Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Дисциплина: Распределённые системы хранения данных

Лабораторная работа 3

Вариант 10

Выполнил:
Кривоносов Егор Дмитриевич

Группа: P33111

Преподаватель:
Николаев Владимир Вячеславович

2022 г.

Санкт-Петербург

Текст задания

Лабораторная работа включает настройку резервного копирования данных с основного узла на резервный, а также несколько сценариев восстановления. Узел из предыдущей лабораторной работы используется в качестве основного; новый узел используется в качестве резервного. В сценариях восстановления необходимо использовать копию данных, полученную на первом этапе данной лабораторной работы.

1. Резервное копирование

1.1 Настроить резервное копирование с основного узла на резервный следующим образом:
Периодические холодные полные копии.
Полная копия **(rsync)** по расписанию **(cron)** раз в сутки. СУБД на время копирования должна отключаться. На резервном узле хранить **14 копий**, после успешного создания пятнадцатой копии, самую старую автоматически уничтожать.
1.2 Подсчитать, каков будет объем резервных копий спустя месяц работы системы, исходя из следующих условий:
Средний объем новых данных в БД за сутки: **~500 МБ**.
1.3 Проанализировать результаты.

2. Потеря основного узла

Этот сценарий подразумевает полную недоступность основного узла.
Необходимо восстановить работу СУБД на резервном узле, продемонстрировать успешный запуск СУБД и доступность данных.

3. Повреждение файлов БД

Этот сценарий подразумевает потерю данных (например, в результате сбоя диска или файловой системы) при сохранении доступности основного узла.
Необходимо выполнить полное восстановление данных из резервной копии и перезапустить СУБД на основном узле.
Ход работы:
3.1 Симулировать сбой: удалить с диска директорию WAL со всем содержимым.
3.2 Проверить работу СУБД, доступность данных, перезапустить СУБД, проанализировать результаты.
3.3 Выполнить восстановление данных из резервной копии, учитывая следующее условие:

Исходное расположение дополнительных табличных пространств недоступно - разместить в другой директории и скорректировать конфигурацию.
3.4 Запустить СУБД, проверить работу и доступность данных, проанализировать результаты.

4. Логическое повреждение данных

Этот сценарий подразумевает частичную потерю данных (в результате нежелательной или ошибочной операции) при сохранении доступности основного узла.
Необходимо выполнить восстановление данных на основном узле следующим способом:
Восстановление с использованием архивных WAL файлов.(СУБД должна работать в режиме архивирования WAL, потребуется задать параметры восстановления).
Ход работы:
4.1 В каждую таблицу базы добавить 2-3 новые строки, зафиксировать результат.
4.2 Зафиксировать время и симулировать ошибку: перезаписать строки любой таблицы “мусором” (INSERT, UPDATE)
4.3 Продемонстрировать результат.
4.4 Выполнить восстановление данных указанным способом.
4.5 Продемонстрировать и проанализировать результат.

Данные для входа

1) ssh s284261@se.ifmo.ru -p 2222 (Пароль: ******)
2) ssh postgres1@pg106 (Пароль: ******)
3) ssh postgres1@pg122 (Пароль: ******)

Выполнение

Создадим супер пользователя для тестов:
CREATE ROLE admin SUPERUSER CREATEDB CREATEROLE LOGIN PASSWORD 'admin';

Задание 1. Резервное копирование

1.1 Настройка резервного копирования

Напишем bash-скрипт, который будет останавливать кластер, отправлять полную копию на резервный узел, после удаленно запускать скрипт (для удаления).

Основной узел:

```
#!/bin/bash

CURRENT_DATE=$(date "+%Y-$m-%d-%H:%M:%S")
BACKUP_DIR_NAME="backup_${CURRENT_DATE}"

#Остановка БД
pg_ctl stop -D $HOME/u08/dnkl3

#Создание копии
/opt/csw/bin/rsync -avv $HOME/u08 $HOME/u03 postgres1@pg122:~/backups/$BACKUP_DIR_NAME --rsync-path="/opt/csw/bin/rsync"

#Запуск БД
postgres -D $HOME/u08/dnkl3 >~/logfile 2>&1 &

#Запуск скрипта на узле копирования
ssh postgres1@pg122 "bash /var/postgres/postgres1/remove_script.sh"

echo "$(date): Backup $BACKUP_DIR_NAME was successfully created in directory ~/backups"
```

Резервный узел:

```
#!/bin/bash
BACKUP_CNT=`ls -1 /var/postgres/postgres0/backups | grep ^d | grep -c backup`
BACKUP_OLDEST=`ls -1 /var/postgres/postgres0/backups | grep ^d | grep backup | awk '{print $9}' | sort | head -1`
MAX_BACKUP_CNT=14

if (($BACKUP_CNT > $MAX_BACKUP_CNT)); then
    echo "$(date) : Backup count is $BACKUP_CNT. Remove the oldest one $BACKUP_OLDEST" >> back_log.log
    rm -rf /var/postgres/postgres1/backups/$BACKUP_OLDEST
else
    echo "$(date) : Backup count is $BACKUP_CNT" >> back_log.log
fi
```

На основном узле создадим cron-файл через команду (***crontab -e)***, в котором опишем правило для запуска нашего скрипта каждую полночь (раз в сутки). После чего запустим с помощью команды **(crontab cronfile)**

Проверим список запланированных задач.

```
bash-3.2$ crontab -l
#every day
0 0 * * * bash /var/postgres/postgres1/backup_script.sh >> log_backup.log
```

Также не забыть сделать для **ОСНОВНОГО** узла подключение к резервному без пароля, чтобы скрипт корректно работал. Для этого нужно выполнить 2 команды:
ssh-keygen -t rsa
Прожимаем везде *Enter.*
ssh-copy-id -i ~/.ssh/id_rsa.pub postgres1@pg122

Проверим работоспособность по логам, запустив вручную наш скрипт командой:
bash ~/backup_script.sh >> log_backup.log

Основной узел **pg106** (**log_backup.log**)

```
bash-3.2$ more log_backup.log
waiting for server to shut down.... done
server stopped
opening connection using: ssh pg122 -l postgres1 /opt/csw/bin/rsync --server -vvlogDtpre.isf . "~/backups/backup_2022--16-00:56:42"
sending incremental file list
created directory /var/postgres/postgres1/backups/backup_2022--16-00:56:42
delta-transmission enabled
u08/
u08/.profile
u08/dnk13/
u08/dnk13/.pg_hba.conf.swn
u08/dnk13/.pg_hba.conf.swo
u08/dnk13/.pg_hba.conf.swp
u08/dnk13/PG_VERSION
u08/dnk13/current_logfiles
u08/dnk13/logfile
u08/dnk13/pg_hba.conf
u08/dnk13/pg_ident.conf
u08/dnk13/postgresql.auto.conf
u08/dnk13/postgresql.conf
u08/dnk13/postmaster.opts
u08/dnk13/base/
u08/dnk13/base/1/
u08/dnk13/base/1/112
u08/dnk13/base/1/113
u08/dnk13/base/1/1247
u08/dnk13/base/1/1247_fsm
u08/dnk13/base/1/1247_vm
u08/dnk13/base/1/1249
u08/dnk13/base/1/1249_fsm
u08/dnk13/base/1/1249_vm
u08/dnk13/base/1/1255
u08/dnk13/base/1/1255_fsm
u08/dnk13/base/1/1255_vm
u08/dnk13/base/1/1259
u08/dnk13/base/1/1259_fsm
u08/dnk13/base/1/1259_vm
u08/dnk13/base/1/12789
u08/dnk13/base/1/12789_fsm
u08/dnk13/base/1/12789_vm
u08/dnk13/base/1/12792
u08/dnk13/base/1/12793
u08/dnk13/base/1/12794
u08/dnk13/base/1/12794_fsm
u08/dnk13/base/1/12794_vm
--More--(3%)
```

```
u08/dnk13/pg_serial/
u08/dnk13/pg_snapshots/
u08/dnk13/pg_stat/
u08/dnk13/pg_stat/db_0.stat
u08/dnk13/pg_stat/db_12971.stat
u08/dnk13/pg_stat/db_16388.stat
u08/dnk13/pg_stat/global.stat
u08/dnk13/pg_stat_tmp/
u08/dnk13/pg_subtrans/
u08/dnk13/pg_subtrans/0000
u08/dnk13/pg_tblspc/
u08/dnk13/pg_tblspc/16386 -> /var/postgres/postgres1/u03/dfl14
u08/dnk13/pg_twophase/
u08/dnk13/pg_wal/
u08/dnk13/pg_wal/000000010000000000000015
u08/dnk13/pg_wal/000000010000000000000016
u08/dnk13/pg_wal/000000010000000000000017
u08/dnk13/pg_wal/000000010000000000000018
u08/dnk13/pg_wal/000000010000000000000019
u08/dnk13/pg_wal/archive_status/
u08/dnk13/pg_xact/
u08/dnk13/pg_xact/0000
total: matches=0  hash_hits=0  false_alarms=0 data=118612089

sent 118698113 bytes  received 25439 bytes  33921014.86 bytes/sec
total size is 118612122  speedup is 1.00
Mon May 16 00:56:45 MSK 2022: Backup backup_2022--16-00:56:42 was successfully created in directory ~/backups
```

Резервный узел **pg122** (**back_log.log**)

```
bash-3.2$ more back_log.log
Mon May 16 00:56:45 MSK 2022 : Backup count is 1
Mon May 16 01:02:17 MSK 2022 : Backup count is 2
Mon May 16 01:02:22 MSK 2022 : Backup count is 3
Mon May 16 01:02:27 MSK 2022 : Backup count is 4
Mon May 16 01:02:32 MSK 2022 : Backup count is 5
Mon May 16 01:02:36 MSK 2022 : Backup count is 6
Mon May 16 01:02:40 MSK 2022 : Backup count is 7
Mon May 16 01:02:44 MSK 2022 : Backup count is 8
Mon May 16 01:02:47 MSK 2022 : Backup count is 9
Mon May 16 01:02:55 MSK 2022 : Backup count is 10
Mon May 16 01:03:03 MSK 2022 : Backup count is 11
Mon May 16 01:03:10 MSK 2022 : Backup count is 12
Mon May 16 01:03:14 MSK 2022 : Backup count is 13
Mon May 16 01:03:18 MSK 2022 : Backup count is 14
Mon May 16 01:03:21 MSK 2022 : Backup count is 15. Remove the oldest one backup_2022--16-00:56:42
Mon May 16 01:03:50 MSK 2022 : Backup count is 15. Remove the oldest one backup_2022--16-01:02:14
```

Резервный узел **pg122 (~/backups)**

```
bash-3.2$ ls -l
total 42
drwxr-xr-x   3 postgres1 postgres        3 мая 16 01:02 backup_2022--16-01:02:19
drwxr-xr-x   3 postgres1 postgres        3 мая 16 01:02 backup_2022--16-01:02:24
drwxr-xr-x   3 postgres1 postgres        3 мая 16 01:02 backup_2022--16-01:02:29
drwxr-xr-x   3 postgres1 postgres        3 мая 16 01:02 backup_2022--16-01:02:33
drwxr-xr-x   3 postgres1 postgres        3 мая 16 01:02 backup_2022--16-01:02:37
drwxr-xr-x   3 postgres1 postgres        3 мая 16 01:02 backup_2022--16-01:02:40
drwxr-xr-x   3 postgres1 postgres        3 мая 16 01:02 backup_2022--16-01:02:44
drwxr-xr-x   3 postgres1 postgres        3 мая 16 01:02 backup_2022--16-01:02:51
drwxr-xr-x   3 postgres1 postgres        3 мая 16 01:02 backup_2022--16-01:02:59
drwxr-xr-x   3 postgres1 postgres        3 мая 16 01:03 backup_2022--16-01:03:07
drwxr-xr-x   3 postgres1 postgres        3 мая 16 01:03 backup_2022--16-01:03:10
drwxr-xr-x   3 postgres1 postgres        3 мая 16 01:03 backup_2022--16-01:03:14
drwxr-xr-x   3 postgres1 postgres        3 мая 16 01:03 backup_2022--16-01:03:18
drwxr-xr-x   3 postgres1 postgres        3 мая 16 01:03 backup_2022--16-01:03:46
```

1.2 Расчет объема

Размер одного бэкапа (изначально)

```
bash-3.2$ du -hs backup_2022--16-02\:06\:08/
  12M   backup_2022--16-02:06:08
```

С помощью арифметической прогрессии вычислим дальнейшие вычисления:

$a1 = 12\ \text{МБ}$

$d = 500\ \text{МБ}$

$n = 30\ \text{дней}$

$S_n = \frac{2 \cdot a_1 + d \cdot (n-1)}{2} \cdot n$ - *частная формула для суммы*

$S_{30} = \frac{2 \cdot 12 + 500 \cdot 29}{2} \cdot 30 = 217860\ MB \approx 213\ GB$

Также нужно учитывать что создаются новые **wal** файлы по 16 МБ каждый (т.к. Изначально мы при создании БД не указывали `--wal-segsize` он по дефолту 16 МБ).

Изменений у нас происходит на примерно 500 МБ, значит будет создано 32 wal файла общим объёмом в 512 МБ.

За месяц таким образом мы получим:

$S_{wal_{30}} = \sum_{x=1}^{30} (512 * x) = 238080\ \text{МБ}$

Если бы у нас в данном пункте была бы включена архивация **wal** файлов, тогда объем не превышал бы 1024 МБ (задается через *max_wal_size*), потому что старые бы файлы выгружались бы через *archive_command*. Но это уже не по условию. В данном случае *achive_mode = off.*

$S_{30} + S_{wal_{30}} = 217860 + 238080 = 455940\ \text{ГБ}$

Объем резервных копий спустя месяц: **~446 ГБ** (это сколько данных ему за месяц придется скопировать с ОСНОВНОГО узла)

По условию у нас может храниться на РЕЗЕРВНОМ узле только 14 копий. Значит в директории спустя месяц будет:
$(S_{30} + S_{wal_{30}}) - (S_{16} + S_{wal_{16}}) = 455940 - 129824 = 326116 \approx 319\,GB$

1.3 Анализ результатов

Результаты не утешительные. Полные холодные копии имеют большой достаточно размер и требуют остановки сервера (в отличии, например, от горячих инкрементальных).

Задание 2. Потеря основного узла

Восстановление

Для восстановления воссоздадим файловую структуру кластера и табличного пространства на РЕЗЕРВНОМ узле из нашего бэкапа.
cp -r ~/backups/backup_2022--16-02:06:08/* ~/

Если бы вместо postgres1 у меня был бы postgres0 на резервном узле, тогда мне еще пришлось бы поправить символическую ссылку на табличное пространство:
ln -s /var/postgres/postgres0/u03/dfl14 ~/u03/dnk13/pg_tblspc/16386

Укажем в postgresql.conf команду для загрузки wal файлов:
restore_command = 'cp /var/postgres/postgres1/wal_archive/%f %p'

Создадим в директории кластера файл, сигнализирующий о восстановлении:
touch ~/u08/dnk13/recovery.signal

Запускаем резервный кластер:
postgres -D $HOME/u08/dnk13 >~/logfile 2>&1 &

Проверим что все запустилось **(pg_ctl status)**:

```
bash-3.2$ postgres -D $HOME/u08/dnk13 >~/logfile 2>&1 &
[1] 16391
bash-3.2$ pg_ctl status
pg_ctl: server is running (PID: 16391)
/var/postgres/14.2/bin/postgres "-D" "/var/postgres/postgres1/u08/dnk13"
```

Проверим наши таблицы:
psql -p 9030 -h pg122 -d whitebunny -U s284261

```
bash-3.2$ psql -p 9030 -d whitebunny -U s284261
Password for user s284261:
psql (14.2)
Type "help" for help.

whitebunny=> \d
              List of relations
 Schema |     Name     |   Type   |   Owner
--------+--------------+----------+-----------
 public | bunny        | table    | postgres1
 public | bunny_id_seq | sequence | postgres1
(2 rows)
```

Задание 3. Повреждение файлов БД

3.1 Симуляция сбоя и проверка работы

Удаляем директорию с WAL-файлами, проверим файлы БД:
rm -rf ~/u08/dnk13/pg_wal
psql -p 9030 -d whitebunny -h pg122 -U s284261

```
bash-3.2$ rm -rf ~/u08/dnk13/pg_wal
bash-3.2$ psql -p 9030 -d whitebunny -h pg122 -U admin
Password for user admin:
psql (14.2)
Type "help" for help.

whitebunny=# \d
              List of relations
 Schema |     Name     |   Type   |   Owner
--------+--------------+----------+-----------
 public | bunny        | table    | postgres1
 public | bunny_id_seq | sequence | postgres1
(2 rows)

whitebunny=#
```

Перезапускаем сервер:

```
bash-3.2$ pg_ctl start
waiting for server to start....2
2022-05-16 03:20:12.607 MSK [807
 stopped waiting
pg_ctl: could not start server
```

```
bash-3.2$ more postgresql-2022-05-16_032012.log
2022-05-16 03:20:12.607 MSK [8070] LOG:  starting PostgreSQL 14.2 on i386-pc-solaris2.10, compiled by gcc (GCC) 3.4.3 (csl-sol210-3_4-branch+
sol_rpath), 64-bit
2022-05-16 03:20:12.608 MSK [8070] LOG:  listening on IPv6 address "::", port 9030
2022-05-16 03:20:12.608 MSK [8070] LOG:  listening on IPv4 address "0.0.0.0", port 9030
2022-05-16 03:20:12.629 MSK [8070] LOG:  listening on Unix socket "/tmp/.s.PGSQL.9030"
2022-05-16 03:20:12.686 MSK [8075] LOG:  database system shutdown was interrupted; last known up at 2022-05-16 03:17:01 MSK
2022-05-16 03:20:12.686 MSK [8075] FATAL:  required WAL directory "pg_wal" does not exist
2022-05-16 03:20:12.688 MSK [8070] LOG:  startup process (PID 8075) exited with exit code 1
2022-05-16 03:20:12.688 MSK [8070] LOG:  aborting startup due to startup process failure
2022-05-16 03:20:12.738 MSK [8070] LOG:  database system is shut down
```

3.2 Восстановление данных

Так как кластер не может запуститься без директории pg_wal, давайте выполним восстановление с последней резервной копии.
rsync -avv $HOME/backups/backup_2022--16-02:06:08/u08/ postgres1@pg106:~/u08 --rsync-path="/opt/csw/bin/rsync"

```
dnk13/pg_tblspc/16386/00000001000000000000020
dnk13/pg_tblspc/16386/00000001000000000000021
dnk13/pg_tblspc/16386/00000001000000000000022
dnk13/pg_tblspc/16386/00000001000000000000023
dnk13/pg_tblspc/16386/00000001000000000000024
dnk13/pg_tblspc/16386/PG_14_202107181/
dnk13/pg_tblspc/16386/PG_14_202107181/16388/
dnk13/pg_tblspc/16386/PG_14_202107181/16388/16398
dnk13/pg_twophase/
dnk13/pg_xact/
dnk13/pg_xact/0000
total: matches=48659  hash_hits=49087  false_alarms=0 data=2930891183

sent 2931514386 bytes  received 319701 bytes  58056120.53 bytes/sec
total size is 2965309008  speedup is 1.01
```

Так как PGDATA не доступен по dnk13 (задание), давайте восстановим его в dnk23:
rsync -avv $HOME/backups/backup_2022--16-05:35:36/u08/dnk13/ postgres1@pg106:~/u08/dnk23 --rsync-path="/opt/csw/bin/rsync"

3.3 Запуск СУБД, проверка результатов

Для запуска кластера с другого $PGDATA, достаточно изменить параметр ключа -D:
pg_ctl -D /var/postgres/postgres1/u08/dnk23 -l logfile start

```
bash-3.2$ pg_ctl -D /var/postgres/postgres1/u08/dnk23 -l logfile start
couldn't set locale correctly
waiting for server to start....couldn't set locale correctly
 done
server started
bash-3.2$ pg_ctl -D $HOME/u08/dnk23 status
pg_ctl: server is running (PID: 22839)
/var/postgres/14.2/bin/postgres "-D" "/var/postgres/postgres1/u08/dnk23"
bash-3.2$
```

Сервер запустился, проверим наши данные:
psql -p 9030 -d whitebunny -h pg122 -U s284261

```
bash-3.2$ psql -p 9030 -d whitebunny -h pg122 -U s284261
Password for user s284261:
psql (14.2)
Type "help" for help.

whitebunny=> \dt
         List of relations
 Schema | Name   | Type  |   Owner
--------+--------+-------+-----------
 public | bunny | table | postgres1
(1 row)

whitebunny=>
```

Задание 4. Логическое повреждение данных

Так как требуется восстановление с использованием архивных WAL-файлов, в начале нужно включить архивирование. Изменим **postgresql.conf**

```
# - Settings -

wal_level = replica


# - Archiving -

archive_mode = on       # enables archiving; off, on, or always
                # (change requires restart)
archive_command = 'scp %p postgres1@pg122:~/wal_lab3/%f'
```

4.1 В каждую таблицу базы добавить 2-3 новые строки

psql -h pg106 -p 9030 -d whitebunny -U admin

```
whitebunny=# select * from bunny;
 id |   name    |      type      | age
----+-----------+----------------+-----
  1 | Заяц1 | мужской |   2
  2 | Заяц2 | мужской |   1
  3 | Заяц3 | женский |   3
  4 | Заяц4 | мужской |   5
  5 | Заяц5 | женский |   1
  6 | Заяц6 | женский |   3
  7 | Заяц7 | мужской |   5
  8 | Заяц8 | женский |   2
(8 rows)
```

- до изменений

Теперь добавим новые данные в нашу таблицу **bunny**:
insert into bunny (id, name) values (31, 'backup1'), (32, 'backup2'), (33, 'backup2'), (34, 'backup3');

update bunny set id = 41 where id = 31;
update bunny set id = 42 where id = 32;
update bunny set id = 43 where id = 33;

```
whitebunny=# insert into bunny (id, name) values (31, 'backup1'), (32, 'backup2'), (33, 'backup2'), (34, 'backup3');
INSERT 0 4
whitebunny=# update bunny set id = 41 where id = 31;
UPDATE 1
whitebunny=# update bunny set id = 42 where id = 32;
UPDATE 1
whitebunny=# update bunny set id = 43 where id = 33;
UPDATE 1
```

```
whitebunny=# select * from bunny;
 id |   name    |      type      | age
----+-----------+----------------+-----
  1 | Заяц1 | мужской |   2
  2 | Заяц2 | мужской |   1
  3 | Заяц3 | женский |   3
  4 | Заяц4 | мужской |   5
  5 | Заяц5 | женский |   1
  6 | Заяц6 | женский |   3
  7 | Заяц7 | мужской |   5
  8 | Заяц8 | женский |   2
 34 | backup3   |                |
 41 | backup1   |                |
 42 | backup2   |                |
 43 | backup2   |                |
(12 rows)
```

4.2 Зафиксировать время и симулировать ошибку

```
whitebunny=# select now();
              now
-------------------------------
 2022-05-26 20:53:35.885133+03
(1 row)
```

2022-05-26 20:53:35.885133+03

```
whitebunny=# delete from bunny where (id % 2) = 1;
DELETE 6
```

4.3 Демонстрация результата

```
whitebunny=# select * from bunny;
 id |   name    |      type      | age
----+-----------+----------------+-----
  2 | Заяц2 | мужской |   1
  4 | Заяц4 | мужской |   5
  6 | Заяц6 | женский |   3
  8 | Заяц8 | женский |   2
 34 | backup3   |                |
 42 | backup2   |                |
(6 rows)
```

4.4 Восстановление данных

Остановим сервер БД:
pg_ctl stop

Скопируем на всякий случай всю **pg_wal**:

```
bash-3.2$ cp ~/lab3/cur1/pg_wal/* ~/pg_wal_lab3/
cp: /var/postgres/postgres1/lab3/cur1/pg_wal/archive_status: is a directory
bash-3.2$ ls pg_wal_lab3
total 31
-rw-------   1 postgres1 postgres 16777216 May 26 21:02 000000010000000000000003
-rw-------   1 postgres1 postgres 16777216 May 26 21:02 000000010000000000000004
```

Сравним то, что у нас заархивировано на pg122:

```
bash-3.2$ ls wal_lab3
total 5692
-rw-------   1 postgres1 postgres 16777216 мая 26 20:38 000000010000000000000001
-rw-------   1 postgres1 postgres 16777216 мая 26 20:45 000000010000000000000002
-rw-------   1 postgres1 postgres 16777216 мая 26 20:58 000000010000000000000003
```

Как мы видим не все файлы у нас заархивированы.

Возьмем наш бэкап и синхронизируем с **lab3**, чтобы восстановить файлы из него все.
rsync -avv $HOME/backups_lab3/backup_2022--26-20:44:40/lab3/ postgres1@pg106:~/lab3/ --rsync-path="/opt/csw/bin/rsync"

Удаляем файлы из папки **pg_wal**:
rm -rf ~/lab3/cur1/pg_wal/*

```
bash-3.2$ rm -rf ~/lab3/cur1/pg_wal/*
bash-3.2$ ls
total 0
```

Скопируем файл, который не заархивирован (обычно это самый последний файл при завершении БД сервера):

```
bash-3.2$ cp 000000010000000000000004 ~/lab3/cur1/pg_wal/
```

Установим время и команду восстановления в **postgresql.conf**

```
restore_command = 'scp postgres1@pg122:~/wal_lab3/%f %p'

recovery_target_time = '2022-05-26 20:53:35.885133+03'

recovery_target_inclusive = false
```

Создадим файл **recovery.signal** - он говорит серверу, что надо стартовать в режиме восстановления
touch ~/lab3/cur1/recovery.signal

Попробуем запустить сервер в режиме восстановления:
pg_ctl start

```
bash-3.2$ pg_ctl start
couldn't set locale correctly
waiting for server to start....couldn't set locale correctly
2022-05-26 21:41:11.205 MSK [15569] LOG:  redirecting log output to logging collector process
2022-05-26 21:41:11.205 MSK [15569] HINT:  Future log output will appear in directory "log".
 done
server started
bash-3.2$ pg_ctl status
pg_ctl: server is running (PID: 15569)
/var/postgres/14.2/bin/postgres
```

ТЕПЕРЬ ОБЯЗАТЕЛЬНО ПРОВЕРИМ ЛОГИ, ВДРУГ ТАМ ЧТО-ТО ВАЖНОЕ ИЗ-ЗА ЧЕГО У НАС МОЖЕТ НЕ ЗАКОНЧИТСЯ ВОССТАНОВЛЕНИЕ!!!

```
2022-05-26 21:41:11.205 MSK [15569] LOG:  listening on IPv6 address "::", port 9030
2022-05-26 21:41:11.205 MSK [15569] LOG:  listening on IPv4 address "0.0.0.0", port 9030
2022-05-26 21:41:11.228 MSK [15569] LOG:  listening on Unix socket "/tmp/.s.PGSQL.9030"
2022-05-26 21:41:11.255 MSK [15581] LOG:  database system was shut down at 2022-05-26 20:45:18 MSK
2022-05-26 21:41:11.255 MSK [15581] LOG:  creating missing WAL directory "pg_wal/archive_status"
couldn't set locale correctly
scp: /var/postgres/postgres1/wal_lab3/00000002.history: No such file or directory
2022-05-26 21:41:11.409 MSK [15581] LOG:  starting point-in-time recovery to 2022-05-26 20:53:35.885133+03
couldn't set locale correctly
2022-05-26 21:41:11.924 MSK [15581] LOG:  restored log file "000000010000000000000003" from archive
2022-05-26 21:41:12.031 MSK [15581] LOG:  consistent recovery state reached at 0/30000A0
2022-05-26 21:41:12.031 MSK [15581] LOG:  redo starts at 0/30000A0
2022-05-26 21:41:12.031 MSK [15581] LOG:  recovery stopping before commit of transaction 747, time 2022-05-26 20:55:51.66303+03
2022-05-26 21:41:12.031 MSK [15581] LOG:  pausing at the end of recovery
2022-05-26 21:41:12.031 MSK [15581] HINT:  Execute pg_wal_replay_resume() to promote.
2022-05-26 21:41:12.031 MSK [15569] LOG:  database system is ready to accept read-only connections
```

ОГО У НАС ПОЯВИЛСЯ **HINT!!!**

После этого не забудем прописать функцию **pg_wal_replay_resume()** - чтобы завершить восстановление. После этого можно спокойно продолжать работать с БД.

P.S. Если не прописать эту функцию, тогда у нас восстановление не завершится и при

перезапуске без файла **recovery.signal** у нас все откатится к последней версии как было после DELETE.

4.5 Демонстрация результата

```
bash-3.2$ psql -h pg106 -p 9030 -d whitebunny -U admin
Password for user admin:
psql (14.2)
Type "help" for help.

whitebunny=# select * from bunny;
 id |   name    |      type      | age
----+-----------+----------------+-----
  1 | Заяц1 | мужской |   2
  2 | Заяц2 | мужской |   1
  3 | Заяц3 | женский |   3
  4 | Заяц4 | мужской |   5
  5 | Заяц5 | женский |   1
  6 | Заяц6 | женский |   3
  7 | Заяц7 | мужской |   5
  8 | Заяц8 | женский |   2
 34 | backup3   |                |
 41 | backup1   |                |
 42 | backup2   |                |
 43 | backup2   |                |
(12 rows)

whitebunny=# \q
bash-3.2$ clear
couldn't set locale correctly
bash-3.2$ psql -h pg106 -p 9030 -d whitebunny -U admin
Password for user admin:
psql (14.2)
Type "help" for help.

whitebunny=# select now();
              now
-------------------------------
 2022-05-26 21:42:36.823991+03
(1 row)
```

Анализ

При восстановлении логических повреждений данных wal - архивация полезна, так как помимо восстановления к самому последнему состоянию позволяет возвратиться в какой-то определенный момент времени.

Вывод

В ходе выполнения данной лабораторной работы я познакомился с прерывным бэкапом postgresql кластера, посмотрел на практике как его настроить и применять при различных сбоях: полной потери основного узла, повреждении файлов БД или логического повреждения данных. Во всех случаях WAL бэкапы решили задачу.