Документация pg_probackup 3.0.0

Документация pg_probackup 3.0.0

Содержание

1. О pg_probackup3 1
Установка pg_probackup3 2
Версионирование 4
2. О резервном копировании и восстановлении с помощью pg_probackup3 5
Функциональные возможности 5
Ограничения 6
3. Настройка резервного копирования и восстановления 8
Инициализация каталога резервных копий 8
Определение копируемого экземпляра 9
Настройка pg_probackup3 9
Указание параметров подключения 10
Настройка кластера баз данных 10
Настройка потокового резервного копирования 11
Настройка непрерывного архивирования WAL 12
Настройка копирования PTRACK 13
Настройка удалённого режима 15
Настройка подключения к хранилищу S3 15
4. Использование 19
Создание резервной копии 19
Монтирование каталога резервных копий с помощью FUSE 21
Восстановление кластера 22
Управление каталогом резервных копий 25
Использование pg_probackup3 в удалённом режиме 32
Запуск pg_probackup3 в параллельных потоках 33
Проверка целостности данных 34
Настройка политики хранения 35
Другие примеры 41
5. Справка 47
pg_probackup3 48
libpgprobackup 71
A. Замечания к выпускам 82
pg_probackup 3.0.0 82
Предметный указатель 83

Глава 1. О pg_probackup3

Содержание

Установка pg_probackup3 2
Версионирование 4

Решение pg_probackup3 создано для управления резервным копированием и восстановлением кластеров баз данных PostgreSQL. Оно поддерживает Postgres Pro и PostgreSQL 15 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.

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 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, доступный исключительно в Postgres Pro Enterprise.
- Объединение цепочек инкрементальных копий. Для экономии места на диске теперь можно объединять цепочки инкрементальных резервных копий.
- Полностью переработанное ядро
- Новая архитектура

- Улучшенная производительность

> **Примечание**
>
> Некоторые функции pg_probackup3 зависят от возможностей, специфичных для Postgres Pro Enterprise. Эти функции автоматически отключаются в версии pg_probackup3, поставляемой с Postgres Pro Standard.

Установка pg_probackup3

pg_probackup3 содержит следующие пакеты:

- `libpgprobackup3` — содержит разделяемую библиотеку, предоставляющую API для создания резервных копий, а также динамическую библиотеку `libpb3_encoder.so`, которая загружается соответствующим расширением сервера.
- `pg_probackup3` — содержит консольную утилиту для работы с резервными копиями.

> **Примечание**
>
> Все пакеты необходимо устанавливать и удалять совместно, так как `pg_probackup3` и `libpgprobackup3` поддерживают только версии Postgres Pro 15 и выше.

> **Примечание**
>
> Перед установкой pg_probackup3 убедитесь, что у вас установлен модуль pgpro_bindump.

Чтобы установить pg_probackup3, выполните следующие действия:

1. **Скачайте архив с пакетами**

   Скачайте архив с пакетами по предоставленной ссылке. В архиве находятся тестовые репозитории сборки pg_probackup3 для следующих операционных систем:

   - Debian 12
   - Astra Linux «Смоленск» 1.8
   - Ubuntu 22.04/24.04
   - RHEL 9

2. **Распакуйте архив**

   Распакуйте архив с помощью следующей команды:

   ```
   tar -xvzf pbk3.tar.gz pbk3/
   ```

3. **Настройте репозиторий**

   1. Установите ключ GPG.

      В архиве есть каталог `keys`, который содержит ключ GPG для подключения репозитория. Установите ключ в зависимости от вашей системы.

      - Для систем на базе Debian:

        ```
        sed -n '/^$/,/=$/p' "/path/to/pbk3/keys/GPG-KEY-
        POSTGRESPRO" | base64 -d | sudo tee "/etc/apt/
        trusted.gpg.d/postgrespro.gpg" > /dev/null
        ```

      - Для систем на базе RHEL:

        ```
        sudo rpm --import /path/to/pbk3/keys/GPG-KEY-POSTGRESPRO
        ```

   2. Подключите репозиторий.

      - Для систем на базе Debian создайте файл `/etc/apt/sources.list.d/pbk3.list` со следующим содержимым.

        - Для Ubuntu 24.04:

          ```
          deb file:///path/to/pbk3/ubuntu noble main
          ```

        - Для Ubuntu 22.04:

          ```
          deb file:///path/to/pbk3/ubuntu jammy main
          ```

      - Для систем на базе RHEL создайте файл `/etc/yum.repos.d/pbk3.repo` со следующим содержимым:

        ```
        [pbk3]
        name=pbk3
        baseurl=file:///path/to/pbk3/rhel/9Server/os/x86_64/rpms
        enabled=1
        gpgcheck=1
        ```

4. **Установите пакеты**

   - Для систем на базе Debian после подключения репозитория обновите список пакетов:

     ```
     sudo apt update
     ```

     Затем установите необходимые пакеты:

     ```
     sudo apt install libpgprobackup3 pg-probackup3
     ```

   - Для систем на базе RHEL выполните установку через `yum`:

     ```
     sudo yum install libpgprobackup3 pg-probackup3
     ```

     Или через `dnf`:

     ```
     sudo dnf install libpgprobackup3 pg-probackup3
     ```

5. **Проверьте установку**

   Убедитесь, что пакеты установлены корректно.

1. Проверьте версию Postgres Pro:

   ```
   /opt/pgpro/ent-16/bin/postgres --version
   ```

2. Проверьте версию pg_probackup3:

   ```
   pg_probackup3 --version
   ```

6. **Настройте Postgres Pro для работы с pg_probackup3**

   Чтобы включить pg_probackup3, добавьте следующие параметры в файл `postgresql.conf`:

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

После завершения установки необходимо перезапустить экземпляр Postgres Pro.

Версионирование

При разработке pg_probackup3 используется семантическое версионирование.

Глава 2. О резервном копировании и восстановлении с помощью pg_probackup3

Содержание

Функциональные возможности 5
Ограничения 6

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.

Функциональные возможности

По сравнению с другими средствами резервного копирования pg_probackup3 имеет следующие преимущества, полезные для реализации различных стратегий резервного копирования и работы с базами данных большого объёма:

- Поддержка S3 для хранения данных в частных облачных хранилищах на базе объектного хранилища MinIO, Amazon S3 и VK Cloud: обеспечивается в Postgres Pro Enterprise. Данные резервного копирования передаются в S3 и обратно без сохранения в промежуточных хранилищах, что устраняет необходимость в большом временном хранилище.
- Поддержка ленточных хранилищ: pg_probackup3 поддерживает работу с системами резервного копирования на ленточные хранилища.
- Поддержка NFS версий 4 и 5: pg_probackup3 позволяет хранить резервные копии в сетевой файловой системе.
- Инкрементальное копирование: выбирая один из трёх режимов инкрементального копирования, вы можете реализовать стратегию резервного копирования, соответствующую вашему профилю транзакционной нагрузки. Это позволяет сэкономить место на диске и создавать копии быстрее, чем при полном копировании. Восстановление инкрементальных копий также осуществляется быстрее, чем воспроизведение файлов WAL.
- 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.
- Каталогизация резервных копий: получение списка резервных копий и соответствующей метаинформации в виде простого текста или JSON.
- Каталогизация архивов WAL: получение списка всех линий времени в WAL и соответствующей метаинформации в виде простого текста или JSON.
- Интеграция с другими приложениями благодаря API, входящему в библиотеку libprobackup.

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. Вы можете хранить резервные копии разных экземпляров в отдельных подкаталогах одного каталога копий.

Используя pg_probackup3, вы можете выполнять полное или инкрементальное резервное копирование:

- 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.

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.

В pg_probackup3 доступны следующие режимы источников данных для резервного копирования:

- DIRECT. Репликационный протокол не используется.
- BASE. Используется протокол pg_basebackup.
- PRO. Режим по умолчанию; используется протокол pg_probackup3. Доступен в Postgres Pro Enterprise.

Ограничения

В настоящее время pg_probackup3 имеет следующие ограничения:

- 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 15 и новее.

Глава 3. Настройка резервного копирования и восстановления

Содержание

Инициализация каталога резервных копий 8
Определение копируемого экземпляра 9
Настройка pg_probackup3 9
Указание параметров подключения 10
Настройка кластера баз данных 10
Настройка потокового резервного копирования 11
Настройка непрерывного архивирования WAL 12
Настройка копирования PTRACK 13
Настройка удалённого режима 15
Настройка подключения к хранилищу S3 15

Установив pg_probackup3, выполните следующие шаги:

- Инициализируйте каталог резервных копий.
- Добавьте копируемый экземпляр в каталог копий.
- Настройте кластер баз данных для использования pg_probackup3.
- Настройте SSH для выполнения операций pg_probackup3 в удалённом режиме, если вы планируете его использовать.
- Настройте параметры S3, если вы планируете использовать pg_probackup3 с хранилищем S3.

Инициализация каталога резервных копий

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

Прежде чем инициализировать каталог резервных копий, убедитесь, что выполняются следующие требования:

- pg_probackup3 подключён к серверу Postgres Pro.
- Пользователь, запускающий pg_probackup3, имеет полный доступ к каталогу *`каталог_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.

Определение копируемого экземпляра

pg_probackup3 может сохранять резервные копии разных кластеров баз данных в одном каталоге резервных копий. Для создания необходимых подкаталогов вы должны определить копируемый экземпляр в каталоге копий для каждого кластера баз данных, копию которого вы будете делать.

Для определения копируемого экземпляра выполните команду:

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

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_удалённого_режима 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_удалённого_режима, они будут добавлены в pg_probackup3.conf.

Более подробно тонкая настройка pg_probackup3 описывается в «Настройка pg_probackup3».

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 возможность Доступ группы, чтобы выполнить копирование мог любой пользователь ОС, включённый в группу владельца кластера. В этом случае пользователь должен иметь права на чтение каталога кластера.

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.

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.

Примечание

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

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_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.

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.

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;
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)
 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**
>
> Для резервного копирования не требуется доступ к каталогу данных `PGDATA`, так как pg_probackup3 использует репликационный протокол (режимы PRO и BASE) для извлечения данных из файлов или выполняет прямое резервное копирование (режим DIRECT). Режим PRO используется по умолчанию.

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 удалённого режима и Настройка копирования PTRACK.

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

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_probackup" archive-push
 -B "каталог_копий" --instance=имя_экземпляра --wal-file-name=%f
 [параметры_удалённого_режима]'
```

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_удалённого_режима 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 «Просмотр оглавления архива WAL».

> **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.

Настройка копирования PTRACK

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

В pg_probackup3 есть два приложения для резервного копирования в режиме PTRACK:

- Приложение PTRACK для создания резервных копий в режиме DIRECT.
- Приложение pb3_ptrack для резервного копирования в режиме PRO.

Если вы намерены использовать режим копирования PTRACK в режиме DIRECT, выполните описанные далее дополнительные действия.

> **Примечание**
>
> Для роли, которая будет выполнять резервное копирование в режиме PTRACK (роль backup в примерах ниже), требуемые права доступа указаны в «Настройка кластера баз данных». 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;
```

Чтобы создавать резервные копии в режиме PRO, задайте для параметра `pb3_ptrack.map_size` положительное целое значение в файле `postgresql.conf` и перезапустите сервер.

Для оптимальной производительности рекомендуется задавать `pb3_ptrack.map_size` равным *`N / 1024`*, где *`N`* — объём кластера Postgres Pro в мегабайтах. Если он будет иметь меньшее значение, это увеличит вероятность наложения информации разных блоков в карте pb3_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 `pb3_ptrack.map_size`, превышающие 1024, не рекомендуется, хотя pb3_ptrack поддерживает большие карты.

> **Примечание**
>
> В случае изменения значения `pb3_ptrack.map_size` ранее созданный файл карты pb3_ptrack очищается, и отслеживание новых блоков начинается с начала. Таким образом, прежде чем создавать новые инкрементальные копии в режиме PTRACK после изменения `pb3_ptrack.map_size` необходимо сделать новую полную копию кластера.

> **Примечание**
>
> Модуль pgpro_bindump должен быть включён перед настройкой pbk3_ptrack.

Предупреждение

Включение обоих приложений PTRACK и pb3_ptrack одновременно приведёт к критическим ошибкам и сбою резервного копирования. Убедитесь, что активировано только необходимое приложение.

Настройка удалённого режима

Программа pg_probackup3 поддерживает удалённый режим, в котором резервное копирование и архивирование WAL могут выполняться удалённо.

Настройте SSH

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 удалённого режима.

pg_probackup3 работает в удалённом режиме по протоколу SSH следующим образом:

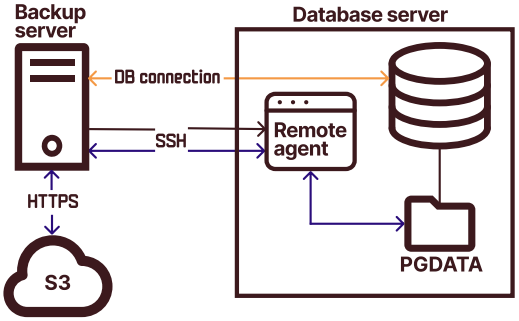
- В удалённом режиме поддерживаются все команды.
- Для работы в удалённом режиме не требуется, чтобы программа pg_probackup3 была установлена и в удалённой системе.

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.

Пример конфигурации с удалённым агентом и облачным хранилищем (S3) показан на Рисунок 3.1.

Рисунок 3.1. Настройка pg_probackup3 с удалённым агентом и S3



На этом рисунке показаны следующие логические компоненты:

Сервер резервного копирования

Сервер, на котором работает основной процесс pg_probackup3 и хранятся локальные резервные копии.

Сервер баз данных

Сервер с экземпляром базы данных, для которого требуется резервное копирование.

Удалённый агент

Вспомогательный процесс pg_probackup3, выполняемый на сервере баз данных. Применимо только к удалённому режиму.

Облачное хранилище

Облачное хранилище резервных копий.

Настройка подключения к хранилищу S3

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

- Создайте для будущих копий в хранилище S3 корзину (bucket) 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 S3 (за подробностями обратитесь к описанию параметра `--s3-config-file` в разделе 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 или список адресов сервера S3 из одного или нескольких адресов, разделённых точкой с запятой. После последнего адреса в списке точка с запятой не ставится. Для каждого адреса можно также указать номер порта через двоеточие. Если номер порта не указан, 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;10.4.13.56:443;172.17.0.1"
```

В этом примере для адреса «127.0.0.1» явно указан порт 9000, для «10.4.13.56» — порт 443, а для адреса «172.17.0.1» будет использоваться порт 80, указанный в `PG_PROBACKUP_S3_PORT`.

Если какой-либо из указанных адресов становится недоступным во время работы pg_probackup3, запросы к хранилищу S3 распределяются между остальными указанными адресами. То есть, когда указано несколько адресов, pg_probackup3 выполняет балансировку нагрузки запросов S3.

`PG_PROBACKUP_S3_PORT`

Порт сервера S3.

`PG_PROBACKUP_S3_REGION`

Регион для выбора сервера S3.

`PG_PROBACKUP_S3_BUCKET_NAME`

Имя корзины на сервере S3.

`PG_PROBACKUP_S3_ACCESS_KEY`, `PG_PROBACKUP_S3_SECRET_ACCESS_KEY`

Безопасные ключи 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: `ON`.

`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.

Глава 4. Использование

Содержание

Создание резервной копии 19
Монтирование каталога резервных копий с помощью FUSE 21
Восстановление кластера 22
Управление каталогом резервных копий 25
Использование pg_probackup3 в удалённом режиме 32
Запуск pg_probackup3 в параллельных потоках 33
Проверка целостности данных 34
Настройка политики хранения 35
Другие примеры 41

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 источников данных BASE и DIRECT не поддерживается CFS.

> **Примечание**
>
> В режимах источников данных BASE и DIRECT 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.

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.

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 [--temp-slot]
```

Необязательный параметр `--temp-slot` обеспечивает наличие необходимых сегментов в случае прокрутки WAL до завершения резервного копирования.

> **Примечание**
>
> Хотя `--temp-slot` и является необязательным флагом, он может влиять на успех резервного копирования.

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.

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`.

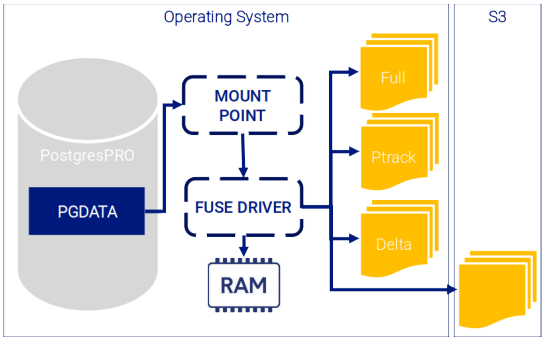
> **Примечание**
>
> Внешние каталоги не поддерживаются в режиме BASE.

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. Это гарантирует, что резервная копия остаётся неизменной, а все операции выполняются в режиме *только для чтения*.

Рисунок 4.1. 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**
>
> CFS (Compressed File System, Сжатая файловая система) и табличные пространства в настоящее время не поддерживаются.

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

Восстановление кластера

> **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.

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 «Выполнение восстановления на момент времени (PITR)».

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
```

Частичное восстановление

Можно восстанавливать определённые базы данных без дополнительной подготовки с помощью параметров частичного восстановления с командой restore и OID этих баз.

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
```

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.

Выполнение восстановления на момент времени (PITR)

Если вы настраивали непрерывное архивирование WAL до создания резервных копий, вы можете восстановить состояние кластера на любой момент времени (до заданной точки восстановления), используя с командой restore параметры точки восстановления.

Для восстановления на момент времени может использоваться копия типа STREAM или ARCHIVE, но при этом обязательно наличие архива WAL с момента создания копии или более раннего.

- Чтобы восстановить состояние кластера на определённый момент времени, укажите это время в параметре `--recovery-target-time`, в формате timestamp. Например:

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

- Чтобы восстановить текущее или последнее возможное состояние кластера, передайте в параметре `--recovery-target-time` значение `current` или `latest`, соответственно:

  ```
  pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра
   --recovery-target-time="latest"
  ```

- Чтобы восстановить состояние кластера до определённой транзакции, воспользуйтесь ключом `--recovery-target-xid`:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра
 --recovery-target-xid=687
```

- Чтобы восстановить состояние кластера до определённой позиции в журнале (LSN), воспользуйтесь ключом `--recovery-target-lsn`:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра
 --recovery-target-lsn=16/B374D848
```

- Чтобы восстановить состояние кластера до заданной именованной точки восстановления, воспользуйтесь ключом `--recovery-target-name`:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра
 --recovery-target-name="before_app_upgrade"
```

- Чтобы восстановить последнее возможное состояние, исходя из содержимого архива WAL, передайте в параметре `--recovery-target-stop` значение `latest`:

```
pg_probackup3 restore -B каталог_копий --instance=имя_экземпляра
 --recovery-target-stop="latest"
```

- Чтобы восстановить самое раннее из возможных согласованное состояние кластера, передайте в параметре `--recovery-target-stop` значение `immediate`:

```
          pg_probackup3 restore -B каталог_копий --
instance=имя_экземпляра --recovery-target-stop='immediate'
```

Управление каталогом резервных копий

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

- Просматривать имеющиеся резервные копии
- Просматривать оглавление архива WAL
- Объединять резервные копии
- Удалять резервные копии

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).
- `pgdata-bytes` — размер файлов данных Postgres Pro на момент копирования. Эффективность инкрементального метода копирования можно оценить, сопоставив объём `pgdata-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"
            }
        ]
    }
]
```


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.

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, 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.

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.

Использование pg_probackup3 в удалённом режиме

Программа pg_probackup3 поддерживает работу в удалённом режиме, то есть может выполнять операцию `backup` удалённо, используя SSH. В

этом режиме каталог резервных копий располагается в локальной системе, а целевой экземпляр Postgres Pro работает в удалённой. При этом pg_probackup3 должен быть установлен в обеих системах.

> **Примечание**
>
> pg_probackup3 рассчитывает на то, что узлы будут взаимодействовать между собой с использованием SSH без пароля.

> **Примечание**
>
> Помимо подключения по SSH, pg_probackup3 использует для управления удалёнными операциями обычное подключение к базе данных. За подробной информацией о настройке подключения к базе данных обратитесь к разделу Настройка кластера базы данных.

Типичная схема его использования выглядит так:

- В системе резервного копирования настройте pg_probackup3, как описывается в подразделе Установка и настройка. Для команд add-instance и set-config необходимо задать параметры удалённого режима, 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 БД в систему резервного копирования, как описано в подразделе Настройка непрерывного архивирования WAL. Для этого в командах archive-push и archive-get требуется задать параметры удалённого режима, указывающие на сервер, где находится каталог резервных копий.
- Запустите команду backup с параметрами удалённого режима **в системе резервного копирования**. pg_probackup3 подключится к удалённой системе по SSH и создаст резервную копию в локальной системе.

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

```
pg_probackup3 backup -B каталог_копий --instance=имя_экземпляра -
b FULL --remote-user=postgres --remote-host=192.168.0.2 --remote-
port=2302
```

Запуск pg_probackup3 в параллельных потоках

Команды backup, restore, merge, delete 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` и `--num-write-threads`. Эти параметры должны быть неотрицательными целыми числами.

Если параметр `--threads` не указан или имеет нулевое значение, pg_probackup3 по умолчанию использует количество ядер процессора. Если определить количество ядер не удастся, будет использоваться один поток.

Если параметр `--num-write-threads` не указан, количество потоков записи будет соответствовать количеству потоков чтения.

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.

> **Важно**
>
> Для выполнения pg_probackup3 в параллельных потоках реализован механизм, исключающий копирование одного файла несколько раз. Но при его использовании в системах с медленными дисками или слишком высокой нагрузкой копирование может завершиться ошибкой. Чтобы исключить такую ситуацию, устраните проблемы своей рабочей среды.

Проверка целостности данных

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.

Проверка резервных копий

pg_probackup3 вычисляет контрольные суммы для всех файлов копии в ходе резервного копирования. Процесс проверки контрольных сумм файлов называется *проверкой целостности копии*. По умолчанию проверка выполняется сразу после создания резервной копии и непосредственно перед восстановлением для выявления возможных повреждений резервных копий.

> **Примечание**
>
> При проверке резервной копии также проверяются контрольные суммы файлов CFS.

Если вы хотите пропустить проверку резервной копии, вы можете передать командам backup и restore флаг `--no-validate`.

Например, чтобы убедиться, что вы можете восстановить кластер баз данных из резервной копии, остановившись на транзакции с идентификатором 4242, выполните команду:

```
pg_probackup3 validate -B каталог_копий --instance=имя_экземпляра
 --recovery-target-xid=4242
```

Если проверка проходит успешно, pg_probackup3 выдаёт сообщение об этом. В случае же неудачи вы получите сообщение об ошибке с указанием точного времени, идентификатора транзакции и значения LSN, до которого возможно восстановление.

Если вы укажете *идентификатор копии* в ключе `-i/--backup-id`, будет проверена только резервная копия с указанным идентификатором. Если *идентификатор копии* указывается вместе с параметрами точки восстановления, команда `validate` проверит, возможно ли восстановить указанную резервную копию до заданной точки.

Например, чтобы убедиться, что можно восстановить кластер баз данных из резервной копии с идентификатором `SBOL6P` до заданного момента времени, выполните команду:

```
pg_probackup3 validate -B каталог_копий --instance=имя_экземпляра -
i SBOL6P --recovery-target-time="2024-04-10 18:18:26+03"
```

Если вы укажете *ид_резервной_копии*, относящийся к инкрементальной копии, будут проверены все нужные ей родительские копии, начиная с полной.

Если вы опустите все параметры, будут проверены все резервные копии.

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.

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.

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.

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 00000001000000000000000A removed
[2025-02-11 15:23:30.446961] [14218] [128670453549440] [info] WAL
 file 00000001000000000000000A.00000028.backup removed
[2025-02-11 15:23:30.448960] [14218] [128670453549440] [info] WAL
 file 00000001000000000000000B removed
[2025-02-11 15:23:30.450991] [14218] [128670453549440] [info] WAL
 file 00000001000000000000000C removed
[2025-02-11 15:23:30.451069] [14218] [128670453549440] [info] WAL
 file 00000001000000000000000C.00000028.backup removed
[2025-02-11 15:23:30.453236] [14218] [128670453549440] [info] WAL
 file 00000001000000000000000D 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 000000010000000000000010 removed
[2025-02-11 15:23:30.459794] [14218] [128670453549440] [info] WAL
 file 000000010000000000000010.00000028.backup removed
[2025-02-11 15:23:30.461135] [14218] [128670453549440] [info] WAL
 file 000000010000000000000011 removed
[2025-02-11 15:23:30.462603] [14218] [128670453549440] [info] WAL
 file 000000010000000000000012 removed
[2025-02-11 15:23:30.462637] [14218] [128670453549440] [info] WAL
 file 000000010000000000000012.00000028.backup removed
[2025-02-11 15:23:30.464003] [14218] [128670453549440] [info] WAL
 file 000000010000000000000013 removed
[2025-02-11 15:23:30.465522] [14218] [128670453549440] [info] WAL
 file 000000010000000000000014 removed
[2025-02-11 15:23:30.465555] [14218] [128670453549440] [info] WAL
 file 000000010000000000000014.00000028.backup removed
[2025-02-11 15:23:30.466910] [14218] [128670453549440] [info] WAL
 file 000000010000000000000015 removed
[2025-02-11 15:23:30.468572] [14218] [128670453549440] [info] WAL
 file 000000010000000000000016 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
```

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`.

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_start_backup(text,
    boolean, boolean) TO backup;
   GRANT EXECUTE ON FUNCTION pg_catalog.pg_stop_backup(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 модуль pg_probackup3 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 5. Справка

Содержание

pg_probackup3 48
libpgprobackup 71

Название

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

Синтаксис

pg_probackup3 version

pg_probackup3 help [*команда*]

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

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]
}
```

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. Вы можете 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.

add-instance

```
pg_probackup3 add-instance -B каталог_копий -D каталог_данных --
instance=имя_экземпляра
[--skip-if-exists] [параметры_s3] [параметры_ssh] [--help]
 [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.

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

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.

set-config

pg_probackup3 set-config -B *каталог_копий* --instance=*имя_экземпляра*
[--help] [--pgdata=*путь_к_pgdata*]
[--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.

Все поддерживаемые параметры описываются в подразделе Параметры.

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

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.

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.

show

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

Показывает содержимое каталога копий. Если заданы *имя_экземпляра* и *ид_резервной_копии*, выводится подробная информация об этой копии. С указанием `--archive` эта команда показывает содержимое архива WAL в данном каталоге.

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.

backup

```
pg_probackup3 backup -B каталог_копий --instance=имя_экземпляра -
b режим_копирования -s источник_данных -i ид_резервной_копии
[--with-file-map] [--help] [-j число_потоков] [--progress]
[--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).

`--backup-threads` *число_потоков*

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

`--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.

`--temp-slot[=true|false|on|off]`

Создаёт *временный* слот физической репликации для передачи WAL с архивируемого экземпляра Postgres Pro. По умолчанию `--temp-slot` включён. Это гарантирует, что все нужные сегменты WAL будут доступны, если в процессе копирования произойдёт переключение сегментов WAL. Этот параметр может использоваться только вместе с параметром `--stream`. По умолчанию имя слота — `pg_probackup_slot`. Чтобы его поменять, воспользуйтесь параметром `--slot`/`-S` и явно укажите `--temp-slot` или `--temp-slot=true|on`.

`-S` *имя_слота*, `--slot=`*имя_слота*

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

`--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.

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

За подробностями обратитесь к подразделу Создание резервной копии.

restore

`pg_probackup3 restore -B` *каталог_копий* `--instance=`*имя_экземпляра*

```
[--help] [-D каталог_данных] [-i ид_резервной_копии]
[--progress] [-T СТАРЫЙ_КАТАЛОГ=НОВЫЙ_КАТАЛОГ]
[--external-mapping=СТАРЫЙ_КАТАЛОГ=НОВЫЙ_КАТАЛОГ] [--skip-external-
dirs]
[--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.

--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 путя-

ми. 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"'`

`--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 удалённого режима, Параметры удалённого архива WAL, Параметры ведения журнала.

За подробностями обратитесь к подразделу Восстановление кластера.

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.

За подробностями обратитесь к подразделу Проверка копии.

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.

delete

```
pg_probackup3 delete -B каталог_копий --instance=имя_экземпляра
[--help] [--progress]
[--dry-run] [--no-sync] [параметры_журнала] [параметры_ssh]
[параметры_s3] [параметры_буферизации]
```

Удаляет резервные копии с указанными *ид_резервной_копии*.

`--no-sync`

Не сбрасывать удаляемые файлы на диск. Использование этого флага может привести к повреждению данных в случае аварии операционной системы или аппаратного сбоя.

За подробностями обратитесь к подразделу Удаление резервных копий.

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] [--compress]
[--archive-timeout=время_ожидания]
[--compress-algorithm=алгоритм_сжатия]
[--compress-level=уровень_сжатия]
[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.)

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.

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.

fuse

```
pg_probackup3 fuse -B каталог_копий --mnt-path=путь_монтирования --
instance=имя_экземпляра
-i ид_резервной_копии [--cache-swap-size=порог_сброса_кеша] [--
help] [ssh_options]
```

[*параметры_журнала*] [*параметры_s3*] [*параметры_буферизации*]

Монтирует каталог резервных копий в виде виртуальной файловой системы и позволяет Postgres Pro работать на ней.

--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.

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

> **Примечание**
>
> Цепочка резервных копий для монтирования должна включать параметр --with-file-map. Этот параметр используется в операциях резервного копирования и объединения.

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] [*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 --s3-config-file, определяющий файл конфигурации S3 с требуемыми параметрами конфигурации, как описано в «Параметры S3».

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

retention

pg_probackup3 retention -B *каталог_копий* --instance=*имя_экземпляра*
[--retention-redundancy] [--retention-window] [--dry-run] [--merge-expired]
[--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.

За подробностями о параметрах команды обратитесь к подразделу Параметры сохранения.

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 «Настройка pg_probackup3».

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`.

`--progress`

Включает вывод прогресса выполнения операций.

`--help`

Выводит подробную информацию по параметрам, которые принимает эта команда.

`-v` *версия*, `--version=`*версия*

Показывает версию pg_probackup3.

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_сообщений`*

Управляет уровнем сообщений, которые будут выводиться в журнал консоли. Допустимые уровни: `verbose`, `log`, `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_сообщений`*

Управляет уровнем сообщений, которые будут выводиться в файл журнала. Допустимые уровни: `verbose`, `log`, `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-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.

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.

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.

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`*

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. Поддерживаются следующие единицы: kB (по умолчанию), MB, GB, TB.

По умолчанию: `0`

`--log-rotation-age=`*`возраст_журнала_для_ротации`*

Максимальное время жизни отдельного файла журнала. Если это значение достигается, файл журнала прокручивается при выполнении какой-либо команды pg_probackup3, за исключением `help` и `version`. Время создания последнего файла журнала сохраняется в `$BACKUP_PATH/log/log_rotation`. Нулевое значение отключает прокрутку по времени. Поддерживаемые единицы: ms (миллисекунды), s (секунды), min (минуты, по умолчанию), h (часы), d (дни).

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

--compress-level=*уровень_сжатия*

Определяет уровень сжатия. Этот параметр можно 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

`--compress`

Задаёт алгоритм сжатия по умолчанию и устанавливает `--compress-level=1`. Алгоритм по умолчанию выбирается среди поддерживаемых Postgres Pro в соответствии с приоритетами: `zstd` (самый высокий) -> `lz4` -> `zlib`. Параметр `--compress` переопределяет параметры `--compression-algorithm` и `--compress-level` и не может быть использован одновременно с ними.

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 вы можете задать параметры удалённого режима и ведения 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.

`--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.

Параметры буферизации

Эти параметры могут использоваться со всеми командами.

`--buffer-size=`*`размер`*

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

Параметры удалённого режима

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 рассказывается в «Настройка удалённого режима» и «Использование pg_probackup3 в удалённом режиме».

`--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.

`--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 восстановления

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

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

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.

`--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`.
- `vk` — хранилище VK Cloud. С этим провайдером используются только адрес узла S3 `hb.vkcs.cloud`, порт 443 и протокол HTTPS. Пользовательские значения узла, порта и протокола игнорируются. Регион по умолчанию — `ru-msk`.
- `aws` — хранилище Amazon S3 от Amazon Web Services (AWS). С этим провайдером используются только адрес узла S3 *`имя_бакета`*`.s3.`*`регион`*`.amazonaws.com`, порт 443 и протокол HTTPS. Пользовательские значения узла, порта и протокола игнорируются. Регион по умолчанию — `us-east-1`.
- `google` —

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.

Когда команда pg_probackup3 запускается с ключом `--s3`, все операции, поддерживающие параллельное выполнение, по умолчанию выполняются параллельно в 10 потоков (за подробностями обратитесь к «Запуск pg_probackup3 в параллельных потоках»). Количество потоков можно изменить с помощью ключа `-j`/`--threads`.

`--s3-config-file=`*`путь_к_файлу_конфигурации`*

Задаёт файл конфигурации S3. Параметры, заданные в этом файле, переопределяют переменные окружения. Если этот параметр опускается, pg_probackup3 ищет файл конфигурации S3 сначала в `/etc/pg_probackup/s3.config`, а затем в `~postgres/.pg_probackup/s3.config`. Ниже приведён пример файла конфигурации S3:

```
access-key = ...
secret-key = ...
s3-host = localhost
s3-port = 9000
s3-bucket = s3demo
s3-region=us-east-1
s3-buffer-size = 32
s3-secure = on | https | http | off
```

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 ниже 2.6.0 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 ко-

пии как скрытые. Добиться такого состояния копии можно, только вручную отредактировав файл `backup.control`. Задать этот параметр можно только в переменных окружения.

`--destroy-all-other-dbs`

По умолчанию pg_probackup3 завершает работу ошибкой при попытке выполнить частичное инкрементальное восстановление, поскольку при этом удаляются базы данных, не включённые в список восстановления. Этот флаг позволяет игнорировать ошибку и продолжать частичное инкрементальное восстановление (например, чтобы снимок тестовой БД находился в том же состоянии, что и снимок производственной БД). Этот параметр можно использовать с командой restore.

> **Важно**
>
> Никогда не используйте этот флаг в производственном кластере.

`PGPROBACKUP_TESTS_SKIP_EMPTY_COMMIT`

Указывает pg_probackup3 пропускать пустые транзакции после pg_backup_stop.

Авторы

Postgres Professional, Москва, Россия.

Название

libpgprobackup — библиотека с API для резервного копирования данных, восстановления из резервных копий, а также проверки и объединения резервных копий

Описание

Библиотека libpgprobackup содержит функции для резервного копирования, восстановления из резервных копий, а также проверки и объединения резервных копий. Предоставляемый API позволяет создать собственное приложение для резервного копирования и восстановления данных вместо использования утилиты командной строки pg_probackup.

Библиотека хранит резервные копии в файлах собственного формата.

Библиотека берёт на себя взаимодействие с сервером баз данных, обработку данных, кодирование и декодирование, а также создание и хранение метаданных для резервных копий.

Приложение, использующее библиотеку, должно предоставить ей доступ к хранилищу, такому как файловая система, хранилище S3 или лента. Однако за операции в файловой системе, такие как чтение и хранение резервных копий, а также хранение метаданных, отвечает приложение. Приложение взаимодействует с libpgprobackup следующим образом:

1. Приложение подготавливает экземпляр базы данных к резервному копированию и вызывает функцию libpgprobackup backup.
2. libpgprobackup преобразует файлы данных и файлы WAL в формат pg_probackup3 и передаёт их приложению.
3. Клиентское приложение отправляет данные в файл в целевом хранилище (файловая система, хранилище S3 или лента) и сохраняет метаданные резервной копии. Промежуточное хранилище не используется.
4. Для восстановления из резервной копии с помощью функции libpgprobackup restore, выполнения проверки целостности резервной копии с помощью функции validate или объединения резервных копий с помощью функции merge, приложение предоставляет библиотеке функции для получения данных и метаданных резервной копии.

libpgprobackup реализована на C++, но может использоваться в приложениях, написанных на различных языках программирования. Интеграция с приложениями на C/C++ должна вызывать наименьшие сложности. Библиотека использует соглашение о вызовах `extern "C"`.

Структуры и функции libprobackup объявлены в файле `probackup_lib.h`.

Функции

Библиотека libprobackup содержит функции, описанные ниже. При вызове функций библиотеки они принимают структуры с параметрами команд и структуру с указателями на функции, которые работают с файлами и метаданными.

bool `backup` (ConnectOptionsLib *`conn_opt`, BackupOptionsLib *`backup_opt`, MetadataProviderLib *`metadata`);

> Подключается к локальному или удалённому серверу и выполняет резервное копирование. Возвращает `true` в случае успеха и `false` в случае ошибки.

Аргументы:

- *conn_opt* — указатель на структуру ConnectOptionsLib, которая задаёт параметры подключения к серверу Postgres Pro, такие как `pghost`, `pgdatabase`, `pgport`, `pguser`, `password`.
- *backup_opt* — указатель на структуру BackupOptionsLib, которая содержит параметры резервного копирования, такие как режим резервного копирования или количество потоков.
- *metadata* — указатель на структуру MetadataProviderLib, которая предоставляет функции-обработчики для метаданных резервных копий и файловых операций.

bool `restore` (RestoreOptionsLib **restore_opt*, MetadataProviderLib **metadata*);

Восстанавливает данные из резервной копии с использованием параметров, переданных в соответствующих структурах, в локальную файловую систему. Возвращает `true` в случае успеха и `false` в случае ошибки.

Аргументы:

- *restore_opt* — указатель на структуру RestoreOptionsLib, содержащую параметры для восстановления из резервной копии.
- *metadata* — указатель на структуру MetadataProviderLib, которая предоставляет функции-обработчики для метаданных резервных копий и файловых операций.

bool `validate` (RestoreOptionsLib **restore_opt*, MetadataProviderLib **metadata*, bool *withParents*);

Проверяет, что все файлы, необходимые для восстановления кластера из резервной копии, присутствуют и не повреждены. Если параметр *withParents* установлен в значение true, также проверяются все резервные копии в цепочке родительских. Возвращает `true` в случае успеха и `false` в случае ошибки.

Аргументы:

- *restore_opt* — указатель на структуру RestoreOptionsLib, содержащую параметры для проверки целостности резервной копии.
- *metadata* — указатель на структуру MetadataProviderLib, которая предоставляет функции-обработчики для метаданных резервных копий и файловых операций.
- *withParents* — логическое значение, определяющее, необходимо ли также проверять родительские резервные копии.

bool `merge` (MergeOptionsLib **merge_opt*, MetadataProviderLib **metadata*);

Объединяет цепочку инкрементных резервных копий в одну полную. Во время объединения создаётся новая полная резервная копия, включающая все родительские. Если у родительских копий нет дополнительных зависимостей, они удаляются после успешного объединения. Возвращает `true` в случае успеха и `false` в случае ошибки.

Аргументы:

- *merge_opt* — указатель на структуру MergeOptionsLib, содержащую параметры для объединения резервных копий.

- *metadata* — указатель на структуру MetadataProviderLib, которая предоставляет функции-обработчики для метаданных резервных копий и файловых операций.

uint64_t identify_system (ConnectOptionsLib **conn_opt*);

Возвращает уникальный идентификатор сервера Postgres Pro. Он используется для управления резервными копиями и их восстановления на системном уровне.

Аргументы:

- *conn_opt* — указатель на структуру ConnectOptionsLib, которая задаёт параметры подключения к серверу Postgres Pro.

void set_probackup_logger (Logger *info*, Logger *warning*, Logger *error*);

Определяет три функции-обработчика записи в журнал, которые libpgprobackup будет использовать для вывода информационных сообщений, предупреждений или сообщений об ошибках. Каждая из этих функций должна принимать строковый параметр с сообщением, передаваемым библиотекой.

Аргументы:

- *info* — указатель на функцию, которая будет обрабатывать информационные сообщения.

- *warning* — указатель на функцию, которая будет обрабатывать предупреждения.

- *error* — указатель на функцию, которая будет обрабатывать сообщения об ошибках.

Структуры для работы с файлами

Библиотека получает обратную связь от приложения через структуру типа MetadataProviderLib, которая должна содержать указатели на функции, работающие с файлами и метаданными.

Используются обработчики из библиотеки для функций, переданных приложением. Указатели на функции передаются в библиотеку через структуру MetadataProviderLib. Операции чтения или записи файлов определяются в структурах CSource и Csink, соответственно.

Для обратной связи от приложения структуры библиотеки содержат указатель void *thisPointer, в котором приложение может хранить указатель на свою собственную функцию или экземпляр класса. Этот указатель передаётся в функции-обработчики при их вызове из библиотеки.

MetadataProviderLib

Указатель на эту структуру передаётся всем основным функциям libprobackup, таким как backup, restore, validate, и merge. Эта структура определяет методы для работы с данными резервных копий, для записи и чтения метаданных резервных копий, а также для получения списка резервных копий.

```
typedef struct
{
    CSink *(*get_sink_for_backup)(const char *backup_id, void
 *thisPointer);
    CSource *(*get_source_for_backup)(const char *backup_id, void
 *thisPointer);

    void (*register_backup)(PgproBackup *backup, void *ptr);
    PgproBackup *(*get_backup_by_id)(const char *backup_id, void
 *thisPointer);
    void (*free_backup)(PgproBackup *backup, void *thisPointer);
    char **(*list_backup_ids)(void *thisPointer);

    bool (*write_backup_status)(const char *backup_id, BackupStatus
 status,
                                void *thisPointer);

    void *thisPointer;
} MetadataProviderLib;
```

Здесь:

get_sink_for_backup

Указатель на функцию, которая возвращает указатель на структуру CSync (см. CSource и Csink для получения более подробной информации). Требуется для записи информации в резервную копию. *backup_id* содержит строку с идентификатором резервной копии. В *thisPointer* приложение получает указатель, который ранее был передан в библиотеку через структуру.

get_source_for_backup

Указатель на функцию, которая возвращает указатель на структуру CSource (см. CSource и Csink для получения более подробной информации). Требуется для чтения информации из резервной копии. *backup_id* содержит строку с идентификатором резервной копии. В *thisPointer* приложение получает указатель, который ранее был передан в библиотеку через структуру.

register_backup

Указатель на функцию, которая сохраняет метаданные резервной копии. Принимает структуру PgproBackup.

get_backup_by_id

Указатель на функцию, которая получает метаданные резервной копии, определяемой параметром *backup_id*.

free_backup

Освобождает память для структуры PgproBackup, возвращаемой функцией `get_backup_by_id`.

list_backup_ids

Возвращает список доступных идентификаторов резервных копий. Список содержит указатели на строки, заканчивающиеся на ноль. Последний указатель в списке также должен быть нулевым. Память для списка должна быть выделена с помощью функции языка C, такой как `malloc`

или `strdup`, поскольку библиотека освобождает память с использованием функции `free`.

`write_backup_status`

Указатель на функцию, которая сохраняет статус резервной копии. Статус резервной копии обновляется отдельно от сохранения остальных метаданных резервной копии.

`void *thisPointer`

Указатель, который передаётся из библиотеки во все функции-обработчики.

CSource и Csink

Эти структуры необходимы для чтения и записи блоков данных в файл резервной копии. Указатели на эти функции должны возвращаться функциями get_source_for_backup и get_sink_for_backup, соответственно. Каждая из этих структур фактически представляет собой файл резервной копии, который открыт соответственно для чтения или записи.

```
/* Support structure */
typedef struct
{
    unsigned char *ptr;
    size_t         len;
} c_buffer_t;

typedef struct
{
    c_buffer_t (*read_one)(void *thisPointer);
    ssize_t (*read)(char *buf, size_t size, void *thisPointer);
    void (*close)(void *thisPointer);
    void *thisPointer;
} CSource;
typedef struct
{
    /* Write one Pb structure. See @PbStructs. */
    size_t (*write_one)(uint8_t *buf, size_t size, void
 *thisPointer);
    ssize_t (*write)(char *buf, size_t size, void *thisPointer);
    /* Close this Sink. No more calls will be done. */
    void (*close)(void *thisPointer);
    void *thisPointer;
} CSink;
```

В поле `thisPointer` приложение может передать в структуру указатель на структуру или класс приложения, который, например, содержит дескриптор открытого файла.

Функции из структуры CSource используются для чтения файла резервной копии, а функции из структуры CSink — для записи. Для этого каждая из этих структур в настоящее время имеет по две функции. Функции `read_one` и `write_one` выполняют операции низкого уровня с блоком данных. `write_one` сохраняет размер блока, в то время как `read_one` сначала читает размер блока, а затем сам блок данных этого размера.

Функции `read` и `write` используются для упрощённой реализации чтения/записи. Их аргументы аналогичны аргументам общих функций для работы с

файлами. Обработка размера блока выполняется внутри библиотеки. Эти функции должны возвращать размер прочитанных/записанных данных или 1 в случае ошибки.

Функции `close` должны закрывать файл.

PgproBackup

В этой структуре передаются метаданные резервной копии.

```
typedef struct
{
    BackupStatus     backup_status;
    BackupMode       backup_mode;      /* Mode - one of
 BACKUP_MODE_xxx */
    char            *backup_id;        /* Identifier of the backup.
 */
    char            *parent_backup_id; /* Identifier of the parent
 backup. */
    BackupTimeLineID tli;       /* timeline of start and stop
 backup lsns */
    BackupXLogRecPtr start_lsn; /* backup's starting transaction
 log location */
    BackupXLogRecPtr stop_lsn; /* backup's finishing transaction
 log location */
    BackupTimestampTz   start_time; /* UTC time of backup creation
 */
    BackupTransactionId minxid;     /* min Xid for the moment of
 backup start */
    BackupMultiXactId
            minmulti; /* min multixact for the moment of backup
 start */

    bool stream; /* Was this backup taken in stream mode? I.e. does
 it include
                    all needed WAL files? */
    bool  from_replica;     /* Was this backup taken from replica
 */
    char *primary_conninfo; /* Connection parameters of the backup
 in the format
                               suitable for recovery.conf */
    char *note;

    /* For compatibility check */
    uint32_t block_size;
    uint32_t wal_block_size;
    char    *program_version;
    int      server_version;

    size_t uncompressed_bytes; ///< Size of data and non-data files
 before
                               ///< compression is applied
    size_t data_bytes; ///< Size of data and non-data files after
 compression is
                       ///< applied
    CompressAlg compress_alg;
    int         compress_level;

    BackupTimestampTz
```

```
            end_time; /* the moment when backup was finished, or
 the moment
                        * when we realized that backup is broken */
    BackupTimestampTz
            end_validation_time; /* UTC time when validation
 finished */

    BackupSource backup_source;/* direct, base or pro backup
 method*/

    size_t wal_bytes;// not used in pb3
} PgproBackup;
```

После выполнения функции backup библиотека вызывает функцию-обработчик register_backup, в которую передаётся заполненная структура PgproBackup. Переданную информацию можно сохранять по своему усмотрению.

Для получения информации о существующей резервной копии библиотека использует функцию-обработчик get_backup_by_id. Из этой функции приложение должно вернуть указатель на структуру PgproBackup с метаданными резервной копии, имеющей указанный идентификатор, или нулевой указатель в случае ошибки.

Чтобы освободить память для структуры PgproBackup приложение должно вызвать функцию-обработчик free_backup.

Общие параметры

В этом разделе перечисляются структуры, используемые для передачи параметров командам. Для получения более подробной информации обратитесь к заголовочному файлу библиотеки `probackup_lib.h`.

connectOptionsLib

Следующая структура определяет параметры для подключения к серверу Postgres Pro:

```
typedef struct connectOptionsLib
{
/* The database name. */
const char *pgdatabase;
/* Name of host to connect to. */
const char *pghost;
/* Port number to connect to at the server host, or socket file
 name
 * extension for Unix-domain connections.*/
const char *pgport;
/* Postgres Pro user name to connect as. */
const char *pguser;
/* Password to be used if the server demands password
 authentication. */
const char *password;
} ConnectOptionsLib;
```

backupOptionsLib

Ниже представлена структура, которая определяет параметры для создания резервной копии. На данный момент поддерживаются только режимы резервного копирования `FULL` и `DELTA`.

```
typedef struct backupOptionsLib
{
    /* Number of threads, if backup mode supports multithreading */
    int num_threads;
    /* Backup mode PAGE, PTRACK, DELTA, AUTO, FULL*/
    BackupMode backup_mode;
    /* Backup source DIRECT, BASE or PRO */
    BackupSource backup_source;
    /* For DIRECT source is required to set up PGDATA. It is not
 required for
     * other sources */
    const char *pgdata;
    /* Backup Id if you wants to use custom id, otherwise it will
 be generated a
     * unique id using datetime of creation */
    const char *backup_id;
    /* Id of parent backup. It is not required for FULL backup mode
 */
    const char *parent_backup_id;
    /* Name of replication slot, if you use custom slot created by
     * pg_create_physical_replication_slot(). Otherwise will be
 used auto
     * generated slot */
    const char *replication_slot;
    bool        create_slot;
    const char *backup_note;
    /* If true, then WAL will be sent in stream mode, otherwise in
 archive mode
     */
    bool stream_wal;
    /* Progress flag. If true progress will be logged */
    bool        progress;
    const char *external_dir_str;
    /* Verify check sums. It is available only if checksums turn on
 on
     * PostgresPro server */
    bool verify_checksums;
    /* Compression algorithm, that is used for sending data between
 PostgresPro
     * server and libpgprobackup */
    CompressAlg compress_alg;
    /* Compression level, that is used for sending data between
 PostgresPro
     * server and libpgprobackup */
    int compress_level;
    /* Wait timeout for WAL segment archiving */
    uint32_t archive_timeout;
} BackupOptionsLib;
```

restoreOptionsLib

Следующая структура определяет параметры для восстановления из резервной копии. Эти параметры также используются для проверки резервной копии:

```
typedef struct restoreOptionsLib
{
    /* Number of threads */
    int         num_threads;
```

```
    /* Path to pgdata for restoration */
    const char *pgdata;
    /* Backup ID for restoration */
    char *backup_id;
    /* Progress flag. If true progress will be logged */
    bool        progress;
    /* Skip external directories */
    bool        skip_external_dirs;
    const char *external_mapping;
    const char *tablespace_mapping;
    /* Checksum page verification */
    bool        verify_checksums;
    /* Restore command to write in postgresql.auto.conf */
    /* https://postgrespro.ru/docs/enterprise/16/runtime-config-
wal#GUC-RESTORE-COMMAND */
    const char *config_content;
    /* Need recovery signal */
    bool need_recovery_signal;
    /* Need standby signal */
    bool need_standby_signal;
    /* No synchronization */
    bool no_sync;
} RestoreOptionsLib;
```

mergeOptionsLib

Следующая структура определяет параметры для объединения резервных копий:

```
typedef struct mergeOptionsLib
{
    /* Number of threads */
    int         num_threads;
    /* Path to pgdata to restore to */
    const char *pgdata;
    /* Incremental backup ID for the merge */
    const char *backup_id;
    /* Target backup ID for the merge */
    const char *target_backup_id;
    /* Progress flag. If true progress will be logged */
    bool        progress;
    /* ID of the last increment */
    const char *merge_from_id;
    /* Time interval within which to merge backups */
    int            interval;
} MergeOptionsLib;
```

Константы

BackupMode

```
/*
 Backup Mode is how backup is taken.
 All DIFF modes require parent backup id passed in the
 BackupOptions.
 Parent backup id is ignored in the FULL mode.

 DIFF_AUTO returns selected mode in the metadata. Tentatively it
 prefers
```

```
 DIFF_PAGE if WAL summarization is available, if not it tries to do
 DIFF_PTRACK if ptrack is enabled and finally it falls back to
 DIFF_DELTA.
 In case when even DIFF_DELTA is not possible (no parent full
 backup exists)
 FULL backup is taken.
*/
typedef enum BackupMode
{
    BACKUP_MODE_INVALID = 0,
    BACKUP_MODE_DIFF_PAGE,   /* incremental page backup */
    BACKUP_MODE_DIFF_PTRACK, /* incremental page backup with ptrack
 system */
    BACKUP_MODE_DIFF_DELTA,  /* incremental page backup with lsn
 comparison */
    BACKUP_MODE_DIFF_AUTO, /* library selects diff backup mode
 automatically */
    BACKUP_MODE_FULL       /* full backup */
} BackupMode;
```

CompressAlg

```
/*
* Compression mode which is used to transfer data between PG server
 and the client lib.
* Default is NONE_COMPRESS.
*/
typedef enum CompressAlg
{
    NONE_COMPRESS = 0,
    ZLIB_COMPRESS,
    LZ4_COMPRESS,
    ZSTD_COMPRESS,
} CompressAlg;
```

BackupSource

```
/**
Backup Source is a method used by the client to access PGDATA.
*/
typedef enum
{
    /*
     * Direct access. The client reads data files directly.
     * Opens normal connection to execute PG_BACKUP_START/STOP.
     * Local file access. Multithreaded. No special PostgresPro
 edition
     * required.
     */
    BACKUP_SOURCE_DIRECT,
    /*
     * Uses pg_basebackup protocol.
     * Opens replication connection.
     * Remote file access. No special PostgresPro edition required.
     */
    BACKUP_SOURCE_BASE_BACKUP,
    /*
     * Uses pg_probackup protocol.
```

```
     * Opens replication connection.
     * Remote file access. Multithreaded. Only works with
 PostgresPro builds.
     * Supported PostgresPro versions start with ent-15.
     */
    BACKUP_SOURCE_PRO_BACKUP
} BackupSource;
```

Приложение A. Замечания к выпускам

Содержание

pg_probackup 3.0.0 82

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, доступный исключительно в Postgres Pro Enterprise.
- Объединение цепочек инкрементальных копий. Для экономии места на диске теперь можно объединять цепочки инкрементальных резервных копий.

Предметный указатель

C

command
- add-instance, 49
- archive-get, 58
- archive-push, 57
- backup, 51
- del-instance, 50
- delete, 56
- fuse, 57
- help, 49
- init, 49
- merge, 55
- restore, 53
- retention, 58
- set-backup, 50
- set-config, 50
- show, 51
- show-config, 51
- validate, 55
- version, 49

L

libpgprobackup, 71
- backup, 71
- identify_system, 73
- merge, 72
- restore, 72
- set_probackup_logger, 73
- validate, 72

P

pg_probackup3, 48