экземпляра --merge-from-
id=объединить_от -i ид_резервной_копии
```

Или задайте идентификатор первой копии, а также интервал времени (в часах), чтобы объединить все копии, созданные за это время:

```
pg_probackup3 merge -B каталог_копий --instance=имя_экземпляра -i ид_резервной_копии --
merge-interval=интервал_объединения
```

Перед объединением pg_probackup3 проверяет все задействуемые резервные копии, чтобы удостовериться в их целостности. Вы можете проверить текущее состояние резервной копии, передав её идентификатор команде show.

```
pg_probackup show -B каталог_копий --instance=имя_экземпляра -i ид_резервной_копии
```

Если процесс объединения ещё не закончен, вы увидите состояние `MERGING`. Для полных копий также можно увидеть состояние `MERGED` в процессе изменения метаданных на последнем этапе объединения. В случае прерывания операции объединения она может быть перезапущена.

> **Предупреждение**
>
> Не рекомендуется принудительно завершать операции объединения — это может нарушить дальнейшую работу команды `merge` и проверку резервных копий.

3.3.4. Удаление резервных копий

Для удаления резервной копии, ставшей ненужной, выполните команду:

```
pg_probackup3 delete -B каталог_копий --instance=имя_экземпляра -i ид_резервной_копии
```

Эта команда удалит резервную копию с заданным *ид_резервной_копии* вместе со всеми инкрементальными копиями, которые от неё зависят (если таковые найдутся). Таким образом вы можете

удалить некоторые последние инкрементальные копии, сохранив предыдущую полную копию и некоторые следующие за ней инкрементальные копии.

Прежде чем удалять резервные копии, вы можете выполнить команду `delete` с параметром `--dry-run` и получить в результате состояние всех имеющихся копий в соответствии с текущей политикой хранения; никакие необратимые действия при этом выполняться не будут.

Для удаления всех копий с определённым состоянием, воспользуйтесь параметром `--status`:

```
pg_probackup delete3 -B каталог_копий --instance=имя_экземпляра --status=ERROR
```

При удалении копий по критерию состояния установленные политики сохранения не учитываются.

3.4. Использование pg_probackup3 в удалённом режиме

pg_probackup3 поддерживает работу в удалённом режиме. Данные для резервного копирования можно получать от сервера Postgres Pro удалённо по встроенному репликационному протоколу, а резервные копии можно сохранять в хранилище S3 или на сервер SSH по протоколу SFTP.

В зависимости от типа операции можно выбрать следующие сценарии:

- Чтобы подключиться к серверу Postgres Pro для создания резервной копии, используйте стандартные параметры подключения: `--pghost`, `--pgport`, `--pgdatabase` и `--pguser`.
- Для сохранения, восстановления и проверки резервных копий в хранилище S3 укажите параметры S3. За подробностями обратитесь к Разделу 2.12.
- Для работы с резервными копиями на сервере SSH по протоколу SFTP укажите параметры SSH. Обратитесь к Разделу 2.11 за подробностями.

Таким образом и каталог резервных копий, и экземпляр Postgres Pro, для которого делается резервная копия, могут располагаться на удалённых серверах.

Типичная схема его использования выглядит так:

- В системе резервного копирования настройте pg_probackup3, как описывается в подразделе Установка и настройка. Для команд init, add-instance, backup и set-config необходимо задать параметры подключения, указывающие на сервер с экземпляром Postgres Pro.
- Если вы хотите в удалённом режиме использовать доставку WAL в режиме ARCHIVE, настройте непрерывное архивирование WAL с сервера баз данных, как описано в подразделе Настройка непрерывного архивирования WAL. Для этого в команде archive-push требуется задать параметры SSH, указывающие на сервер, где находится каталог резервных копий.

> **Примечание**
>
> Для работы в режиме непрерывного архивирования файлов WAL исполняемые файлы pg_probackup3 должны находиться на сервере Postgres Pro. Вызов pg_probackup3 с командами `archive-push` и `archive-get` осуществляется сервером Postgres Pro в соответствии с параметрами *`archive_command`* и *`restore_command`* в его конфигурации.

Например, чтобы создать полную архивную **локальную** копию кластера Postgres Pro, работающего в удалённой системе с адресом `192.168.0.2`, подключившись к серверу через порт `2302` с именем пользователя `postgres`, выполните:

```
pg_probackup3 backup -B каталог_копий --instance=имя_экземпляра -b FULL --
pguser=postgres --pghost=192.168.0.2 --pgport=2302
```

Чтобы создать резервную копию на сервере SSH с адресом `10.0.3.77` для пользователя `ubuntu`, выполните:

```
pg_probackup3 backup -B каталог_копий --instance=имя_экземпляра -b FULL --
pguser=postgres --pghost=192.168.0.2 --pgport=2302 --remote-host=10.0.3.77 --remote-
user=ubuntu
```

Скрипт для создания резервной копии в хранилище S3:

```
export PG_PROBACKUP_S3_PORT=9000
export PG_PROBACKUP_S3_ACCESS_KEY=admin
export PG_PROBACKUP_S3_SECRET_ACCESS_KEY=password
export PG_PROBACKUP_S3_REGION=us-west-2
export PG_PROBACKUP_S3_HOST=10.0.3.77
export PG_PROBACKUP_S3_BUCKET_NAME=test
pg_probackup3 backup -B каталог_копий --instance=имя_экземпляра -b FULL --
pguser=postgres --pghost=192.168.0.2 --pgport=2302 --s3
```

Чтобы настроить непрерывное архивирование WAL на сервере SSH с адресом `10.0.3.77`, выполните следующую команду:

```
pg_probackup3 archive-push -B каталог_копий --instance=имя_экземпляра --wal-file-name=
%f --compress-algorithm=zstd --remote-host=10.0.3.77 --remote-user=ubuntu
```

Чтобы настроить непрерывное архивирование WAL в хранилище S3, выполните следующую команду:

```
pg_probackup3 archive-push -B каталог_копий --instance=имя_экземпляра --compress-
algorithm=zstd --wal-file-name=%f --s3 --config-file=s3.config
```

При этом параметры подключения должны быть сохранены в файле `s3.config` в каталоге данных Postgres Pro.

3.5. Удалённое восстановление

Удалённое восстановление в pg_probackup3 — это функциональность, позволяющая восстанавливать резервные копии напрямую на удалённый сервер без необходимости копировать файлы архива вручную. Такой подход существенно сокращает время восстановления и количество ручных операций при аварийном восстановлении, миграции или автоматическом развёртывании.

Функциональность удалённого восстановления состоит из следующих основных компонентов:

- Команда send-backup в утилите pg_probackup3 для передачи данных через указанный порт на удалённый сервер с использованием многопоточности.
- Утилита pgpro_backupstream, запускаемая вручную на удалённом сервере, для получения, распаковки и восстановления данных.

Для выполнения удалённого восстановления необходимо соблюдать следующие условия:

- В локальной системе:
  - Должны быть установлены утилита pg_probackup3 и библиотека `libpgprobackup`.
  - Должен быть обеспечен доступ к каталогу резервных копий.
- В удалённой системе:
  - Должна быть установлена утилита pgpro_backupstream.

> **Примечание**
>
> pgpro_backupstream запускается вручную.

  - Каталог `PGDATA` должен быть пустым и открытым для записи.

> **Примечание**
>
> На данный момент инкрементальное восстановление в удалённом режиме не поддерживается.

- Выбранный порт должен быть открыт и доступен для входящих подключений.

Чтобы восстановить экземпляр на удалённый сервер, выполните следующие шаги:

1. В локальной системе выполните команду send-backup через утилиту pg_probackup3, чтобы отправить данные резервной копии на удалённый сервер по указанному порту:

   ```
   pg_probackup3 send-backup -B каталог_копий --instance=имя_экземпляра -
   i ид_резервной_копии -p порт -h сервер [--no-merge]
   ```

   Флаг `--no-merge` отключает слияние цепочки резервных копий перед передачей данных. В противном случае цепочка резервных копий будет автоматически объединена во временный файл, который удалится после завершения передачи.

2. В удалённой системе запустите команду `restore` через утилиту pgpro_backupstream для приёма и восстановления данных:

   ```
   pgpro_backupstream restore -D путь_для_восстановления [-p порт]
   ```

   Если порт не указан, используется STDIN.

> **Важно**
>
> Запустите утилиту pgpro_backupstream в удалённой системе **до** того, как начать передачу данных с помощью команды send-backup.

3.6. Запуск pg_probackup3 в параллельных потоках

Команды backup, restore, merge, delete, catchup и validate могут выполняться в несколько параллельных потоков. Это может существенно ускорять работу pg_probackup3 при наличии достаточных ресурсов (ядер процессора, производительности дисковой подсистемы и сети).

Параллельным выполнением управляют ключи командной строки `-j`/`--threads`, `--num-write-threads` и `--num-validate-threads`. Эти параметры должны быть неотрицательными целыми числами.

Если эти параметры не указаны или имеют нулевое значение, pg_probackup3 по умолчанию использует количество ядер процессора. Если определить количество ядер не удастся, будет использоваться один поток.

Когда параметры `--num-write-threads` и `--num-validate-threads` указаны, они переопределяют значение `-j`.

Если запрошенное количество потоков превышает системное ограничение (например, указанное в `/proc/sys/kernel/threads-max`), будет выведено предупреждение и вместо запрошенного значения будет использовано системное ограничение. Если ограничение не обнаружено, будет применено значение, указанное пользователем.

В режиме PRO количество потоков чтения должно быть меньше значения серверного параметра `max_wal_senders`.

Например, чтобы запустить резервное копирование в четыре параллельных потока, выполните:

```
pg_probackup3 backup -B каталог_копий --instance=имя_экземпляра -b FULL -j 4
```

> **Примечание**
>
> Восстановление происходит в параллельном режиме только на этапе копирования данных из каталога копий в каталог данных кластера. При запуске сервера Postgres Pro он должен будет воспроизвести записи из WAL, а это может происходить только последовательно.

3.7. Монтирование каталога резервных копий с помощью FUSE

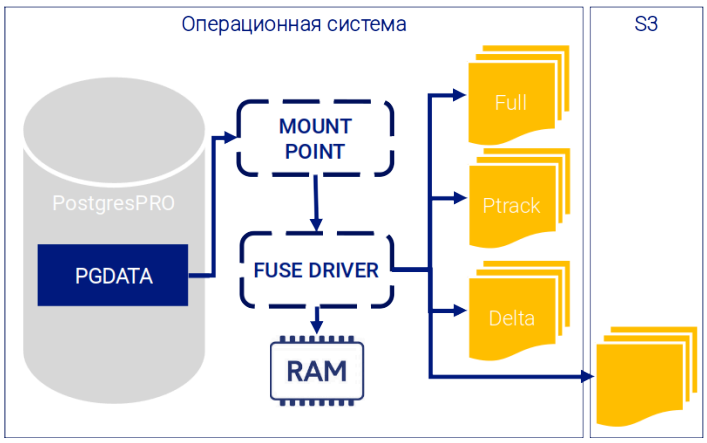
pg_probackup3 позволяет запускать экземпляр базы данных напрямую из резервной копии, проверять и восстанавливать отдельные данные без необходимости полного восстановления, используя команду fuse.

Эта команда задействует механизм FUSE (Filesystem in User Space, Файловая система в пользовательском пространстве), монтируя виртуальное представление каталога резервных копий. Postgres Pro взаимодействует с этим смонтированным каталогом как с реальным каталогом `PGDATA`, при этом все запросы к файловой системе перенаправляются к файлам резервной копии. Так как изменения пишутся в кеш, а не в резервную копию, исходная резервная копия остаётся неизменной, а все операции выполняются в режиме *только для чтения*.

> **Примечание**
>
> Операции `fuse` доступны только в версиях и редакциях Postgres Pro Enterprise.

> **Предупреждение**
>
> Механизм FUSE не предоставляет всех возможностей, необходимых для работы в производственной среде. Поэтому рекомендуется использовать `fuse` только для разовых операций. Производительность операций `fuse` может значительно варьироваться в зависимости от используемого типа хранилища (S3, NFS, SSD).

Рисунок 3.1. Механизм FUSE pg_probackup3



Основные сценарии использования команды fuse:

- Восстановить удалённые данные с определённой даты (например, с помощью pg_dump).
- Проверить данные на определённый момент времени.
- Обеспечить среду, идентичную рабочей, в режиме только для чтения, когда полное восстановление заняло бы слишком много времени.

- Выполнить откат на определённый момент времени для тестирования и отладки сбоев приложения.
- Генерировать отчёты на основе резервной копии без затрат на полное восстановление, в качестве альтернативы репликации.
- Поддерживать пользовательские базы данных на FUSE без необходимости полного восстановления большого объёма данных.

> **Примечание**
>
> В ALT Linux пользователь, запускающий pg_probackup3, должен быть членом группы `fuse`. За подробностями обратитесь к *документации ALT Linux*.

Чтобы использовать смонтированную резервную копию как `PGDATA`, укажите *`путь_монтирования`* в качестве пути для параметра `-D` при запуске Postgres Pro командой `pg_ctl start`.

Чтобы обеспечить достаточное дисковое пространство для операций FUSE, укажите пользовательский каталог для хранения кеша через параметр `--cache-dir`.

Монтирование цепочки резервных копий требует наличия заранее созданных файлов сопоставления. Чтобы включить создание файлов сопоставления, используйте один из следующих методов:

- Используйте параметр `--with-file-map` с командой backup или merge.
- Выполните команду file-map для существующей цепочки резервных копий. Обратите внимание, что это заменит все ранее созданные файлы сопоставления.

Для целей автоматизации файловую систему FUSE можно запустить в фоновом режиме с помощью параметра `--detach`.

Чтобы демонтировать файловую систему FUSE, используйте команду `fuse` с параметром `--unmount`:

```
pg_probackup3 fuse --unmount --mnt-path=путь_монтирования
```

За подробной информацией о команде `fuse` и её параметрах обратитесь к Подразделу «Команды».

3.8. Проверка целостности данных

3.8.1. Проверка страниц

Когда в кластере БД включены *контрольные суммы*, pg_probackup3 использует их для проверки целостности файлов данных в процессе резервного копирования. При чтении каждой страницы pg_probackup3 проверяет, совпадает ли вычисленная сумма с контрольной суммой, хранящейся в заголовке страницы. Это гарантирует, что в кластере Postgres Pro и самой резервной копии не содержатся испорченные страницы. Заметьте, что pg_probackup3 читает файлы данных непосредственно из файловой системы, поэтому при активной записи в момент копирования возможны ложные выявления некорректных контрольных сумм из-за частичной записи. В случае несовпадения контрольной суммы страница считывается повторно, и контрольная сумма проверяется ещё раз.

Страница признаётся испорченной, если проверка контрольной суммы не проходит более 300 раз. В этом случае резервное копирование прерывается.

Даже если контрольные суммы не включены, pg_probackup3 всегда проверяет целостность заголовков страниц.

3.9. Настройка политики хранения

Используя pg_probackup3, вы можете реализовать политики хранения резервных копий, в соответствии с которыми могут удаляться лишние копии, очищаться ненужные сегменты WAL. Кроме того, можно закреплять определённые копии, чтобы они сохранялись независимо от политики, как описано ниже. Эти подходы можно комбинировать произвольным образом.

3.9.1. Удаление ненужных копий

По умолчанию все резервные копии, которые создаёт pg_probackup3, сохраняются в предназначенном для них каталоге. Для экономии дискового пространства вы можете настроить политику сохранения копий, чтобы ненужные копии удалялись.

Чтобы настроить политику хранения, задайте одну или несколько следующих переменных в файле `pg_probackup3.conf` с помощью команды set-config:

`--retention-redundancy=`*`избыточность`*

Определяет **количество полных резервных копий**, которое должно сохраняться в каталоге копий.

`--retention-window=`*`окно`*

Определяет самый ранний момент времени, на который pg_probackup3 может выполнить восстановление. В этом параметре задаётся **количество дней** от текущего момента. Например, если `retention-window=6`, должна сохраниться минимум одна копия старее шести дней, вместе с соответствующими файлами WAL и всеми последующими копиями.

Если установлены параметры `--retention-redundancy` и `--retention-window`, при очистке каталога от ненужных копий принимаются во внимание оба заданных ими условия. Например, если задать параметры `--retention-redundancy=2` и `--retention-window=6`, программа pg_probackup3 должна сохранить две полные резервные копии, а также все копии, необходимые для восстановления данных, за последние шесть дней:

```
pg_probackup3 set-config -B каталог_копий --instance=имя_экземпляра --retention-
redundancy=2 --retention-window=6
```

Рекомендуется всегда хранить как минимум две последние полные родительские копии, чтобы избежать ошибок при создании инкрементальных резервных копий.

Чтобы очистить каталог копий в соответствии с политикой хранения, нужно запустить команду retention с флагами сохранения, как показано ниже.

Например, чтобы удалить все резервные копии, считающиеся ненужными согласно установленной политике хранения, нужно выполнить следующую команду с флагом `--delete-expired`:

```
pg_probackup3 retention -B каталог_копий --instance=имя_экземпляра --delete-expired
```

Если вы хотите также удалить файлы WAL, которые больше не требуются ни для каких копий, укажите дополнительно `--delete-wal`:

```
pg_probackup3 retention -B каталог_копий --instance=имя_экземпляра --delete-expired --
delete-wal
```

Вы также можете установить или переопределить текущую политику хранения, добавив параметры `--retention-redundancy` и `--retention-window` непосредственно при выполнении команды `retention`:

```
pg_probackup3 retention -B каталог_копий --instance=имя_экземпляра --delete-expired --
retention-window=6 --retention-redundancy=2
```

Так как для инкрементальных копий требуется наличие всех родительских полных копий и всех предыдущих инкрементальных копий, даже по истечении срока их хранения эти копии нельзя удалить, пока минимум одна инкрементальная копия в цепочке удовлетворяет политике хранения. Чтобы не хранить устаревшие копии, которые всё ещё нужны для восстановления нужной инкрементальной копии, вы можете объединить их с ней, воспользовавшись ключом `--merge-expired` команды retention.

Предположим, что вы заархивировали экземпляр *`узла`* в *`каталог_копий`* со значением параметра `--retention-window`, равным `6`, и значением параметра `--retention-redundancy`, равным `2`, и на 11 февраля 2025 г. у вас есть следующие копии:

```
BACKUP INSTANCE 'dev', version 3
================================================================================================
Instance Version ID        End time                 Mode  WAL Mode TLI Duration Data
 WAL Zalg Zratio Start LSN  Stop LSN   Status
================================================================================================
dev      17      full-1    2024-10-18 21:02:28+0000 FULL  ARCHIVE  1            87MB  -
   none 1.00   0/10000028 0/10000128 OK
dev      17      delta-1-1 2024-11-11 00:36:01+0000 DELTA ARCHIVE  1            23MB  -
   none 1.00   0/12000028 0/12000128 OK
dev      17      delta-1-2 2024-11-15 15:43:01+0000 DELTA ARCHIVE  1            22MB  -
   none 1.00   0/14000028 0/14000128 OK
dev      17      full-2    2024-11-22 14:24:04+0000 FULL  ARCHIVE  1            98MB  -
   none 1.00   0/17000028 0/17000128 OK
dev      17      delta-2-1 2024-11-23 18:10:55+0000 DELTA ARCHIVE  1            23MB  -
   none 1.00   0/19000028 0/19000128 OK
 ------------------------------------------------------retention
 window------------------------------------------------------------
dev      17      delta-2-2 2025-02-06 23:44:33+0000 DELTA ARCHIVE  1            33MB  -
   none 1.00   0/1C000028 0/1C000128 OK
dev      17      full-3    2025-02-08 03:31:33+0000 FULL  ARCHIVE  1            120MB -
   none 1.00   0/1F000028 0/1F000128 OK
dev      17      delta-3-1 2025-02-09 07:18:31+0000 DELTA ARCHIVE  1            23MB  -
   none 1.00   0/21000028 0/21000128 OK
dev      17      delta-3-2 2025-02-10 11:05:17+0000 DELTA ARCHIVE  1            23MB  -
   none 1.00   0/23000028 0/23000128 OK
dev      17      full-4    2025-02-11 15:00:38+0000 FULL  ARCHIVE  1   1s       123MB -
   none 1.00   0/25000028 0/25000128 OK
```

При запуске команды retention с флагом `--delete-expired` резервные копии с идентификаторами `full-1`, `delta-1-1` и `delta-1-2` будут удалены как устаревшие по критериям окна хранения и избыточности (требуемый набор полных резервных копий уже сохранён). `delta-1-1` и `delta-1-2` также будут удалены, поскольку базовая полная копия потеряла актуальность.

При запуске команды retention с флагом `--merge-expired` резервные копии `full-2` и `delta-2-1` будут объединены с `delta-2-2`. Объединение произойдёт именно с `delta-2-2`, так как это первая актуальная разностная копия, которую можно объединить с неактуальной разностной копией `delta-2-1` и неактуальной полной копией `full-2`.

```
pg_probackup3 retention -B каталог_копий --instance=узел --delete-expired --merge-
expired
pg_probackup3 show -B каталог_копий
```

```
BACKUP INSTANCE 'dev', version 3
================================================================================================
Instance Version ID                       End time                 Mode  WAL Mode TLI
 Duration Data  WAL Zalg Zratio Start LSN  Stop LSN   Status
================================================================================================
dev      17      2025-02-11-11-14-18-254 2025-02-11 11:14:18+0000 FULL  ARCHIVE  1
       108MB  -  none 1.00   0/17000028 0/19000128 OK
dev      17      full-3                   2025-02-08 03:31:33+0000 FULL  ARCHIVE  1
       120MB -   none 1.00   0/1F000028 0/1F000128 OK
dev      17      delta-3-1                2025-02-09 07:18:31+0000 DELTA ARCHIVE  1
       23MB  -   none 1.00   0/21000028 0/21000128 OK
dev      17      delta-3-2                2025-02-10 11:05:17+0000 DELTA ARCHIVE  1
       23MB  -   none 1.00   0/23000028 0/23000128 OK
dev      17      full-4                   2025-02-11 15:00:38+0000 FULL  ARCHIVE  1   1s
       123MB -   none 1.00   0/25000028 0/25000128 OK
```

Поле `Duration` (Время) для объединённой копии показывает время, которое заняла процедура объединения.

3.9.2. Закрепление резервных копий

Если вам нужно хранить отдельные копии дольше, чем допускает установленная политика хранения, вы можете дополнительно закрепить их на определённое время. Например:

```
pg_probackup3 set-backup -B каталог_копий --instance=имя_экземпляра -
i ид_резервной_копии --ttl=30d
```

Эта команда задаёт срок хранения заданной резервной копии 30 дней, начиная с момента, указанного в атрибуте `recovery-time` этой копии.

Вы также можете явно задать время истечения срока хранения копии, воспользовавшись ключом `--expire-time`. Например:

```
pg_probackup3 set-backup -B каталог_копий --instance=имя_экземпляра -
i ид_резервной_копии --expire-time="2027-04-09 18:21:32+00"
```

Также можно воспользоваться ключами `--ttl` и `--expire-time` команды backup и сразу закрепить создаваемую копию:

```
pg_probackup3 backup -B каталог_копий --instance=имя_экземпляра -b FULL --ttl=30d
pg_probackup3 backup -B каталог_копий --instance=имя_экземпляра -b FULL --expire-
time="2027-04-09 18:21:32+00"
```

Определить, закреплена ли копия, можно с помощью команды show:

```
pg_probackup show -B каталог_копий --instance=имя_экземпляра -i ид_резервной_копии
```

Если копия закреплена, у неё есть атрибут `expire-time`, содержащий время окончания срока её хранения:

```
...
recovery-time = '2024-04-09 18:21:32+00'
expire-time = '2027-04-09 18:21:32+00'
data-bytes = 22288792
...
```

Отменить закрепление копии можно, передав в параметре `--ttl` ноль:

```
pg_probackup3 set-backup -B каталог_копий --instance=имя_экземпляра -
i ид_резервной_копии --ttl=0
```

> **Примечание**
>
> Для закреплённой инкрементальной копии неявным образом закрепляются все нужные ей родительские копии. Если вы позже снимите с неё закрепление, последние также автоматически перестанут быть закреплёнными.

3.9.3. Настройка политики хранения архива WAL

Когда осуществляется непрерывное архивирование WAL, сегменты WAL могут занимать много места на диске. Даже если вы время от времени удаляете старые резервные копии, с флагом `--delete-wal` могут быть удалены только те сегменты WAL, которые не относятся ни к какой копии из остающихся в каталоге. Однако если возможность восстановления на момент времени нужна только для последних копий, можно настроить политику хранения архива WAL, чтобы ограничить глубину архива и сэкономить место на диске.

Предположим, что вы заархивировали экземпляр `node` в *каталог_копий* и настроили непрерывное архивирование WAL:

```
pg_probackup3 show -B каталог_копий --instance=узел
```

```
BACKUP INSTANCE 'dev', version 3
================================================================================================
Instance Version ID                         End time                 Mode  WAL Mode TLI
 Duration Data  WAL Zalg Zratio Start LSN  Stop LSN   Status
================================================================================================
dev      17      2025-02-11-15-13-36-756 2025-02-11 15:13:37+0000 FULL  ARCHIVE  1   1s
       38MB  -   none 1.00   0/17000028 0/19000128 OK
dev      17      2025-02-11-14-51-12-937 2025-02-06 23:44:33+0000 DELTA ARCHIVE  1
       33MB  -   none 1.00   0/1C000028 0/1C000128 OK
dev      17      2025-02-11-14-51-33-367 2025-02-08 03:31:33+0000 FULL  ARCHIVE  1
       120MB -   none 1.00   0/1F000028 0/1F000128 OK
dev      17      2025-02-11-14-51-51-220 2025-02-09 07:18:31+0000 DELTA ARCHIVE  1
       23MB  -   none 1.00   0/21000028 0/21000128 OK
dev      17      2025-02-11-14-51-57-473 2025-02-10 11:05:17+0000 DELTA ARCHIVE  1
       23MB  -   none 1.00   0/23000028 0/23000128 OK
dev      17      2025-02-11-15-00-37-815 2025-02-11 15:00:38+0000 FULL  ARCHIVE  1   1s
       123MB -   none 1.00   0/25000028 0/25000128 OK
```

Вы можете проверить состояние архива WAL, запустив команду show с ключом `--archive`:

```
pg_probackup3 show -B каталог_копий --instance=node --archive
```

```
BACKUP INSTANCE 'dev', version 3
================================================================================================
TLI Parent TLI Switchpoint Min Segno                Max Segno                 N segments
 Size  Zratio N backups Status
================================================================================================
1              0/0         000000010000000000000001 000000010000000000000025 37
 592MB 1.41    6         OK
```

Для удаления всех старых файлов WAL, которые не нужны для восстановления никаких из оставшихся резервных копий, выполните следующую команду:

```
pg_probackup retention -B каталог_копий --instance=node --delete-wal
```

```
[2025-02-11 15:23:30.422696] [14218] [128670453549440] [info] command: ./pg_probackup3
 retention -B /work/backup --instance dev --delete-wal
[2025-02-11 15:23:30.422738] [14218] [128670453549440] [info] execute command:
 'retention', instance 'dev'
[2025-02-11 15:23:30.426167] [14218] [128670453549440] [info] WAL file
 000000010000000000000001 removed
[2025-02-11 15:23:30.428095] [14218] [128670453549440] [info] WAL file
 000000010000000000000002 removed
[2025-02-11 15:23:30.429776] [14218] [128670453549440] [info] WAL file
 000000010000000000000003 removed
[2025-02-11 15:23:30.431838] [14218] [128670453549440] [info] WAL file
 000000010000000000000004 removed
[2025-02-11 15:23:30.434124] [14218] [128670453549440] [info] WAL file
 000000010000000000000005 removed
[2025-02-11 15:23:30.434196] [14218] [128670453549440] [info] WAL file
 000000010000000000000005.00000028.backup removed
[2025-02-11 15:23:30.435852] [14218] [128670453549440] [info] WAL file
 000000010000000000000006 removed
[2025-02-11 15:23:30.437579] [14218] [128670453549440] [info] WAL file
 000000010000000000000007 removed
[2025-02-11 15:23:30.441360] [14218] [128670453549440] [info] WAL file
 000000010000000000000008 removed
[2025-02-11 15:23:30.441815] [14218] [128670453549440] [info] WAL file
 000000010000000000000008.00000028.backup removed
[2025-02-11 15:23:30.444488] [14218] [128670453549440] [info] WAL file
 000000010000000000000009 removed
```

```
[2025-02-11 15:23:30.446902] [14218] [128670453549440] [info] WAL file
 0000000100000000000000A removed
[2025-02-11 15:23:30.446961] [14218] [128670453549440] [info] WAL file
 0000000100000000000000A.00000028.backup removed
[2025-02-11 15:23:30.448960] [14218] [128670453549440] [info] WAL file
 0000000100000000000000B removed
[2025-02-11 15:23:30.450991] [14218] [128670453549440] [info] WAL file
 0000000100000000000000C removed
[2025-02-11 15:23:30.451069] [14218] [128670453549440] [info] WAL file
 0000000100000000000000C.00000028.backup removed
[2025-02-11 15:23:30.453236] [14218] [128670453549440] [info] WAL file
 0000000100000000000000D removed
[2025-02-11 15:23:30.455291] [14218] [128670453549440] [info] WAL file
 0000000100000000000000E removed
[2025-02-11 15:23:30.455462] [14218] [128670453549440] [info] WAL file
 0000000100000000000000E.00000028.backup removed
[2025-02-11 15:23:30.458088] [14218] [128670453549440] [info] WAL file
 0000000100000000000000F removed
[2025-02-11 15:23:30.459755] [14218] [128670453549440] [info] WAL file
 00000001000000000000010 removed
[2025-02-11 15:23:30.459794] [14218] [128670453549440] [info] WAL file
 00000001000000000000010.00000028.backup removed
[2025-02-11 15:23:30.461135] [14218] [128670453549440] [info] WAL file
 00000001000000000000011 removed
[2025-02-11 15:23:30.462603] [14218] [128670453549440] [info] WAL file
 00000001000000000000012 removed
[2025-02-11 15:23:30.462637] [14218] [128670453549440] [info] WAL file
 00000001000000000000012.00000028.backup removed
[2025-02-11 15:23:30.464003] [14218] [128670453549440] [info] WAL file
 00000001000000000000013 removed
[2025-02-11 15:23:30.465522] [14218] [128670453549440] [info] WAL file
 00000001000000000000014 removed
[2025-02-11 15:23:30.465555] [14218] [128670453549440] [info] WAL file
 00000001000000000000014.00000028.backup removed
[2025-02-11 15:23:30.466910] [14218] [128670453549440] [info] WAL file
 00000001000000000000015 removed
[2025-02-11 15:23:30.468572] [14218] [128670453549440] [info] WAL file
 00000001000000000000016 removed
[2025-02-11 15:23:30.468600] [14218] [128670453549440] [info] 30 WAL files removed.
```

Вы можете проверить состояние архива WAL, запустив команду show с ключом `--archive`:

```
pg_probackup3 show -B каталог_копий --instance=node --archive

BACKUP INSTANCE 'dev', version 3
============================================================================================================
TLI Parent TLI Switchpoint Min Segno                Max Segno                N segments
 Size  Zratio N backups Status
============================================================================================================
1              0/0         000000010000000000000017 000000010000000000000025 15
 240MB 1.47   6         OK
```

3.10. Клонирование и синхронизация экземпляра Postgres Pro

В pg_probackup3 реализована команда catchup, которая позволяет создать копию экземпляра сервера Postgres Pro напрямую, не используя каталог резервных копий. Эта команда может быть полезна:

- Для добавления нового ведомого сервера.

  Если каталог данных целевого экземпляра пуст, команда `catchup` работает аналогично команде `backup` в режиме PRO, но может выполняться быстрее при запуске в параллельном режиме.

> **Примечание**
>
> Операции `catchup` на данный момент поддерживаются только в режиме PRO и не требуют прямого доступа к `PGDATA`.

- Для синхронизации отставшего ведомого сервера с ведущим.

  При активной записи на ведущем сервере реплики могут не успевать воспроизводить WAL с достаточной скоростью и в результате отставать. Обычно в этом случае создаётся новая реплика, а для этого нужно передать и сохранить на диске большой объём данных. Команда `catchup` напрямую переносит различия с ведущего сервера, что позволяет намного быстрее обновить данные на уже существующей реплике.

Особенности и ограничения `catchup`:

- Не требуется каталог резервных копий.
- Копирование внешних каталогов не поддерживается.
- Нет поддержки удалённого режима.
- Одновременно с `catchup` нельзя выполнять команды *DDL CREATE TABLESPACE*/*DROP TABLESPACE*.
- При синхронизации `catchup` берёт файлы конфигурации, такие как `postgresql.conf`, `postgresql.auto.conf` или `pg_hba.conf`, с исходного сервера и заменяет ими соответствующие файлы на целевом сервере. Чтобы оставить файлы конфигурации без изменений, используйте параметр `--exclude-path`.

Чтобы подготовить экземпляр Postgres Pro к клонированию/синхронизации, настройте исходный сервер следующим образом:

- Настройте кластер баз данных для копирования экземпляра сервера.
- Для использования режима PTRACK настройте копирование PTRACK.

Перед клонированием/синхронизацией экземпляра Postgres Pro убедитесь, что исходный сервер запущен и принимает подключения. Чтобы клонировать/синхронизировать экземпляр Postgres Pro, в системе с целевым сервером выполните команду catchup:

```
pg_probackup3 catchup -b режим_синхронизации --destination-
pgdata=путь_к_целевому_каталогу_данных --stream [параметры_подключения]
```

Здесь *режим_синхронизации* может принимать следующие значения:

- `FULL` — создаётся полная копия экземпляра Postgres Pro, для этого целевой каталог данных БД должен быть пустым.
- `DELTA` — считываются все файлы данных в каталоге данных и создаётся инкрементальная копия для страниц, изменённых с момента остановки целевого экземпляра.
- `PTRACK` — изменения в страницах отслеживаются на лету, считываются и копируются только страницы, изменённые с точки расхождения исходного и целевого экземпляров.

Указав параметр `--stream`, можно задать режим STREAM, при котором все необходимые файлы WAL передаются с исходного сервера по протоколу репликации.

Для подключения к исходному кластеру базы данных можно использовать параметры connection_options.

Если в исходной базе данных есть табличные пространства, которые должны располагаться в других каталогах в целевой системе, задайте также параметр `--tablespace-mapping`:

```
pg_probackup3 catchup -b режим_синхронизации --destination-
pgdata=путь_к_целевому_каталогу_данных --stream --tablespace-
mapping=СТАРЫЙ_КАТАЛОГ=НОВЫЙ_КАТАЛОГ
```

Чтобы операция `catchup` выполнялась в несколько параллельных потоков, задайте число потоков с помощью параметра `--threads` или `--num-write-threads` и `--num-validate-threads`:

```
pg_probackup3 catchup -b режим_синхронизации --destination-
pgdata=путь_к_целевому_каталогу_данных --stream --threads=число_потоков
```

3.11. Другие примеры

Все приведённые ниже примеры предполагают использование удалённого режима работы через SSH. Если вы планируете выполнять резервное копирование и восстановление локально, пропустите шаг «Настройка SSH-подключения без пароля» и не используйте параметры `--remote-*`.

Примеры основаны на Ubuntu 22.04, Postgres Pro 17 и pg_probackup3

- `backup` — роль в Postgres Pro, используемая для подключения к кластеру Postgres Pro.
- `backupdb` — база данных, через которую выполняется подключение к кластеру Postgres Pro.
- `backup_host` — система, где находится каталог резервных копий.
- `backup_user` — пользователь в системе `backup_host`, от имени которого выполняются все операции pg_probackup3.
- `/mnt/backups` — путь к каталогу резервных копий в системе `backup_host`.
- `postgres_host` — система, в которой работает Postgres Pro.
- `postgres` — пользователь в системе `postgres_host`, от имени которого запускаются процессы кластера Postgres Pro.
- `/var/lib/pgpro/std-17/data` — путь к каталогу данных Postgres Pro в системе `postgres_host`.

3.11.1. Минимальная настройка

Данный сценарий показывает настройку самодостаточного полного или разностного резервного копирования.

1. **Настройте подключение по SSH с** `backup_host` **к** `postgres_host`**:**

   ```
   [backup_user@backup_host] ssh-copy-id postgres@postgres_host
   ```

2. **Настройте кластер Postgres Pro.**

   В целях безопасности для резервного копирования рекомендуется использовать отдельную базу данных.

   ```
   postgres=#
   CREATE DATABASE backupdb;
   ```

   Подключитесь к базе данных `backupdb`, создайте роль `probackup` и выдайте этой роли следующие права:

   ```
   backupdb=#
   BEGIN;
   CREATE ROLE backup WITH LOGIN REPLICATION;
   GRANT USAGE ON SCHEMA pg_catalog TO backup;
   GRANT EXECUTE ON FUNCTION pg_catalog.current_setting(text) TO backup;
   GRANT EXECUTE ON FUNCTION pg_catalog.set_config(text, text, boolean) TO backup;
   GRANT EXECUTE ON FUNCTION pg_catalog.pg_is_in_recovery() TO backup;
   GRANT EXECUTE ON FUNCTION pg_catalog.pg_backup_start(text, boolean, boolean) TO
    backup;
   GRANT EXECUTE ON FUNCTION pg_catalog.pg_backup_stop(boolean, boolean) TO backup;
   ```

```
GRANT EXECUTE ON FUNCTION pg_catalog.pg_create_restore_point(text) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_switch_wal() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_last_wal_replay_lsn() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current_snapshot() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_snapshot_xmax(txid_snapshot) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_control_checkpoint() TO backup;
COMMIT;
```

Добавьте плагин pgpro_bindump в файл `postgresql.conf`:

```
echo "shared_preload_libraries = 'pgpro_bindump'" >> "/var/lib/pgpro/std-17/data/
postgresql.conf"
echo "walsender_plugin_libraries = 'pgpro_bindump' " >> "/var/lib/pgpro/std-17/
data/postgresql.conf"
echo "wal_level = 'replica'" >> "/var/lib/pgpro/std-17/data/postgresql.conf"
```

3. **Инициализируйте каталог резервных копий:**

```
[backup_user@backup_host]$ pg_probackup3 init -B /mnt/backups
2024-12-09 07:40:27.198881] [363926] [135107950659968] [info] Backup catalog '/mnt/
backups' successfully initialized
```

4. **Определите копируемый экземпляр** `pg-17` **в каталоге резервных копий:**

```
[backup_user@backup_host]$ pg_probackup3 add-instance -B /mnt/backups --instance
 pg-17 --remote-host=postgres_host --remote-user=postgres -D var/lib/pgpro/std-17/
data
[2024-12-09 07:47:56.595727] [364390] [138813944502656] [info] Instance 'pg-17'
 successfully initialized
```

5. **Создайте полную резервную копию:**

```
[backup_user@backup_host] pg_probackup3 backup -B /mnt/backups --instance pg-17
 -b FULL --stream --remote-host=postgres_host --remote-user=postgres -U backup -d
 backupdb --backup-id=1-full
[2024-12-09 23:44:49.602026] [425177] [123209585379712] [info] START BACKUP
 COMMAND= PGPRO_CALL_PLUGIN pgpro_bindump start_backup(LABEL '1-full');
[2024-12-09 23:44:49.645450] [425177] [123209585379712] [info] PG_PROBACKUP
 0/4000028 tli=1
[2024-12-09 23:44:49.652048] [425177] [123209585379712] [info] Created replication
 slot. Name='pg_probackup3_wal_streaming_425181', consistent point=0/0, snapshot
 name=, output plugin=
[2024-12-09 23:44:49.652185] [425177] [123209585379712] [info] BACKUP COMMAND
 PGPRO_CALL_PLUGIN pgpro_bindump copy_files(VERIFY_CHECKSUMS true, COMPRESS_ALG
 'none', COMPRESS_LVL 1);
[2024-12-09 23:44:49.652468] [425177] [123209573729984] [info] Starting new segment
 4
[2024-12-09 23:44:49.769640] [425177] [123209585379712] [info] BACKUP COMMAND
 PGPRO_CALL_PLUGIN pgpro_bindump stop_backup(STREAM true, COMPRESS_ALG 'none',
 COMPRESS_LVL 1);
[2024-12-09 23:44:49.805112] [425177] [123209573729984] [info] Stopping segment 4
[2024-12-09 23:44:49.805316] [425177] [123209573729984] [info] finished streaming
 WAL at 0/5000000 (timeline 1)
[2024-12-09 23:44:49.805343] [425177] [123209573729984] [info] WAL streaming: WAL
 streaming stop requested at 0/4000138, stopping at 0/5000000
[2024-12-09 23:44:49.805935] [425177] [123209585379712] [info] PG_PROBACKUP-STOP
 0/4000138 tli=1 bytes written=39430093 bytes compressed=39430093
[2024-12-09 23:44:49.806484] [425177] [123209585379712] [info] Backup time 206
[2024-12-09 23:44:49.806515] [425177] [123209585379712] [info] Backup 1-full
 completed successfully.
INFO: Backup 1-full completed successfully.
```

```
[2024-12-09 23:44:49.806592] [425177] [123209585379712] [info] Start validate  1-
full ...
[2024-12-09 23:44:49.807204] [425177] [123209585379712] [info] Validating backup 1-
full
[2024-12-09 23:44:49.912115] [425177] [123209585379712] [info] Validate time 104
[2024-12-09 23:44:49.912398] [425177] [123209585379712] [info] INFO: Backup 1-full
 is valid
```

6. **Посмотрим на каталог резервных копий:**

```
[backup_user@backup_host] pg_probackup3 show -B /mnt/backups --instance pg-17

BACKUP INSTANCE 'pg-17', version 3
============================================================================================
 Instance Version ID     End time                 Mode WAL Mode TLI Duration Data
 WAL Zalg Zratio Start LSN Stop LSN  Status
============================================================================================
pg-17      16      1-full 2024-12-09 23:44:49+0000 FULL STREAM   1            38MB
 -   none 1.00   0/4000028 0/4000138 OK
```

7. **Создайте инкрементальную резервную копию в режиме DELTA:**

```
[backup_user@backup_host] pg_probackup3 backup -B /mnt/backups --instance pg-17 -
b delta --stream --remote-host=postgres_host --remote-user=postgres -U backup -d
 backupdb --parent-backup-id=1-full --backup-id=1-delta
[2024-12-10 01:00:50.804867] [430043] [130779551140224] [info] This PostgreSQL
 instance was initialized with data block checksums. Data block corruption will be
 detected
[2024-12-10 01:00:50.805233] [430043] [130779551140224] [info] START BACKUP
 COMMAND= PGPRO_CALL_PLUGIN pgpro_bindump start_backup(LABEL '1-delta', START_LSN
 '0/4000028');
[2024-12-10 01:00:50.843249] [430043] [130779551140224] [info] PG_PROBACKUP
 0/6000028 tli=1
[2024-12-10 01:00:50.850799] [430043] [130779551140224] [info] Created replication
 slot. Name='pg_probackup3_wal_streaming_430047', consistent point=0/0, snapshot
 name=, output plugin=
[2024-12-10 01:00:50.850898] [430043] [130779551140224] [info] BACKUP COMMAND
 PGPRO_CALL_PLUGIN pgpro_bindump copy_files(VERIFY_CHECKSUMS true, START_LSN
 '0/4000028', COMPRESS_ALG 'none', COMPRESS_LVL 1);
[2024-12-10 01:00:50.851124] [430043] [130779470366400] [info] Starting new segment
 6
[2024-12-10 01:00:50.877932] [430043] [130779551140224] [info] BACKUP COMMAND
 PGPRO_CALL_PLUGIN pgpro_bindump stop_backup(STREAM true, COMPRESS_ALG 'none',
 COMPRESS_LVL 1);
[2024-12-10 01:00:50.913070] [430043] [130779470366400] [info] Stopping segment 6
[2024-12-10 01:00:50.913284] [430043] [130779470366400] [info] finished streaming
 WAL at 0/7000000 (timeline 1)
[2024-12-10 01:00:50.913302] [430043] [130779470366400] [info] WAL streaming: WAL
 streaming stop requested at 0/6000138, stopping at 0/7000000
[2024-12-10 01:00:50.913497] [430043] [130779551140224] [info] PG_PROBACKUP-STOP
 0/6000138 tli=1 bytes written=786310 bytes compressed=786310
[2024-12-10 01:00:50.913868] [430043] [130779551140224] [info] Backup time 110
[2024-12-10 01:00:50.913884] [430043] [130779551140224] [info] Backup 1-delta
 completed successfully.
INFO: Backup 1-delta completed successfully.
[2024-12-10 01:00:50.913918] [430043] [130779551140224] [info] Start validate  1-
delta ...
[2024-12-10 01:00:50.914269] [430043] [130779551140224] [info] Validating backup 1-
delta
[2024-12-10 01:00:50.934892] [430043] [130779551140224] [info] Validate time 20
```

```
[2024-12-10 01:00:50.935188] [430043] [130779551140224] [info] INFO: Backup 1-delta
 is valid
```

8. **Добавим несколько параметров в файл конфигурации pg_probackup3, чтобы их не надо было каждый раз указывать в командной строке:**

```
[backup_user@backup_host] pg_probackup3 set-config -B /mnt/backups --instance pg-17
 --remote-host=postgres_host --remote-user=postgres -U backup -d backupdb
[2024-12-10 01:03:18.173698] [430208] [125541616851328] [info] Instance 'pg-17'
 successfully updated
```

9. **Сделайте ещё одну инкрементальную копию в режиме DELTA, опустив некоторые из предыдущих параметров:**

```
[backup_user@backup_host] pg_probackup3 backup -B /mnt/backups --instance pg-17 -b
 delta --stream --parent-backup-id=1-delta --backup-id=2-delta
[2024-12-10 01:26:33.325658] [431695] [135663496210816] [info] This PostgreSQL
 instance was initialized with data block checksums. Data block corruption will be
 detected
[2024-12-10 01:26:33.326140] [431695] [135663496210816] [info] START BACKUP
 COMMAND= PGPRO_CALL_PLUGIN pgpro_bindump start_backup(LABEL '2-delta', START_LSN
 '0/6000028');
[2024-12-10 01:26:33.365430] [431695] [135663496210816] [info] PG_PROBACKUP
 0/8000028 tli=1
[2024-12-10 01:26:33.372681] [431695] [135663496210816] [info] Created replication
 slot. Name='pg_probackup3_wal_streaming_431699', consistent point=0/0, snapshot
 name=, output plugin=
[2024-12-10 01:26:33.372762] [431695] [135663496210816] [info] BACKUP COMMAND
 PGPRO_CALL_PLUGIN pgpro_bindump copy_files(VERIFY_CHECKSUMS true, START_LSN
 '0/6000028', COMPRESS_ALG 'none', COMPRESS_LVL 1);
[2024-12-10 01:26:33.372966] [431695] [135663483619008] [info] Starting new segment
 8
[2024-12-10 01:26:33.407073] [431695] [135663496210816] [info] BACKUP COMMAND
 PGPRO_CALL_PLUGIN pgpro_bindump stop_backup(STREAM true, COMPRESS_ALG 'none',
 COMPRESS_LVL 1);
[2024-12-10 01:26:33.441125] [431695] [135663483619008] [info] Stopping segment 8
[2024-12-10 01:26:33.441303] [431695] [135663483619008] [info] finished streaming
 WAL at 0/9000000 (timeline 1)
[2024-12-10 01:26:33.441318] [431695] [135663483619008] [info] WAL streaming: WAL
 streaming stop requested at 0/8000138, stopping at 0/9000000
[2024-12-10 01:26:33.441497] [431695] [135663496210816] [info] PG_PROBACKUP-STOP
 0/8000138 tli=1 bytes written=786310 bytes compressed=786310
[2024-12-10 01:26:33.441809] [431695] [135663496210816] [info] Backup time 117
[2024-12-10 01:26:33.441822] [431695] [135663496210816] [info] Backup 2-delta
 completed successfully.
INFO: Backup 2-delta completed successfully.
[2024-12-10 01:26:33.441850] [431695] [135663496210816] [info] Start validate  2-
delta ...
[2024-12-10 01:26:33.442115] [431695] [135663496210816] [info] Validating backup 2-
delta
[2024-12-10 01:26:33.463554] [431695] [135663496210816] [info] Validate time 21
[2024-12-10 01:26:33.463728] [431695] [135663496210816] [info] INFO: Backup 2-delta
 is valid
```

10. **Проверим конфигурацию экземпляра:**

```
[backup_user@backup_host] pg_probackup3 show-config -B /mnt/backups --instance
 pg-17

# Backup instance information
system-identifier = 7446313657913924966
```

```
# Connection parameters
pguser = backup
pgdatabase = backupdb
pgdata = /var/lib/pgpro/std-17/data
# Logging parameters
log-level-console = info
log-level-file = off
log-format-console = plain
log-format-file = plain
log-filename = pg_probackup.log
log-rotation-size = 0
# Compression parameters
compress-algorithm = none
compress-level = 0
# Retention parameters
retention-redundancy = 0
retention-window = 0
wal-depth = 0
```

11. **Посмотрим на каталог резервных копий:**

```
[backup_user@backup_host] pg_probackup3 show -B /mnt/backups --instance pg-17
BACKUP INSTANCE 'pg-17', version 3
============================================================================================
 Instance Version ID      End time                 Mode  WAL Mode TLI Duration Data
  WAL Zalg Zratio Start LSN Stop LSN  Status
============================================================================================
pg-17      16      1-full  2024-12-09 23:44:49+0000 FULL  STREAM   1
 38MB  -   none 1.00   0/4000028 0/4000138 OK
pg-17      16      1-delta 2024-12-10 01:00:50+0000 DELTA STREAM   1
 768kB -   none 1.00   0/6000028 0/6000138 OK
pg-17      16      2-delta 2024-12-10 01:26:33+0000 DELTA STREAM   1
 768kB -   none 1.00   0/8000028 0/8000138 OK
```

Глава 4. Справка

pg_probackup3

pg_probackup3 — управление резервным копированием и восстановлением кластеров баз данных Postgres Pro

Синтаксис

pg_probackup3 add-instance -B *каталог_копий* -D *каталог_данных* --instance *имя_экземпляра* --skip-if-exists

pg_probackup3 archive-get -B *каталог_копий* --instance *имя_экземпляра* --wal-file-path *путь_файлов_wal* --wal-file-name *имя_файла_wal* [*параметр*...]

pg_probackup3 archive-push -B *каталог_копий* --instance *имя_экземпляра* --wal-file-path *путь_файлов_wal* --wal-file-name *имя_файла_wal* [*параметр*...]

pg_probackup3 backup -B *каталог_копий* --instance *имя_экземпляра* -b *режим_копирования* [*параметр*...]

pg_probackup3 catchup -b *режим_синхронизации* --destination-pgdata=*путь_к_целевому_каталогу_данных* [*параметр*...]

pg_probackup3 del-instance -B *каталог_копий* --instance *имя_экземпляра*

pg_probackup3 delete -B *каталог_копий* --instance *имя_экземпляра* -i *ид_резервной_копии*

pg_probackup3 file-map -B *каталог_копий* --instance *имя_экземпляра* -i *ид_резервной_копии* [*параметр*...]

pg_probackup3 fuse -B *каталог_копий* --mnt-path *путь_монтирования* --instance *имя_экземпляра* -i *ид_резервной_копии* --cache-swap-size *порог_сброса_кеша* --cache-dir *каталог_кеша* [*параметр*...]

pg_probackup3 help [*команда*]

pg_probackup3 init -B *каталог_копий* --skip-if-exists

pg_probackup3 merge -B *каталог_копий* --instance *имя_экземпляра* -i *ид_резервной_копии* [*параметр*...]

pg_probackup3 restore -B *каталог_копий* --instance *имя_экземпляра* [*параметр*...]

pg_probackup3 retention -B *каталог_копий* --instance *имя_экземпляра* { --delete-wal | --delete-expired | --merge-expired } [*параметр*...]

pg_probackup3 send-backup -B *каталог_копий* --instance *имя_экземпляра* -i *ид_резервной_копии* -p *порт* -h *сервер* [*параметр*...]

pg_probackup3 server-info [*параметр*...]

pg_probackup3 set-backup -B *каталог_копий* --instance *имя_экземпляра* -i *ид_резервной_копии* [*параметр*...]

pg_probackup3 set-config -B *каталог_копий* --instance *имя_экземпляра* [*параметр*...]

pg_probackup3 show -B *каталог_копий* [*параметр*...]

pg_probackup3 show-config -B *каталог_копий* --instance *имя_экземпляра* [*параметр*...]

```
pg_probackup3 validate -B каталог_копий [параметр...]

pg_probackup3 version
```

Справка по командной строке

Команды

В этом подразделе описываются команды pg_probackup3. Необязательные параметры этих команд заключаются в квадратные скобки. В подробностях все параметры описываются в подразделе Параметры.

add-instance

```
pg_probackup3 add-instance -B каталог_копий -D каталог_данных --instance=имя_экземпляра
[--skip-if-exists] [--wal-archive-dir=путь_каталога] [--help]
[параметры_s3] [параметры_ssh] [параметры_журнала]
[параметры_подключения] [параметры_сжатия] [параметры_сохранения]
[параметры_буферизации]
```

Инициализирует новый копируемый экземпляр в каталоге *каталог_копий* и создаёт файл конфигурации pg_probackup3.conf, управляющий параметрами pg_probackup3, относящимися к кластеру в указанном *каталоге_данных*. Если каталог уже был инициализирован, сообщение об ошибке можно отключить, указав --skip-if-exists.

По умолчанию архив WAL хранится в каталоге *каталог_копий*/wal/*имя_экземпляра*. Чтобы указать пользовательский каталог, используйте параметр --wal-archive-dir. Тип хранилища (локальная файловая система, SFTP или S3) должен соответствовать конфигурации экземпляра. Каталог будет создан автоматически, если он не существует.

За подробностями обратитесь к подразделам Общие параметры и Определение копируемого экземпляра.

archive-get

```
pg_probackup3 archive-get -B каталог_копий --instance=имя_экземпляра --wal-file-
path=путь_файлов_wal --wal-file-name=имя_файла_wal
[--help] [-j | --threads=число_потоков] [--batch-size=размер_порции] [--prefetch-
dir=путь] [--wal-archive-dir=путь_каталога]
[параметры_ssh] [параметры_журнала] [параметры_s3] [параметры_буферизации]
```

Копирует файлы WAL из соответствующего подкаталога каталога резервных копий в каталог журнала предзаписи кластера. Эта команда автоматически устанавливается программой pg_probackup3 в значении параметра restore_command при восстановлении архивных копий с применением архива WAL. Устанавливать её вручную не нужно, если вы используете для копий локальное хранилище или удалённый режим.

Если вы используете интерфейс S3, для обеспечения доступа сервера Postgres Pro к файлам WAL во время восстановления вы можете указать параметр --config-file, определяющий файл конфигурации S3 с требуемыми параметрами конфигурации, как описано в Подразделе «Общие параметры».

По умолчанию архив WAL хранится в каталоге *каталог_копий*/wal/*имя_экземпляра*. Чтобы указать пользовательский каталог, используйте параметр --wal-archive-dir. Тип хранилища (локальная файловая система, SFTP или S3) должен соответствовать конфигурации экземпляра. Указывать необходимо уже существующий каталог.

Postgres Pro запрашивает сегменты WAL по одному. Для ускорения восстановления вы можете воспользоваться параметром --batch-size, определяющим размер порции из нескольких копируемых сегментов WAL. Вместе с параметром --batch-size также можно применить указание -j, чтобы порции сегментов копировались в несколько потоков.

За подробностями обратитесь к подразделам Общие параметры, Параметры архивирования и Параметры сжатия.

archive-push

```
pg_probackup3 archive-push -B каталог_копий --instance=имя_экземпляра
--wal-file-name=имя_файла_wal [--wal-file-path=путь_файлов_wal]
[--help] [--no-sync] [--overwrite] [--wal-archive-dir=путь_каталога]
[--archive-timeout=время_ожидания]
[--compress-algorithm=алгоритм_сжатия]
[--compress-level=уровень_сжатия]
[-j | --threads= число_потоков] [--batch-size=размер_порции]
[параметры_ssh] [параметры_журнала]
[параметры_s3] [параметры_буферизации]
```

Копирует файлы WAL в соответствующий подкаталог каталога копий, проверяя целевой экземпляр по *имени_экземпляра* и значению `system-identifier`. Если параметры экземпляра резервной копии и кластера не совпадают, операция копирования не выполняется, и выдаётся ошибка: `Refuse to push WAL segment segment_name into archive. Instance parameters mismatch`. (Отказано в помещении сегмента имя_сегмента в архив. Параметры экземпляра не совпадают.)

По умолчанию архив WAL хранится в каталоге *каталог_копий*/wal/*имя_экземпляра*. Чтобы указать пользовательский каталог, используйте параметр `--wal-archive-dir`. Тип хранилища (локальная файловая система, SFTP или S3) должен соответствовать конфигурации экземпляра. Указывать необходимо уже существующий каталог.

Если файлы, которые требуется копировать, уже имеются в каталоге копий, pg_probackup3 вычисляет и сравнивает их контрольные суммы. В случае совпадения контрольных сумм `archive-push` пропускает соответствующий файл и выдаёт код успешного завершения. Если же они не совпадают, операция `archive-push` завершается с ошибкой.

Содержимое каждого файла копируется во временный файл с расширением `.part`. Если такой временный файл уже существует, pg_probackup3 ждёт, что он исчезнет в течение заданного параметром `archive_timeout` времени, а если этого не происходит, отбрасывает его. После переноса содержимого выполняется атомарная операция переименования. Тем самым гарантируется, что в случае ошибки команды `archive-push` непрерывное архивирование не остановится и что при параллельном архивировании WAL из разных источников в один архив повреждение архива исключено.

Postgres Pro запрашивает сегменты WAL по одному. Для ускорения архивирования вы можете воспользоваться параметром `--batch-size`, определяющим размер порции из нескольких копируемых сегментов WAL. Вместе с параметром `--batch-size` также можно применить указание `-j`, чтобы порции копировались в несколько потоков.

Сегменты WAL, копируемые в архив, по умолчанию (без указания флага `--no-sync`) гарантированно сбрасываются на диск.

Команду `archive-push` можно использовать в значении параметра *archive_command* Postgres Pro при настройке непрерывного архивирования WAL.

За подробностями обратитесь к подразделам Общие параметры, Параметры архивирования и Параметры сжатия.

backup

```
pg_probackup3 backup -B каталог_копий --instance=имя_экземпляра -b режим_копирования -
s источник_данных -i ид_резервной_копии
[--with-file-map] [--help] [--progress] [-j число_потоков]
[--num-write-threads число_потоков] [--num-validate-threads число_потоков]
[--num-segments] [--create-slot] [--transfer-mode]
[--no-validate] [--skip-block-validation]
```

[--archive-timeout=*время_ожидания*] [--external-dirs=*путь_внешнего_каталога*]
[--no-sync] [--note=*заметка_к_копии*]
[*параметры_подключения*] [*параметры_сжатия*] [*параметры_ssh*]
[*параметры_закрепления*] [*параметры_журнала*] [*параметры_s3*] [*параметры_буферизации*]

Создаёт копию экземпляра Postgres Pro.

-b *режим*
--backup-mode=*режим*

Выбирает режим резервного копирования. Поддерживаются следующие режимы: FULL, DELTA и PTRACK.

-s *источник_копирования*
--backup-source=*источник_копирования*

Указывает источник данных для резервного копирования. Возможные значения: DIRECT, BASE и PRO.

--num-segments *количество_сегментов*

Задаёт количество сегментов резервной копии при её создании или объединении. Должен быть положительным целым числом.

> **Примечание**
>
> Если указанное значение превышает системное ограничение на количество одновременно открытых файлов, процесс завершится с ошибкой «too many open files» (слишком много открытых файлов).

--num-write-threads *число_потоков*

Указывает количество потоков для копирования файлов. Переопределяет параметр j/--threads для копирования файлов.

--num-validate-threads *число_потоков*

Указывает количество потоков для проверки резервной копии. Переопределяет параметр j/--threads для проверки резервной копии.

-C
--smooth-checkpoint

Растягивает выполнение контрольной точки во времени. По умолчанию pg_probackup3 пытается произвести контрольную точку максимально быстро.

--stream

Создаёт потоковую резервную копию (STREAM), включая в неё все необходимые файлы WAL, получаемые от сервера по протоколу репликации.

--backup-pg-log

Включает в резервную копию каталог log. Этот каталог обычно содержит журналы сообщений сервера. По умолчанию каталог log в копию не включается.

-E *путь_внешнего_каталога*
--external-dirs=*путь_внешнего_каталога*

Включает в создаваемую копию указанный каталог, рекурсивно копируя его содержимое в отдельный подкаталог каталога резервной копии. Этот параметр полезен для архивирования скриптов, SQL-дампов и файлов конфигурации, расположенных вне каталога данных. Если вы хотите архивировать несколько внешних каталогов, их пути нужно разделять двоеточием в Linux или точкой с запятой в Windows.

`--archive-timeout=`*время_ожидания*

Задаёт тайм-аут для архивирования сегментов WAL и потоковой передачи (в секундах). По умолчанию pg_probackup3 ждёт выполнения этих операций 300 секунд.

`--skip-block-validation`

Отключает проверку контрольных сумм на уровне блоков в процессе резервного копирования.

`--no-validate`

Пропускает автоматическую проверку созданной резервной копии. Этот ключ может быть полезен, если вы регулярно проверяете резервные копии и хотите сократить время создания копии.

Рекомендуется использовать этот флаг при создании резервной копии в хранилище S3. Из-за некоторых особенностей хранилищ S3 автоматическая проверка в этом случае может оказаться некорректной. Пропустите её, а затем выполните проверку с помощью отдельной команды validate.

`--no-sync`

Не сбрасывать копируемые файлы на диск. Этот флаг позволяет несколько ускорить процесс копирования. Использование этого флага может привести к повреждению данных в случае аварии операционной системы или аппаратного сбоя. Если вы используете его, рекомендуется выполнить команду validate сразу после завершения копирования, чтобы убедиться в отсутствии проблем.

`--note=`*заметка_к_копии*

Задаёт текстовую заметку для резервной копии. Если *заметка_к_копии* содержит символы перевода строки, сохранена будет только подстрока до первого перевода строки. Максимальный размер заметки равен 1 КБ. Значение `'none'` удаляет текущую заметку.

`--with-file-map`

Включает создание файлов сопоставления. Необходим для команды fuse.

`--transfer-mode=`*режим_передачи*

Указывает способ передачи данных с сервера в приложение.

> **Примечание**
>
> Этот параметр доступен только в режиме источника данных PRO.

Возможные значения:

- `raw` — данные передаются в несжатом виде блоками произвольного размера.
- `packed` — данные передаются в упакованном виде блоками по 128 КБ с общим заголовком.

`packed` — значение по умолчанию (рекомендуемое).

За подробной информацией о параметрах команды обратитесь к подразделам Общие параметры, Параметры соединения, Параметры закрепления, Параметры SSH, Параметры сжатия и Параметры ведения журнала.

За подробностями обратитесь к подразделу Создание резервной копии.

catchup

```
pg_probackup3 catchup -b режим_синхронизации
--destination-pgdata=путь_к_целевому_каталогу_данных
[--temp-slot] [-P | --perm-slot] [-S | --slot=имя_слота] [-j | --threads=число_потоков]
```

```
[--num-write-threads число_потоков] [--num-validate-threads число_потоков]
[-x | --exclude-path=префикс_пути]
[-T СТАРЫЙ_КАТАЛОГ=НОВЫЙ_КАТАЛОГ] [параметры_подключения] [параметры_журнала]
 [параметры_сжатия]
```

Создаёт копию экземпляра Postgres Pro, не используя каталог резервных копий.

`-b` *режим_синхронизации*
`--backup-mode=`*режим_синхронизации*

Выбирает режим синхронизации. Поддерживаются следующие режимы: `FULL`, `DELTA` и `PTRACK`.

`--destination-pgdata=`*путь_к_целевому_каталогу_данных*

Задаёт путь к локальному целевому каталогу данных.

`-j` *число_потоков*
`--threads=`*число_потоков*

Задаёт число параллельных потоков для процесса `catchup`.

`--num-write-threads` *число_потоков*

Указывает количество потоков для копирования файлов. Переопределяет параметр `j`/`--threads` для копирования файлов.

`--num-validate-threads` *число_потоков*

Указывает количество потоков для проверки резервной копии. Переопределяет параметр `j`/`--threads` для проверки резервной копии.

`--stream`

Копирует экземпляр в режиме доставки WAL STREAM, при котором все необходимые файлы WAL передаются с исходного сервера по протоколу репликации.

`-x=`*префикс_пути*
`--exclude-path=`*префикс_пути*

Определяет префикс для файлов, которые не будут копироваться при синхронизации экземпляров Postgres Pro. Такой префикс должен содержать путь относительно каталога данных экземпляра. Если в префиксе указан каталог, ни один файл в этом каталоге не будет копирован.

> **Предупреждение**
>
> Используйте этот параметр с осторожностью, поскольку исключение файлов может привести к неполной синхронизации.

`--temp-slot`

Создаёт *временный* слот физической репликации для передачи WAL с копируемого экземпляра Postgres Pro. Это гарантирует, что все нужные сегменты WAL будут доступны, если в процессе копирования произойдёт переключение сегментов WAL. Этот параметр нельзя использовать с параметром `--perm-slot`. По умолчанию имя временного слота — `pg_probackup3_wal_streaming_`*идентификатор_процесса*, где *идентификатор_процесса* — PID Postgres Pro. Чтобы его поменять, воспользуйтесь параметром `--slot`/`-S` и явно укажите `--temp-slot`.

`-P`
`--perm-slot`

Создаёт *постоянный* слот физической репликации для передачи WAL с копируемого экземпляра Postgres Pro. Этот параметр нельзя использовать с параметром `--temp-slot`. По умолчанию

имя постоянного слота — `pg_probackup_perm_slot`, но его можно поменять, воспользовавшись параметром `--slot`/`-S`.

`-S` *имя_слота*
`--slot=`*имя_слота*

Задаёт, к какому слоту репликации подключаться для передачи WAL. Этот параметр можно указать только вместе с параметром `--stream`.

`-T` *СТАРЫЙ_КАТАЛОГ=НОВЫЙ_КАТАЛОГ*
`--tablespace-mapping=`*СТАРЫЙ_КАТАЛОГ=НОВЫЙ_КАТАЛОГ*

Перемещает табличное пространство из каталога *СТАРЫЙ_КАТАЛОГ* в *НОВЫЙ_КАТАЛОГ* во время восстановления. И *СТАРЫЙ_КАТАЛОГ*, и *НОВЫЙ_КАТАЛОГ* должны задаваться абсолютными путями. Если путь содержит знак равно (=), экранируйте этот знак обратной косой чертой. Данный параметр может указываться неоднократно для перемещения нескольких табличных пространств.

del-instance

```
pg_probackup3 del-instance -B каталог_копий --instance=имя_экземпляра [параметры_s3]
 [--help]
[параметры_ssh] [параметры_журнала] [параметры_буферизации]
```

Удаляет все резервные копии и файлы WAL, связанные с указанным экземпляром.

За подробностями о параметрах команды обратитесь к подразделу Общие параметры.

delete

```
pg_probackup3 delete -B каталог_копий --instance=имя_экземпляра -i ид_резервной_копии
[--help] [--progress] [--status=статус_резервной_копии]
[--dry-run] [параметры_журнала] [параметры_ssh]
[параметры_s3] [параметры_буферизации]
```

Удаляет резервные копии с указанными *ид_резервной_копии*.

`--dry-run`

Выполняет пробный запуск команды `delete`, который не вносит никаких реальных изменений: файлы на диске не удаляются. Этот флаг также позволяет проверить правильность всех параметров команды и её готовность к запуску.

`--status`

Позволяет удалять все резервные копии с определённым состоянием.

За подробностями обратитесь к подразделу Удаление резервных копий.

file-map

```
pg_probackup3 file-map -B каталог_копий --instance=имя_экземпляра -i ид_резервной_копии
```

Включает создание файлов сопоставления для существующей цепочки резервных копий.

Существующие файлы сопоставления для указанных резервных копий будут заменены новыми версиями.

fuse

```
pg_probackup3 fuse -B каталог_копий --mnt-path=путь_монтирования --
instance=имя_экземпляра
-i ид_резервной_копии [--cache-swap-size=порог_сброса_кеша] [--cache-dir=каталог_кеша]
 [--detach] [--unmount] [--help]
[параметры_ssh] [параметры_журнала] [параметры_s3] [параметры_буферизации]
```

Монтирует каталог резервных копий в виде виртуальной файловой системы и позволяет Postgres Pro работать на ней. Более детальное описание процесса представлено в Раздел 3.7.

`--cache-swap-size`

Задаёт объём данных (в МБ), хранящихся в оперативной памяти. Значение по умолчанию — 128 МБ. При превышении этого размера изменения сбрасываются на ближайший диск. Это позволяет работать со снимком состояния базы данных, не изменяя исходную резервную копию. Кеш очищается после остановки сервера Postgres Pro.

`--cache-dir=`*`каталог_кеша`*

Указывает путь к каталогу кеша FUSE. Если этот параметр опущен, используется системный временный каталог.

`--detach`

Запускает файловую систему FUSE в фоновом режиме (по умолчанию работает не в фоновом режиме). Это полезно для автоматизации.

`--unmount`

Демонтирует файловую систему FUSE, которая была ранее смонтирована с помощью команды `fuse`. Для этого параметра требуется указать только путь монтирования (параметр `--mnt-path`), который должен совпадать с путём, использовавшимся при монтировании.

help

`pg_probackup3 help [`*`command`*`]`

Выдаёт справку по командам pg_probackup3. Если в параметрах задаётся одна из команд pg_probackup3, выводит подробную информацию по параметрам, которые принимает эта команда.

init

`pg_probackup3 init -B` *`каталог_копий`* `[--skip-if-exists] [`*`параметры_s3`*`] [--help] [`*`параметры_ssh`*`][`*`параметры_журнала`*`] [`*`параметры_буферизации`*`]`

Инициализирует *`каталог_копий`*, в котором будут храниться резервные копии, архив WAL и метаинформация о скопированных кластерах баз данных. Если заданный *`каталог_копий`* уже существует, он должен быть пустым. В противном случае pg_probackup3 выдаст соответствующее сообщение об ошибке. Можно отключить вывод этого сообщения, указав `--skip-if-exists`. Хотя каталог не будет инициализирован, приложение вернёт 0.

За подробностями обратитесь к подразделу Инициализация каталога резервных копий. Более подробно о параметрах команды рассказывается в подразделе Общие параметры.

merge

`pg_probackup3 merge -B` *`каталог_копий`* `--instance=`*`имя_экземпляра`* `-i` *`ид_резервной_копии`* `--merge-from-id=`*`объединить_от`* `--merge-interval=`*`интервал_объединения`*
`[-t | --target-backup-id=`*`ид_резервной_копии`*`] [-j` *`число_потоков`*`] [--progress] [--no-validate] [--no-sync]`
`[--keep-backups] [--dry-run] [--help][`*`параметры_журнала`*`] [`*`параметры_ssh`*`] [`*`параметры_s3`*`] [`*`параметры_буферизации`*`]`

Объединяет копии, относящиеся к одной цепочке инкрементальных копий. Если выбрана полная копия, она будет объединена с первой инкрементальной копией после неё. Если выбрана инкрементальная копия, она будет объединена с родительской полной копией, включая все инкрементальные копии между ними. После выполнения команды полная копия содержит все объединённые данные, а инкрементальные копии удаляются как ненужные. Вы также можете объединять цепочки инкрементальных копий, указав первую и последнюю резервные копии или интервал (в часах) от момента создания первой резервной копии.

`--no-validate`

Пропускает автоматическую проверку до и после объединения.

`--no-sync`

Не сбрасывать объединяемые файлы на диск. Этот флаг позволяет несколько ускорить процесс объединения. Использование этого флага может привести к повреждению данных в случае аварии операционной системы или аппаратного сбоя.

`-t`
`--target-backup-id`

Задаёт идентификатор объединённой резервной копии.

`--keep-backups`

Сохраняет исходные резервные копии после объединения.

`--merge-from-id`

Указывает идентификатор первой резервной копии в цепочке копий для объединения.

`--merge-interval`

Задаёт время (в часах) перед объединением цепочки инкрементальных резервных копий.

`--with-file-map`

Включает создание файлов сопоставления. Необходим для команды fuse.

За подробностями обратитесь к подразделам Общие параметры и Объединение резервных копий.

restore

`pg_probackup3 restore -B` *`каталог_копий`* `--instance=`*`имя_экземпляра`*
`[--help] [-D` *`каталог_данных`*`] [-i` *`ид_резервной_копии`*`] [--wal-archive-dir=`*`путь_каталога`*`]`
`[--progress] [-T` *`СТАРЫЙ_КАТАЛОГ=НОВЫЙ_КАТАЛОГ`*`]`
`[--external-mapping=`*`СТАРЫЙ_КАТАЛОГ=НОВЫЙ_КАТАЛОГ`*`] [--skip-external-dirs]`
`[-j` *`число_потоков`*`] [--num-validate-threads` *`число_потоков`*`]`
`[-R | --restore-as-replica] [--no-validate] [--skip-block-validation]`
`[--no-sync] [--restore-command=`*`команда`*`]`
`[--primary-conninfo=`*`строка_подключения`*`]`
`[ --primary-slot-name=`*`имя_слота`*`]`
[*`параметры_точки_восстановления`*] [*`параметры_журнала`*]
[*`параметры_ssh`*] [*`параметры_s3`*] [*`параметры_буферизации`*]

Восстанавливает экземпляр Postgres Pro из резервной копии, расположенной в каталоге *`каталог_копий`*.

> **Примечание**
>
> В то время как файлы резервных копий могут передаваться из разных источников (файловая система, S3 или SSH SFTP), восстановление каталога данных `PGDATA` сервера Postgres Pro производится в локальную файловую систему.

> **Примечание**
>
> Команда `restore` пока не поддерживает параметр `--threads`. Число потоков равняется числу сегментов резервной копии.

`-R`
`--restore-as-replica`

Создаёт минимальный файл конфигурации восстановления для облегчения настройки ведомого сервера. Если для соединения репликации требуется пароль, его нужно дополнительно за-

дать вручную в параметре *primary_conninfo*. pg_probackup3 сохраняет эти параметры в файле `probackup_recovery.conf` и при запуске кластера включает их в `postgresql.auto.conf`.

`--primary-conninfo=`*`строка_подключения`*

Устанавливает заданное значение для параметра *primary_conninfo*. Это значение учитывается только при использовании флага `-R`.

Пример: `--primary-conninfo="host=192.168.1.50 port=5432 user=foo password=foopass"`

`--primary-slot-name=`*`имя_слота`*

Устанавливает заданное значение для параметра *primary_slot_name*. Это значение учитывается только при использовании флага `-R`.

`-T` *`СТАРЫЙ_КАТАЛОГ=НОВЫЙ_КАТАЛОГ`*
`--tablespace-mapping=`*`СТАРЫЙ_КАТАЛОГ=НОВЫЙ_КАТАЛОГ`*

Перемещает табличное пространство из каталога *`СТАРЫЙ_КАТАЛОГ`* в *`НОВЫЙ_КАТАЛОГ`* во время восстановления. И *`СТАРЫЙ_КАТАЛОГ`*, и *`НОВЫЙ_КАТАЛОГ`* должны задаваться абсолютными путями. Если путь содержит знак равно (=), экранируйте этот знак обратной косой чертой. Данный параметр может указываться неоднократно для перемещения нескольких табличных пространств.

`--external-mapping=`*`СТАРЫЙ_КАТАЛОГ=НОВЫЙ_КАТАЛОГ`*

Перемещает внешний каталог, включённый в резервную копию, из каталога *`СТАРЫЙ_КАТАЛОГ`* в *`НОВЫЙ_КАТАЛОГ`* во время восстановления. И *`СТАРЫЙ_КАТАЛОГ`*, и *`НОВЫЙ_КАТАЛОГ`* должны задаваться абсолютными путями. Если путь содержит знак равно (=), экранируйте этот знак обратной косой чертой. Данный параметр может указываться неоднократно для нескольких каталогов.

`--skip-external-dirs`

Пропускать внешние каталоги, включённые в резервную копию указанием `--external-dirs`. Содержимое этих каталогов не будет восстановлено.

`--skip-block-validation`

Отключает проверку контрольных сумм на уровне блоков для ускорения проверки целостности. При автоматической проверке перед восстановлением будут проверяться только контрольные суммы на уровне файлов.

`--no-validate`

Пропускает проверку резервного копирования. Этот ключ может быть полезен, если вы регулярно проверяете копии и хотите сократить время восстановления данных.

`--restore-command=`*`команда`*

Задаёт значение для параметра *restore_command*. Например: `--restore-command='cp /mnt/server/archivedir/%f "%p"'`

`--wal-archive-dir=`*`путь_каталога`*

Задаёт каталог для архива WAL.

По умолчанию архив WAL хранится в каталоге *`каталог_копий`*`/wal/`*`имя_экземпляра`*. Чтобы указать пользовательский каталог, используйте параметр `--wal-archive-dir`. Тип хранилища (локальная файловая система, SFTP или S3) должен соответствовать конфигурации экземпляра. Указывать необходимо уже существующий каталог.

`--no-sync`

Не сбрасывать восстанавливаемые файлы на диск. Этот флаг позволяет несколько ускорить процесс восстановления. Использование этого флага может привести к повреждению данных в

случае аварии операционной системы или аппаратного сбоя. Если такое событие произойдёт, вам потребуется запустить команду restore ещё раз.

За подробной информацией о параметрах команды обратитесь к подразделам Общие параметры, Параметры точки восстановления, Параметры SSH, Параметры удалённого архива WAL, Параметры ведения журнала.

За подробностями обратитесь к подразделу Восстановление кластера.

retention

```
pg_probackup3 retention -B каталог_копий --instance=имя_экземпляра
[--retention-redundancy] [--retention-window] [--dry-run] [--merge-expired] [--wal-
archive-dir=путь_каталога]
[--delete-expired] [--delete-wal] [параметры_закрепления]
[параметры_ssh] [параметры_s3] [параметры_буферизации]
```

Устанавливает политику хранения резервных копий в экземпляре или каталоге и запускает процесс удаления или объединения резервных копий согласно указанным параметрам.

По умолчанию архив WAL хранится в каталоге *каталог_копий*/wal/*имя_экземпляра*. Чтобы указать пользовательский каталог, используйте параметр `--wal-archive-dir`. Тип хранилища (локальная файловая система, SFTP или S3) должен соответствовать конфигурации экземпляра. Указывать необходимо уже существующий каталог.

За подробностями о параметрах команды обратитесь к подразделу Параметры сохранения.

send-backup

```
pg_probackup3 send-backup -B каталог_копий --instance=имя_экземпляра
-i ид_резервной_копии -p порт -h сервер [--no-merge]
```

Передаёт данные резервных копий в удалённую систему через указанный порт с использованием многопоточности.

Выполняется на локальной машине, содержащей каталог резервных копий.

Перед передачей данных цепочка резервных копий объединяется во временный файл, который удаляется после завершения передачи. Объединение можно отключить с помощью флага `--no-merge` — в этом случае данные будут передаваться в исходном виде.

За подробностями обратитесь к разделу Удалённое восстановление.

server-info

```
pg_probackup3 server-info [--format=plain|json] [параметры_подключения]
```

Отображает доступность параметров резервного копирования для текущего экземпляра Postgres Pro. Включает основную информацию об экземпляре и версию pg_probackup3. Пример вывода:

```
CAN_USE_API = true
CAN_USE_CLI = true
CAN_USE_S3 = true
CAN_USE_CFS = true
CAN_USE_FUSE = true
CAN_USE_DIRECT = true
CAN_USE_BASE = true
CAN_USE_PRO = true
CAN_USE_MULTITHREAD = true
IS_PRO_MODE_CONFIGURED = true
IS_DELTA_AVAILABLE = true
IS_PTRACK_CONFIGURED = true
```

```
IS_PAGE_AVAILABLE = false
IS_WALSUM_AVAILABLE = false
IS_PGDATA_ACCESS = true
SERVER_CHECKSUM_ENABLED = true
PG_VERSION = 170006
PG_VERSION_STRING = "Postgres Pro (enterprise) 17.6.1 on aarch64-apple23.6.0, compiled
 by Homebrew clang version 21.1.2, 64-bit"
PG_EDITION_STRING = "enterprise"
PGDATA = "/Users/username/projects/postgrespro/tmp_install/data"
TIMELINE = 1
SYSTEM_ID = 75800766633445207
PG_MAX_WAL_SENDERS = 20
PG_BLOCK_SIZE = 8192
PG_BLOCKS_IN_SEGMENT = 131072
PG_WAL_BLOCK_SIZE = 8192
PG_WAL_SEGMENT_SIZE = 16777216
PG_TABLESPACE_MAP = "[]"
APP_VERSION = 3.2.0
```

По умолчанию вывод отображается в виде простого текста. Укажите параметр `--format=json`, чтобы получить результат в формате JSON.

Параметры типа `CAN_USE_*` отражают лицензионные ограничения и показывают, какие функции доступны в текущей версии Postgres Pro.

set-backup

```
pg_probackup3 set-backup -B каталог_копий --instance=имя_экземпляра -
i ид_резервной_копии
{--ttl=время_жизни | --expire-time=время}
[--note=заметка_к_копии] [параметры_ssh]
[параметры_s3] [--help] [параметры_журнала] [параметры_буферизации]
```

Устанавливает заданные для конкретной резервной копии параметры в конфигурационном файле `backup.control` или изменяет ранее определённые значения.

`--note=`*заметка_к_копии*

Задаёт текстовую заметку для резервной копии. Если *заметка_к_копии* содержит символы перевода строки, сохранена будет только подстрока до первого перевода строки. Максимальный размер заметки равен 1 КБ. Значение *'none'* удаляет текущую заметку.

За подробностями о параметрах команды обратитесь к подразделам Общие параметры и Параметры закрепления.

set-config

```
pg_probackup3 set-config -B каталог_копий --instance=имя_экземпляра
[--help] [--pgdata=путь_к_pgdata] [--wal-archive-dir=путь_каталога]
[--retention-redundancy=избыточность][--retention-window=окно]
[параметры_сжатия] [параметры_подключения]
[--archive-timeout=время_ожидания] [--external-dirs=путь_внешнего_каталога]
[параметры_журнала] [параметры_ssh] [параметры_буферизации]
```

Добавляет заданные параметры соединения, сжатия, хранения, ведения журнала и указания внешних каталогов в конфигурационный файл `pg_probackup3.conf` либо изменяет ранее заданные значения.

По умолчанию архив WAL хранится в каталоге *каталог_копий*`/wal/`*имя_экземпляра*. Чтобы указать пользовательский каталог, используйте параметр `--wal-archive-dir`. Тип хранилища (локальная файловая система, SFTP или S3) должен соответствовать конфигурации экземпляра. Указывать необходимо уже существующий каталог.

Все поддерживаемые параметры описываются в подразделе Параметры.

Редактировать `pg_probackup3.conf` вручную **не рекомендуется**.

show

```
pg_probackup3 show3 -B каталог_копий
[--help] [--instance=имя_экземпляра [-i ид_резервной_копии | --archive [--wal-archive-
dir=путь_каталога]]]
[--show-log] [--format=plain|json] [--no-color] [--format=plain|json|tree]
[параметры_s3] [параметры_ssh]
[параметры_журнала] [параметры_буферизации]
```

Показывает содержимое каталога резервных копий. Если заданы *имя_экземпляра* и *ид_резервной_копии*, выводится подробная информация об этой копии. С указанием флага `--archive` эта команда отображает содержимое архива WAL, который по умолчанию находится в *каталог_копий*`/wal/`*имя_экземпляра*. Для указания пользовательского каталога используйте параметр `--wal-archive-dir`.

По умолчанию содержимое каталога представляется в виде обычного текста. Вы можете передать параметр `--format=json`, чтобы получить результат в формате JSON. С параметром `--no-color` выводимые сообщения не выделяются цветом. Можно также использовать параметр `--format=tree`, чтобы посмотреть список резервных копий в виде дерева.

Более подробно использование этой команды описывается в подразделах Управление каталогом резервных копий и Просмотр оглавления архива WAL.

show-config

```
pg_probackup3 show-config -B каталог_копий --instance имя_экземпляра
[--format=plain|json] [параметры_s3] [параметры_ssh]
[параметры_журнала] [параметры_буферизации]
```

Выводит все текущие параметры конфигурации pg_probackup3, в том числе те, что содержатся в файле `pg_probackup3.conf`, размещённом в каталоге *каталог_копий*`/backups/`*имя_экземпляра*, и те, что были заданы в командной строке. По умолчанию параметры конфигурации выводятся обычным текстом.

Чтобы изменить содержимое `pg_probackup3.conf`, используйте команду set-config.

validate

```
pg_probackup3 validate -B каталог_копий
[--help] [--instance=имя_экземпляра] [-i ид_резервной_копии]
[-j число_потоков] [--progress]
[--skip-block-validation] [параметры_буферизации]
[параметры_журнала] [параметры_ssh][параметры_s3]
```

Проверяет наличие и целостность всех файлов, необходимых для восстановления кластера. Если *имя_экземпляра* не задаётся, pg_probackup3 проверяет все резервные копии, имеющиеся в каталоге копий.

Если указать параметр `--progress`, в процессе проверки будет выводиться список файлов и каталогов резервной копии.

version

```
pg_probackup3 version
```

Выводит версию pg_probackup3.

При указании `--format=json` вывод будет получен в формате JSON. Это может потребоваться для внутренней интеграции с приложениями на основе JSON, например PPEM. Пример вывода JSON:

```
pg_probackup3 version --format=json
{
    "pg_probackup3":
    {
        "version": "3.0.0",
    },
    "compressions": [zlib, lz4, zstd]
}
```

Параметры

В этом подразделе описываются параметры командной строки для команд pg_probackup3. Если какое-либо значение параметра может быть получено из переменной окружения, имя этой переменной указывается в верхнем регистре ниже параметра командной строки. Некоторые значения могут быть получены из файла конфигурации `pg_probackup3.conf`, находящегося в каталоге копий.

За подробностями обратитесь к Разделу 2.4.

Если некоторый параметр задаётся несколькими способами, значение в командной строке имеет наивысший приоритет, а значение в `pg_probackup3.conf` — наименьший.

Общие параметры

Ниже приведён список параметров общего характера.

`--dry-run`

Выполняет пробный запуск нужной команды, который не вносит никаких реальных изменений: не создаются, не удаляются и не перемещаются файлы на диске. Этот флаг также позволяет проверить правильность всех параметров команды и её готовность к запуску. С `--dry-run` пропускается потоковая трансляция WAL.

`-B` *каталог*
`--backup-path=`*каталог*
`BACKUP_PATH`

Задаёт абсолютный путь к каталогу копий. Каталог копий — это каталог, в котором хранятся все файлы резервных копий и метаинформация. Поскольку это расположение необходимо задавать почти для всех команд pg_probackup3, имеет смысл указать его один раз в переменной окружения `BACKUP_PATH`. В этом случае каждый раз указывать этот путь в командной строке не нужно.

`-D` *каталог*
`--pgdata=`*каталог*
`PGDATA`

Задаёт абсолютный путь к каталогу данных кластера. Этот параметр является обязательным только для команды add-instance. Другие команды могут получать этот путь из переменной окружения `PGDATA` или из файла конфигурации `pg_probackup3.conf`.

`-i` *ид_резервной_копии*
`--backup-id=`*ид_резервной_копии*

Задаёт уникальный идентификатор резервной копии.

`--parent-backup-id=`*ид_родительской_копии*

Указывает уникальный идентификатор родительской копии (используется при инкрементальном копировании).

`--from-full`

Создаёт инкрементальную резервную копию от последней родительской полной копии.

`-j` *число_потоков*
`--threads=`*число_потоков*

Задаёт число параллельных потоков, запускаемых командами `backup`, `restore`, `merge`, `validate` и `archive-push`. По умолчанию равно числу ядер процессора.

`--num-validate-threads` *число_потоков*

Задаёт число параллельных потоков, запускаемых во время проверки резервных копий, например, при выполнении команд `backup` или `restore`.

Ключ `--no-validate` отключает действие параметра `--num-validate-threads`.

`--progress`

Включает вывод прогресса выполнения операций.

`--help`

Выводит подробную информацию по параметрам, которые принимает эта команда.

`-v` *версия*
`--version=`*версия*

Показывает версию pg_probackup3.

`--config-file=`*имя_файла*

Задаёт файл конфигурации S3 или SSH. Параметры, заданные в этом файле, переопределяют переменные окружения.

Чтобы создать файл конфигурации, выполните команду set-config с параметром `--config-file`.

В созданном файле будут указаны все явно заданные параметры S3 или SSH, при этом пароли исключаются и отображаются в виде звёздочек.

Пример файла конфигурации S3:

```
access-key=admin
s3=on
s3-bucket=test
s3-host=127.0.0.1
s3-port=9000
s3-region=us-west-2
secret-key=***
```

Если файл конфигурации содержит параметры и S3, и SSH, будут использоваться параметры S3.

Если параметр `--config-file` опускается, pg_probackup3 ищет файлы конфигурации S3 и SSH сначала в `/etc/pg_probackup/s3.config` или `/etc/pg_probackup/ssh.config`, а затем в `~postgres/.pg_probackup/s3.config` или `~postgres/.pg_probackup/ssh.config` соответственно.

Параметры точки восстановления

Если настроено непрерывное архивирование WAL, вы можете передать один из этих параметров с командой restore, чтобы указать момент, до которого должен быть проверен кластер баз данных.

`--recovery-target-stop=immediate|latest`

Определяет, когда остановить восстановление:

- Со значением `immediate` восстановление завершается сразу после достижения согласованного состояния выбранной копиин. Такое поведение по умолчанию применяется для копий типа STREAM.

- Со значением `latest` восстановление продолжается до тех пор, пока не будут применены все имеющиеся в архиве сегменты WAL. При указании этого значения для параметра `--recovery-target` такое же значение задаётся для параметра `--recovery-target-timeline`.

`--recovery-target-timeline=`*`линия_времени`*

Выбирает линию времени для восстановления:

- `current` — линия времени указанной резервной копии. Это значение по умолчанию.
- `latest` — линия времени последней доступной резервной копии.
- Числовое значение.

`--recovery-target-lsn=`*`lsn`*

Указывает последовательный номер в журнале предзаписи, до которого будет производиться восстановление.

`--recovery-target-name=`*`имя_цели_восстановления`*

Указывает именованную точку сохранения, вплоть до которой будет восстановлен кластер.

`--recovery-target-time=`*`время`*`|current|latest`

Указывает точку времени, вплоть до которой будет производиться восстановление. Если часовой пояс не указывается, подразумевается местное время.

Например: `--recovery-target-time="2027-04-09 18:21:32+00"`

`--recovery-target-xid=`*`ид_транзакции`*

Указывает идентификатор транзакции, вплоть до которой будет производиться восстановление.

`--recovery-target-inclusive=`*`boolean`*

Указывает на необходимость остановки сразу после (`true`) либо до (`false`) достижения целевой точки. Этот параметр можно использовать только вместе с параметром `--recovery-target-time`, `--recovery-target-lsn` или `--recovery-target-xid`. Значение по умолчанию определяется параметром *recovery_target_inclusive*.

`--recovery-target-action=pause|promote|shutdown`

Задаёт действие (*recovery_target_action*), которое должен выполнить сервер по достижении цели восстановления.

По умолчанию: `pause`

Параметры сохранения

Эти параметры используются с командой retention.

Подробнее о политике хранения рассказывается в подразделе Настройка политики хранения.

`--retention-redundancy=`*`избыточность`*

Указывает, сколько полных резервных копий должно сохраняться в каталоге данных. Должно быть неотрицательным целым числом. Ноль отключает сохранение.

По умолчанию: `0`

`--retention-window=`*`окно`*

Указывает количество дней, в течение которого возможно восстановление. Должно быть неотрицательным целым числом. При нулевом значении окно восстановления отсутствует.

По умолчанию: `0`

`--delete-wal`

Удаляет файлы WAL, которые не являются необходимыми для восстановления кластера из имеющихся резервных копий.

`--delete-expired`

Удаляет резервные копии, не удовлетворяющие политике сохранения, определённой в файле конфигурации `pg_probackup3.conf`.

`--merge-expired`

Объединяет самую старую инкрементальную копию, удовлетворяющую требованиям политики хранения, с её родительскими копиями, срок хранения которых истёк.

Параметры закрепления

Эти параметры могут использоваться с командами backup, set-backup и retention.

За подробностями обратитесь к подразделу Закрепление резервных копий.

`--ttl=`*`время_жизни`*

Задаёт время, на которое закрепляется резервная копия. Значение должно быть неотрицательным целым. Нулевое значение отменяет установленное ранее закрепление резервной копии. Поддерживаются следующие единицы измерения: ms (миллисекунды), s (секунды), min (минуты), h (часы), d (дни). По умолчанию подразумеваются секунды.

Например: `--ttl=30d`

`--expire-time=`*`время`*

Определяет момент времени, до которого будет храниться резервная копия. Время должно задаваться в формате ISO-8601. Если часовой пояс не указывается, подразумевается местное время.

Например: `--expire-time="2027-04-09 18:21:32+00"`

Параметры ведения журнала

Эти параметры могут использоваться с любой командой.

`--no-color`

Отключает цветовое выделение сообщений уровней `warning` и `error` в консоли.

`--log-level-console=`*`уровень_протоколирования`*

Управляет уровнем сообщений, которые будут выводиться в журнал консоли. Допустимые уровни: `trace`, `debug`, `info`, `warning`, `error` и `off`. Каждый уровень включает все последующие, и с каждым последующим уровнем объём сообщений уменьшается. Вариант `off` отключает вывод в журнал консоли.

По умолчанию: `info`

> **Примечание**
>
> Все выводимые в консоль сообщения журнала передаются через stderr, чтобы их можно было отделить от вывода команд show и show-config.

`--log-level-file=`*`уровень_протоколирования`*

Управляет уровнем сообщений, которые будут выводиться в файл журнала. Допустимые уровни: `trace`, `debug`, `info`, `warning`, `error` и `off`. Каждый уровень включает все последующие, и с каждым последующим уровнем объём сообщений уменьшается. Вариант `off` отключает вывод в файл журнала.

По умолчанию: `off`

`--log-backup=`*`уровень_протоколирования`*

Управляет уровнем сообщений, которые будут выводиться в файл журнала резервного копирования (создаваемый в каталоге резервных копий) во время выполнения команды `backup`. Допустимые уровни: `trace`, `debug`, `info`, `warning`, `error` и `off`. Каждый уровень включает все последующие, и с каждым последующим уровнем объём сообщений уменьшается. Вариант `off` отключает вывод в файл журнала.

По умолчанию: `info`

`--log-filename=`*`файл_журнала`*

Определяет имена для создаваемых файлов журналов. Имена файлов обрабатываются по шаблону `strftime`, так что вы можете использовать спецкоды с % для выбора имён файлов, зависящих от времени.

> **Примечание**
>
> Начиная с PostgreSQL 17, в shell-командах, таких как *`archive_command`*, применяется более строгая проверка местозаполнителей со знаком процента. В этом контексте вместо одинарного `%` следует использовать `%%`. Для получения дополнительной информации обратитесь к *соответствующему коммиту PostgreSQL*.

По умолчанию: `pg_probackup.log`

Например, если задать шаблон `pg_probackup-%%u.log`, pg_probackup3 будет записывать журнал в отдельные файлы по дням недели, и символы `%%u` в имени будут заменяться соответствующим десятичным номером: `pg_probackup-1.log` в понедельник, `pg_probackup-2.log` во вторник и т. д.

Также можно использовать счётчик файлов (`%%N`) и формат `strftime` (pg_probackup-%%Y-%%m-%%d_%%H%%M%%S.log). Примеры показаны в таблице ниже.

Таблица 4.1. Примеры шаблонов имён файлов

Шаблон	**Развёрнутая версия**
file_%%N.log	file_1.log, file_2.log...
file_%%3N.log	file_001.log, file_002.log...
file_%%Y%%m%%d.log	file_20080705.log, file_20080706.log...
file_%%Y-%%m-%%d_%%H-%%M-%%S.%%N.log	file_2008-07-05_13-44-23.1.log, file_2008-07-06_16-00-10.2.log...

Этот параметр действует, если включена запись в журнал (параметром `--log-level-file`).

`--error-log-filename=`*`файл_журнала_событий`*

Определяет имена только для файлов журналов ошибок. Имена файлов обрабатываются по шаблону `strftime`, так что вы можете использовать спецкоды с % для выбора имён файлов, зависящих от времени.

> **Примечание**
>
> Начиная с PostgreSQL 17, в shell-командах, таких как *archive_command*, применяется более строгая проверка местозаполнителей со знаком процента. В этом контексте вместо одинарного `%` следует использовать `%%`. Для получения дополнительной информации обратитесь к *соответствующему коммиту PostgreSQL*.

По умолчанию: none

Например, если задать шаблон `error-pg_probackup-%%u.log`, pg_probackup3 будет записывать журнал в отдельные файлы по дням недели, и символы `%%u` в имени будут заменяться соответствующим десятичным номером: `error-pg_probackup-1.log` в понедельник, `error-pg_probackup-2.log` во вторник и т. д.

Этот параметр полезен для диагностики и решения возникающих проблем.

`--log-directory=`*каталог_журнала*

Определяет каталог, в котором будут создаваться файлы журналов. Вы должны задать в этом параметре абсолютный путь. Этот каталог создаётся только при необходимости, когда в журнал выводится первое сообщение.

Обратите внимание, что каталог для файлов журнала всегда создаётся локально, даже если резервные копии создаются в хранилище S3. Поэтому при необходимости обязательно указывайте локальный путь в *каталоге_журнала*.

По умолчанию: `$BACKUP_PATH/log/`

`--log-format-console=`*формат_журнала*

Определяет формат журнала консоли. Устанавливается только из командной строки. Обратите внимание, что этот параметр нельзя указать в файле конфигурации `pg_probackup3.conf` посредством команды set-config и что команда backup также воспринимает указание этого параметра в конфигурационном файле как ошибку. Этот параметр может иметь следующие значения:

- Если `plain`, то журнал выводится на консоль в формате обычного текста.
- Если `json`, то журнал выводится на консоль в формате JSON.

По умолчанию: `plain`

`--log-format-file=`*формат_журнала*

Определяет используемый формат файлов журнала. Этот параметр может иметь следующие значения:

- Если `plain`, то файлы журнала записываются в формате обычного текста.
- Если `json`, то файлы журнала записываются в формате JSON.

По умолчанию: `plain`

`--log-rotation-size=`*размер_журнала_для_ротации*

Определяет максимальный размер отдельного файла журнала. Если это значение достигается, файл журнала прокручивается при выполнении какой-либо команды pg_probackup3, за исключением `help` и `version`. Нулевое значение отключает прокрутку в зависимости от размера. Допустимые единицы измерения: B, kB (по умолчанию), MB, GB, TB. Используйте счётчик файлов (%N) в шаблоне имени журнала.

По умолчанию: `0`

Параметры подключения

Эти параметры могут использоваться с командой backup.

pg_probackup3 поддерживает все *переменные окружения libpq*.

`-d` *имя_бд*
`--pgdatabase=`*имя_бд*
`PGDATABASE`

Задаёт имя базы данных для подключения. Это подключение используется только для управления процессом резервного копирования, так что вы можете подключиться к любой существующей базе. Если этот параметр не задаётся в командной строке, переменной окружения `PGDATABASE` или в конфигурационном файле `pg_probackup3.conf`, pg_probackup3 принимает в качестве имени базы значение переменной окружения `PGUSER` или имя текущего пользователя, если переменная `PGUSER` не задана.

`-h` *сервер*
`--pghost=`*сервер*
`PGHOST`

Указывает имя системы, в которой работает сервер. Если значение начинается с косой черты, оно определяет каталог Unix-сокета.

По умолчанию: `localhost`

`-p` *порт*
`--pgport=`*порт*
`PGPORT`

Указывает TCP-порт или расширение файла локального Unix-сокета, через который сервер принимает подключения.

По умолчанию: `5432`

`-U` *имя_пользователя*
`--pguser=`*имя_пользователя*
`PGUSER`

Имя пользователя для подключения.

`-w`
`--no-password`

Не выдавать запрос на ввод пароля. Если сервер требует аутентификацию по паролю и пароль не доступен с помощью других средств, таких как файл *.pgpass* или переменная окружения `PGPASSWORD`, попытка соединения не удастся. Этот параметр может быть полезен в пакетных заданиях и скриптах, где нет пользователя, который вводит пароль.

`-W`
`--password`

Запрашивать пароль. (Устаревший параметр.)

Параметры сжатия

Эти параметры могут использоваться с командами backup и archive-push.

`--compress-algorithm=`*алгоритм_сжатия*

Определяет алгоритм, который будет использоваться для сжатия файлов данных. Возможные значения: `zlib`, `lz4`, `zstd` и `none`. Любое значение, отличное от `none`, включает сжатие. При этом сжимаются и файлы данных, и файлы WAL. По умолчанию сжатие отключено.

По умолчанию: `none`

> **Предупреждение**
>
> Значение `lz4` для параметра `--compress-algorithm` в настоящее время не поддерживается в командах `archive-push` и `archive-get`.

`--compress-level=`*`уровень_сжатия`*

Определяет уровень сжатия.

> **Примечание**
>
> Этот параметр необходимо использовать вместе с параметром `--compress-algorithm`.

Возможные значения зависят от указанного алгоритма сжатия:

- 0 — 9 для `zlib`
- 0 — 12 для `lz4`
- 0 — 22 для `zstd`

При значении 0 устанавливается уровень сжатия по умолчанию для указанного алгоритма:

- 6 для `zlib`
- 9 для `lz4`
- 3 для `zstd`

> **Примечание**
>
> Обычный алгоритм lz4 имеет только один уровень сжатия — 1. Так что, если указан алгоритм сжатия `lz4` и значение `--compress-level` больше 1, фактически используется алгоритм lz4hc, который работает намного медленнее, но обеспечивает лучшее сжатие.

По умолчанию: `1`

Параметры архивации

Эти параметры могут использоваться с командой archive-push в значении параметра *archive_command* и с командой archive-get в значении *restore_command*.

Дополнительно можно задать параметры SSH и параметры журнала.

`--wal-file-path=`*`путь_файлов_wal`*

Задаёт путь файла WAL в *`archive_command`* и *`restore_command`*. В качестве значения для данного параметра укажите `%p` или явно задайте путь к файлу вне каталога данных. Если этот параметр не задан, используется путь, заданный в файле `pg_probackup3.conf`.

`--wal-file-name=`*`имя_файла_wal`*

Задаёт имя файла WAL в *`archive_command`* и *`restore_command`*. В качестве значения для данного параметра укажите `%f` для правильной его обработки. Если значением параметра `--wal-file-path` является путь вне каталога данных, следует явно указывать имя файла.

`--overwrite`

Разрешает перезапись файлов WAL в архиве. Этот ключ действует при выполнении команды archive-push, когда указанный подкаталог каталога резервных копий уже содержит данный

файл WAL, и его нужно заменить новой копией. Без этого ключа `archive-push` сообщит, что сегмент WAL уже существует, и прервёт операцию. Если заменяемый файл не изменился, `archive-push` пропускает его, независимо от указания `--overwrite`.

`--batch-size=`*размер_порции*

Используется для ускорения процесса архивирования при выполнении `archive-push` или процесса восстановления при выполнении `archive-get`. Задаёт максимальное количество файлов WAL, которое может быть скопировано в архив одним процессом `archive-push` или из архива одним процессом `archive-get`.

`--archive-timeout=`*время_ожидания*

Задаёт интервал, по истечении которого существующие файлы `.part` будут считаться потерянными. По умолчанию pg_probackup3 ждёт исчезновения этих файлов 300 секунд. Этот параметр можно использовать только с командой archive-push.

`--no-sync`

Не сбрасывать копируемые файлы WAL на диск. Этот флаг позволяет несколько ускорить процесс архивации. Использование этого флага может привести к повреждению архива WAL в случае аварии операционной системы или аппаратного сбоя. Данный параметр можно указать только с командой archive-push.

`--prefetch-dir=`*путь*

Каталог, в котором будут храниться предзагружаемые сегменты WAL при использовании параметра `--batch-size`. Этот каталог должен находиться в той же файловой системе и ниже той же точки монтирования, что и `PGDATA/pg_wal`. По умолчанию эти сегменты размещаются в каталоге `PGDATA/pg_wal/pbk_prefetch`. Данный параметр может применяться только с командой archive-get.

Параметры буферизации

Эти параметры могут использоваться со всеми командами.

`--buffer-size=`*размер*

Задаёт размер буфера для операций чтения и записи. Должно быть неотрицательным целым числом. При нулевом значении этот параметр отключается. Значение по умолчанию — 0.

`--buffer-read-size=`*размер*

Задаёт размер отдельного буфера для операций чтения. Должно быть неотрицательным целым числом. При нулевом значении этот параметр отключается. Значение по умолчанию — 0.

`--buffer-write-size=`*размер*

Задаёт размер расширенного буфера для операций записи. Должно быть неотрицательным целым числом. При нулевом значении этот параметр отключается. Значение по умолчанию — 0.

Вы можете явно указать единицы измерения для любого из параметров буфера. Допустимые значения: B, kB, MB, GB, TB. Если единица измерения не указана, по умолчанию используются байты.

Параметры S3

В этом разделе описываются параметры, которые необходимо задать для размещения копий в частном облачном хранилище. Эти параметры могут задаваться с любыми командами, которые pg_probackup3 выполняет через интерфейс S3.

За подробностями обратитесь к подразделу Настройка подключения к хранилищу S3.

`--s3=`*провайдер_интерфейса_s3*

Задаёт провайдера, поддерживающего интерфейс S3. Возможные значения:

- `minio` — объектное хранилище MinIO, совместимое с облачным хранилищем S3. С этим провайдером могут указываться пользовательские параметры сервера S3. По умолчанию используется протокол HTTP, порт 9000 и регион `us-east-1`.
- `off` — полностью отключает функциональность S3. Это значение по умолчанию для параметра `--s3`.

При указании `--s3=minio` pg_probackup3 будет работать с хранилищем VK Cloud, если правильно заданы адрес узла S3, порт и протокол (адрес узла — `hb.vkcs.cloud` или указанный в соответствующем разделе профиля VK Cloud, порт 443 и протокол HTTPS). Не указывайте `--s3=minio` при использовании хранилища Amazon S3.

`--s3-host=`*`имя_сервера`*

Задаёт адрес сервера S3. Можно также указать номер порта через двоеточие. Если номер порта не указан в строке адреса, используется значение `--s3-port`. Добавляйте двоеточие только при указании номера порта.

`--s3-port=`*`номер_порта`*

Задаёт порт сервера S3.

`--s3-region=`*`регион`*

Задаёт регион сервера S3. Значение по умолчанию — `us-east-1`.

`--s3-bucket=`*`бакет`*

Задаёт имя бакета на сервере S3.

`--access-key=`*`ключ_доступа`*

Задаёт ключ доступа для хранилища S3.

`--secret-key=`*`пароль`*

Задаёт секретный ключ доступа для хранилища S3.

`--s3-secure=`*`протокол`*

Указывает используемый протокол. Допустимые значения:

- `ON` или `HTTPS` — используется HTTPS.
- `HTTP` — используется HTTP. Это режим по умолчанию.

`--s3-retries=`*`число_повторных_попыток`*

Задаёт максимальное количество попыток выполнения запроса к S3 при возникновении сбоев. Значение по умолчанию — `3`.

`--s3-timeout=`*`время_ожидания`*

Задаёт максимальное время выполнения HTTP-запроса к серверу S3 в секундах. Значение по умолчанию — `300`.

`--s3-ignore-cert-ver=ON|OFF`

Позволяет не проверять сертификат узла и узла-партнёра. По умолчанию — `OFF`.

`--s3-ca-certificate=`*`сертификат_ца`*

Указывает путь к каталогу файла с сертификатом от доверенного центра сертификации (ЦА).

`--s3-ca-path=`*`каталог_ца`*

Указывает каталог, в котором должны храниться сертификаты доверенного ЦС.

`--s3-client-cert=`*клиентский_сертификат*

Устанавливает клиентский сертификат SSL.

`--s3-client-key=`*клиентский_ключ*

Задаёт файл закрытого ключа для клиентских сертификатов TLS и SSL.

`--s3-versioning=enabled|suspended|off`

Включает управление версиями объектов в бакете S3. По умолчанию — `off`.

`--s3-http-compression=true|false`

Устанавливает HTTP-заголовок «Accept-Encoding» и выполняет распаковку полученного содержимого. По умолчанию — `false` (отключено).

Ниже описываются параметры производительности S3.

`--s3-buffer-size=`*размер* `[`*единица_измерения*`]`

Задаёт размер буфера чтения/записи для взаимодействия с S3. Вы можете явно указать единицы измерения. Допустимые значения: B, kB, MB, GB, TB. Если единица не указана, по умолчанию используется размер в байтах.

> **Примечание**
>
> Значение параметра `--s3-buffer-size` не должно быть меньше 5 МБ. При указании меньших значений будет выдано предупреждение, а значение автоматически изменится на `5MB`.

Параметры SSH

В этом подразделе описываются параметры, связанные с работой pg_probackup3 в удалённом режиме через SSH. Эти параметры могут использоваться со всеми командами.

Подробнее о настройке и использовании удалённого режима через SSH рассказывается в Разделе 2.11 и Разделе 3.4.

`--remote-host=`*целевой_адрес*

Задаёт имя или IP-адрес целевого удалённого сервера.

`--remote-port=`*порт*

Задаёт целевой порт на удалённом сервере.

По умолчанию: `22`

`--remote-user=`*имя_пользователя*

Задаёт имя пользователя на удалённом сервере для SSH-соединения. В отсутствие этого параметра используется имя текущего пользователя, устанавливающего SSH-соединения.

`--remote-path=`*путь*

Задаёт каталог, в котором pg_probackup3 установлен на удалённой системе.

`--ssh-options=`*параметры_ssh*

Задаёт строку параметров командной строки для SSH. Например, следующим образом можно установить свойства *keep-alive* для SSH-подключений, которые будет открывать pg_probackup3: `--ssh-options="-o ServerAliveCountMax=5 -o ServerAliveInterval=60"`. Полный список всех параметров можно найти в *руководстве по ssh_config*.

`--ssh-password=`*пароль*

Задаёт пароль для подключения по SSH.

Параметры удалённого архива WAL

В этом подразделе описываются параметры, позволяющие задать аргументы для использования удалённого режима через SSH.

`--archive-host=`*целевой_адрес*

Задаёт значение аргумента `--remote-host` для команды `archive-get`.

`--archive-port=`*порт*

Задаёт значение аргумента `--remote-port` для команды `archive-get`.

По умолчанию: `22`

`--archive-user=`*имя_пользователя*

Задаёт значение аргумента `--remote-user` для команды `archive-get`. В отсутствие этого указания используется имя пользователя, запускающего кластер Postgres Pro.

По умолчанию: пользователь Postgres Pro

Параметры инкрементального восстановления

В этом подразделе описываются параметры, связанные с инкрементальным восстановлением кластера. Эти параметры могут передаваться с командой restore.

`--I` *инкрементальный_режим*
`--incremental-mode=`*инкрементальный_режим*

Выбирает инкрементальный режим. Поддерживаются следующие режимы:

- `CHECKSUM` — заменять только страницы с неподходящей контрольной суммой и LSN.
- `LSN` — заменять только те страницы, LSN которых больше точки расхождения.
- `NONE` — обычное восстановление.

Параметры частичного резервного копирования и восстановления

В этом подразделе описываются параметры, связанные с частичным резервным копированием и восстановлением кластера. Эти параметры могут передаваться как с командой backup, так и с restore.

`--db-exclude-oid=`*dboid*

Задаёт OID базы данных, которая должна быть исключена из числа восстанавливаемых. Все остальные базы данных в кластере будут восстанавливаться, включая `template0` и `template1`. Этот параметр можно задать несколько раз, таким образом исключив несколько баз данных.

`--db-include-oid=`*dboid*

Задаёт OID базы данных, которая должна восстанавливаться. Все остальные базы данных восстанавливаться не будут, за исключением `template0` и `template1`. Этот параметр можно задать несколько раз, таким образом выбрав для восстановления несколько баз данных.

Эти параметры используются с командой restore.

`--db-exclude-name=`*имя_бд*

Задаёт имя базы данных, которая должна быть исключена из числа восстанавливаемых. Все остальные базы данных в кластере будут восстанавливаться, включая `template0` и `template1`. Этот параметр можно задать несколько раз, таким образом исключив несколько баз данных.

`--db-include-name=`*`имя_бд`*

Задаёт имя базы данных, которая должна восстанавливаться. Все остальные базы данных восстанавливаться не будут, за исключением `template0` и `template1`. Этот параметр можно задать несколько раз, таким образом выбрав для восстановления несколько баз данных.

> **Предупреждение**
>
> Параметры `--db-exclude-oid` и `--db-include-oid`, так же как и параметры `--db-exclude-name` и `--db-include-name`, использовать вместе нельзя.

Параметры тестирования и отладки

В этом подразделе описываются параметры, полезные лишь при тестировании или разработке.

`--cfs-nondatafile-mode`

Указывает команде backup выполнить резервное копирование CFS в режиме предыдущих версий. Этот параметр позволяет настраивать совместимость с версиями pg_probackup3 и предназначен в основном для тестирования.

`PGPROBACKUP_TESTS_SKIP_HIDDEN`

Указывает pg_probackup3 игнорировать копии, помеченные как скрытые. Заметьте, что сама утилита pg_probackup3 никогда не помечает копии как скрытые. Добиться такого состояния копии можно, только вручную отредактировав файл `backup.control`. Задать этот параметр можно только в переменных окружения.

`PGPROBACKUP_TESTS_SKIP_EMPTY_COMMIT`

Указывает pg_probackup3 пропускать пустые транзакции после pg_backup_stop.

Авторы

Postgres Professional, Москва, Россия.

Приложение A. Замечания к выпускам

A.1. pg_probackup 3.2.0

Дата выпуска: 2025-12-30

Этот выпуск основан на pg_probackup3 3.1.1, в нём повышена стабильность и производительность и улучшена совместимость pg_probackup3. Основные изменения перечислены ниже:

- WAL и инкрементальное резервное копирование:
  - Добавлена обработка WAL.
  - Реализовано ожидание полной загрузки WAL-сегментов.
  - Добавлено игнорирование WAL-файлов, отсутствующих в последней инкрементальной резервной копии, при восстановлении для корректного запуска сервера.
  - Добавлена проверка WAL для всей цепочки резервных копий при объединении.
  - Добавлена установка `start_lsn` в соответствии с последней инкрементальной резервной копией после объединения.
  - Ограничено число потоков резервного копирования доступными процессами walsender.
  - Реализовано сохранение WAL-файлов только для последней инкрементальной резервной копии во время объединения.
  - Добавлен параметр `--wal-archive-dir`, позволяющий использовать отдельный каталог для архивации WAL-файлов.
  - Исправлены уровни протоколирования команд архивации WAL.
  - Исправлена ошибка, которая возникала при указании нескольких `walsender_plugin_libraries`.
- Резервное копирование и восстановление:
  - Исправлена проверка уровня сжатия для поддержки уровней выше 9, когда это позволяет выбранный алгоритм.
  - Исправлена обработка табличных пространств по умолчанию при инкрементальном восстановлении: их содержимое больше не перезаписывается.
  - Исправлено формирование пути в `restore_command`.
  - Добавлены сообщения о завершении проверки резервных копий и о выбранном источнике резервного копирования, если он не указан явно.
  - Улучшены сообщения, отображаемые при отсутствии файлов резервного копирования.
  - Добавлена проверка файлов с одинаковыми идентификаторами.
  - Исправлено восстановление из смешанных цепочек инкрементальных резервных копий.
  - Упрощена проверка перед восстановлением: для восстановления не из последней резервной копии больше не требуется чтение метаданных всех последующих инкрементальных резервных копий.
  - Исправлена проверка всей цепочки резервных копий при объединении нескольких инкрементальных копий в одну интервальную.
  - Добавлены параметры `--db-include-oid` и `--db-exclude-oid` для команд `backup` и `restore` для резервного копирования и восстановления отдельных баз данных.

- `catchup`:
  - Реализована корректная обработка ошибок и пропуск файлов.
  - Добавлен параметр `--exclude-path` для исключения определённых файлов и каталогов.
  - Добавлена поддержка параметров `--threads`, `--temp-slot`, `--perm-slot`, `--slot-name` и `--tablespace-mapping`.
- CFS:
  - Реализована поддержка резервного копирования и восстановления файлов CFS в режиме DIRECT.
  - Добавлена проверка блоков CFS с проверкой контрольных сумм.
  - Добавлено сообщение о том, что в режиме BASE резервное копирование CFS не поддерживается.
- pgpro_backupstream (восстановление на удалённый сервер):
  - Отменено создание файла `recovery.signal` в случае ошибки.
  - Исправлена логика потоков в `send-backup`: потоки теперь завершают работу только после того, как данные полностью прочитаны и обработаны утилитой pgpro_backupstream.
- S3:
  - Сделан обязательным параметр бакета.
  - Исправлена совместимость с хранилищами, которые не поддерживают версионированные бакеты.
  - Реализовано повторное подключение с соответствующими статус-сообщениями при разрыве соединения.
  - Ускорена запись в S3 за счёт выбора сегментов с наименьшей нагрузкой.
  - Установлен лимит на количество составных (multipart) загрузок.
- FUSE:
  - Добавлен параметр `--unmount` для размонтирования файловой системы FUSE.
  - Добавлен параметр `--detach` для запуска процесса в фоновом режиме.
  - Добавлена поддержка команд `chmod` и `chown`.
- TDE (Прозрачное защитное преобразование данных):
  - Добавлена поддержка TDE.
  - Реализовано хранение статуса TDE в метаданных резервной копии.

A.2. pg_probackup 3.1.1

Дата выпуска: 2025-10-28

Этот выпуск основан на pg_probackup3 3.1.0, в нём повышена стабильность и производительность и улучшена совместимость pg_probackup3. Основные изменения перечислены ниже:

- Основные улучшения:
  - Реализовано сохранение файлов истории при потоковой передаче WAL.
  - Устранена проблема с удалённым восстановлением.
- Прочие улучшения:

- Исправлена ошибка при выполнении команды `add-instance`, которая могла возникать из-за того, что обработка параметра конфигурации `pgpro_edition` зависела от локализованного текста ошибки.
- Добавлено сопоставление идентификатора системы из файла `pg_control` с идентификатором системы в `PGDATA` перед восстановлением из резервной копии, сделанной в режиме источника данных `DIRECT`. Это позволяет избежать восстановления из неподходящей резервной копии.
- Добавлено предупреждение, которое отображается после выполнения команды `validate --instance`, если в ходе валидации были обнаружены ошибки.
- Исправлена логика объединения интервалов объединяемых резервных копий и добавлено отображение идентификатора родительской копии для объединённых интервалов.
- Улучшена процедура обработки исключений при работе с S3-хранилищем. Добавлена процедура безопасной очистки ресурсов.
- Оптимизирована обработка данных в формате CBOR (RFC 8949 Concise Binary Object Representation, Сжатое представление двоичных объектов) за счёт использования буфера напрямую, без дополнительного копирования.
- Устранена проблема с резервным копированием в режиме источника данных `BASE` для PostgreSQL 14.
- Улучшена обработка переменной окружения `BACKUP_PATH`. Теперь, чтобы задать переменную, не требуется указывать каталог резервных копий в параметре `-B`.

A.3. pg_probackup 3.1.0

Дата выпуска: 2025-09-25

Этот выпуск основан на pg_probackup3 3.0.2, в нём добавлены новые возможности, улучшена производительность и исправлены некоторые ошибки. Важные изменения перечислены ниже.

- Новые возможности:
  - Реализована поддержка инкрементального восстановления с помощью параметра `-I | --incremental-mode` в режиме PRO.
  - Добавлен параметр `--smooth-checkpoint` для команды `backup`. Этот параметр позволяет растягивать время выполнения контрольной точки, давая время командам штатно завершить работу.
  - Добавлены параметры `--batch-size` и `--threads` для команды `archive-push` для многопоточного копирования WAL.
  - Добавлена утилита pgpro_backupstream, которая позволяет выполнить восстановление на удалённый сервер. В данный момент утилита представлена в режиме бета-тестирования. Некоторые возможности, такие как восстановление из резервных копий в режиме ARCHIVE и инкрементальное восстановление, пока недоступны.
- Исправления ошибок:
  - Исправлена проблема, возникающая при попытке копирования без указания параметров экземпляра и каталога резервных копий.
  - Исправлено отображение режимов доставки WAL — STREAM и ARCHIVE.
  - Исправлена ошибка при попытке резервного копирования кластера с CFS в режиме BASE при отсутствии `shared_preload_libraries` в файле конфигурации.
  - Исправлено отображение идентификаторов резервных копий, созданных с использованием pg_probackup 2.X.

- Исправлен список доступных значений для параметра `--log_rotation_size` команды `set-config`.
- Исправлены проблемы с подключением при использовании параметра `-W | --password`.

- Прочие улучшения:
  - Добавлена поддержка баз данных без включённых контрольных сумм для корректной работы с Shardman.
  - Добавлена проверка повторного архивирования WAL-файлов. При наличии файла с таким же именем производится сравнение нового и уже существующего файла. Если файл отличается, то на поведение влияет параметр `--overwrite` — файл переписывается или выдаётся ошибка.
  - Добавлен запрет на создание резервных копий с разными значениями `--backup-source` (BASE, DIRECT, PRO) в рамках одной цепочки.

A.4. pg_probackup 3.0.2

Дата выпуска: 2025-07-10

Этот выпуск основан на pg_probackup3 3.0.0, в нём добавлены новые возможности, улучшена производительность и исправлены некоторые ошибки. Важные изменения перечислены ниже.

- FUSE:
  - Добавлена возможность работы с несжатыми табличными пространствами.
  - Исправлена работа FUSE с CFS.
  - Добавлена команда `file-map`, которая позволяет создавать файлы сопоставления для существующей цепочки резервных копий, начиная с выбранной инкрементальной копии.
  - Исправлена ошибка FUSE, возникавшая при нехватке места на диске. Добавлен параметр `--cache-dir` для указания пути к каталогу кеша FUSE (по умолчанию используется временный каталог ОС).
- S3:
  - Добавлены параметры для управления сертификатами HTTPS.
  - Реализована возможность задавать параметры подключения в файле конфигурации с помощью параметра `--config-file`.
- Прочие улучшения:
  - В команду `delete` добавлен параметр `--status` для удаления резервных копий с определённым состоянием.
  - Сообщения уровня WARNING в журналах теперь выделяются жёлтым цветом. Для сообщений уровней `debug` и `trace` также добавлена цветовая подсветка.
  - Добавлены параметры `--db-include` и `--db-exclude` для команды `restore`. Теперь включать и исключать базы данных можно как по OID (параметры `--db-include-oid` и `--db-exclude-oid`), так и по имени.
  - Добавлена возможность указывать значения параметров в удобных размерных единицах: `B`, `KB`, `MB`, `GB`, `TB` (по умолчанию используются байты).
  - Добавлен параметр `--num-validate-threads` для задания пользовательского количества потоков для проверки.

A.5. pg_probackup 3.0.0

Дата выпуска: 2025-03-28

Это первая версия pg_probackup3.

Решение pg_probackup3 основано на *pg_probackup*, где реализована большая часть функциональности.

Основные возможности перечислены ниже:

- Версионная независимость. Одна и та же версия pg_probackup3 теперь может использоваться с различными версиями Postgres Pro или PostgreSQL, обеспечивая совместимость и адаптивность.
- Интеграция с API. pg_probackup3 может интегрироваться с различными системами резервного копирования через API, что обеспечивает централизованное управление резервным копированием.
- Работа без SSH. pg_probackup3 может работать без SSH-соединения, гарантируя более эффективную и безопасную передачу данных.
- `FUSE`. В pg_probackup3 реализована команда `fuse`, которая с помощью механизма FUSE (Filesystem in User Space, Файловая система в пользовательском пространстве) обеспечивает работу с базой данных прямо из резервной копии, минуя этап полного восстановления.
- Запуск от имени непривилегированных пользователей. pg_probackup3 может запускаться пользователями, не имеющими прав доступа к PGDATA. Это повышает уровень безопасности и снижает риск возможных ошибок.
- Новый формат резервных копий. Каждая резервная копия теперь хранится в виде единого файла, что облегчает управление и хранение резервных копий.
- Поддержка pg_basebackup. В режиме источника данных BASE теперь возможно использовать репликационный протокол pg_basebackup для повышения скорости и эффективности резервного копирования.
- Режим PRO. В режиме источника данных PRO pg_probackup3 использует собственный репликационный протокол.
- Объединение цепочек инкрементальных копий. Для экономии места на диске теперь можно объединять цепочки инкрементальных резервных копий.

Приложение B. Известные проблемы и устранение неполадок

B.1. Ошибка `libpq.so.5: no version information available`

Если вы устанавливаете PostgreSQL из RPM-пакетов репозитория *PostgreSQL Yum Repository*, при запуске pg_probackup3 может возникнуть следующая ошибка:

```
libpq.so.5: no version information available
```

Эти пакеты заменяют системную версию библиотеки `libpq` по умолчанию, что может привести к некорректной работе приложений (не только pg_probackup3).

Для решения этой проблемы переключитесь на использование стандартной версии `libpq`, выполнив следующие команды:

```
alternatives --install /etc/ld.so.conf.d/postgresql-pgdg-libs.conf pgsql-ld-conf /dev/
null 1320
alternatives --set pgsql-ld-conf /dev/null
ldconfig
```

Приложение C. Совместимость версий

При разработке pg_probackup3 используется *семантическое* версионирование.

Для режима источника данных резервного копирования PRO требуется плагин *pgpro_bindump* (первоначально называвшийся contrib/pg_probackup3), доступный начиная со следующих версий Postgres Pro:

- Postgres Pro Standard: 14.18.1, 15.10.1, 16.6.1, 17.0.1, 17.2.1, 18.0.1
- Postgres Pro Enterprise: 14.18.1, 15.10.1, 16.6.1, 17.2.1, 18.0.1

pg_probackup3 поддерживает следующие версии и редакции Postgres Pro:

Версия Postgres Pro	**Соответствующая версия pg_probackup3**
15.10.1–15.12.1 16.6.1–16.8.1 17.2.1–17.4.1	3.0.0
14.18.1 15.13.1 16.9.1 17.5.1	3.0.1, 3.0.2
14.19.1 15.14.1 16.10.1 17.6.1 17.7.1	3.1.0, 3.1.1, 3.2.0

> **Примечание**
>
> Обратная совместимость поддерживается для всех новых выпусков, поэтому новые версии pg_probackup3 остаются совместимыми с предыдущими версиями Postgres Pro.

Предметный указатель

C

P