RuBackup

Система резервного копирования
и восстановления данных

POSTGRESQL, TANTOR И JATOBA

ВЕРСИЯ 2.9.0.0.0

Содержание

Глава 1. Назначение

Резервное копирование и восстановление СУБД *PostgreSQL*, *Tantor Special Edition* и *Jatoba* выполняется с помощью модуля PostgreSQL Universal, входящего в состав СРК RuBackup.

Таблица 1. Поддерживаемые версии СУБД

PostgreSQL	11, 12, 13, 14, 15, 16, 17, 18
Tantor Special Edition	15, 1C 15, 16, 17, 18
Jatoba	6

Таблица 2. Подмодули

`postgresql`	Использует низкоуровневый API СУБД для выполнения резервного копирования
`pg_probackup`	Использует утилиту `pg_probackup` для выполнения резервного копирования СУБД
`superb`	Резервное копирование и восстановление СУБД в режиме непрерывного резервного копирования и резервного копирования архивных журналов транзакций

Глава 2. Резервируемые данные

Осуществляется резервное копирование всей СУБД и архивных журналов транзакций (WAL).

При выполнении резервного копирования данных для подмодулей `postgresql` и `superb` создаются резервные копии:

- файлов данных СУБД, расположенных в директории, которая определяется автоматически или задается параметром `data_directory` в файле `postgresql.conf` (Глава 10);
- архивных журналов транзакций (WAL-файлов), расположенных в директории, которая определяется параметром `archive_catalog` в файле `/opt/rubackup/etc/rb_module_postgresql.conf` (Глава 10);
- файлов настроек СУБД, независимо от их расположения:
  - `postgresql.conf`. Если в конфигурационный файл `postgresql.conf` включены другие конфигурационные файлы (с помощью директив `include`, `include_if_exists`, `include_dir`), то будет создана резервная копия этих файлов;
  - `pg_hba.conf`;
  - `pg_ident.conf`.

Глава 3. Типы резервного копирования

Модуль поддерживает следующие типы резервного копирования СУБД:

- полное,
- инкрементальное.

Глава 4. Типы восстановления данных

Модуль поддерживает полное восстановление СУБД из резервных копий с развертыванием и без развертывания на целевом ресурсе.

Глава 5. Способы резервного копирования

Модуль поддерживает резервное копирование СУБД с помощью:

- приложений:
  - Tucana (рекомендуемый способ),
  - Менеджер администратора RuBackup (RBM),
  - Менеджер клиента RuBackup (RBC),
- утилит командной строки.

Глава 6. Способы восстановления данных

Модуль поддерживает следующие способы восстановления СУБД из резервных копий:

- Централизованное восстановление с помощью:
  - приложений:
    - Tucana (рекомендуемый способ),
    - Менеджер администратора RuBackup (RBM),
  - утилит командной строки.
- Локальное восстановление на клиенте резервного копирования с помощью:
  - приложения Менеджер клиента RuBackup (RBC),
  - утилит командной строки.

Глава 7. Способы аутентификации в СУБД

Модуль поддерживает три способа аутентификации в СУБД.

1. Базовый. Для аутентификации используются учетные данные пользователя СУБД из конфигурационного файла модуля.
2. Через Unix-сокет с помощью метода `peer`. Для аутентификации используются полномочия пользователя ОС.
3. С помощью внешнего хранилища секретов. Для аутентификации используются учетные данные пользователя СУБД в зашифрованном виде.

Таблица 3. Хранилища секретов

Хранилище секретов	Операционные системы
HashiCorp Vault 1.16.3	Astra Linux 1.7
	Ubuntu 20.04
	RHEL 8

Глава 8. Комплект поставки

Дистрибутив модуля поставляется в виде deb-пакета с именем `rubackup-postgresql_<version>_amd64_signed.deb` и в виде rpm-пакета с именем `rubackup-postgresql-<version>.x86_64.rpm`, где `<version>` — номер версии поставляемого модуля.

Пакет доступен для скачивания на официальном сайте https://www.rubackup.ru/go/.

Глава 9. Ограничения

Запустить одновременно две операции резервного копирования или восстановления для модуля на одном хосте невозможно. Выполнение резервного копирования СУБД производится методом, не предусматривающим параллельных задач резервного копирования и восстановления для одной и той же СУБД. Если параллельная задача будет запущена, она завершится с ошибкой.

Во время выполнения задачи на создание инкрементальной резервной копии в RuBackup производится анализ WAL-файлов. Если анализ показывает, что в текущем WAL-файле изменилась линия времени по сравнению с последним WAL-файлом из предыдущей итерации создания резервной копии, то вместо инкрементальной копии создается полная резервная копия. Это актуально для подмодулей `postgresql` и `superb`.

При выполнении восстановления с развертыванием существующий кластер баз данных СУБД *PostgreSQL* будет уничтожен, а на его месте будет восстановлен кластер баз данных из резервной копии. Перед операцией восстановления рекомендуется принудительно остановить работу всех клиентов с СУБД и выполнить базовое (полное) резервное копирование!

Рекомендуем *отключить возможность централизованного восстановления* СУБД на клиенте и выполнять восстановление из резервной копии только со стороны клиента под контролем администратора СУБД.

Централизованное восстановление и восстановление с развертыванием рекомендуем предварительно выполнять на резервном хосте (виртуальной машине) для проверки корректности восстановления СУБД.

Если на хосте, который не является лидером, в поле **Ресурс** отсутствуют необходимые ресурсы, то выберите клиент, который является лидером.

Сделать инкрементальную резервную копию одного ресурса в разные пулы при включенном параметре `auto_remove_wal` невозможно, потому что файлы журналов удаляются. Чтобы создать инкрементальную резервную копию одного ресурса в разных пулах, установите в конфигурационном файле `/opt/rubackup/etc/rb_module_postgresql.conf` параметр `auto_remove_wal` в значение `no`.

При создании инкрементальной резервной копии ресурса следует учитывать метод получения секрета при выполнении предыдущих резервных копий выбранного ресурса. Созданные РК будут находится в одном репозитории при условии использования одного и того же метода получения секрета для подключения к резервируемому ресурсу. В случае выполнения инкрементальной РК ресурса с другим методом получения секрета, который был использован для получения

предыдущей РК, будет выполнено полное резервное копирование ресурса и созданная РК будет помещена в другой репозиторий.

Глава 10. Подготовка СУБД PostgreSQL

Управление резервным копированием выполняется командами с сервера клиенту. Взаимодействие с СУБД выполняется модулем PostgreSQL Universal на узле клиента (локально).

Резервное копирование СУБД может быть выполнено, если

- обеспечена сетевая доступность узла СУБД с сервера и сервера — с клиента;
- в настройках СУБД разрешено локальное подключение к СУБД пользователя, указанного в настройках модуля PostgreSQL Universal;
- в настройках СУБД настроен режим архивирования.

Модуль PostgreSQL Universal поддерживает три способа аутентификации в СУБД.

Для аутентификации с помощью Unix-сокета (метод `peer`) выполните подготовку СУБД по инструкции из Раздел 10.1.

Для базовой аутентификации или аутентификации с использованием внешнего хранилища секретов выполните подготовку СУБД по инструкции из Раздел 10.2.

10.1. Подготовка СУБД для аутентификации через Unix-сокет

Аутентификация через Unix-сокет с использованием метода `peer` позволяет пользователю ОС получить доступ к СУБД без ввода пароля.

Настройте доступ к СУБД из-под учетной записи пользователя `root`.

1. В файле `pg_ident.conf` укажите соответствие системного пользователя `root` и пользователя базы данных `postgres`.

   Пример 1. Правило соответствия

   ```
   root_to_postgres   root  postgres
   ```

2. Добавьте в файл `pg_hba.conf` строку, разрешающую локальные подключения к СУБД через Unix-сокет (метод `peer`) с использованием созданного соответствия.

   ```
   local   all    all  peer  map=root_to_postgres
   ```

10.2. Подготовка СУБД для аутентификации через пользователя СУБД

10.2.1. Создание и настройка прав пользователя СУБД

Если для доступа к СУБД используется учетная запись не администратора СУБД, то для создания базовой резервной копии пользователь должен обладать правами на вызов функций начала и окончания резервного копирования экземпляра СУБД.

1. Создайте в СУБД пользователя `rubackup_backuper`.

```
create user rubackup_backuper password '12345'; (1)
alter role rubackup_backuper with login;
```

(1) Вместо `12345` укажите желаемый пароль.

2. Если вы планируете выполнять потоковое резервное копирование журналов транзакций (режим STREAM), то выдайте пользователю `rubackup_backuper` права на выполнение репликации.

```
alter role rubackup_backuper with replication;
```

3. Настройте права пользователя в СУБД для доступа.

Пример 2. Предоставление прав на выполнение функций в PostgreSQL версии 15 и выше

```
grant execute on function pg_backup_start(text,bool) to
rubackup_backuper; (1)
grant execute on function pg_backup_stop(bool) to rubackup_backuper; (1)
grant execute on function pg_switch_wal() to rubackup_backuper;
grant pg_read_all_settings to rubackup_backuper;
```

(1) См. официальную документацию PostgreSQL.

Пример 3. Предоставление прав на выполнение функций в PostgreSQL версии 14 и ниже

```
grant execute on function pg_start_backup(text,bool,bool) to
rubackup_backuper; (1)
grant execute on function pg_stop_backup(bool,bool) to
rubackup_backuper; (1)
grant execute on function pg_switch_wal() to rubackup_backuper;
grant pg_read_all_settings to rubackup_backuper;
```

1 См. официальную документацию PostgreSQL.

Если вы планируете использовать базовый тип аутентификации в СУБД, то в дальнейшем при настройке модуля в конфигурационном файле укажите имя созданного пользователя (`username`) и его пароль (`password`).

Если вы планируете использовать внешнее хранилище секретов для аутентификации в СУБД, то в дальнейшем добавьте учетные данные созданного пользователя в такое хранилище.

10.2.2. Доступ пользователя к СУБД

Правила доступа к СУБД задаются в файле `pg_hba.conf`. Файл устанавливает метод подключения (`local` — локально, `host` — по TCP/IP), ограничивает доступ базой данных или задачей (`replication`), предоставляет доступ пользователю с заданных адресов указанным методом аутентификации.

Добавьте в файл `pg_hba.conf` строки, указывающие на право пользователя `rubackup_backuper` на локальный (`localhost`, `127.0.0.1`) доступ по TCP/IP к СУБД для резервного копирования и репликации. Закомментируйте строки, предоставляющие доступ неограниченному кругу лиц (`host all all`).

```
host    backupdb     rubackup_backuper  127.0.0.1/32  md5
host    replication rubackup_backuper  127.0.0.1/32  md5
```

Пример 4. Пример файла `pg_hba.conf`

```
# TYPE DATABASE    USER              ADDRESS       METHOD

# "local" is for Unix domain socket connections only
local  all         all                             md5

# IPv4 local connections:
# host   all         all               127.0.0.1/32  md5

# IPv6 local connections
# host   all         all               ::1/128       md5

# Allow replication connections from localhost, by a user with the
# replication privilege
local  replication all                             md5
host   replication all               127.0.0.1/32  md5
host   replication all               ::1/128       md5
host   backupdb    rubackup_backuper  127.0.0.1/32  md5
```

```
host    replication rubackup_backuper  127.0.0.1/32  md5
```

Если вы хотите подключаться из локальной сети (`192.168.1.*`) непосредственно к СУБД, добавьте в `pg_hba.conf` строку:

```
host    any    any    192.168.1.1/32  md5
```

Для доступа к СУБД необходимо открыть порт `5432` (или иной заданный в `postgresql.conf`) в фаерволе узла клиента. Пользователь, для которого требуется доступ к СУБД, должен существовать и иметь необходимые полномочия.

10.3. Настройка SSL-соединения

Настройка SSL-соединения осуществляется на этапе подготовки СУБД *PostgreSQL* (см. Глава 10).

10.3.1. Настройка SSL-соединения на сервере PostgreSQL

1. Из Центра сертификации скопируйте в папку `/etc/postgresql/16/main` на сервер *PostgreSQL* подготовленные:
   - сертификат Центра сертификации (например, `ca.crt`);
   - подписанный сертификат сервера *PostgreSQL* (например, `server.crt`);
   - сгенерированный закрытый ключ сервера *PostgreSQL* (например, `server.key`).
2. Для сертификатов и закрытого ключа установите:
   - доступ на чтение и запись только для владельцев:

     ```
     chmod 600 ca.crt server.crt server.key
     ```

   - владельца и группу пользователя `postgres`:

     ```
     chown postgres:postgres ca.crt server.crt server.key
     ```

3. В конфигурационном файле `/etc/postgresql/12/main/postgresql.conf`[1]:
   - включите поддержку зашифрованных соединений:

```
ssl = on
```

- укажите путь к файлу сертификата Центра сертификации:

```
ssl_ca_file = '/etc/postgresql/16/main/ca.crt'
```

- укажите путь к файлу сертификата сервера *PostgreSQL*:

```
ssl_cert_file = '/etc/postgresql/16/main/server.crt'
```

- укажите путь к файлу закрытого ключа сервера *PostgreSQL*:

```
ssl_key_file = '/etc/postgresql/16/main/server.key'
```

4. Для доступа модуля к серверу *PostgreSQL* только по доверенному сертификату в конфигурационном файле `/etc/postgresql/12/main/pg_hba.conf`[1] укажите:

   - тип подключения `hostssl`;
   - метод аутентификации по сертификату `cert`, если не требуется дополнительная аутентификация по паролю;
   - параметр аутентификации `clientcert`. При значении:
     - `verify-full` сервер *PostgreSQL* проверяет, подписан ли сертификат доверенным центром (CA) и совпадает ли имя клиента с именем в сертификате (CN);
     - `verify-ca` сервер *PostgreSQL* проверяет, подписан ли сертификат клиента доверенным центром (CA).

     Пример 5. Пример настройки подключения

     ```
     # TYPE    DATABASE    USER                ADDRESS
     METHOD

     hostssl   all         rubackup_backuper   127.0.0.1/32        cert
     clientcert=verify-ca
     ```

     Если параметр `clientcert` не указан, сервер не требует и не проверяет сертификат клиента.

5. Для применения изменений перезапустите сервер:

```
sudo systemctl restart postgresql
```

10.3.2. Настройка SSL-соединения на узле с модулем *PostgreSQL*

1. Из Центра сертификации перенесите в любую папку на узле с модулем *PostgreSQL*:
   - сертификат Центра сертификации (например, `ca.crt`);
   - сертификат для модуля *PostgreSQL* (например, `postgresql.crt`);
   - сгенерированный закрытый ключ для модуля *PostgreSQL* (например, `postgresql.key`).
2. Для сертификатов и закрытого ключа установите:
   - доступ на чтение и запись только для владельцев:

     ```
     chmod 600 ca.crt postgresql.crt postgresql.key
     ```

   - владельца и группу пользователя:

     ```
     chown suser:suser ca.crt postgresql.crt postgresql.key
     ```

     где `suser` — пользователь, от имени которого будет запущен клиент СРК с установленным модулем *PostgreSQL*.
3. В дальнейшем при настройке модуля настройте SSL-соединение с СУБД в конфигурационном файле `/opt/rubackup/etc/rb_module_postgresql.conf` (см. [reference-configuration-file]). Укажите:
   - режим SSL в параметре `sslmode`;
   - путь до сертификата Центра сертификации в параметре `sslrootcert`;
   - путь до сертификата для модуля *PostgreSQL* в параметре `sslcert`;
   - путь до сгенерированного закрытого ключа для модуля *PostgreSQL* в параметре `sslkey`.

10.4. Режимы доставки журналов транзакций ARCHIVE и STREAM

Режимы доставки ARCHIVE и STREAM в PostgreSQL и производных от нее СУБД отражают разное поведение СУБД при работе с журналами транзакций.

В режиме ARCHIVE транзакции БД пишутся в журналы, а журналы транзакций

архивируются в соответствии с настройками СУБД.

В режиме STREAM выполняется репликация БД путем отправки журналов транзакций на узел, который находится в ожидающем режиме на случай отказа основной БД. Модуль PostgreSQL Universal в этом режиме получает изменившиеся за заданный период журналы транзакций с узла СУБД.

10.4.1. Настройка архивирования журналов транзакций

СУБД на основе PostgreSQL хранят журналы транзакций. Эти журналы позволяют восстановить БД к консистентному состоянию путем последовательного применения сохраненных транзакций. Архивирование и резервное копирование этих журналов позволит восстановить БД как из любого неконсистентного состояния, так и восстановить БД на любой заданный момент времени (point-in-time recovery, PITR).

Чтобы включить режим архивирования журналов транзакций (ARCHIVE):

1. Установите *на ведущем сервере* параметру `wal_level` значение `replica`.
2. Установите параметру `archive_mode` значение `on` или `always` (*на ведомом сервере* — всегда `always`).
3. Задайте команды сохранения (`archive_command`) и восстановления (`restore_command`) журналов транзакций.

Пример 6. Настройка архивирования журналов транзакций в `postgresql.conf` через утилиту `cp`

```
wal_level = replica
archive_mode = on
archive_command = 'cp %p /opt/rubackup/mnt/postgresql_archives/%f'
restore_command = 'cp /opt/rubackup/mnt/postgresql_archives/%f %p'
```

Пример 7. Настройка архивирования журналов транзакций в `postgresql.conf` через утилиту `rb_module_postgresql`

```
wal_level = replica
archive_mode = on
archive_command = '/opt/rubackup/modules/rb_module_postgresql pgsql-
archive-push %p'
restore_command = '/opt/rubackup/modules/rb_module_postgresql pgsql-
archive-get %f %p'
```

Пример 8. Настройка архивирования журналов транзакций в `postgresql.conf` через утилиту `rb_module_postgresql` при включенном режиме работы с несколькими инстансами БД (см.Конфигурационный файл модуля)

```
wal_level = replica
archive_mode = on
archive_command = '/opt/rubackup/modules/rb_module_postgresql pgsql-
archive-push %p "частный_конфигурационный_файл"'
restore_command = '/opt/rubackup/modules/rb_module_postgresql pgsql-
archive-get %f %p "частный_конфигурационный_файл"'
```

Параметр восстановления `restore_command` используется только для версии PostgreSQL 12 и выше.

Допускается сжатие архивных файлов транзакций во время архивирования или восстановления из архива.

Пример 9. Команды сжатия и распаковки архивных журналов транзакций

```
archive_command = 'gzip < %p >
/opt/rubackup/mnt/postgresql_archives/%f.gz'
restore_command = 'gunzip < /opt/rubackup/mnt/postgresql_archives/%f.gz
> %p'
```

4. Выдайте пользователю `postgres` права на чтение и запись в каталог, в котором хранятся архивные журналы транзакций (WAL-файлы).

Пример 10. Выдача прав доступа пользователю `postgres` на каталог `/opt/rubackup/mnt/postgresql_archives/`

```
sudo chown postgres:postgres /opt/rubackup/mnt/postgresql_archives/
```

Каталог также должен быть доступен для записи и чтения пользователю, под контролем которого работает клиент RuBackup.

На папку, хранящую архив журналов транзакций, необходимо указать в файле настроек модуля (параметр `archive_catalog`).

10.4.2. Настройка потокового резервного копирования журналов транзакций

Для настройки потокового резервного копирования (STREAM) внесите изменения в файл настроек СУБД (`postgresql.conf`).

1. Установите для параметра `max_wal_senders` достаточно большое значение, предусматривающее минимум одно подключение для процесса резервного копирования.
2. Задайте параметру `wal_level` значение выше `minimal`.
3. Перезапустите СУБД.

Теперь модуль PostgreSQL Universal может использовать потоковую доставку WAL (STREAM) для получения журналов транзакций.

10.5. Режим PAGE

При инкрементальном копировании в режиме PAGE изменения отслеживаются по транзакционным журналам: архив журналов считывается из папки архива журналов (`/var/lib/postgresql/15/main/pg_wal`) и создаётся «карта» измененных страниц БД.

Во время резервного копирования сохраняются только страницы, отмеченные на «карте» как изменившиеся с момента предыдущей резервной копии.

10.5.1. Требования

- Валидная предыдущая резервная копия.
- Включено непрерывное архивирование транзакционных журналов.
- В архиве должны быть доступны все транзакционные журналы с момента предыдущей резервной копии.

Установите параметр `pg_waldump` ([reference-configuration-file]).

10.6. Режим PTRACK

PTRACK — сокращение от Physical Truncate Tracking. PTRACK используется PostgreSQL для эффективного управления и отслеживания изменений в файлах базы данных.

PTRACK хранит сведения о том, какие страницы файлов были изменены и требуют записи на диск. СУБД отслеживает изменения в памяти и группирует их для последующей записи. При каждом изменении страницы добавляется отметка в PTRACK.

PTRACK используется при включенной опции `full_page_writes` (`postgresql.conf`).

Для корректной работы подмодуля необходимо расширение `ptrack` версии 2.6.0 или выше.

10.6.1. Настройка копирования в режиме PTRACK

Перед выполнением инкрементальной резервной копии в режиме PTRACK настройте PostgreSQL на работу с PTRACK.

1. Установите в конфигурационном файле `postgresql.conf` параметру `shared_preload_libraries` значение `ptrack`.
2. Добавьте в конец конфигурационного файла `postgresql.conf` параметр `ptrack.map_size` и установите его значение равным `N` ÷ 1024, где `N` — объём кластера в мегабайтах. Увеличивать значение `ptrack.map_size` сверх этого не рекомендуется. Максимально допустимое значение `ptrack.map_size` — `1024`.

   Если до этих изменений была сделана полная резервная копия, то после вступления изменений в силу необходимо сделать новую полную резервную копию, иначе инкрементальное резервное копирование в режиме PTRACK завершится с ошибкой.

3. Перезапустите сервис.

```
sudo systemctl restart <database_service>
```

4. Войдите в БД от имени администратора СУБД.

```
sudo -u postgres psql -d <backupdb>
```

5. Загрузите расширение `ptrack`.

```
CREATE EXTENSION ptrack;
```

Теперь модуль может выполнять инкрементальное резервное копирование в режиме PTRACK.

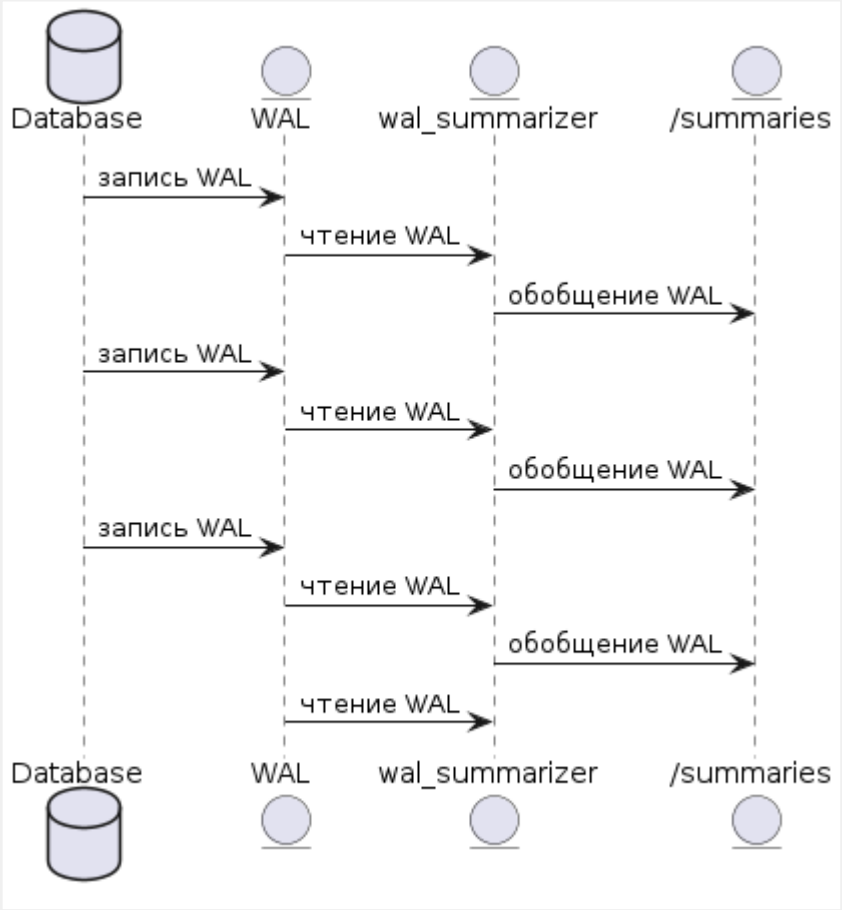
10.7. Режим `wal_summarizer`

В PostgreSQL 17 доступен новый фоновый процесс, который создает «обобщения» для журналов транзакций (WAL) в папке `pg_wal/summaries`. Эти «обобщенные» файлы содержат только информацию, необходимую для инкрементальных резерв-

ных копий.

Для работы с этими «обобщенными» файлами поставляется утилита `pg_walsummary`.

Режим `wal_summarizer` доступен только начиная с PostgreSQL 17 и только для подмодуля `postgresql`.



10.7.1. Настройка `wal_summarizer`

`postgresql.conf`

```
wal_level = 'replica' # или 'logical'
summarize_wal = on
```

[1] Расположение файла может отличаться в зависимости от дистрибутива Linux и версии *PostgreSQL*.

Глава 11. Установка модуля

Модуль резервного копирования PostgreSQL Universal устанавливается на сервер с БД, на котором уже установлен и настроен клиент резервного копирования, подключенный к основному серверу СРК RuBackup.

1. На подготовленном узле клиента СРК подключите публичный репозиторий и установите модуль PostgreSQL Universal `rubackup-postgresql` с помощью пакетного менеджера вашей ОС.

   Установка в ОС Astra Linux

   ```
   apt install rubackup-postgresql
   ```

2. После запуска команды установки модуля выполняются:
   - распаковка пакета модуля PostgreSQL Universal;
   - настройка пакета `rubackup-postgresql`.
3. В результате установки пакета модуля PostgreSQL Universal созданы:

   `/opt/rubackup/etc/rb_module_postgresql.conf`
   : Конфигурационный файл модуля PostgreSQL Universal

     - При обновлении модуля на новую версию обновляется и его конфигурационный файл. При обновлении конфигурационного файла все комментарии из него удаляются.
     - При обновлении модуля в конфигурационный файл могут добавиться новые обязательные параметры. Модуль сообщит об отсутствии обязательных параметров в файле настроек.

   `/opt/rubackup/modules/rb_module_postgresql`
   : Утилита резервного копирования и восстановления данных модуля PostgreSQL Universal

4. Для применения настроек перезапустите сервис клиента СРК RuBackup на узле, на котором установлен клиент СРК и модуль PostgreSQL Universal, выполнив команду:

   ```
   sudo systemctl restart rubackup_client
   ```

Критерием успешности установки и настройки модуля PostgreSQL Universal будет являться запись о его успешной проверке клиентом резервного копирования («*... module PostgreSQL universal was checked successfully*») в журнале событий `/opt/rubackup/log/RuBackup.log`.

Пример 11. Сообщение об успешной проверке модуля PostgreSQL Universal в `RuBackup.log`

```
Execute OS command: /opt/rubackup/modules/rb_module_postgresql -t 2>&1
Info: Patroni node type for backup not set. Processing as a standalone
node...
Warning: Please define pg_probackup
Info: Unable to work with engine pg_probackup
Info: Module version: 2.6.0.03d7888
Info: Initiate connection with database
Info: Connected to demo
Info: PostgreSQL version: 13.21
Info: PostgreSQL data directory: /var/lib/pgsql/13/data
Info: Enable to work with engine postgresql
Info: Module version: 2.6.0.03d7888
Info: Initiate connection with database
Info: Connected to demo
Info: PostgreSQL version: 13.21
Info: PostgreSQL data directory: /var/lib/pgsql/13/data
Info: Enable to work with engine superb
... module 'PostgreSQL universal' was checked successfully
```

В случае, если в журнале событий `/opt/rubackup/log/RuBackup.log` Администратор СРК видит ошибку о неправильной конфигурации модуля PostgreSQL Universal, то необходимо проверить настройки конфигурационного файла `/opt/rubackup/etc/rb_module_postgresql.conf` в ручном режиме, выполнив в терминале клиента СРК команду:

```
/opt/rubackup/modules/rb_module_postgresql -t
```

Пример 12. Сообщение об ошибке при установке модуля

```
Error: Please define pg_ctl or postgresql_service_name in
rb_module_postgresq;.conf
Error: bool rb_module_postgresql::RbPostgresqlUniversal::Init(init,char**)
: Can`t load module config : base
Error: Wrong combination of options
```

Пример 13. Сообщение о предупреждении при установке модуля

```
Warning: Configuration file error: "auto_remove_wal" parameter has
```

```
incompatible value "". Using default value: yes
Warning: Configuration file error: "wal_wait_timeout" parameter has
incompatible value "" bad lexical cast: source type value could not be
interpreted as target. Using default value: 10
Warning: Configuration file error: "wal_check_period" parameter has
incompatible value "" bad lexical cast: source type value could not be
interpreted as target. Using default value: 1
```

== Настройка модуля

После настройки СУБД для резервного копирования и установки модуля PostgreSQL Universal задайте в конфигурационном файле минимальные настройки модуля, при которых резервное копирование и восстановление должны выполняться без ошибок.

1. Укажите метод аутентификации пользователя (`authentication_method`) для взаимодействия с СУБД.
2. Если используется базовый метод аутентификации (`authentication_method` равен `basic`), укажите имя и пароль пользователя (`username`, `password`), от имени которого будет выполняться резервное копирование.
3. Укажите адрес и порт узла (`host`, `port`), на котором располагается резервируемая СУБД.
4. Укажите имя пользователя-администратора СУБД (`postgresql_admin`).
5. Укажите имя сервиса СУБД (`postgresql_service_name`) или полный путь к `pg_ctl` (`pg_ctl`).
6. Укажите папку, в которой хранятся архивные журналы транзакций (`archive_catalog`).

С этими настройками модуля для настроенной СУБД будет выполняться резервное копирование и восстановление.

После любого изменения настроек СУБД или модуля PostgreSQL Universal перезапустите модуль: `systemctl restart rubackup_client.service`.

== Проверка настройки

Для проверки настроек перезапустите клиента RuBackup. В результате клиент должен сообщить, что модуль резервного копирования готов к работе.

```
rubackup_client stop
rubackup_client start
```

```
Try to check module: PostgreSQL...
Execute OS command: /opt/rubackup/modules/rb_module_postgresql -t 2>&1
... module PostgreSQL was checked successfully
```

```
/opt/rubackup/modules/rb_module_postgresql -t
```

При отсутствии обязательных параметров в файле настроек СУБД запуск модуля завершится с ошибкой.

Сообщение об ошибке

```
Wed Feb 15  12:10:34  2023: Try to check module: 'PostgreSQL universal'
...
Wed Feb 15  12:10:34  2023: Execute OS command:
/opt/rubackup/modules/rb_module_postgresql -t 2>&1
Wed Feb 15  12:10:34  2023: Module version: 2.0
Wed Feb 15  12:10:34  2023: PostgreSQL version: 12.13 (Ubuntu 12.13-
0ubuntu0.20.04.1)
Wed Feb 15  12:10:34  2023: [2023-02-15 12:10:34] Error: You MUST define
restore_command in the /etc/postgresql/12/main/postgresql.conf
Wed Feb 15  12:10:34  2023: ... unable to use module 'PostgreSQL
universal' at this client
```

```
[root@centos_vm modules]# ./rb_module_postgresql -t
[2025-07-30 11:45:16] rb_module_postgresql[2069] Info: Patroni node type
for backup not set. Processing as a standalone node...
[2025-07-30 11:45:16] rb_module_postgresql[2069] Warning: Please define
pg_probackup
[2025-07-30 11:45:16] rb_module_postgresql[2069] Info: Unable to work with
engine pg_probackup
[2025-07-30 11:45:16] rb_module_postgresql[2069] Info: Module version:
2.6.0.03d7888
[2025-07-30 11:45:16] rb_module_postgresql[2069] Info: Initiate connection
with database
[2025-07-30 11:45:16] rb_module_postgresql[2069] Info: Connected to sammy
[2025-07-30 11:45:16] rb_module_postgresql[2069] Info: PostgreSQL version:
13.14
[2025-07-30 11:45:16] rb_module_postgresql[2069] Info: PostgreSQL data
directory: /var/lib/pgsql/13/data
```

```
[2025-07-30 11:45:16] rb_module_postgresql[2069] Info: Enable to work with
engine postgresql
[2025-07-30 11:45:16] rb_module_postgresql[2069] Info: Module version:
2.6.0.03d7888
[2025-07-30 11:45:16] rb_module_postgresql[2069] Info: Initiate connection
with database
[2025-07-30 11:45:16] rb_module_postgresql[2069] Info: Connected to sammy
[2025-07-30 11:45:16] rb_module_postgresql[2069] Info: PostgreSQL version:
13.14
[2025-07-30 11:45:16] rb_module_postgresql[2069] Info: PostgreSQL data
directory: /var/lib/pgsql/13/data
[2025-07-30 11:45:16] rb_module_postgresql[2069] Info: Enable to work with
engine superb
```

== Удаление модуля

1. Остановите сервис `rubackup_client`: `systemctl stop rubackup_client`.
2. Удалите `rubackup-postgresql` через пакетный менеджер ОС.
3. Запустите сервис `rubackup_client`: `systemctl start rubackup_client`.

== Конфигурационный файл

Обязательные к заполнению параметры обозначены символом ✱.

Таблица 4. Параметры конфигурационного файла `/opt/rubackup/etc/rb_module_postgresql.conf`

Параметр	Назначение
`dbname`	Имя базы данных, к которой будет происходить подключение для резервного копирования всей СУБД **По умолчанию** `postgres`
`username`✱	Имя пользователя в СУБД, обладающего правами выполнять резервное копирование **По умолчанию** `rubackup_backuper` Параметр обязателен только при использовании базового метода аутентификации (если `authentication_method` равен `basic`).

Параметр	Назначение
`password`*	Пароль для пользователя, указанного в параметре `username` **По умолчанию** `12345` Параметр обязателен только при использовании базового метода аутентификации (если `authentication_method` равен `basic`).
`host`*	IP-адрес или доменное имя локального хоста, на котором СУБД принимает подключения **По умолчанию** `localhost` Необходим для взаимодействия с СУБД, резервное копирование которой выполняется
`port`*	Порт для соединения с СУБД **По умолчанию** `5432`
`authentication_method`	Метод аутентификации пользователя для взаимодействия с СУБД **`basic`** Аутентификация с использованием учетных данных пользователя СУБД, которые хранятся в конфигурационном файле модуля. **`peer`** Аутентификация через Unix-сокет с помощью метода `peer` (используются полномочия пользователя ОС). **`secret`** Аутентификация с использованием учетных данных пользователя СУБД, которые хранятся во внешнем хранилище секретов в зашифрованном виде. **По умолчанию** `basic`
`postgresql_admin`*	Login администратора PostgreSQL в операционной системе **По умолчанию** `postgres`

Параметр	Назначение
`pg_ctl`*	Путь до утилиты `pg_ctl` **По умолчанию** `/usr/lib/postgresql/12/bin/pg_ctl` Параметр используется при восстановлении с развертыванием для остановки и запуска сервиса СУБД Используется, если не задано значение параметра `postgresql_service_name`
`postgresql_service_name`	Имя `systemd` службы СУБД **По умолчанию** `postgresql` Параметр используется при восстановлении с развертыванием для остановки и запуска сервиса СУБД через `systemctl` Используется, если не задано значение параметра `pg_ctl`
`auto_remove_wal`*	Удалять ли архивные WAL-файлы из каталога, указанного в параметре `archive_catalog` **`yes`** Архивные WAL-файлы удаляются из каталога `archive_catalog` после выполнения резервного копирования (если они включены в резервную копию). **`no`** Архивные WAL-файлы не удаляются. **По умолчанию** `yes`
`wal_wait_timeout`*	Период (в секундах) ожидания окончания архивации последнего WAL-файла, сгенерированного во время создания резервной копии **По умолчанию** `10`
`wal_check_period`*	Период (в секундах) проверки окончания архивации последнего WAL-файла, сгенерированного во время создания резервной копии **По умолчанию** `1`

Параметр	Назначение

`keep_wal_chain`	Сохранять цепочку WAL-файлов даже при полном резервном копировании
	
	Возможные значения
	`yes`, `no`
	
	По умолчанию
	`no`

`auto_switch_full`	Автоматически переключать на полное резервное копирование, если произошла ошибка выполнения инкрементального резервного копирования
	
	Возможные значения
	`yes`, `no`
	
	По умолчанию
	`yes`

`operation_timeout`	Таймаут (в секундах) запуска/остановки сервиса СУБД **По умолчанию** `0`
`version_override`	Принудительное указание версии ресурса для случаев, когда СУБД, основанная на PostgreSQL, возвращает некорректную или нестандартную версию. Указывается мажорная и минорная версия через точку (`.`) **По умолчанию** `99.99`

Параметр	Назначение
`instance_name`	Изменение переменной части имени ресурса **По умолчанию** `custom_name` По умолчанию имя ресурса имеет формат `PostgreSQL (<port>)`, где `PostgreSQL` - постоянная часть имени, `(<port>)` - переменная часть имени. Параметр `instance_name` позволяет задать переменную часть имени ресурса в свободном формате Значение может содержать: • не более 128 символов; • заглавные буквы; • строчные буквы; • цифры; • спецсимволы: `()[]._`. Используется, если параметру `legacy_instance_name` установлено значение `no`

`legacy_instance_name`	Использовать формирование имени ресурса, совместимое с прошлыми версиями
	
	Возможные значения
	`yes`, `no`
	
	По умолчанию
	`no`
	
	По умолчанию имя ресурса имеет новый формат `PostgreSQL (<port>)`, где `PostgreSQL` - постоянная часть имени, `(<port>)` - переменная часть имени.
	
	Значение `yes` параметра `legacy_instance_name` позволяет использовать старый способ именования ресурса в формате `PostgreSQL <Версия Major>.<Версия Minor>`. Значение параметра `instance_name` игнорируется

`multi_instance`	Режим работы с несколькими инстансами БД **`yes`** Режим работы с несколькими инстансами БД включен. Для работы в этом режиме создайте вручную частный конфигурационный файл по аналогии с `rb_module_postgresql.conf`.

Частный конфигурационный файл:

- должен быть создан отдельно для каждого инстанса;
- должен находиться в одной директории с основным конфигурационном файлом `rb_module_postgresql.conf`;
- должен иметь идентичные права на чтение, что и основной конфигурационный файл;
- должен следовать схеме наименования `rb_module_postgresql<SUFFIX>.conf`, где `<SUFFIX>` — это произвольное значение;
- переопределяет значения параметров из основного конфигурационного файла.

no
Режим работы с несколькими инстансами БД выключен и файлы частных конфигурационных файлов игнорируются.

По умолчанию
`no`

| archive_catalog✱ | Каталог для хранения архивных WAL-файлов

По умолчанию
`/opt/rubackup/mnt/postgresql_archives`

| move_on_archive_get | Перенос WAL-файлов вместо копирования

yes
WAL-файлы будут перемещены.

no
Будут созданы копии WAL-файлов.

По умолчанию
`yes`

| num_threads_for_wal_archiving | Количество потоков для архивации WAL-файлов

По умолчанию
`1`

| batch_size_for_wal_archiving | Количество обрабатываемых WAL-файлов, которые одновременно отправляются в архивный каталог

По умолчанию

`1`

| wal_archive_files_size | Максимальный размер архивных WAL-файлов, хранимых локально (в МБ)

По умолчанию

`0`

При превышении размера архива запускается фоновая очистка. Во время резервного копирования очистка блокируется.

При значении `0` размер не ограничен

Может использоваться только если команды архивирования и восстановления WAL-файлов выполняются через утилиту `rb_module_postgresql` (см. Раздел 10.4.1).

| cleanup_wait_timeout | Максимальное время ожидания (в секундах) освобождения архивного каталога, которое ждет процесс резервного копирования

По умолчанию

`1000`

| make_archiving_check | Проверка работы архивации WAL-файлов до запуска длительного резервного копирования

yes

Выполняется проверка команд архивации перед резервным копированием. Если проверка завершилась успешно, то резервное копирование происходит в штатном режиме. Если проверка завершилась неудачно, то резервное копирование не запускается, в журнале появляется соответствующая запись.

no

Проверка команд архивации перед резервным копированием не выполняется.

По умолчанию

`yes`

| restore_target_action | Восстановление целевого действия

pause

Восстановление будет приостановлено.

`promote`
: Восстановление будет продолжено.

`shutdown`
: Восстановление не производится.

По умолчанию
: `promote`

| wal_sum_max_retries | Попытки проверки, что wal summarizer успел внести все изменения в summary файлы

По умолчанию
: `20`

| replication_catalog | Абсолютный путь до каталога с репликационными данными

По умолчанию
: `/opt/rubackup/mnt/postgresql_replica`


Используется только для доставки WAL-файлов в режиме `stream`

| pg_receivewal | Путь до утилиты `pg_receivewal`

По умолчанию
: `/usr/lib/postgresql/10/bin/pg_receivewal`

Утилита `pg_receivewal` обрабатывает ошибки и сохраняет файлы


Используется только для доставки WAL-файлов в режиме `stream`

| drop_slot | Выполняет удаление физического слота

`yes`
: Физический слот будет удален.

`no`
: Физический слот удален не будет.

По умолчанию
: `no`


Используется только для доставки WAL-файлов в режиме `stream`

| slot_name | Имя, с которым необходимо создать физический слот

По умолчанию

`my_slot_name`

Если значение не задано, то используется физический слот, заданный в системе.

Используется только для доставки WAL-файлов в режиме `stream`

| patroni_host | IP-адрес, на котором Patroni принимает входящие запросы REST API

По умолчанию

`localhost`

Параметр необходим только для взаимодействия модуля с REST API локального процесса Patroni.

Если значение параметра не указано, то оно определяется через утилиту `lsof`

| patroni_port | Порт, на котором локальный процесс Patroni слушает запросы REST API

По умолчанию

`8008`

Параметр необходим только для взаимодействия модуля с REST API локального процесса Patroni.

Если значение параметра не указано, то оно определяется через утилиту `lsof`

| patroni_node_type_for_backup | Роль узла в кластере Patroni, при которой ресурс будет доступен

`leader`

Ресурс будет доступен только при условии, что узел на котором установлен модуль с таким значением имеет роль *leader* в кластере Patroni.

`sync`

Ресурс будет доступен только при условии, что узел, на котором установлен модуль с таким значением, имеет роль *sync stanby* в кластере Patroni.

`async`

Ресурс будет доступен только при условии, что узел, на котором установлен модуль с таким значением, имеет роль *replica* в кластере Patroni.

По умолчанию

`leader`

| patroni_https | Позволяет использовать режим `HTTPS` при работе с Patroni

`yes`
Режим HTTPS может использоваться.

`no`
Режим HTTPS не может использоваться.

По умолчанию
`no`

| patroni_ca | Путь к корневому сертификату, если требуется проверка сертификата сервера Patroni

По умолчанию
`/etc/ssl/ca.crt`

| pg_waldump | Путь до утилиты `pg_waldump`

По умолчанию
`/usr/lib/postgresql/12/bin/pg_waldump`

Используется для работы подтипа инкрементального резервного копирования `page`. Параметр необходим, если наблюдаются проблемы с запуском или остановкой сервиса `postgresql` через `pg_ctl`.

Местонахождение `pg_waldump` зависит от используемой СУБД

| num_threads_for_wal_processing | Количество процессов, выделенных для обработки архивных WAL-файлов

По умолчанию
`8`

| restore_target | В каком состоянии будет восстановлена БД

`immediate`
БД будет восстановлена в самом раннем состоянии.

`latest`
БД будет восстановлена в самом позднем состоянии, исходя из содержимого архива WAL.

По умолчанию
`immediate`

Используется только для подмодуля `pg_probackup`

| pg_probackup | Абсолютный путь до утилиты `pg_probackup`

По умолчанию

`/opt/pgpro/std-13/bin/pg_probackup`

Используется только для подмодуля `pg_probackup`

| probackup_catalog_copies | Абсолютный путь до каталога для хранения резервных копий

По умолчанию

`/opt/rubackup/mnt/pg_probackup`

Используется только для подмодуля `pg_probackup`

| probackup_instance_name | Имя подкаталога для хранения резервных копий

По умолчанию

`data`

Используется только для подмодуля `pg_probackup`

| check_probackup_version | Нужно ли проверять версию `probackup`

Возможные значения

`yes`, `no`

По умолчанию

`yes`

Используется только для подмодуля `pg_probackup`

| s3_interface | Интерфейс облачного хранилища

`minio`

Будет использоваться объектное хранилище MinIO, совместимое с облачным хранилищем S3.

`vk`

Будет использоваться хранилище VK.

По умолчанию

`minio`

Используется только для подмодуля `pg_probackup`

| pg_binary | Путь к исполняемому файлу `postgres`

По умолчанию

`/usr/bin/custom/postgres`

Необходим при использовании утилиты `pg_ctl` для запуска СУБД при восстановлении с развертыванием

| pg_log | Путь до файла, в который будет направляться вывод сообщений сервера

По умолчанию

`/tmp/postgres.log`

Необходим при использовании утилиты `pg_ctl` для запуска СУБД при восстановлении с развертыванием

.Пример листинга конфигурационного файла `/opt/rubackup/etc/rb_module_postgresql.conf`
[source,.conf] ---- # Symbol "" at the beginning of the line treats as a comment # "" in the middle of the line treats as a parameter value # So please do not use comments in one line with parameter # # # ----- General ----- # #dbname postgres #username rubackup_backuper #password 12345 #host localhost #port 5432 # Доступные значения authentication_method: basic, peer, secret authentication_method basic #postgresql_admin postgres # Specify this path according to the installed version #pg_ctl /usr/lib/postgresql/12/bin/pg_ctl #postgresql_service_name postgresql #auto_remove_wal yes # Timeout period for the last WAL file generated during backup(in seconds) #wal_wait_timeout 10 # Availability check period for last WAL file generated during backup(in seconds) #wal_check_period 1 #keep wal files chain even for full backup #keep_wal_chain no # Переключаться или нет на полный бэкап если невозможно выполнить инкрементальный/дифф #auto_switch_full yes # Таймаут старт/стоп БД. В секундах. 0 - использовать системный дефолт. Обычно 60 сек. #operation_timeout 0 # Переопределение версии БД #version_override 99.99 # Имя инстанса #instance_name custom_name # Использовать формирование имени ресурса совместимое с прошлыми версиями #legacy_instance_name no # # ----- Wal Archiving ----- # latest] #restore_target immediate # Абсолютный путь до утилиты pg_probackup #pg_probackup /opt/pgpro/std-13/bin/pg_probackup # Абсолютный путь до каталога, в котором хранятся резервные копии #probackup_catalog_copies /opt/rubackup/mnt/pg_probackup # Имя инстанса. Имя подкаталогов, в которых будут храниться копии #probackup_instance_name data # Нужно ли проверять версию probackup #check_probackup_version yes # # ----- S3 ----- # # Возможные значения для s3_interface: [minio vk] s3_interface minio # # ----- Custom PostgreSQL build ----- # # Specify if custom built PostgreSQL binary is required # pg_binary /usr/bin/custom/postgres # Specify if server output should be redirected to the file # pg_log /tmp/postgres.log # # Variables value username password port postgresql_admin pg_ctl auto_remove_wal wal_wait_timeout wal_check_period legacy_instance_name archive_catalog ----

:!navtitle:

:docname: reference-tweaks :page-module: ROOT :page-relative-src-path: reference-tweaks.adoc :page-origin-url: http://10.177.32.32/rubackup/docs/module-postgresql :page-origin-start-path: :page-origin-refname: master :page-origin-reftype: branch :page-origin-refhash: 97d9f10191b09d5d1874d73d79c8f8458bccb359 [#reference-tweaks] == Тонкие настройки модуля при резервном копировании

:button-settings: [.nowrap# (**Настройки**)] :button-ok: **ОК** :button-save: **Сохранить** :button-table-columns-setup: [.nowrap (**Настройка колонок**)] :sign-column-sorted: image:../../../General/2.9.0.0.0/ui/_images/common_gui/rc/images/Filter/Filter-small.svg[pdfwidth="14",width="20px"] :button-clean-filter: image:../../../General/2.9.0.0.0/ui/_images/common_gui/rc/images/Filter/No-filter.svg[pdfwidth="14",width="20px"] :button-clean-filters:

В LVM volume groups, в которых расположены тома LVM, должно быть не менее 10% свободного места для возможности создания моментальных снимков LVM.

В ходе выполнения задания резервного копирования в журнальном файле задания резервного копирования (файл с номером задачи в `/opt/rubackup/log`) можно проконтролировать реальную утилизацию созданного снимка:

```
Snapshot '/dev/mapper/vg0-var--lib.snap' was used for: 0.92 %
The snapshot was removed: /dev/mapper/vg0-var--lib.snap
```

В том случае, если это значение при реальном резервном копировании близко к 100%, то необходимо увеличить размер свободного места в LVM группе и увеличить `lvm_snapshot_size`.

По умолчанию

`10`

incremental_subtype

Выбор подтипа инкрементального резервного копирования.

`archive_wal`

Режим инкрементального резервного копирования, при котором модуль PostgreSQL Universal выполняет резервное копирование архивных WAL-файлов.

`page` (страничное копирование)

В этом режиме модуль сканирует все файлы WAL в архиве с момента создания предыдущей полной или инкрементальной копии. Новая резервная копия будет содержать только страницы, на которые указывает журнал транзакций. Необходимо, чтобы в архиве WAL сохранялись все журналы транзакций, записанные после предыдущей резервной копии. Если размер этих файлов сравним с общим размером файлов базы данных, ускорение будет невелико, но размер резервной копии будет меньше.

Для работы в этом режиме необходимо настроить непрерывное архивирование журналов транзакций.

Для работы подтипа `page` необходимо настроить параметр конфигурационного файла `pg_waldump` ([reference-configuration-file]).

`delta` **(разностное копирование)**

В этом режиме модуль считывает все файлы данных в каталоге данных и копирует только те страницы, которые изменились со времени предыдущего копирования. Для использования данного режима непрерывное архивирование не требуется. В этом режиме нагрузка на ввод-вывод может быть сопоставима с полным резервным копированием.

`ptrack` **(копирование изменений)**

В этом режиме модуль отслеживает изменения на лету. Этот режим не требует непрерывного архивирования журналов транзакций. Этот режим добавляет нагрузку на сервер, но значительно ускоряет инкрементальное резервное копирование.

`wal_summarizer`

В этом режиме используется функция обобщения журналов транзакций, появившаяся в PostgreSQL 17.

connection_monitoring

Мониторинг соединения с базой данных

По умолчанию

Вкл

compression_type

Сжатие данных.

Возможные варианты

`none`, `fast`, `optimal`, `best`

=== Подмодуль `pg_probackup`

stream Выбор режима доставки WAL.

`Вкл` **(STREAM)**

Копии типа STREAM включают все сегменты WAL, необходимые для восстановления согласованного состояния кластера на момент создания копии.

	`Выкл` **(ARCHIVE)** В режиме ARCHIVE целостность копий обеспечивается посредством непрерывного архивирования (подробнее см. на https://postgrespro.ru/docs/enterprise/). **По умолчанию** `Вкл`
pg_pro_threads	Выбор количества параллельных потоков при резервном копировании.
pg_pro_backup_mode	Выбор подтипа инкрементального резервного копирования. `delta` **(разностное копирование)**

В этом режиме модуль считывает все файлы данных в каталоге данных и копирует только те страницы, которые изменились со времени предыдущего копирования. Для использования данного режима непрерывное архивирование не требуется. В этом режиме нагрузка на ввод-вывод может быть сопоставима с полным резервным копированием.

`page` **(страничное копирование)**

В этом режиме модуль сканирует все файлы WAL в архиве с момента создания предыдущей полной или инкрементальной копии. Новая резервная копия будет содержать только страницы, на которые указывает журнал транзакций. Необходимо, чтобы в архиве WAL сохранялись все журналы транзакций, записанные после предыдущей резервной копии. Если размер этих файлов сравним с общим размером файлов базы данных, ускорение будет невелико, но размер резервной копии будет меньше.

Для работы в этом режиме необходимо настроить непрерывное архивирование журналов транзакций.

`ptrack` **(копирование изменений)**

В этом режиме модуль отслеживает изменения на лету. Этот режим не требует непрерывного архивирования журналов транзакций. Этот режим добавляет нагрузку на сервер, но значительно ускоряет инкрементальное резервное копирование.

По умолчанию

`ptrack`

=== Подмодуль `superb`

entire_snapshot_backup — При активировании переключателя будет выполняться резервное копирование всего снимка состояния диска (логического тома LVM), на котором располагается СУБД. Если переключатель не активирован, то будет только выполняться резервное копирование данных СУБД.

snapshot_size — Выбор размера снимка в процентах от размера Logical Volume тома, на котором расположены файлы базы данных, для которой выполняется резервное копирование, в случае LVM. Либо размер снимка в процентах от размера устройства, на котором расположены файлы базы данных, для которой выполняется резервное копирование, в случае использования `dattobd`.

В LVM volume groups, в которых расположены тома LVM, должно быть не менее 10% свободного места для возможности создания моментальных снимков LVM.

В ходе выполнения задания резервного копирования в журнальном файле задания резервного копирования (файл с номером задачи в `/opt/rubackup/log`) можно проконтролировать реальную утилизацию созданного снимка:

```
Snapshot '/dev/mapper/vg0-var--lib.snap' was used for: 0.92 %
The snapshot was removed: /dev/mapper/vg0-var--lib.snap
```

В том случае, если это значение при реальном резервном копировании близко к 100%, то необходимо увеличить размер свободного места в LVM группе и увеличить `lvm_snapshot_size`.

По умолчанию

`10`

snapshot_type

Выбор типа снимка.

`lvm`

Использование снимков LVM.

СУБД должна располагаться в файловой системе, которая использует том LVM.

Модуль поддерживает СУБД с дополнительными табличными пространствами (tablespaces). Табличные пространства должны располагаться в файловых системах, которые используют тома LVM.

`dattobd`

Использование модуля ядра dattobd, позволяющего делать снимки блочного устройства.

`tatlin`

Использование Tatlin Unified Storage, позволяющего делать мгновенные снимки состояния данных СУБД, которая располагается в системе хранения данных Tatlin Unified Storage.

Перед выполнением резервного копирования создаётся мгновенный снимок состояния (в зависимости от способа мгновенного снимка состояния: логического тома LVM, данных на блочном устройстве, системы хранения данных Tatlin Unified Storage), по окончанию резервного копирования мгновенный снимок состояния будет удалён.

При использовании Tatlin Unified Storage необходимо предварительно на хосте, на котором развёрнут модуль, установить утилиты `multipath` и `sg3_utils`.

По умолчанию

`lvm`

== Восстановление резервных копий

=== Восстановление резервных копий в Tucana

В Tucana произведите настройку, следуя указаниям из документа Восстановление резервной копии.

Для восстановления БД из РК:

1. Из списка **Восстановить на клиенте** выберите узел клиента резервного копирования.
2. В **Каталог распаковки** нажмите **[...]** и укажите каталог для распаковки БД из РК и файла с метаданными этой РК.
3. В **Параметры восстановления для модуля** нажмите **[...]** и определите тон-

кие настройки модуля (см. [reference-restore-thin-settings]).

4. Включите **Восстановить на целевом ресурсе** для восстановления БД на целевом ресурсе (на узле с СУБД) после распаковки РК во временный каталог (**Каталог распаковки**). После восстановления БД каталог распаковки будет очищен.

   Если при этом включен параметр `enable_direct_restore` в тонких настройках модуля, то восстановление БД будет происходить без использования временного каталога распаковки. Во временный каталог распаковываются только метаданные РК.

   Если флаг **Восстановить на целевом ресурсе** выключен, то резервная копия БД распаковывается во временный каталог. После распаковки РК для восстановления БД перенесите файлы из временного каталога в целевые каталоги. Например:

   a. если файлы располагаются в `/restore_dir/number.rest/var/lib/postgresql/11/main` (`number` — это номер резервной копии), то переместите их в `/var/lib/postgresql/11/main`;

   b. если wal-файлы располагаются в `/restore_dir/number.rest/opt/rubackup/mnt/postgresql_archives` (`number` — это номер резервной копии), то переместите их в `/opt/rubackup/mnt/postgresql_archives/`.

=== Восстановление со стороны клиента

Для операции восстановления можно использовать утилиту командной строки `rb_archives`.

Использование утилиты командной строки `rb_archives` позволяет посмотреть список резервных копий:

```
root@postgresql:~# rb_archives
Id | Ref ID | Resource          | Resource type         | Backup type | Created
| Crypto  | Signed | Status
---+--------+-------------------+-----------------------+-------------+
-----------------------+---------+--------+-------------
1  |        | PostgreSQL 12.13 | PostgreSQL universal | full        | 2023-
02-16 11:11:03+03 | nocrypt | True   | Not Verified
```

В первой колонке указаны идентификаторы резервных копий. Чтобы восстановить резервную копию без развертывания, нужно использовать команду:

```
sudo rb_archives -X -d /path_to_restore
```

Опция `-X` указывает, что нужно выполнить операцию восстановления без развертывания

Опция `-d` указывает путь, в который нужно восстановить резервную копию. Если не используется опция `-d`, резервная копия будет восстановлена в каталог для временных операций с резервными копиями, либо, если клиент настроен на использование временной NFS-папки от сервера резервного копирования, восстановление произойдет в эту NFS-папку. В случае восстановления резервной копии без развертывания всегда рекомендуется использовать опцию `-d` с указанием каталога на клиенте, в котором есть достаточно места для восстановления резервной копии.

В том случае, если необходимо выполнить восстановление резервной копии с развертыванием, выполните команду:

```
sudo rb_archives -x -d /path_to_restore
```

Опция `-x` указывает, что нужно восстановить резервную копию с развертыванием.

Для восстановления резервной копии необходимо ввести пароль клиента (задается при первом использовании `rb_archives`). Если вы не знаете пароль, обратитесь к системному администратору.

Проконтролировать выполнение задачи восстановления можно при помощи утилиты командной строки `rb_tasks`:

```
root@postgresql:~# rb_tasks
Id | Task type     | Resource         | Backup type | Status | Created
---+---------------+------------------+-------------+--------+
-----------------------
1  | Backup global | PostgreSQL 12.13 | full        | Done   | 2023-02-16
11:10:42+03
2  | Restore       | PostgreSQL 12.13 | full        | Done   | 2023-02-16
11:21:20+03
```

Так же можно получить детальную информацию о ходе восстановления из журнального файла задачи:

```
root@ubuntu-server:~# cat /opt/rubackup/log/task_2.log
Wed Feb 15 12:30:18 2023: Media server ql has 'New' task in the queue. Task
ID: 2. Task type: Backup global
Wed Feb 15 12:30:18 2023: Task ID: 2. New status: Assigned
Wed Feb 15 12:30:18 2023: Task ID: 2. New status: At_Client
Wed Feb 15 12:30:18 2023: Task ID: 2. New status: Execution
Wed Feb 15 12:30:19 2023: Set unlimited bandwidth for task ID: 2
Wed Feb 15 12:30:22 2023: Task ID: 2. New status: Start Transfer
Wed Feb 15 12:30:22 2023: Set unlimited bandwidth for task ID: 2
Wed Feb 15 12:30:23 2023: Transfer of snapshot client2 TaskID 2
NORuleOrStrategy_0 D2023_2_15H09 30 18 BackupType 1 ResourceType 11 has
succeeded. Task ID: 2
Wed Feb 15 12:30:23 2023: Task ID: 2. New record ID was created in
repository: 2
Wed Feb 15 12:30:23 2023: Task ID: 2. New status: Transmission
Wed Feb 15 12:30:24 2023: Task ID: 2. New status: Done
Thu Feb 16 11:21:20 2023: Media server ubuntu-server has 'New' task in the
queue. Task ID: 2. Task type: Restore
Thu Feb 16 11:21:20 2023: Task ID: 2. New status: Assigned
Thu Feb 16 11:21:21 2023: Task ID: 2. New status: At_Client
Thu Feb 16 11:21:21 2023: Task ID: 2. New status: Start-Transfer
Thu Feb 16 11:21:21 2023: Task ID: 2. New status: Transmission
Thu Feb 16 11:21:21 2023: Set unlimited bandwidth for task ID: 2
Thu Feb 16 11:21:24 2023: Blocks are ready, time: 2
Thu Feb 16 11:21:26 2023: Task ID: 2. New status: Done
```

=== Тонкие настройки для восстановления резервной копии

В таблице описаны тонкие настройки модуля для восстановления резервной копии (см. [howto-tucana-restore]).

Таблица 5. Тонкие настройки модуля PostgreSQL Universal для восстановления резервной копии

Параметр	Описание
Использовать настройки по умолчанию	Использование значений по умолчанию При значении: • `true` для параметров используются значения по умолчанию; • `false` значения параметров можно изменить. **По умолчанию** `true`

Параметр	Описание
Ручная настройка времени восстановления	Позволяет настроить произвольный момент времени восстановления состояния БД из инкрементальной РК Используется только для подмодуля `postgresql` При значении: • `true` при восстановлении цепочки резервных копий состояние БД будет восстановлено до указанного момента времени; • `false` при восстановлении цепочки резервных копий состояние БД будет восстановлено до момента создания последней РК в цепочке. **По умолчанию** `false` Используется вместе с параметром **recovety_target_time**
recovety_target_time	Произвольный момент времени восстановления состояния БД из резервной копии. Указанный момент должен быть позднее момента создания полной РК в цепочке. Задается по нажатию в виде даты и времени Только для подмодуля `postgresql` Используется вместе с параметром **Ручная настройка времени восстановления**
secret_method	Метод получения секрета (информации для подключения к СУБД PostgreSQL) (см. 2.9.0.0.0@TucanaGuide:ROOT:page$secret-storages.adoc) **По умолчанию** `No secret method`
enable_direct_restore	Прямое восстановление БД **Возможные значения** `true`, `false` **По умолчанию** `true` При значении `true` восстановление БД из резервной копии происходит напрямую в целевое местоположение без использования временного каталога распаковки. Используется только при восстановлении с развертыванием на целевом ресурсе (см. [howto-tucana-restore]).

== Резервное копирование с использованием подмодуля `pg_probackup`

Для корректной работы подмодуля необходима утилита `pg_probackup` версии 2.6 или выше.

=== Подготовка к использованию `pg_probackup`

Для выполнения резервного копирования с подмодулем `pg_probackup` выполните следующие действия:

1. Запустите RBM командой:

   ```
   rbm
   ```

   После этого в открывшемся окне (Рисунок 1) введите наименование сервера RuBackup, имя пользователя и пароль.

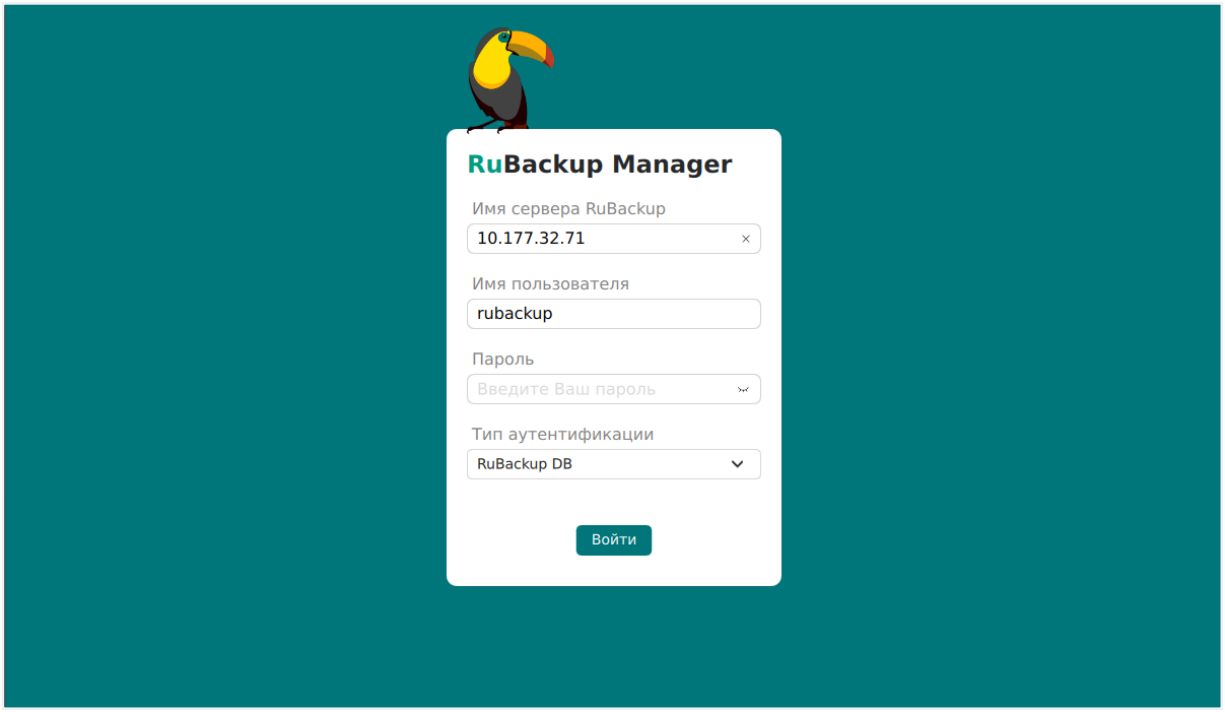


Рисунок 1. Окно входа RBM

2. Добавьте пул типа `Client defined` с помощью RBM либо командной строки (см. подробнее в документе «Руководство системного администратора RuBackup»);
3. Настройте Клиентское хранилище с помощью RBM либо командной строки (см. подробнее в документе «Руководство системного администратора RuBackup»);

1. Настройте на клиентском хосте базу данных PostgreSQL;
2. Установите на клиентском хосте модуль для PostgreSQL Universal (`rb_module_postgresql`);
3. Разверните клиент резервного копирования, сконфигурируйте и подключите его к серверу RuBackup на хосте, где установлен модуль PostgreSQL Universal;

4. Создайте в S3-хранилище MinIO папку, в которую будут помещаться резервные копии;
5. Настройте PTRACK на клиенте для создания и восстановления инкрементальных копий табличных пространств CFS;
6. Настройте утилиту `pg_probackup` на клиенте для работы с S3-хранилищем;

Для работы с S3-хранилищем MinIO в утилите `pg_probackup` нужно использовать ключ `--s3=minio`

1. Для использования режима постраничного копирования (PAGE) настройте непрерывное архивирование WAL с сервера БД в RuBackup.

=== Инициализация каталога резервных копий

Для корректной работы подмодуля `pg_probackup` выполните последовательно следующие команды от имени администратора СУБД.

```
pg_probackup init -B каталог_копий [--skip-if-exists] [параметры_s3] [
--help]

pg_probackup add-instance -B каталог_копий -D каталог_данных --instance
имя_экземпляра [--skip-if-exists] [параметры_s3] [--help]
```

`имя_экземпляра` должно совпадать с именем `каталог_данных`. Например, если добавляется каталог данных `/var/lib/pgpro/std-13/data/`, именем экземпляра будет `data`.

=== Доступ к папке `pg_probackup`

Папка `/opt/rubackup/mnt/pg_probackup` и все вложенные в неё папки должны быть доступны для записи и чтения пользователю, указанному в параметре `postgresql_admin` в конфигурационном файле модуля, а также пользователю, от имени которого работает клиент RuBackup.

Задание прав доступа к папке `pg_probackup`

```
sudo chgrp postgres -R /opt/rubackup/mnt/pg_probackup
sudo chmod g+rwx -R /opt/rubackup/mnt/pg_probackup
```

=== Настройка непрерывного архивирования WAL для `pg_probackup`

Если установлен `pg_probackup`, то команды архивирования журналов транзакций могут использовать эту утилиту с предоставляемой ею функциональностью ([pg_probackup----howto-s3-storage]).

Настройка команд выполняется редактированием файла настроек СУБД (`postgresql.conf` для PostgreSQL).

1. Задайте команду архивирования журналов транзакций.

```
archive_command = 'pg_probackup archive-push
  -B /opt/rubackup/mnt/pg_probackup
  --wal-file-path %p
  --wal-file-name %f
  --instance имя_экземпляра ❶
  [параметры_удалённого_режима] ❷
  "[частный_конфигурационный_файл]"' ❸
```

❶ `имя_экземпляра` должно указывать на уже проинициализированный для данного кластера БД копируемый экземпляр.

❷ `параметры_удалённого_режима` должны задаваться только в случае расположения архива WAL в удалённой системе.

❸ `"частный_конфигурационный_файл"` укажите частный конфигурационный файл при включенном режиме работе с несколькими инстансами БД в конфигурационном файле модуля.

2. Задайте параметром `restore_command` команду восстановления журналов транзакций из архива.

```
restore_command = 'путь_инсталляции/pg_probackup archive-get
  -B /opt/rubackup/mnt/pg_probackup
  --instance имя_экземпляра
  --wal-file-path=%p
  --wal-file-name=%f
  [параметры_удалённого_режима]
  "[частный_конфигурационный_файл]"'
```

=== Настройка хранения резервных копий в S3

`pg_probackup` позволяет хранить резервные копии в облачном хранилище MinIO, VK Cloud и AWS.

Для использования облачного хранилища

- задайте параметр `s3_interface`;
- передайте `pg_probackup` параметры `--s3` и (опционально) `--s3-config-file` (см. документацию).

```
pg_probackup add-instance \ ❶
  -D /var/lib/pgpro/ent-16/data \ ❷
  -B /opt/rubackup/mnt/pg_probackup \ ❸
  --instance=example \ ❹
  --s3 ❺
  [ --s3-config-file=путь_к_файлу_конфигурации ] ❻
```

❶ `add-instance` создает запись о резервном копировании СУБД.

❷ `-D` или `--pgdata` — папка с данными кластера БД. Необходим доступ на запись в эту папку.

❸ `-B` или `--backup-path` — папка, в которой хранятся резервные копии. В этой папке будут создаваться дополнительные подпапки.

❹ `--instance` — имя подпапки кластера в основной папке резервных копий. Для `--instance=example` в папке, указанной для `--backup-path`, будут созданы папки `/backups/example` и `/wal/example`.

❺ Использовать облачное хранение S3.

❻ Если этот параметр не указан, `pg_probackup` ищет файл конфигурации S3 сначала в `/etc/pg_probackup/s3.config`, а затем в `~postgres/.pg_probackup/s3.config`.

В случае работы с подмодулем `pg_probackup` при выборе типа хранилища Cloud вам будет предложено ввести данные для работы с облачным хранилищем: хост облака, порт облака, бакет, безопасность облака, ID ключа доступа, секретный ключ доступа. Данная информация будет передаваться модулю во время работы и перезаписывать конфигурационный файл `/etc/pg_probackup/s3.config`. Указав данные в RBM, вы можете обновить данные для всех клиентов, которые будут использовать данный пул.

=== Пример использования подмодуля `pg_probackup` в Менеджере администратора RuBackup (RBM)

Для регулярного резервного копирования СУБД необходимо создать правило в глобальном расписании.

1. Перейдите в раздел **Глобальное расписание** и нажмите кнопку (**Добавить**). Откроется окно добавления правила глобального расписания.
2. В поле **Клиент** выберите клиента, на котором установлен модуль PostgreSQL Universal.
3. В **Тип ресурса** выберите тип ресурса `PostgreSQL Universal`.

4. Откройте тонкие настройки модуля нажатием **[...]**. Выберите подмодуль (**engine**) `pg_probackup`.
5. Настройте число параллельных потоков (**pg_pro_threads**) резервного копирования или восстановления.
6. Выберите режим резервного копирования (**pg_pro_backup_mode**), если требуется инкрементальная резервная копия.

Перед выполнением резервного копирования в режиме PAGE необходима [pg_probackup----howto-continuous-wal-archiving].

Перед выполнением резервного копирования в режиме PTRACK необходима Раздел 10.6.

1. Выберите режим доставки WAL флагом **stream**: `STREAM` (вкл., по умолчанию) или `ARCHIVE` (выкл.). Нажмите **OK** для сохранения.
2. Выберите тип резервной копии.
3. Нажмите ✓ **Применить**.

=== Настройка копируемого кластера баз данных для использования `pg_probackup`

Для выполнения резервного копирования в защищённом режиме необходимо создать роль с ограниченными правами и базу данных резервного копирования. Имя роли, базы данных и пароль условны и могут быть изменены по усмотрению пользователя.

1. Создайте базу данных резервного копирования. Данная операция производится от имени пользователя postgres:

```
su postgres
createdb backupdb
```

2. Далее описан процесс создания роли для выполнения резервного копирования. В целях обеспечения безопасности копируемых данных, создаваемая роль будет обладать минимальными правами, необходимыми для выполнения резервного копирования экземпляра Postgres Pro. В этом примере такой ролью будет `rubackup_backuper`.
   a. Выполните подключение к базе данных `backupdb` от имени пользователя `postgres`:

```
sudo -u postgres psql -d backupdb
```

b. В `psql` создайте роль `rubackup_backuper` и задайте пароль, а также задайте разрешения (только в базе данных, к которой производится подключение).

```
BEGIN;
CREATE ROLE rubackup_backuper WITH LOGIN;
ALTER USER rubackup_backuper WITH PASSWORD '12345';
GRANT USAGE ON SCHEMA pg_catalog TO rubackup_backuper;
GRANT EXECUTE ON FUNCTION pg_catalog.current_setting(text) TO
rubackup_backuper;
GRANT EXECUTE ON FUNCTION pg_catalog.set_config(text, text, boolean) TO
rubackup_backuper;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_is_in_recovery() TO
rubackup_backuper;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_backup_start(text, boolean) TO
rubackup_backuper;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_backup_stop(boolean) TO
rubackup_backuper;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_create_restore_point(text) TO
rubackup_backuper;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_switch_wal() TO rubackup_backuper;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_last_wal_replay_lsn() TO
rubackup_backuper;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current() TO rubackup_backuper;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current_snapshot() TO
rubackup_backuper;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_snapshot_xmax(txid_snapshot) TO
rubackup_backuper;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_control_checkpoint() TO
rubackup_backuper;
ALTER ROLE rubackup_backuper WITH REPLICATION;
COMMIT;
```

1. Создайте файл `.pgpass` в домашнем каталоге пользователя, указанного в параметре `postgresql_admin` в конфигурационном файле модуля. В файле `.pgpass` укажите данные для подключения к ранее созданной базе данных и для репликации.

   Это распространённый способ хранения информации о соединении в PostgreSQL вместо ввода пароля при каждой совершённой операции с `pg_probackup`. Этот файл должен содержать строки в таком формате:

```
сервер:порт:база_данных:имя_пользователя:пароль
```

```
localhost:5432:backupdb:rubackup_backuper:12345
localhost:5432:replication:rubackup_backuper:12345
```

Файл `.pgpass` обязательно должен находиться в домашнем каталоге пользователя, указанного в параметре `postgresql_admin` в конфигурационном файле модуля.

Маска разрешений файла должна соответствовать маске `0600`.

1. Далее необходимо произвести изменения в файле `pg_hba.conf`. Найти конфигурационный файл, относящийся к настраиваемому кластеру, можно так:

```
sudo -u postgres psql
psql -c 'show hba_file'
```

Вызовите `psql` при помощи команды:

```
sudo -u postgres psql
```

Если для пользователя `postgres` не установлен пароль, установите его, изменив `12345` на подходящий:

```
alter user postgres with password `12345`;
```

В конец файла `pg_hba.conf` добавьте следующие строки:

```
host backupdb rubackup_backuper 127.0.0.1/32 md5
host replication rubackup_backuper 127.0.0.1/32 md5
```

При использовании HAProxy, необходимо в файл `pg_hba.conf` добавить следующие строки:

```
host backupdb rubackup_backuper <IP-адрес хоста HAProxy> md5
host replication rubackup_backuper <IP-адрес хоста HAProxy> md5
```

Вместо `peer` везде установите `md5`. Выполните повторную загрузку конфигурации БД или перезапустите СУБД.

```
psql -c 'select * from pg_hba_file_rules'

psql -c 'select pg_reload_conf()'
```

Теперь модуль может выполнять полное резервное копирование и инкрементальное резервное копирование в режиме DELTA, используя режим доставки WAL по умолчанию (ARCHIVE).

=== Резервное копирование кластера Patroni

Если планируется использование модуля для резервного копирования кластеров СУБД в составе Patroni, то необходимо выполнить следующую команду:

```
pg_probackup set-config \
  -B /opt/rubackup/mnt/pg_probackup/ \
  --instance=patroni \
  --pghost ip_кластера (1)
```

(1) `ip_кластера` — это IP-адрес копируемого экземпляра Patroni.

== HashiCorp Vault

Аутентификационная информация (далее — *секрет*) для подключения к СУБД PostgreSQL может храниться в конфигурационном файле. Это небезопасно, т.к. злоумышленники могу получить доступ к содержимому файла, если он недостаточно защищён.

Безопаснее использовать инструмент управления секретами, одним из которых является HashiCorp Vault. Эта система позволяет шифровать и безопасно хранить секреты.

RuBackup взаимодействует с хранилищем секретов HashiCorp Vault по REST API.

Интеграция RuBackup с хранилищем секретов HashiCorp Vault поддерживается только при создании и восстановлении полных и инкрементальных резервных копий.

=== Подготовка к использованию

Предварительно получите от администратора хранилища секретов:

- подтверждение, что секрет в хранилище секретов HashiCorp Vault создан. Один секрет может содержать несколько наборов аутентификационной информации в формате JSON;
- токен для доступа к хранилищу секретов HashiCorp Vault;
- метод доступа к секрету.

1. Проверьте, что сервер хранилища секретов доступен: IP-адрес внесен в `/etc/hosts` или в локальный DNS организации.
2. Проверьте номер и доступность порта, по которому выполняется защищенное HTTPS-подключение к хранилищу секретов. По умолчанию: `:8200`.
3. Получите цепочку HTTPS-сертификатов сервера (хранилища секретов) в формате PEM.

```
sudo openssl s_client \
  -connect hostname:8200 \ ❶ ❷
  -showcerts \ ❸
  -servername hostname \ ❹
  > ~/hashicorp.pem ❺
```

❶ Флаг `-connect` показывает диагностическую информацию об SSL-подключении к серверу.

❷ `hostname` — имя сервера хранилища секретов.

❸ Флаг `-showcerts`, добавленный к `-connect`, показывает всю цепочку сертификатов сервера в формате PEM.

❹ Включение SNI.

❺ Экспорт полученной цепочки сертификатов в файл.

Сохраните файл `~/hashicorp.pem`, он потребуется при добавлении хранилища секретов.

1. В конфигурационном файле модуля установите параметру `authentication_method` значение `secret` и закомментируйте те параметры, которые планируется передавать из хранилища секретов.

=== Добавление хранилища секретов

1. Перейдите в раздел **Администрирование** → подраздел **Настройки хранилища секретов** → **Список хранилищ секретов**. Нажмите **Добавить**.
2. В **Имя хранилища секретов** укажите отображаемое имя хранилища.

3. Укажите **Тип хранилища секретов** `HashiCorp Vault`.
4. (опционально) При использовании защищенного HTTPS-соединения с хранилищем секретов добавьте сертификат стандарта X.509 в текстовом формате.
5. (опционально) Добавьте описание хранилища секретов.
6. Нажмите ✓ **Применить**.

Добавленное хранилище будет показано в списке хранилищ секретов.

=== Добавление метода получения секрета

1. Перейдите в раздел **Администрирование** → подраздел **Настройки хранилища секретов** → **Список методов получения секрета**. Нажмите ⊞ **Добавить**.
2. Задайте **Имя метода получения секрета**.
3. Выберите из выпадающего списка **Имя хранилища секретов** доступное (созданное на предыдущем шаге) хранилище секретов, к которому будет выполняться подключение при выборе создаваемого метода для запроса секрета.
4. Введите **Токен** (идентификатор для получения секрета), предварительно полученный у администратора хранилища секретов.
5. Укажите в **Метод получения секретов** путь до секрета, предварительно полученный у администратора хранилища секретов.

Таблица 6. Методы получения аутентификационной информации для подключения к СУБД PostgreSQL в хранилище секретов

Метод получения секретов	HTTPS (при создании хранилища указан сертификат)	HTTP
Формат пути к секрету	`hostname:8200/v1/vault/data/postgresql`	`hostname:8201/v1/vault/data/postgresql`

Метод получения секретов	HTTPS (при создании хранилища указан сертификат)	HTTP

Описание формата метода	`hostname`
	IP-адрес или имя хоста, указанное в сертификате и используемом для подключения к хранилищу, на котором развёрнуто хранилище секретов HashiCorp Vault.
	`8200`
	Порт (по умолчанию) для прослушивания запросов к хранилищу секретов HashiCorp Vault по HTTPS.
	`/v1/vault/data/postgresql`
	Путь до папки, содержащей секрет.
		`hostname`
		IP-адрес или имя хоста, на котором развёрнуто хранилище секретов HashiCorp Vault, секрет которого запрашивается.
		`8201`
		Порт (по умолчанию) для прослушивания запросов к хранилищу секретов HashiCorp Vault по HTTP.
		`/v1/vault/data/postgresql`
		Путь до папки, содержащей секрет.

6. (опционально) Введите описание метода получения секрета.
7. Нажмите ✓ **Применить**.

Добавленный метод будет отображён в **Списке методов получения секрета**.

=== Настройка доступа пользователей к хранилищу секретов

Суперпользователь может назначить *Супервайзеру* или *Администратору* доступ к выбранному секрету посредством ассоциации пользователя с методом получения секрета.

Таблица 7. Права доступа пользователей RuBackup к секретам хранилища HashiCorp Vault

Операция	Роль				
	Суперпользователь	Администратор	Аудитор	Сопровождающий	Супервайзер
Редактирование данных хранилища секретов	✓	✗	✗	✗	✗
Добавление данных хранилища секретов	✓	✗	✗	✗	✗
Удаление данных хранилища секретов	✓	✗	✗	✗	✗
Добавление методов получения секретов	✓	✗	✗	✗	✗
Просмотр методов получения секретов	✓	✓	✗	✗	✓
Редактирование методов получения секретов	✓	✗	✗	✗	✗
Удаление методов получения секретов	✓	✗	✗	✗	✗

Управление доступом к методам получения секретов	✓	✗	✗	✗	✗

— доступ на выбранный метод назначает Суперпользователь

1. Перейдите в раздел **Администрирование** → подраздел **Настройки хранилища секретов** → **Доступ пользователей к методам**. Нажмите **Добавить**.
2. Выберите из выпадающего списка **Имя хранилища секретов** хранилище секретов.
3. Выберите из выпадающего списка **Имя метода получения секрета** метод, с которым будет ассоциирован пользователь.
4. Выберите из списка пользователей (с ролью *Супервайзер* и *Администратор*), которым будет назначен доступ к методу получения секрета в выбранном хранилище секретов.
5. Нажмите **Применить**.

Добавленный пользователь и ассоциированный с ним метод будет показан в окне **Доступ пользователей к методам**.

== Резервное копирование и восстановление СУБД PostgreSQL в кластере Patroni

Резервное копирование выполняется только на клиенте с ролью, указанной в параметре `patroni_node_type_for_backup` файла настроек модуля.

Для выполнения резервного копирования в кластере Patroni необходимо в файле настроек на каждом клиенте, входящем в кластер, задать параметры `patroni_host`, `patroni_port`, `patroni_node_type_for_backup`.

Если выполняется резервное копирование с узла кластера Patroni, не являющегося лидером, параметр `archive_mode` *на всех узлах кластера* должен иметь значение `always`.

=== Создание группы Patroni на сервере RuBackup

При регистрации в RuBackup все клиенты помещаются в группу *No group*. Для корректной работы с кластером Patroni нужно создать отдельную группу клиентов, переместить в неё все клиенты кластера, а также установить группе атрибуты **Разделяемая группа** и **Кластерная группа**.

=== Восстановление в режиме Point in Time Recovery (PITR)

1. Определите лидера кластера Patroni командой, выполненной от пользователя `postgres` на клиенте, входящем в кластер.

```
patronictl -d etcd://адрес_хоста_etcd list patroni_cluster_1
```

2. Отключите элементы кластера с ролью *replica*.
3. Отключите элемент кластера с ролью *leader*.
4. Запустите процесс восстановления без развертывания в каталог для восстановления на клиенте с ролью *leader*.
5. После завершения задачи по восстановлению скопируйте файлы из каталога для восстановления в целевые каталоги:
   - из каталога для восстановления `/restore_dir/number.rest/var/lib/postgresql/11/main` (где `number` — номер резервной копии) в `/var/lib/postgresql/11/main`, предварительно удалив файлы из целевого каталога;
   - из каталога для восстановления `/restore_dir/number.rest/opt/rubackup/mnt/postgresql_archives` (где `number` — номер резервной копии) WAL-файлы в `/opt/rubackup/mnt/postgresql_archives/`.
6. Убедитесь, что у скопированных файлов назначены владелец и группа `postgres`.
7. На клиенте с ролью *leader* от пользователя `postgres` удалите кластер командой:

   ```
   patronictl -c /etc/patroni/config.yml remove patroni_cluster_1
   ```

   Подтвердите удаление кластера: на первый запрос введите имя кластера, на второй запрос ввести `Yes I am aware`.

```
+--------+------+------+-------+----+-----------+
| Member | Host | Role | State | TL | Lag in MB |
+ Cluster: patroni_cluster_1 (7158598221360414 82+
+--------+------+------+-------+----+-----------+
Please confirm the cluster name to remove: patroni_cluster_1
You are about to remove all information in DCS for patroni_cluster_1, please
type: "Yes I am aware": Yes I am aware
```

1. Запустите элемент кластера на клиенте с ролью *leader*.
2. Запустите элементы кластера на клиенте с ролью *replica*.
3. Убедитесь, что все элементы имеют статус *running* (от имени пользователя

`postgres` на клиенте, входящем в кластер).

```
patronictl -d etcd://адрес_хоста_etcd list patroni_cluster_1
```

4. Отключите элементы кластера с ролью *replica*.
5. Отключите элемент кластера с ролью *leader*.
6. Повторно скопируйте файлы из каталога для восстановления `/restore_dir/number.rest/var/lib/postgresql/11/main` (где `number` —номер резервной копии) в `/var/lib/postgresql/11/main`, предварительно удалив файлы из целевого каталога.
7. В файле `postgresql.conf` установите значение параметру `recovery_target_time = '2023-03-27 15:29:00'` и закомментируйте остальные параметры `recovery_target*`, если они не используются.

```
# recovery_target = ' '
# recovery_target_lsn = ''
# recovery_target_name = ''
recovery_target_time = '2023-03-27 15:29:00'
# recovery_target_timeline = 'latest'
# recovery_target_xid = ''
```

1. В файле `postgresql.auto.conf` установите параметры `restore_command`, `recovery_target_time`, `recovery_target_action`.

```
recovery_target_time = '2023-03-27 15:29:00'
recovery_target_action='promote'
restore_command = 'cp /opt/rubackup/mnt/postgresql_archives/%f %p'
```

1. Запустите СУБД.

```
sudo -u postgres путь_к_bin/postgres -D
путь_к_data_содержащей_конфиг_файлы
```

```
sudo -u postgres /usr/local/pgsql/bin/postgres -D
/usr/local/pgsql/data/patroni_cluster_1/data/
```

1. После успешного запуска СУБД проверьте в логах, что СУБД готова принимать подключения.

=== Восстановление без развертывания

1. Определите лидера кластера Patroni командой, выполненной от пользователя `postgres` на клиенте, входящем в кластер.

```
patronictl -d etcd://адрес_хоста_etcd list patroni_cluster_1
```

2. Отключите элементы кластера с ролью *replica*.
3. Отключите элемент кластера с ролью *leader*.
4. Запустите процесс восстановления без развертывания в каталог для восстановления на клиенте с ролью *leader*.
5. После завершения задачи по восстановлению переместите файлы из каталога для восстановления в целевые каталоги:
   - из каталога для восстановления `/restore_dir/number.rest/var/lib/postgresql/11/main` (где `number` — номер резервной копии) в `/var/lib/postgresql/11/main` с заменой файлов;
   - из каталога для восстановления `/restore_dir/number.rest/opt/rubackup/mnt/postgresql_archives` (где `number` — номер резервной копии) WAL-файлы в `/opt/rubackup/mnt/postgresql_archives/`.
6. Убедитесь, что у перемещенных файлов назначены владелец и группа `postgres`.
7. На клиенте с ролью *leader* от пользователя postgres удалите кластер командой:

```
patronictl -c /etc/patroni/config.yml remove patroni_cluster_1
```

   Подтвердите удаление кластера: на первый запрос введите имя кластера, на второй запрос ввести `Yes I am aware`.

```
+--------+------+------+-------+----+-----------+
| Member | Host | Role | State | TL | Lag in MB |
+ Cluster: patroni_cluster_1 (7158598221360414824+
+--------+------+------+-------+----+-----------+
Please confirm the cluster name to remove: patroni_cluster_1
```

```
You are about to remove all information in DCS for patroni_cluster_1, please
type: "Yes I am aware": Yes I am aware
```

1. Запустите элемент кластера на клиенте с ролью *leader*.
2. Запустите элементы кластера на клиенте с ролью *replica*.
3. Убедитесь, что все элементы имеют статус *running* (от имени пользователя `postgres` на клиенте, входящем в кластер).

   ```
   patronictl -d etcd://адрес_хоста_etcd list patroni_cluster_1
   ```

== Решение проблем

=== Настройка SELinux

В некоторых случаях SELinux может блокировать выполнение резервного копирования. На это может указывать ошибка в журнале:

```
cp: cannot create regular file
'/opt/rubackup/mnt/postgresql_archives/00000001000004B500000077':
Permission denied
```

Чтобы устранить ошибку, выполните следующие шаги:

1. Установите инструменты управления SELinux: пакет `policycoreutils-python-utils` (или `policycoreutils-python`, в зависимости от дистрибутива).

   RHEL, CentOS, Fedora

   ```
   sudo yum install policycoreutils-python-utils
   ```

   Debian, Ubuntu

   ```
   sudo apt install policycoreutils-python-utils
   ```

2. Создайте пользовательский модуль политики SELinux.

   Сначала создайте файл для определения вашей пользовательской политики. Например, создайте файл с именем `my_custom_policy.te`:

```
vi my_custom_policy.te
```

Добавьте в этот файл следующее содержимое, заменив `/path/to/your/folder` фактическим путем к каталогу, который вы хотите исключить из ограничений, а `my_custom_t` — пользовательским типом.

```
module my_custom_policy 1.0;
require
{
  type unconfined_t;
  type my_custom_t;
}
type my_custom_t;
allow unconfined_t my_custom_t:file { read write execute };
allow unconfined_t my_custom_t:dir { read write add_name remove_name };
```

3. Скомпилируйте модуль политики.

```
checkmodule -M -m -o my_custom_policy.mod my_custom_policy.te
semodule_package -o my_custom_policy.pp -m my_custom_policy.mod
```

4. Установите модуль политики.

```
sudo semodule -i my_custom_policy.pp
```

5. Пометьте каталог пользовательским типом SELinux:

   Используйте команду `semanage fcontext`, чтобы добавить контекст, и команду `restorecon`, чтобы его применить.

```
sudo semanage fcontext -a -t my_custom_t "/path/to/your/folder(/.*)?"
sudo restorecon -R -v /path/to/your/folder
```

=== Резервное копирование в Astra Linux

При настройке резервного копирования в ОС Astra Linux SE 1.7 необходимо в файле `/etc/parsec/mswitch.conf` для параметра `zero_if_notfound` установить значение `yes` и перезагрузить СУБД.

=== Восстановление с развертыванием в Alt Linux

Для корректного восстановления с развертыванием в ОС Alt Linux в файле `/etc/passwd` необходимо:

1. Добавить строку:

```
postgres:x:46:46:PostgreSQL Server:/var/lib/pgsql:/bin/bash
```

2. Закомментировать строку:

```
postgres:x:46:46:PostgreSQL Server:/var/lib/pgsql:/dev/null
```

=== Настройка ведомых серверов в кластерах Patroni

Проблема

Резервное копирование не с узла-лидера в кластере Patroni завершается с ошибкой.

Решение

Установите в настройках кластера `wal_level` = `always` для ведомых серверов.