RuBackup
Система резервного копирования и восстановления данных

Резервное копирование и восстановление СУБД Postgres Pro

Версия 1.5

2021 г.

Содержание

Введение

Система резервного копирования (СРК) RuBackup позволяет осуществлять резервное копирование и восстановление данных СУБД Postgres Pro версии 13.

Принцип резервного копирования данных СУБД Postgres Pro 13 с использованием RuBackup состоит в периодическом создании базовых резервных копий экземпляра СУБД по определённому расписанию.

В репозитории RuBackup базовые резервные копии будут храниться как **полные резервные копии** (full), а файлы `backup.control` и `backup_content.control`, созданные после базовой резервной копии, - как **snap-файл** с расширением `.snap.tar`. На основе полной резервной копии и snap-файла создаётся **дифференциальная резервная копия** (differential). Инкрементальное резервное копирование данных СУБД Postgres Pro 13 не предусмотрено. В случае попытки создания правила в глобальном расписании RuBackup для выполнения инкрементальной резервной копии будет создано правило для дифференциального резервного копирования.

После успешного выполнения резервного копирования следует производить периодическую очистку каталога хранения архивных файлов WAL. Автоматическая очистка не предусмотрена.

После окончания операции резервного копирования будут созданы два файла - архивный и снимок состояния - на медиасервере, которому принадлежит пул, указанный в правиле резервного копирования. Точное расположение файлов указано в записи репозитория системы резервного копирования RuBackup.

При необходимости архивный файл может быть преобразован при помощи алгоритма защитного преобразования на клиенте и сжат. Снимок состояния не преобразовывается, так как в нём располагается только информация о ресурсе, о режиме, в котором была сделана резервная копия и время старта и окончания резервного копирования. В снимке состояния отсутствуют значимые данные СУБД.

Для выполнения резервного копирования данных СУБД Postgres Pro 13 на хосте клиента должно быть достаточно свободного места для создания резервной копии. Локальное местоположение временного каталога для создания резервных копий определено в файле `/opt/rubackup/etc/config.file` (параметр `use-local-backup-directory`). Если на хосте клиента недостаточно места для создания резервной копии, ему может быть предоставлена сетевая файловая система NFS с сервера резервного копирования во временное пользование (см. «Руководство системного администратора RuBackup»).

Для выполнения резервного копирования администратор RuBackup может настраивать правила глобального расписания в оконном Менеджере Администратора RuBackup (RBM).

Клиенты RuBackup могут осуществлять восстановление данных резервных копий и создание срочных резервных копий при помощи оконного Менеджера Клиента RuBackup (RBC), а также при помощи утилит командной строки RuBackup.

Подготовка хоста СУБД Postgres Pro

Для возможности резервного копирования данных СУБД Postgres Pro при помощи СРК RuBackup на сервер с СУБД следует установить следующие пакеты:

- `rubackup-client.deb` - клиент резервного копирования,
- `rubackup-postgres-pro-13.deb` - модуль резервного копирования данных Postgres Pro 13,
- `pg-probackup-std-13.deb` - утилита для управления резервным копированием и восстановлением кластеров баз данных Postgres Pro,
- `python3-psycopg2` - драйвер PostgreSQL, совместимый с DB API 2.0.

Установка клиента RuBackup

Для осуществления резервного копирования и восстановления данных СУБД Postgres Pro 13 при помощи RuBackup на сервер должен быть установлен клиент RuBackup со всеми необходимыми модулями. Клиент RuBackup представляет собой фоновое системное приложение (демон или сервис), обеспечивающее взаимодействие с серверной группировкой RuBackup.

Для выполнения резервного копирования ресурсов СУБД Postgres Pro клиент RuBackup должен работать от имени суперпользователя (`root` в Linux и Unix).

Подробно процедуру установки клиента RuBackup см. «Руководство по установке RuBackup».

Установка пакета модулей резервного копирования

Установка пакета модулей резервного копирования RuBackup производится из учётной записи с административными правами на узле СУБД Postgres Pro **после** установки на него клиента RuBackup.

Для установки пакета модулей используйте следующий вызов:

```
$ sudo dpkg -i rubackup-postgres-pro-13.deb
Выбор ранее не выбранного пакета rubackup-postgres-pro-13.
```

```
(Чтение базы данных … на данный момент установлено 137334 файла и
каталога.)
Подготовка к распаковке rubackup-postgres-pro-13.deb …
Распаковывается rubackup-postgres-pro-13 (2021-02-20) …
Настраивается пакет rubackup-postgres-pro-13 (2021-02-20) …
```

Установка дополнительных пакетов

Пакет `pg-probackup-std-13.deb` содержит утилиту `pg_probackup` и входит в репозиторий СУБД Postgres Pro 13.

Пакет `python3-psycopg2` содержит драйвер PostgreSQL, совместимый с DB API 2.0, и входит в стандартный репозиторий.

Для установки этих пакетов используйте команды:

```
$ sudo dpkg -i rubackup-postgres-pro-13.deb
$ sudo apt install python3-psycopg2
```

Подготовка к использованию pg_probackup

RuBackup использует утилиту `pg_probackup` для выполнения резервного копирования данных СУБД Postgres Pro 13. Перед её использованием необходимо выполнить следующие действия:

- Инициализировать каталог резервных копий,
- Добавить копируемый экземпляр в каталог копий.

Для инициализации каталога резервных копий используйте следующую команду:

```
# pg_probackup init -B /opt/rubackup/mnt/pg_probackup ,
```

Внимание! Если каталог /opt/rubackup/mnt/pg_probackup существует, то он должен быть пустым. В противном случае вызов pg_probackup выдаст ошибку.

В каталоге резервных копий `/opt/rubackup/mnt/pg_probackup` утилита `pg_probackup` создаст следующие подкаталоги:

- `wal` - каталог для архивов WAL,
- `backups` - каталог для файлов резервных копий.

Утилита `pg_probackup` может сохранять резервные копии разных кластеров баз данных в одном каталоге резервных копий. Для создания необходимых подкаталогов вам следует добавить копируемый экземпляр в

каталоге копий для каждого кластера баз данных, копию которого вы будете делать.

Для определения копируемого экземпляра выполните команду:

```
# pg_probackup add-instance -B /opt/rubackup/mnt/pg_probackup -D
каталог_данных --instance имя_экземпляра
```

Здесь *каталог_данных* - это каталог, в котором хранятся все данные кластера баз данных. Например, для кластера postgres каталогом данных будет `/var/lib/pgpro/std-13/data/`. Параметр *имя_экземпляра* должен соответствовать имени каталога данных.

Внимание! Имя экземпляра должно совпадать с именем каталога данных. Например, для каталога данных /var/lib/pgpro/std-13/data/, имя экземпляра - data.

Внимание! Директория /opt/rubackup/mnt/pg_probackup и все вложенные в неё каталоги должны быть доступны для записи и чтения пользователю postgres, а также пользователю, под управлением которого работает клиент RuBackup.

Обеспечить доступ можно следующим образом:

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

Удаление клиента RuBackup

При необходимости вы можете удалить с сервера клиент RuBackup и установленные модули резервного копирования.

Удаление клиента RuBackup возможно из учётной записи с административными правами.

Для удаления сервиса `rubackup-client` используйте команды:

```
# systemctl disable rubackup-client
# systemctl daemon-reload
```

Для удаления клиента RuBackup и модуля **rubackup-postgres-pro-13** используйте команды:

```
# apt remove rubackup-postgres-pro-13
# apt remove rubackup-client
```

При необходимости удалить клиент RuBackup из конфигурации системы резервного копирования, это может сделать системный администратор RuBackup с помощью оконного Менеджера Администратора (RBM).

После удаления клиента RuBackup в ОС Astra Linux SE 1.6 с активированным режимом защитной программной среды следует:

1. Выполнить команду:

```
$ sudo update-initramfs -u -k all
```

2. Перезагрузить операционную систему

```
$ init 6
```

Подготовка СУБД Postgres Pro

Чтобы подготовить СУБД Postgres Pro к выполнению резервного копирования при помощи СРК RuBackup необходимо выполнить следующие действия.

Подготовка сервера с СУБД Postgres Pro

Для выполнения любого типа резервного копирования СУБД Postgres Pro 13 в режиме доставки WAL ARCHIVE и для дифференциального резервного копирования в режиме PAGE требуется включить архивирование WAL.

Для этого в конфигурационном файле копируемого кластера СУБД Postgres Pro 13 `/var/lib/pgpro/std-13/data/postgresql.conf` (расположение файла может отличаться в зависимости от дистрибутива Linux, версии Postgres Pro и кластера баз данных) настройте следующие параметры:

```
wal_level = replica
archive_mode = on
archive_command = '/opt/pgpro/std-13/bin/pg_probackup archive-push
-B /opt/rubackup/mnt/pg_probackup --instance имя_экземпляра --wal-
file-path %p --wal-file-name %f'
```

Значение параметра `archive_command` должно содержать каталог, в котором будут храниться архивы WAL.

После внесения изменений перезапустите сервис Postgres Pro командой:

```
$ sudo systemctl restart postgrespro-std-13.service
```

В конец конфигурационного файла pg_hba.conf добавьте строки:

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

Вместо значения `peer` везде установите `md5`. Итоговый файл должен выглядеть следующим образом:

```
# 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        backup          127.0.0.1/32            md5
host    replication     backup          127.0.0.1/32            md5
```

Перечитываем конфигурацию:

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

Проверяем на наличие опечаток:

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

После внесения изменений перезапустите сервис Postgres Pro командой:

```
$ sudo systemctl restart postgrespro-std-13.service
```

Перед выполнением дифференциальной резервной копии в режиме PTRACK следует произвести подготовительные действия:

1. В конфигурационном файле `postgresql.conf` (в каталоге кластера БД) измените значение параметра `shared_preload_libraries` на `'ptrack'`. Например:

```
# - Shared Library Preloading -
shared_preload_libraries = 'ptrack'
```

2. Добавьте в конец того же конфигурационного файла параметр `ptrack.map_size` и установите его значение согласно следующим правилам:

Для оптимальной производительности рекомендуется задавать значение `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 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 неизмененные блоки. Увеличивать значение `ptrack.map_size` сверх рекомендуемого не имеет большого практического смысла. Максимально допустимое значение — 1024.

Пример:

```
#------------------------------------------
# CUSTOMIZED OPTIONS
#------------------------------------------

#Add settings for extensions here
max_connections = 98
shared_buffers = 981MB # 25% of RAM
effective_cache_size = 3GB
work_mem = 256kB
maintenance_work_mem = 256MB
```

```
max_wal_size = 4GB
min_wal_size = 2GB
checkpoint_completion_target = 0.9
effective_cache_size = 2944MB # 75% of RAM
wal_buffers = 16MB
default_statistics_target = 100
ptrack.map_size = 512
```

3. Перезапустите сервис Postgres Pro:

```
$ sudo systemctl restart postgrespro-std-13.service
```

4. Зайдите от имени администратора БД в bakupdb:

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

5. Выполните запрос:

```
# CREATE EXTENSION ptrack;
```

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

Подготовка СУБД

Для безопасного выполнения базовой резервной копии данных Postgres Pro 13 в СУБД необходимо создать базу данных и пользователя.

Создание базы данных

Для создания базы данных выполните следующие действия:

1. Вызовите psql при помощи команды:

```
$ sudo -u postgres psql
```

2. Если для пользователя postgres не установлен пароль, установите его (в команде ниже измените ‘12345’ на подходящий пароль):

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

3. Проверьте значение для параметра `listen_addresses`. Если его значение не равняется '*', то исправьте это следующим образом:

```
# show listen_addresses;
# alter system set listen_addresses='*';
```

4. Выйдите из psql.

5. Перезапустите демон postgres-pro:

```
$ sudo systemctl restart postgrespro-std-13.service
```

6. Создайте базу данных (вместо *backupdb* задайте своё имя):

```
$ su - postgres
```

```
# createdb backupdb
```

Создание пользователя СУБД

Пользователь для выполнения операции создания базовой резервной копии должен обладать правами на выполнение функций начала и окончания резервного копирования экземпляра Postgres Pro 13. Для настройки выполните следующие действия.

1. Выполните подключение к ранее созданной базе данных от имени пользователя postgres:

```
# su - postgres
# psql -d backupdb
```

2. В psql создайте пользователя *rubackup_backuper* и наделите его особыми правами (вместо '12345' задайте желаемый пароль):

```
BEGIN;
CREATE ROLE rubackup_backuper WITH LOGIN REPLICATION 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.pg_is_in_recovery() TO
rubackup_backuper;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_start_backup(text,
boolean, boolean) TO rubackup_backuper;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_stop_backup(boolean,
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;
GRANT pg_read_all_settings TO rubackup_backuper;
COMMIT;
```

Вместо пользователя *rubackup_backuper* вы можете создать пользователя с другим именем и таким же набором прав.

В файле конфигурации модуля `/root/.pgpass` необходимо указать данные для подключения к ранее созданной базе данных и для репликации. Этот файл должен содержать строки следующего формата:

`сервер:порт:база_данных:имя_пользователя:пароль`

Например:

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

Внимание! Конфигурационный файл обязательно должен находиться в домашнем каталоге суперпользователя (/root) и иметь имя «.pgpass». Также для него должны быть определены права только для суперпользователя (chmod 0600). Если хотя бы одно из этих условий будет нарушено, то выполнение резервной копии будет прервано ошибкой.

После подготовки сервера СУБД Postgres Pro 13 к выполнению резервного копирования необходимо перезапустить клиента RuBackup:

```
# rubackup_client stop
# rubackup_client start
```

В результате клиент должен сообщить о том, что модуль для резервного копирования Postgres Pro 13 готов к работе:

```
Try to check module: Postgres Pro 13 ...
Execute OS command:
/opt/rubackup/modules/rb_module_postgres_pro_13 -t 2>&1
... module Postgres Pro 13 was checked successfully
```

Принцип полного резервного копирования Postgres Pro

В репозитории RuBackup базовые резервные копии будут храниться как полные резервные копии. В ходе базового резервного копирования данных СУБД Postgres Pro СРК RuBackup использует утилиту `pg_probackup`.

Для резервного копирования в режиме доставки WAL ARCHIVE используется команда:

```
# pg_probackup backup -B /opt/rubackup/mnt/pg_probackup --
instance=имя_экземпляра --backup-mode=FULL -j 1 --
pguser=rubackup_backuper --pgdatabase=backupdb --no-password
```

Для резервного копирования в режиме доставки WAL STREAM используется команда:

```
# pg_probackup backup -B /opt/rubackup/mnt/pg_probackup --
instance=имя_экземпляра --backup-mode=FULL --stream -j 1 --
pguser=rubackup_backuper --pgdatabase=backupdb --no-password
```

Параметры команды следующие:

`-j` - количество потоков, которые использует программа;

`--pguser` - ранее созданный пользователь базы данных;

`--pgdatabase` - ранее созданная база данных;

`--no-password` - исключаем пользовательский ввод во время выполнения резервного копирования.

Принцип дифференциального резервного копирования Postgres Pro

В ходе дифференциального резервного копирования данных СУБД Postgres Pro СРК RuBackup использует утилиту `pg_probackup`.

Выполнение дифференциального резервного копирования данных Postgres Pro 13 может производиться в трёх режимах: DELTA, PAGE и PTRACK.

В режиме DELTA:

Для дифференциального резервного копирования в режиме доставки WAL ARCHIVE используется команда:

```
# pg_probackup backup -B /opt/rubackup/mnt/pg_probackup --
instance=имя_экземпляра --backup-mode=DELTA -j 1 --
pguser=rubackup_backuper --pgdatabase=backupdb --no-password
```

Для дифференциального резервного копирования в режиме доставки WAL STREAM используется команда:

```
# pg_probackup backup -B /opt/rubackup/mnt/pg_probackup --
instance=имя_экземпляра --backup-mode=DELTA --stream -j 1 --
pguser=rubackup_backuper --pgdatabase=backupdb --no-password
```

В режиме PAGE:

Внимание! Сначала необходимо настроить режим доставки WAL ARCHIVE.

Для дифференциального резервного копирования в режиме доставки WAL ARCHIVE используется команда:

```
# pg_probackup backup -B /opt/rubackup/mnt/pg_probackup --
instance=имя_экземпляра --backup-mode=PAGE -j 1 --
pguser=rubackup_backuper --pgdatabase=backupdb --no-password
```

Для дифференциального резервного копирования в режиме доставки WAL STREAM используется команда:

```
# pg_probackup backup -B /opt/rubackup/mnt/pg_probackup --
instance=имя_экземпляра --backup-mode=PAGE --stream -j 1 --
pguser=rubackup_backuper --pgdatabase=backupdb --no-password
```

В режиме PTRACK

Внимание! Дифференциальное резервное копирование в режиме PTRACK происходит только на основе полной резервной копии, выполненной в режиме ARCHIVE.

Для дифференциального резервного копирования в режиме доставки WAL ARCHIVE используется команда:

```
# pg_probackup backup -B /opt/rubackup/mnt/pg_probackup --
instance=имя_экземпляра --backup-mode=PTRACK -j 1 --
pguser=rubackup_backuper --pgdatabase=backupdb --no-password
```

Для дифференциального резервного копирования в режиме доставки WAL STREAM используется команда:

```
# pg_probackup backup -B /opt/rubackup/mnt/pg_probackup --
instance=имя_экземпляра --backup-mode=PAGE --PTRACK -j 1 --
pguser=rubackup_backuper --pgdatabase=backupdb --no-password
```

Параметры команды следующие:

`-j` - количество потоков, которые использует программа;

`--pguser` - ранее созданный пользователь базы данных;

`--pgdatabase` - ранее созданная база данных;

`--no-password` - исключаем пользовательский ввод во время выполнения резервного копирования.

Принцип восстановления резервной копии Postgres Pro

Перед восстановлением базы данных рекомендуется сделать резервную копию всех имеющихся файлов в каталоге кластера баз данных.

Для восстановления данных СУБД Postgres Pro 13 выполните следующие действия:

1. Остановите службу Postgres Pro 13:

```
$ sudo systemctl stop postgrespro-std-13.service
```

2. Сделайте резервную копию файлов каталога кластера баз данных, для возможности отката (в примере ниже использован каталог `~/emergency_copy`, в нём должно быть достаточно места для выполнения данной операции):

```
$ sudo -iu postgres (cd /var/lib/pgpro/std-13/data && tar cfv - *)
| (cd ~/emergency_copy && tar xf - )
```

3. Очистите каталог кластера баз данных:

```
$ sudo -iu postgres rm -rf /var/lib/pgpro/std-13/data/*
```

4. Восстановите данные из резервных копий (например, установить значение `no` для параметра `direct_restore` в файле `/opt/rubackup/etc/rb_module_postgres_pro_13.conf` и выполнить восстановление резервной копии в какой-либо каталог при помощи Менеджера Клиента RuBackup (RBC) или утилиты командной строки `rb_archives`). Важно, чтобы все файлы сохранили свои изначальные разрешения и владельцев.

5. Запустите восстановление Postgres Pro (*инкрементальный_режим* может принимать значения CHECKSUM или NONE, подробнее см. Руководство Postgres Pro):

```
$ pg_probackup restore -B /postgreBackups/ --instance=postgres -i
ID_резервной_копии --no-validate -I режим_восстановления
```

6. Восстановите права пользователя и группы `postgres`:

```
$ sudo chown -R postgres:postgres /var/lib/pgpro/std-13/data/
```

7. Запустите сервис Postgres Pro:

```
$ sudo systemctl start postgrespro-std-13.service
```

Данный ручной метод может быть использован при ручном восстановлении служебной базы данных RuBackup, если для её работы используется СУБД Postgres Pro 13 и выполнялось её резервное копирование.

Мастер-ключ

В ходе установки клиента RuBackup будет создан мастер-ключ для защитного преобразования резервных копий, а также ключи для электронной подписи, если предполагается использовать электронную подпись.

Внимание! При утере ключа вы не сможете восстановить данные из резервной копии, если она была преобразована с помощью защитных алгоритмов.

Важно! Ключи рекомендуется после создания скопировать на внешний носитель, а также распечатать бумажную копию и убрать эти копии в надёжное место.

Мастер-ключ рекомендуется распечатать при помощи утилиты hexdump, так как он может содержать неотображаемые на экране символы:

```
$ hexdump /opt/rubackup/keys/master-key
0000000 79d1 4749 7335 e387 9f74 c67e 55a7 20ff
0000010 6284 54as 83a3 2053 4818 e183 1528 a343
0000020
```

Защитное преобразование резервных копий

При необходимости, сразу после выполнения резервного копирования архивы могут быть преобразованы на хосте клиента. Таким образом, важные данные будут недоступны для администратора RuBackup или других лиц, которые могли бы получить доступ к резервной копии (например, на внешнем хранилище картриджей ленточной библиотеки или на площадке провайдера облачного хранилища для ваших резервных копий).

Защитное преобразование осуществляется входящей в состав RuBackup утилитой `rbcrypt`. Ключ для защитного преобразования резервных копий располагается на хосте клиента в файле `/opt/rubackup/keys/master-key`. Защитное преобразование данных при помощи `rbcrypt` возможно с длиной ключа 256 бит (по умолчанию), а также 128, 512 или 1024 бита в зависимости от выбранного алгоритма преобразования.

Если для правила глобального расписания необходимо выбрать особый режим защитного преобразования с длиной ключа, отличной от 256 бит, и с ключом, расположенным в другом месте, то вы можете сделать это при помощи скрипта, выполняющегося после выполнения резервного копирования (определяется в правиле глобального расписания администратором RuBackup). При этом необходимо, чтобы имя преобразованного файла осталось таким же, как и ранее, иначе задача завершится с ошибкой. Провести обратное преобразование такого файла после восстановления его из архива следует вручную при помощи утилиты `rbcrypt`. При таком режиме работы нет необходимости указывать алгоритм преобразования в правиле резервного копирования, иначе архив будет повторно преобразован с использованием мастер-ключа.

Алгоритмы защитного преобразования

Для выполнения защитного преобразования доступны следующие алгоритмы:

Таблица 1. Алгоритмы защитного преобразования, доступные в утилите `rbcrypt`.

Алгоритм	Длина ключа, бит	Примечание
Anubis	128, 256	
Aria	128, 256	
CAST6	128, 256	
Camellia	128, 256	
Kalyna	128, 256, 512	Украинский национальный стандарт ДСТУ 7624:2014
Kuznyechik	256	Российский национальный стандарт ГОСТ Р 34.12-2015
MARS	128, 256	
Rijndael	128, 256	Advanced Encryption Standard (AES)
Serpent	128, 256	
Simon	128	
SM4	128	Китайский национальный стандарт для беспроводных сетей
Speck	128, 256	
Threefish	256, 512, 1024	
Twofish	128, 256	

Менеджер Администратора RuBackup (RBM)

Оконное приложение Менеджер Администратора RuBackup (RBM) предназначено для администрирования серверной группировки RuBackup, включая управление клиентами, глобальным расписанием, хранилищами резервных копий и другими параметрами RuBackup. Системный администратор RuBackup может запустить RBM на основном сервере резервного копирования RuBackup.

Для запуска RBM следует выполнить команду:

```
# ssh -X user@rubackup_server
# /opt/rubackup/bin/rbm&
```

Пользователь, запускающий RBM, должен входить в группу `rubackup`.

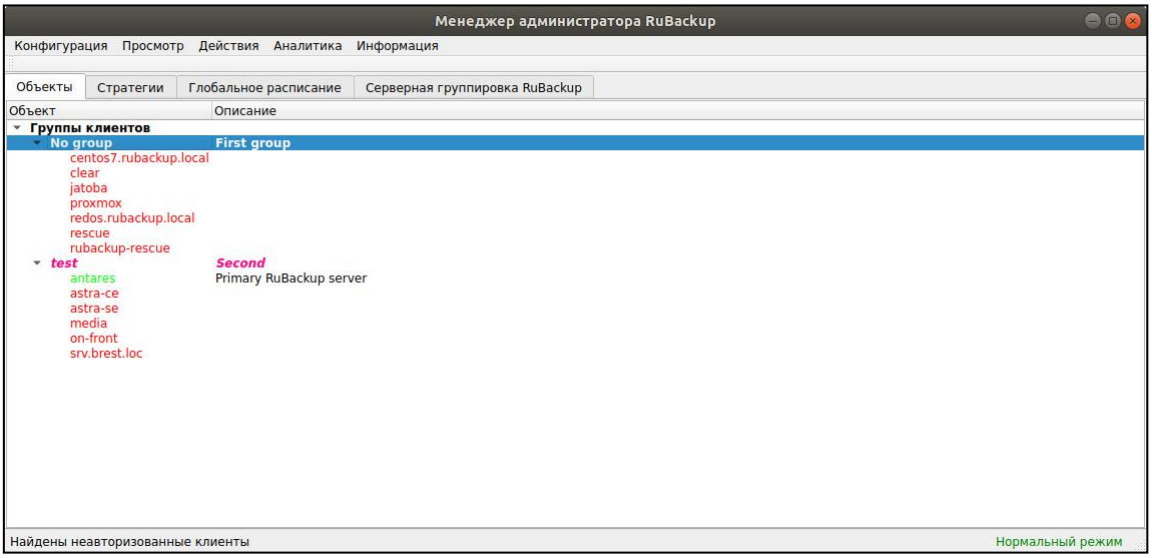


Рис. 1. Менеджер Администратора RuBackup.

Для резервного копирования данных СУБД Postgres Pro 13 на хосте должен быть установлен клиент RuBackup и необходимые модули. Клиент должен быть авторизован администратором RuBackup.

Если клиент RuBackup установлен, но не авторизован, в нижней части окна RBM появится сообщение о том, что найдены неавторизованные клиенты.

Все новые клиенты должны быть авторизованы в системе резервного копирования RuBackup.

Для авторизации неавторизованного клиента в RBM выполните следующие действия:

1. Откройте меню **Действия > Клиенты > Авторизовать клиентов**.

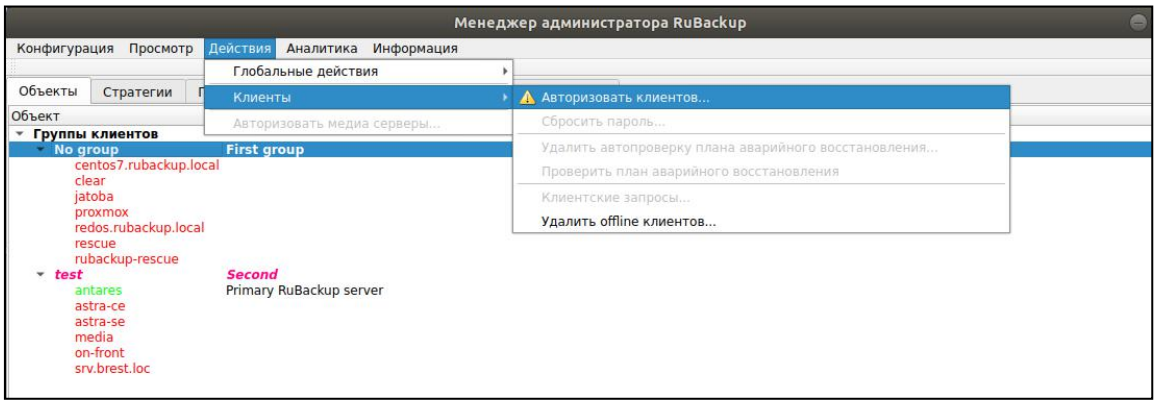


Рис. 2. Меню авторизации клиентов.

2. Выберите нужного неавторизованного клиента и нажмите **Авторизовать**.

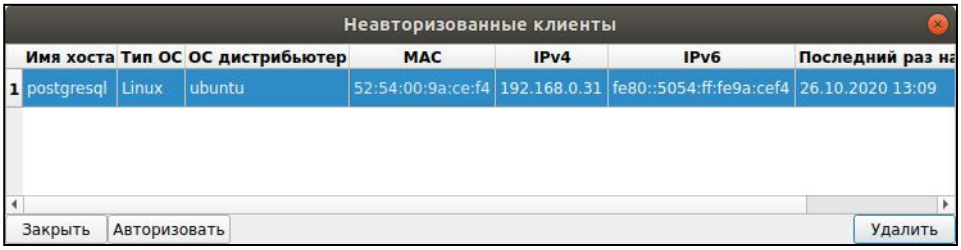


Рис. 3. Список неавторизованных клиентов в RBM.

После авторизации новый клиент будет виден в главном окне RBM:

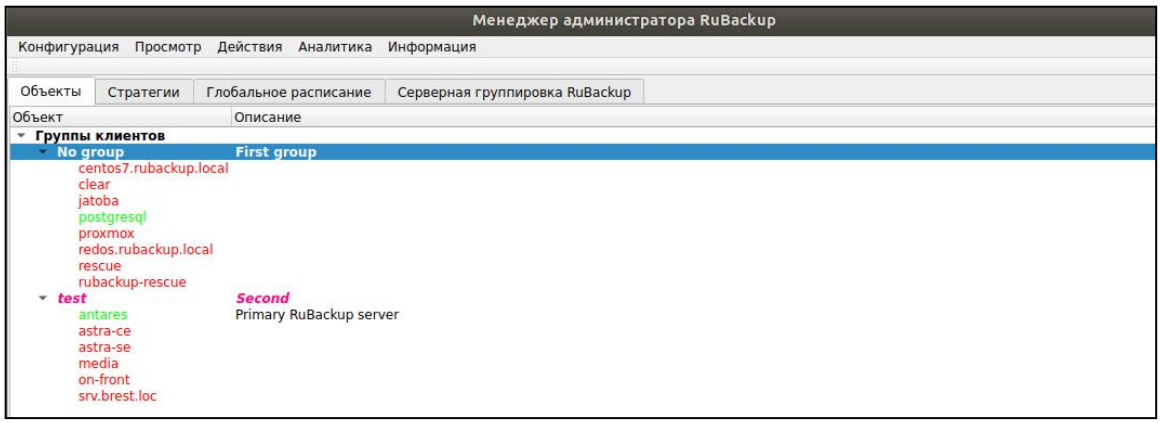


Рис. 4. Авторизованный клиент в RBM.

Клиенты могут быть сгруппированы администратором по какому-либо общему признаку. В случае необходимости восстановить резервные копии на другом хосте клиенты должны принадлежать к разделяемой группе (такая группа отмечается *курсивным шрифтом*).

Чтобы выполнять регулярное резервное копирование данных СУБД Postgres Pro 13, необходимо создать правило в глобальном расписании. Для этого выполните следующие действия:

1. Выберите хост клиента, на котором находится СУБД Postgres Pro 13, и добавьте правило резервного копирования.

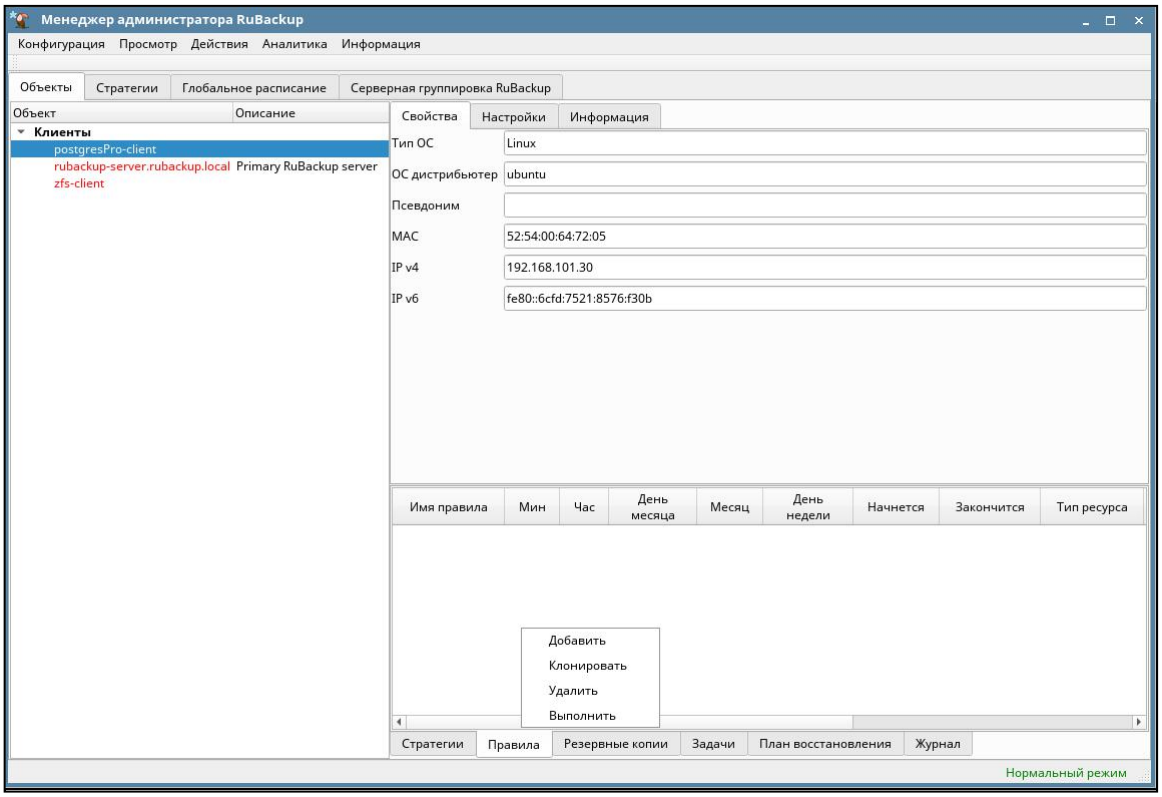


Рис. 5. Добавление правила резервного копирования.

2. Выберите тип ресурса: “**Postgres Pro 13**”.

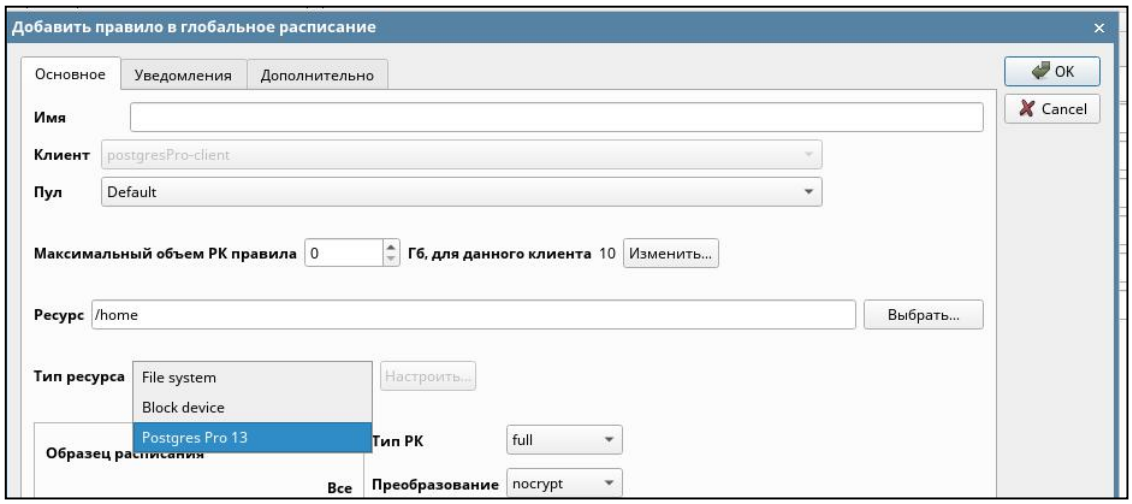


Рис. 6. Выбор типа ресурса для правила резервного копирования.

В качестве ресурса будет автоматически подставлен путь до каталога данных кластера postgres. Если необходимо сделать резервную копию другого ресурса, укажите полный путь до него вручную или выберите, нажав **Выбрать**.

3. Установите настройки правила: название правила, пул хранения данных, максимальный объём для резервных копий правила (в ГБ), тип резервного копирования, расписание резервного копирования, срок хранения и необязательный временной промежуток проверки резервной копии.

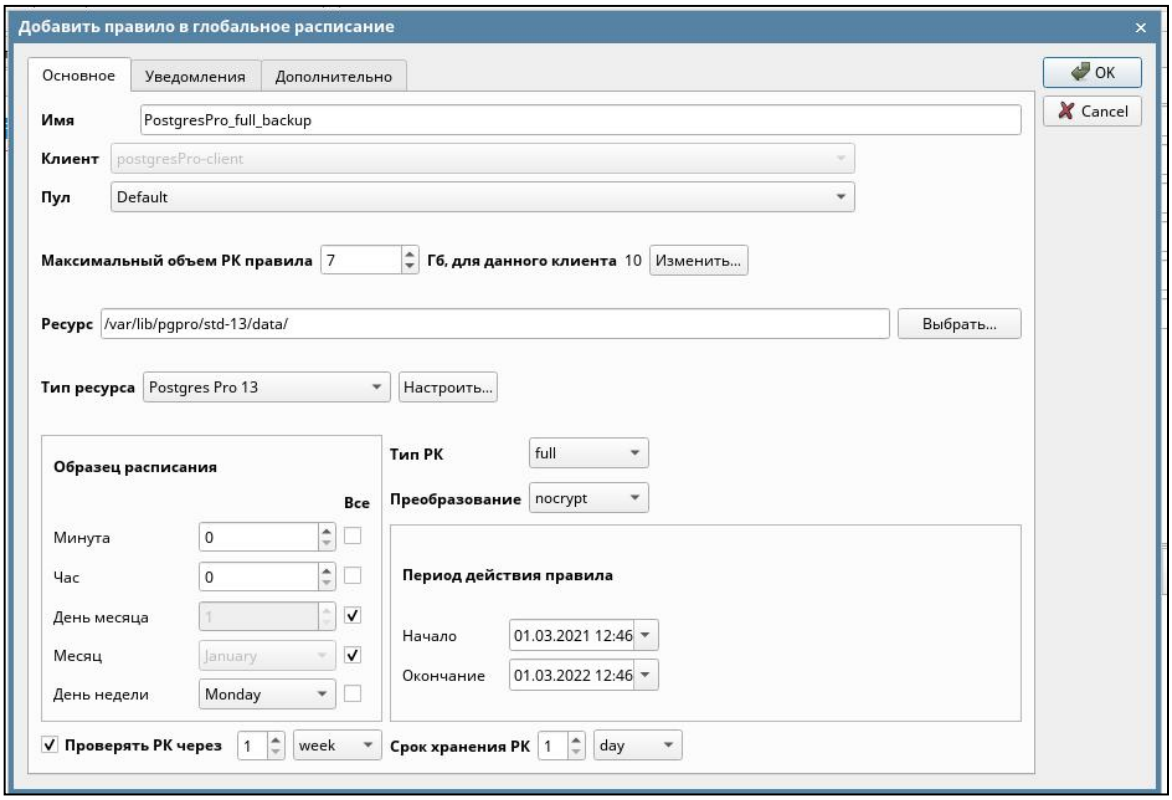


Рис. 7. Настройки правила резервного копирования.

4. Нажмите кнопку **Настроить** и настройте дополнительные параметры ресурса Postgres Pro 13

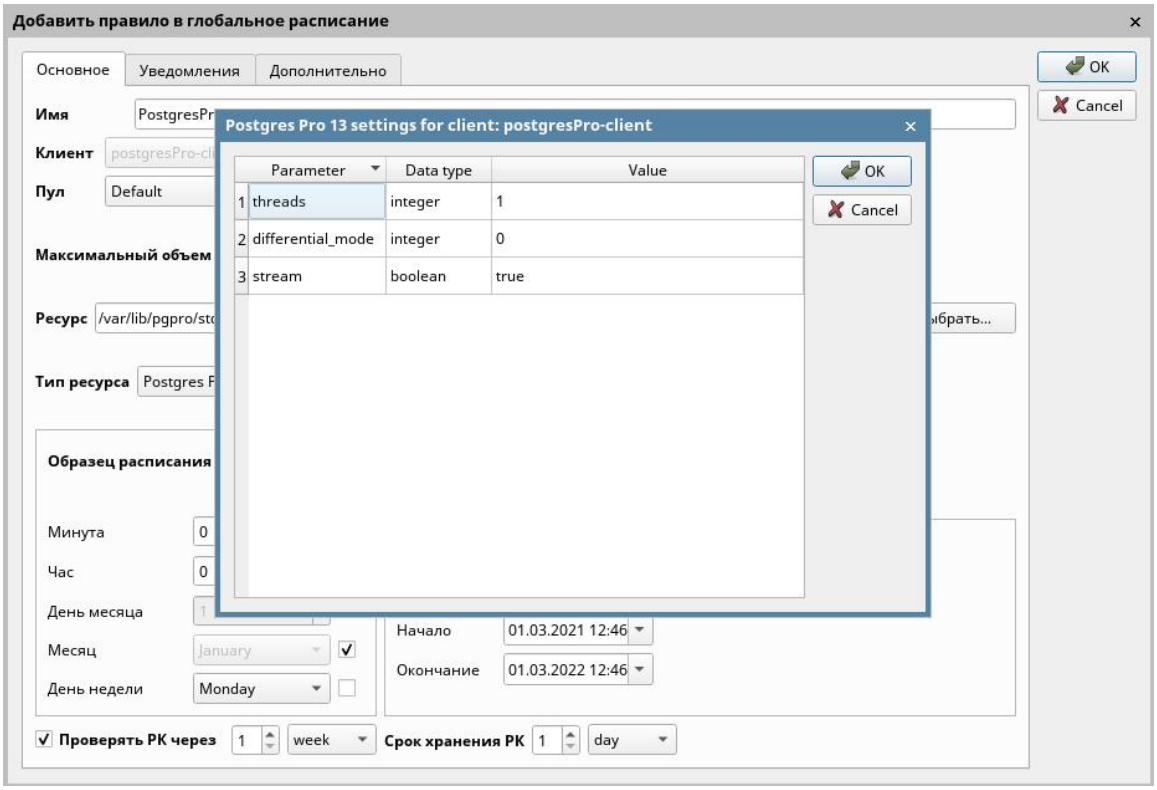


Рис. 8. Настройки ресурса Postgres Pro 13.

Здесь параметру `threads` присваивается значение, равное количеству потоков, в которые будет выполняться резервное копирование. Параметр `differential_mode` задаёт режим, в котором будет производиться дифференциальное резервное копирование (0 = DELTA, 1 = PAGE, 2 = PTRACK). Параметр `stream` определяет режим доставки WAL (false = ARCHIVE, true = STREAM).

5. На вкладке «Дополнительно» можно настроить автоматическое удаление устаревших резервных копий, определить условие их перемещения в другой пул и установить разрешение для клиента удалять резервные копии.

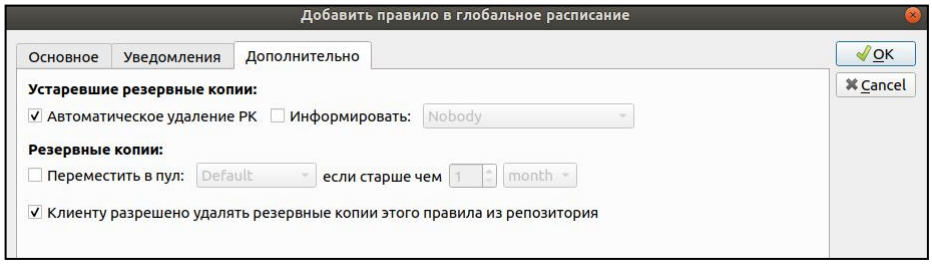


Рис. 9. Дополнительные параметры правила резервного копирования.

Вновь созданное правило будет иметь статус `wait`. Это означает, что оно не будет порождать задач на выполнение резервного копирования, пока администратор RuBackup не запустит его (тогда его статус сменится на `run`). При необходимости, администратор может приостановить работу правила или немедленно запустить его (т.е. инициировать немедленное создание задачи при статусе правила `wait`).

Правила глобального расписания имеют срок жизни, определяемый при их создании, а также предоставляют следующие возможности:

- Выполнить скрипт на клиенте перед началом резервного копирования.
- Выполнить скрипт на клиенте после успешного окончания резервного копирования
- Выполнить скрипт на клиенте после неудачного завершения резервного копирования
- Выполнить защитное преобразование резервной копии на клиенте
- Выполнить сжатие резервной копии на клиенте или на сервере после передачи ему резервной копии
- Периодически выполнять проверку целостности резервной копии
- Хранить резервные копии определённый срок, по окончании которого удалять их из хранилища резервных копий и из записей репозитория, либо уведомлять клиента об окончании срока хранения.
- Через определённый срок после создания резервной копии автоматически переместить её в другой пул хранения резервных копий, например, на картридж ленточной библиотеки.
- Уведомлять пользователей системы резервного копирования о результатах выполнения тех или иных операций, связанных с правилом глобального расписания.

При создании задачи RuBackup она появляется в главной очереди задач. Отслеживать выполнение правил может как администратор (при помощи RBM или утилит командной строки), так и клиент (при помощи RBC или утилиты командной строки `rb_tasks`).

После успешного завершения резервного копирования резервная копия будет помещена в хранилище резервных копий, а информация о ней будет размещена в репозитории RuBackup.

Менеджер Клиента RuBackup (RBC)

Принцип взаимодействия Менеджера Клиента RuBackup (RBC) с системой резервного копирования состоит в том, что клиент может сформировать ту или иную задачу (желаемое действие) и отправить её серверу резервного копирования RuBackup. Взаимодействие клиента с сервером резервного копирования производится через клиента RuBackup (фоновый процесс). RBC отправляет команду клиенту RuBackup, который отправляет её серверу. Если действие допустимо, то сервер RuBackup отдаст команду клиенту RuBackup и, при необходимости, перенаправит её медиа серверу RuBackup для дальнейшей обработки. Это означает, что, как правило, RBC не ожидает завершения того или иного действия, но ожидает ответа от клиента RuBackup, что задание принято. Это позволяет инициировать параллельные запросы процесса клиента RuBackup к серверу, но требует от клиента самостоятельно контролировать отсутствие «встречных» операций, при которых происходит восстановление данных, и в этот же момент эти же данные требуются для создания новой резервной копии. После того, как клиент отдал какую-либо команду при помощи RBC, он может просто закрыть приложение, все действия будут выполнены системой резервного копирования (тем не менее, стоит дождаться сообщения о том, что задание принято к исполнению, и проконтролировать это на вкладке «Задачи»).

Графический интерфейс RBC поддерживает русский и английский языки.

Для запуска RBC следует выполнить команды:

```
# ssh -X user@postgresql-host
# /opt/rubackup/bin/rbc&
```

Пользователь, запускающий RBC, должен входить в группу `rubackup`.

При первом запуске RBC необходимо задать пароль, при помощи которого впоследствии можно будет запросить восстановление резервной копии. Без ввода пароля получить резервную копию для клиента из хранилища невозможно. Хеш пароля восстановления хранится в базе данных сервера RuBackup. При необходимости клиент может изменить пароль при помощи RBC (меню **Конфигурация > Изменить пароль**).

Главная страница RBC содержит вкладки, которые позволяют управлять резервными копиями и расписанием резервного копирования, а также просматривать текущие задачи клиента, локальное расписание и ограничения.

Вкладка «Резервные копии»

Вкладка «Резервные копии» содержит таблицу с информацией обо всех резервных копиях клиента, которые хранятся в репозитории RuBackup. Дифференциальные резервные копии ссылаются на полные резервные копии. Инкрементальные резервные копии ссылаются на полные резервные копии или предыдущие инкрементальные. При необходимости восстановить данные можно одной командой инициировать восстановление всей цепочки резервных копий.

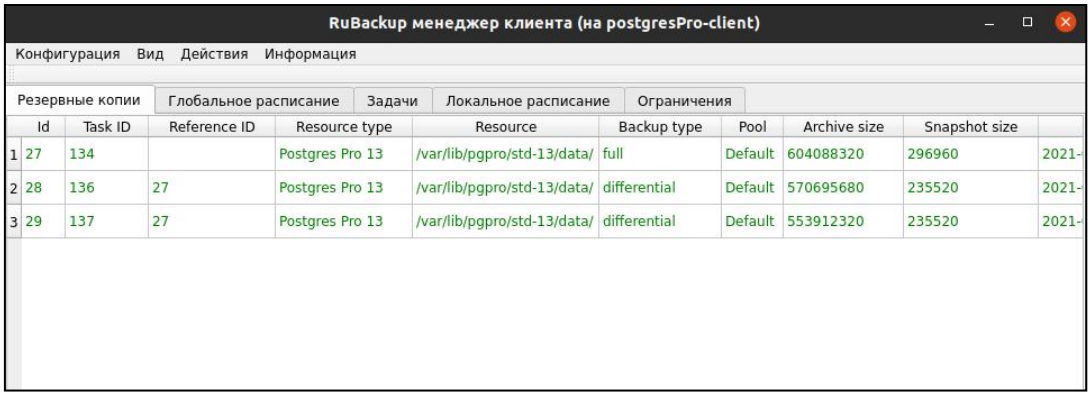


Рис. 10. Вкладка Резервные копии.

На этой вкладке клиенту доступны следующие действия:

- Удалить выбранную резервную копию. Это действие возможно в том случае, если в правиле глобального расписания есть соответствующее разрешение. При удалении резервной копии потребуется вести пароль клиента.
- Восстановить цепочку резервных копий. Это действие запускает процесс восстановления цепочки резервных копий на локальной файловой системе клиента. При восстановлении резервной копии или цепочки резервных копий клиент должен выбрать место для восстановления файлов резервной копии. Рекомендуется использовать временный каталог для операций с резервными копиями (например, `/rubackup-tmp`). Если в файле `/opt/rubackup/etc/rb_module_postgres_pro_13.conf` параметр `direct_restore` имеет значение `yes`, то произойдёт остановка сервиса Postgres Pro, очистка каталога кластера баз данных, перемещение восстановленной полной резервной копии в каталог кластера баз данных и будут выполнены все необходимые настройки для восстановления СУБД при старте службы `postgresql`. Запуск службы `postgresql` необходимо выполнить в ручном режиме. Если в файле `/opt/rubackup/etc/rb_module_postgres_pro_13.conf` параметр `direct_restore` имеет значение `no`, то восстановленные резервные копии будут расположены в выбранном для восстановления каталоге и далее вы сможете провести восстановление СУБД в ручном

режиме.
RBC не ожидает окончания восстановления всех резервных копий. Клиент должен проконтролировать на вкладке «Задачи» успешное завершение созданных задач на восстановление данных завершились успешно (статус задач `Done`). Для успешного выполнения этого действия требуется наличие достаточного свободного места в каталоге, предназначенном для создания и временного хранения резервных копий (см. параметр `use-local-backup-directory`).

- Проверить резервную копию. Это действие инициирует создание задачи проверки резервной копии. Если резервная копия была подписана цифровой подписью, то будут проверены размер файлов резервной копии, md5 сумма и проверена сама резервная копия. Если резервная копия не была подписана цифровой подписью, то будут проверены размер файлов резервной копии и md5 сумма.

Вкладка «Глобальное расписание»

Вкладка «Глобальное расписание» содержит таблицу с информацией обо всех правилах в глобальном расписании RuBackup для этого клиента.

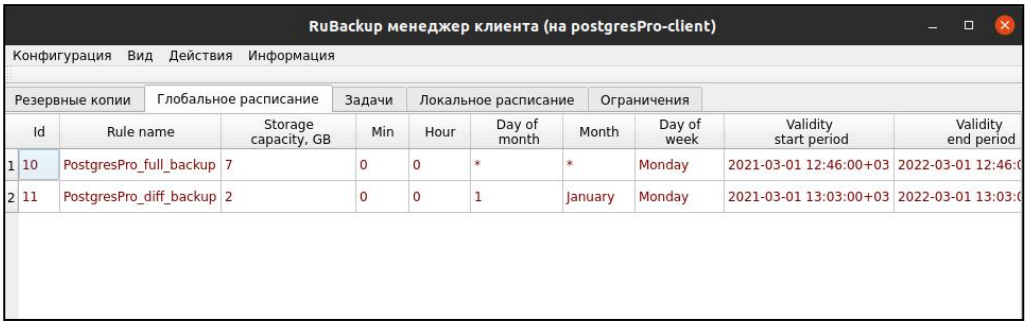


Рис. 11. Вкладка Глобальное расписание.

На этой вкладке клиенту доступны следующие действия:

- Запросить новое правило. Это действие вызывает диалог подготовки нового правила в глобальном расписании RuBackup для клиента. Запрос на добавление правила требует одобрения администратора RuBackup, одобрение может быть сделано в RBM.
- Запросить удаление правила из глобального расписания. Это действие формирует запрос к администратору RuBackup об удалении выбранного пользователем правила из глобального расписания RuBackup. Запрос на удаление правила требует одобрения администратора RuBackup, одобрение может быть сделано в RBM.

Вкладка «Задачи»

Вкладка «Задачи» содержит таблицу информацией обо всех задачах в главной очереди заданий RuBackup для этого клиента.

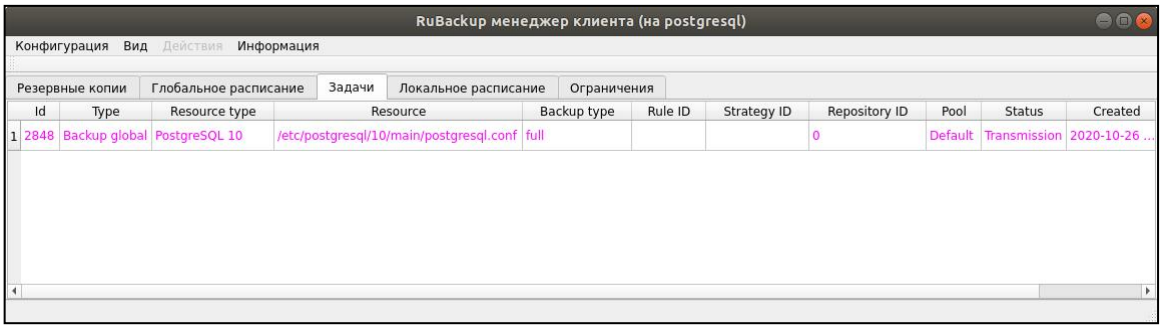


Рис. 12. Вкладка Задачи.

В зависимости от настроек сервера RuBackup выполненные задачи и задачи, завершившиеся неудачно, через какое-то время могут быть автоматически удалены из главной очереди задач. Информация о выполнении заданий фиксируется в специальном журнале задач сервера RuBackup. При необходимости статус любой задачи, даже удалённой из очереди, можно уточнить у администратора RuBackup. Также информация о выполнении задач клиента заносится в локальный файл журнала на хосте клиента. В RBC можно открыть окно отслеживания журнального файла (меню **Информация > Журнальный файл**).

Вкладка «Локальное расписание»

На вкладке «Локальное расписание» можно определить правила, задаваемые клиентом для каких-либо локальных ресурсов. Для работы локального расписания эта возможность должна быть включена для клиента администратором RuBackup.

Вкладка «Ограничения»

На вкладке «Ограничения» можно определить локальные ресурсы, резервное копирование которых нежелательно. Для работы локальных ограничений эта возможность должна быть включена для клиента администратором RuBackup.

Утилиты командной строки клиента

RuBackup

Для управления RuBackup со стороны клиента, помимо RBC, можно использовать утилиты командной строки. Пользователь, запускающий утилиты командной строки, должен входить в группу `rubackup`.

Ознакомиться с функциями утилит командной строки можно при помощи команды `man` и в руководстве «Утилиты командной строки RuBackup».

rb_archives

Эта утилита предназначена для просмотра списка резервных копий клиента в системе резервного копирования, создания срочных резервных копий, их удаления, проверки и восстановления. Ниже представлен пример.

```
# rb_archives
Id | Ref ID | Resource                  | Resource type   | Backup type  | Created             | Crypto  | Signed | Status
---+--------+---------------------------+-----------------+--------------+---------------------+---------+--------+--------
27 |        | /var/lib/pgpro/std-13/data/ | Postgres Pro 13 | full         | 2021-03-01 13:03:00 | nocrypt | True   | Trusted
28 | 27     | /var/lib/pgpro/std-13/data/ | Postgres Pro 13 | differential | 2021-03-01 13:05:08 | nocrypt | True   | Trusted
29 | 27     | /var/lib/pgpro/std-13/data/ | Postgres Pro 13 | differential | 2021-03-01 13:07:11 | nocrypt | True   | Trusted
```

rb_schedule

Эта утилита предназначена для просмотра имеющихся правил клиента в глобальном расписании резервного копирования. Ниже представлен пример.

```
#rb_schedule
Id | Name                    | Resource type   | Resource                  | Backup type  | Status
---+-------------------------+-----------------+---------------------------+--------------+-------
10 | PostgresPro_full_backup | Postgres Pro 13 | /var/lib/pgpro/std-13/data/ | full         | wait
11 | PostgresPro_diff_backup | Postgres Pro 13 | /var/lib/pgpro/std-13/data/ | differential | wait
```

rb_tasks

Эта утилита предназначена для просмотра задач клиента, которые присутствуют в главной очереди задач системы резервного копирования.

```
#rb_tasks
Id  | Task type     | Resource                    | Backup type  | Status | Created
----+---------------+-----------------------------+--------------+--------+----------------------
134 | Backup global | /var/lib/pgpro/std-13/data/ | full         | Done   | 2021-03-01 13:03:45+03
135 | Backup global | /var/lib/pgpro/std-13/data/ | differential | Done   | 2021-03-01 13:05:14+03
136 | Backup global | /var/lib/pgpro/std-13/data/ | differential | Done   | 2021-03-01 13:08:07+03
```

Восстановление резервной копии

СУБД Postgres Pro

Ход восстановления резервной копии СУБД Postgres Pro 13 зависит от значения параметра `direct_restore` в файле конфигурации модуля резервного копирования `/opt/rubackup/etc/rb_module_postgres_pro_13.conf`.

Если параметр `direct_restore` имеет значение `yes`, то произойдёт остановка сервиса `postgresql`, очистка каталога кластера баз данных, перемещение восстановленной полной резервной копии в каталог кластера баз данных, и будут выполнены все необходимые настройки для восстановления СУБД при старте службы `postgresql`.

Если параметр `direct_restore` имеет значение `no`, то восстановленные резервные копии будут расположены в выбранном для восстановления каталоге, и восстановление СУБД можно будет провести вручную.

Клиент может осуществить восстановление данных резервной копии в оконном Менеджере Клиента RuBackup (RBC), либо при помощи утилиты командной строки `rb_archives`.

В случае восстановления дифференциальной резервной копии будет сформирована цепочка восстановления: вначале будет восстановлена полная резервная копия, на которую будут наложены изменения из дифференциальной резервной копии.

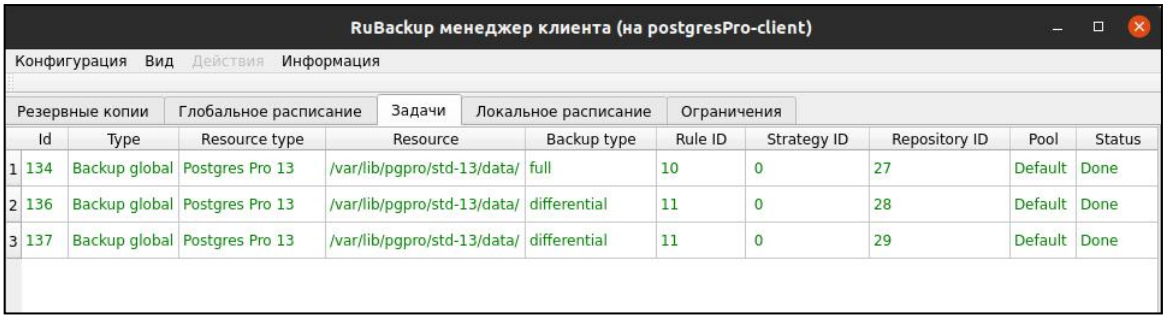
Восстановление резервной копии в RBC

Для восстановления данных резервной копии в оконном Менеджере Клиента RuBackup (RBC) выполните следующие действия.

1. Выделите нужную резервную копию и в контекстном меню выберите **Восстановить**.
2. Для восстановления потребуется ввести пароль клиента. Затем RBC выведет информационное сообщение о дальнейших действиях.
3. Укажите в качестве временного места восстановления резервных копий каталог, отдельный от каталога кластера баз данных (`/var/lib/pgpro/std-13/data/`).

4. RBC выведет информационное сообщение о создании задачи на восстановление.

Для контроля процесса восстановления RBC автоматически переключится на вкладку «Задачи», в которой можно проконтролировать результат:



	Id	Type	Resource type	Resource	Backup type	Rule ID	Strategy ID	Repository ID	Pool	Status
1	134	Backup global	Postgres Pro 13	/var/lib/pgpro/std-13/data/	full	10	0	27	Default	Done
2	136	Backup global	Postgres Pro 13	/var/lib/pgpro/std-13/data/	differential	11	0	28	Default	Done
3	137	Backup global	Postgres Pro 13	/var/lib/pgpro/std-13/data/	differential	11	0	29	Default	Done

Рис. 13. Вкладка Задачи в RBC.

Восстановление при помощи утилиты rb_archives

Для восстановления резервных копий клиент может использовать утилиту командной строки `rb_archives`. Вызов следующий:

```
# rb_archives
Id | Ref ID | Resource                  | Resource type   | Backup type  | Created             | Crypto  | Signed | Status
---+--------+---------------------------+-----------------+--------------+---------------------+---------+--------+--------
27 |        | /var/lib/pgpro/std-13/data/ | Postgres Pro 13 | full         | 2021-03-01 13:03:00 | nocrypt | True   | Trusted
28 | 27     | /var/lib/pgpro/std-13/data/ | Postgres Pro 13 | differential | 2021-03-01 13:05:08 | nocrypt | True   | Trusted
29 | 27     | /var/lib/pgpro/std-13/data/ | Postgres Pro 13 | differential | 2021-03-01 13:07:11 | nocrypt | True   | Trusted
```

В приведённом примере в системе резервного копирования присутствуют три резервные копии с идентификаторами 27, 28 и 29. Для восстановления резервной копии 28 необходимо выполнить команду:

```
# rb_archives -x 28
Password:
----> Restore archive chain: 27 28  < ----
Record ID: 27 has status: Trusted
Record ID: 28 has status: Trusted
TASK WAS ADDED TO QUEUE:140_141
```

В случае успешно принятой задачи команда вернёт список созданных задач, а восстановление будет происходить в фоновом режиме.

Проконтролировать процесс восстановления можно при помощи утилиты `rb_tasks`:

```
#rb_tasks
Id  | Task type     | Resource                  | Backup type  | Status | Created
----+---------------+---------------------------+--------------+--------+----------------------
134 | Backup global | /var/lib/pgpro/std-13/data/ | full         | Done   | 2021-03-01 13:03:45+03
135 | Backup global | /var/lib/pgpro/std-13/data/ | differential | Done   | 2021-03-01 13:05:14+03
136 | Backup global | /var/lib/pgpro/std-13/data/ | differential | Done   | 2021-03-01 13:08:07+03
140 | Restore       | /var/lib/pgpro/std-13/data/ | full         | Done   | 2021-03-01 13:44:51+03
141 | Restore       | /var/lib/pgpro/std-13/data/ | differential | Done   | 2021-03-01 13:44:51+03
```

Вы можете проконтролировать процесс восстановления в файле журнала при помощи вызова:

```
# tail -f /opt/rubackup/log/RuBackup.log
Mon Mar 1 13:45:01 2021: Module version: 1.4
Mon Mar 1 13:45:14 2021: WARNING: Backup QPAC02 is used without validation.
Mon Mar 1 13:45:14 2021: INFO: Restoring the database from backup at 2021-03-01 13:04:50+03
Mon Mar 1 13:45:15 2021: INFO: Start restoring backup files. PGDATA size: 575MB
Mon Mar 1 13:45:15 2021: INFO: Backup files are restored. Transfered bytes: 575MB, time elapsed: 0
Mon Mar 1 13:45:15 2021: INFO: Restore incremental ratio (less is better): 100% (575MB/575MB)
Mon Mar 1 13:45:15 2021: INFO: Syncing restored files to disk
Mon Mar 1 13:45:20 2021: INFO: Restored backup files are synced, time elapsed: 5s
Mon Mar 1 13:45:20 2021: INFO: Restore of backup QPAC02 completed.
Mon Mar 1 13:45:27 2021: Task was done. ID: 141
```