5.2

Создаем стенд

Текущий стенд состоит из 1-ого узла: `vs01.sXXXXXX`, (XXXXXX - ваш номер студента).
Работа стенда 6 часов. 2 попытки запуска стенда.

1. Над текстом нажмите кнопку "Создать стенд". Каждый стенд создан под определенную тему курса. Сейчас вы находитесь в теме - **5. Резервное копирование и восстановление с помощью сторонних инструментов** , этот стенд не подойдет под практические занятия из других пунктов курса.

Запуск обычно идёт до 10 минут, в редких случаях - до 30 минут.

2. После создания стенда вам нужно зайти по SSH на **sbox.slurm.io** с логином и паролем из личного кабинета (https://edu.slurm.io/):

```
ssh s000001@sbox.slurm.io
```

s000001 нужно заменить на ваш номер студента

Далее вам нужно перейти на сам стенд (sbox является просто jump-хостом и не понадобится кроме как для входа):

```
ssh vs01.s000001
```

s000001 снова нужно заменить на ваш личный номер студента

На хосте vs01 у вас есть полные права (включая беспарольный sudo):

```
sudo -i
```

Будьте осторожны и не сломайте стенд до прохождения всех заданий :)

Практика: Создание и восстановление резервной копии с помощью wal-g

Можете запустить скрипт на стенде `~/practice/lecture2/1.wal-g.sh`, который запускал спикер в видео.

Или пройтись по шагам текущей практики
- https://gitlab.slurm.io/postgres/slurm_course/-/blob/main/practice/lecture2/1.wal-g.md, также представленной ниже по тексту.

Инициализация кластера PostgreSQL

1. Мы зашли на стенд. Теперь все действия с СУБД PostgreSQL мы будем выполнять от локального пользователя postgres. Для этого выполним команду

```
student$ sudo -u postgres -i
```

2. Проверяем установленные пакеты

```
postgres$ yum list installed | grep postgres
```

Должен быть примерно такой результат

```
postgresql12.x86_64              12.6-1PGDG.rhel7            @pgdg12
postgresql12-contrib.x86_64      12.6-1PGDG.rhel7            @pgdg12
postgresql12-docs.x86_64         12.6-1PGDG.rhel7            @pgdg12
postgresql12-libs.x86_64         12.6-1PGDG.rhel7            @pgdg12
postgresql12-server.x86_64       12.6-1PGDG.rhel7            @pgdg12
postgresql12-pglogical.x86_64    2.3.3-1.el7                 @barman
postgresql12-server.x86_64       12.6-1PGDG.rhel7            @pgdg12
```

3. Инициализируем кластер. Мы это делаем специально для того, чтобы потом было удобно разворачивать другие экземпляры кластера на других портах для прохождения практики. К тому же нам нужно включить контрольные суммы. Этот важный параметр. О нем мы будем говорить чуть позже.

```
postgres$ pg_ctl initdb "-D" "/var/lib/pgsql/12/main5432" -o "--data-checksums"

The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are enabled.

creating directory /var/lib/pgsql/12/main5432 ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... UTC
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    /usr/pgsql-12/bin/pg_ctl -D /var/lib/pgsql/12/main5432 -l logfile start
```

4. Последняя команда показывает, как надо запустить кластер. Меняем немного команду, для нашего удобства. Путь до файлов кластера баз данных должен быть заканчиваться на main5432, номер порта на котором будет запущен кластер. И флаг `-l logfile` нам не нужен, будем писать в стандартный файл логов postgres-а, путь до которого можно посмотреть в конфигурационном файле `postgresql.conf`. Запускаем кластер

```
postgres$ pg_ctl -D /var/lib/pgsql/12/main5432 start

waiting for server to start....2021-03-30 10:41:21.319 UTC [18476] LOG:  starting
PostgreSQL 12.6 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat
4.8.5-44), 64-bit
2021-03-30 10:41:21.319 UTC [18476] LOG:  listening on IPv4 address "127.0.0.1", port
5432
2021-03-30 10:41:21.322 UTC [18476] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
2021-03-30 10:41:21.330 UTC [18476] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5432"
2021-03-30 10:41:21.343 UTC [18476] LOG:  redirecting log output to logging collector
process
2021-03-30 10:41:21.343 UTC [18476] HINT:  Future log output will appear in directory
"log".
 done
server started
```

5. Проверяем состояние базы данных

```
postgres$ pg_ctl -D /var/lib/pgsql/12/main5432 status

pg_ctl: server is running (PID: 18476)
/usr/pgsql-12/bin/postgres "-D" "/var/lib/pgsql/12/main5432"
```

Работа с wal-g

6. Проверяем, что пакеты wal-g установлены

```
postgres$ ls -l /usr/local/bin/wal-g
-rwxrwxr-x 1 2000 2000 40014512 Nov 30 09:18 /usr/local/bin/wal-g*
```

7. Создаем файл конфигурации для wal-g

```
postgres$ nano ~/.walg.json
```

8. Заносим в файл следующую информацию

```
{
    "WALG_FILE_PREFIX": "/var/lib/pgsql/12/backups/wal-g",
    "WALG_COMPRESSION_METHOD": "brotli",
    "WALG_DELTA_MAX_STEPS": "5",
    "PGDATA": "/var/lib/pgsql/12/main5432",
    "PGHOST": "/var/run/postgresql/.s.PGSQL.5432"
}
```

Видим, что в файле конфигурации параметры для wal-g нужно передавать в ввиде json, что вначале может показаться неудобным.

В параметре `WALG_FILE_PREFIX` мы задали хранение на локальный диск сервера PostgreSQL. Это не очень удобный вариант использования wal-g. На текущий момент утилита позволяет хранить резервные копии в удаленных хранилищах

AWS, Azure, Google Cloud Storage и YA.
Параметр `WALG_COMPRESSION_METHOD` указывает метод сжатия при передаче данных. При локальном создании копии не сильно полезен, но при удаленном хранении резервных копий – очень полезный параметр.
Параметр `WALG_DELTA_MAX_STEPS` указывает на максимальное количество дельта копий между полными копиями БД.
Параметры `PGDATA` и `PGHOST` показывают, где находятся каталог данных PostgreSQL и сокет для подключения к БД.

9. Создадим каталог для резервных копий и саму резервную копию

```
postgres$ mkdir /var/lib/pgsql/12/backups/wal-g
```

10. Изменяем конфигурацию `postgresql.conf`

```
postgres$ echo "wal_level=replica" >> /var/lib/pgsql/12/main5432/postgresql.auto.conf
postgres$ echo "archive_mode=on" >> /var/lib/pgsql/12/main5432/postgresql.auto.conf
postgres$ echo "archive_command='wal-g wal-push \"%p\" >>
/var/lib/pgsql/12/main5432/log/archive_command.log 2>&1' " >>
/var/lib/pgsql/12/main5432/postgresql.auto.conf
postgres$ echo "archive_timeout=60" >>
/var/lib/pgsql/12/main5432/postgresql.auto.conf
postgres$ echo "restore_command='wal-g wal-fetch \"%f\" \"%p\" >>
/var/lib/pgsql/12/main5432/log/restore_command.log 2>&1' " >>
/var/lib/pgsql/12/main5432/postgresql.auto.conf

postgres$ cat ~/12/main5432/postgresql.auto.conf
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
wal_level=replica
archive_mode=on
archive_command='wal-g wal-push "%p" >>
/var/lib/pgsql/12/main5432/log/archive_command.log 2>&1'
archive_timeout=60
restore_command='wal-g wal-fetch "%f" "%p" >>
/var/lib/pgsql/12/main5432/log/restore_command.log 2>&1'
```

11. Перезапускаем кластер PostgreSQL

```
postgres$ pg_ctl -D /var/lib/pgsql/12/main5432/ restart
```

12. Создадим новую базу данных, таблицу в ней и заполним данными

```
postgres$ psql
5432=> CREATE DATABASE wal_g;
CREATE DATABASE
5432=> \c wal_g
You are now connected to database "wal_g" as user "postgres".
5432=> CREATE TABLE test(id BIGINT PRIMARY KEY, name TEXT);
CREATE TABLE
5432=> INSERT INTO test(id, name) VALUES (1, 'Value 1'), (2, 'Value 2'), (3, 'Value
3');
INSERT 0 3
5432=> \q
```

13. Запускаем команду создания резервной копии

```
postgres$ wal-g backup-push /var/lib/pgsql/12/main5432
INFO: 2021/04/05 06:28:48.890232 Couldn't find previous backup. Doing full backup.
INFO: 2021/04/05 06:28:48.902586 Calling pg_start_backup()
```

```
INFO: 2021/04/05 06:28:49.225463 Walking ...
INFO: 2021/04/05 06:28:49.225770 Starting part 1 ...
INFO: 2021/04/05 06:28:49.663335 Finished writing part 1.
INFO: 2021/04/05 06:28:49.663366 Starting part 2 ...
INFO: 2021/04/05 06:28:49.663379 /global/pg_control
INFO: 2021/04/05 06:28:49.664117 Finished writing part 2.
INFO: 2021/04/05 06:28:49.667893 Calling pg_stop_backup()
INFO: 2021/04/05 06:28:50.726188 Starting part 3 ...
INFO: 2021/04/05 06:28:50.726239 backup_label
INFO: 2021/04/05 06:28:50.726248 tablespace_map
INFO: 2021/04/05 06:28:50.726492 Finished writing part 3.
INFO: 2021/04/05 06:28:50.731314 Wrote backup with name base_000000010000000000000002
```

14. Посмотрим нашу резервную копию

```
postgres$ wal-g backup-list
name                          last_modified        wal_segment_backup_start
base_000000010000000000000002 2021-04-05T06:28:50Z 000000010000000000000002
```

15. Давайте обновим одну из записей таблице test

```
postgres$ psql -d wal_g
5432=> update test set name = 'Value 4' where id = 1;
UPDATE 1
5432=> \q
```

16. Создадим резервную копию

```
postgres$ wal-g backup-push /var/lib/pgsql/12/main5432
INFO: 2021/04/05 06:30:26.751923 Delta backup from base_000000010000000000000002 with
LSN 2000028.
INFO: 2021/04/05 06:30:26.761953 Calling pg_start_backup()
INFO: 2021/04/05 06:30:26.986952 Walking ...
INFO: 2021/04/05 06:30:26.988078 Starting part 1 ...
INFO: 2021/04/05 06:30:27.008489 Finished writing part 1.
INFO: 2021/04/05 06:30:27.008521 Starting part 2 ...
INFO: 2021/04/05 06:30:27.008536 /global/pg_control
INFO: 2021/04/05 06:30:27.009101 Finished writing part 2.
INFO: 2021/04/05 06:30:27.010012 Calling pg_stop_backup()
INFO: 2021/04/05 06:30:28.068764 Starting part 3 ...
INFO: 2021/04/05 06:30:28.068843 backup_label
INFO: 2021/04/05 06:30:28.068857 tablespace_map
INFO: 2021/04/05 06:30:28.069290 Finished writing part 3.
INFO: 2021/04/05 06:30:28.073783 Wrote backup with name
base_000000010000000000000005_D_000000010000000000000002
```

17. Теперь посмотрим на наши резервные копии

```
postgres$ wal-g backup-list
name                                                                  last_modified
wal_segment_backup_start
base_000000010000000000000002                                         2021-04-05T06:28:50Z
000000010000000000000002
base_000000010000000000000005_D_000000010000000000000002 2021-04-05T06:30:28Z
000000010000000000000005
```

18. Восстановим теперь ее на другом порту 5433

```
postgres$ wal-g backup-fetch /var/lib/pgsql/12/main5433 LATEST
INFO: 2021/04/05 06:38:43.990358 LATEST backup is:
'base_000000010000000000000005_D_000000010000000000000002'
INFO: 2021/04/05 06:38:44.003763 Delta from base_000000010000000000000002 at LSN
2000028
INFO: 2021/04/05 06:38:44.023523 Finished decompression of part_003.tar.br
```

```
INFO: 2021/04/05 06:38:44.023537 Finished extraction of part_003.tar.br
INFO: 2021/04/05 06:38:55.252890 Finished decompression of part_001.tar.br
INFO: 2021/04/05 06:38:55.252964 Finished extraction of part_001.tar.br
INFO: 2021/04/05 06:38:55.253362 Finished extraction of pg_control.tar.br
INFO: 2021/04/05 06:38:55.253372
Backup extraction complete.
INFO: 2021/04/05 06:38:55.253392 base_000000010000000000000002 fetched. Upgrading
from LSN 2000028 to LSN 5000028
INFO: 2021/04/05 06:38:55.260607 Finished decompression of pg_control.tar.br
INFO: 2021/04/05 06:38:55.274337 Finished decompression of part_003.tar.br
INFO: 2021/04/05 06:38:55.274351 Finished extraction of part_003.tar.br
INFO: 2021/04/05 06:38:55.292880 Finished decompression of part_001.tar.br
INFO: 2021/04/05 06:38:55.292895 Finished extraction of part_001.tar.br
INFO: 2021/04/05 06:38:55.302957 Finished decompression of pg_control.tar.br
INFO: 2021/04/05 06:38:55.302969 Finished extraction of pg_control.tar.br
INFO: 2021/04/05 06:38:55.302977
Backup extraction complete.
```

19. Создаем файл для восстановления файлов wal из архива, меняем порт с 5432 на 5433 и запускаем кластер PostgreSQL

```
postgres$ touch "/var/lib/pgsql/12/main5433/recovery.signal"
postgres$ sed -i 's/#port = 5432/port = 5433/'
/var/lib/pgsql/12/main5433/postgresql.conf
postgres$ pg_ctl -D /var/lib/pgsql/12/main5433/ start
waiting for server to start....2021-04-05 06:39:12.419 UTC [3055] LOG:  starting
PostgreSQL 12.6 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat
4.8.5-44), 64-bit
2021-04-05 06:39:12.420 UTC [3055] LOG:  listening on IPv4 address "127.0.0.1", port
5433
2021-04-05 06:39:12.424 UTC [3055] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5433"
2021-04-05 06:39:12.432 UTC [3055] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5433"
2021-04-05 06:39:12.449 UTC [3055] LOG:  redirecting log output to logging collector
process
2021-04-05 06:39:12.449 UTC [3055] HINT:  Future log output will appear in directory
"log".
 done
server started
```

20. Проверяем работу кластера и данные в таблице test

```
postgres$ psql -p5433 -d wal_g
5433=> select * from test;
id |  name
----+---------
  2 | Value 2
  3 | Value 3
  1 | Value 4
(3 rows)
5433=> \q
```

5.2

Практика: Создание и восстановление резервной копии с помощью barman

Можете запустить скрипт на стенде `~/practice/lecture2/2.barman.sh`, который спикер запускал в видео.

Или пройтись по шагам текущей практики
- https://gitlab.slurm.io/postgres/slurm_course/-/blob/main/practice/lecture2/2.barman.md, также представленной ниже по тексту.

21. Утилита barman содержит два типа конфиг. Свой и для каждого кластера PostgreSQL. Посмотрим на них

```
postgres$ cat /etc/barman.conf
postgres$ ls /etc/barman.d/
total 20
drwxr-xr-x   2 root root 4096 Apr  5 05:55 ./
drwxr-xr-x. 82 root root 4096 Apr  5 05:55 ../
-rw-r--r--   1 root root  947 Nov  4 08:55 passive-server.conf-template
-rw-r--r--   1 root root 1565 Nov  4 08:55 ssh-server.conf-template
-rw-r--r--   1 root root 1492 Nov  4 08:55 streaming-server.conf-template
```

22. Создадим роль в PostgreSQL для выполнения бекапов и дадим ему соответствующие права

```
postgres$ psql
psql (12.6)
Type "help" for help.
5432=>
CREATE ROLE backup WITH PASSWORD 'backup' LOGIN REPLICATION SUPERUSER;
CREATE ROLE
```

23. Выйдем из-под пользователя postgres и зайдем под пользователем root

```
postgres$ exit
student$ sudo -i
```

24. Создадим файл backup и внесем туда данные

```
root$ nano /etc/barman.d/pg_backup.conf
[pg_backup]
description = "PostgreSQL DB"
conninfo = host=localhost port=5432 user=backup password=backup dbname=template1
streaming_archiver = true
streaming_conninfo = host=localhost port=5432 user=backup password=backup
backup_method = postgres
slot_name = pg_backup
```

25. Проверим работу barman

```
root$ barman check pg_backup
Server pg_backup:
    WAL archive: FAILED (please make sure WAL shipping is setup)
```

```
    PostgreSQL: OK
    superuser or standard user with backup privileges: OK
    PostgreSQL streaming: OK
    wal_level: OK
    replication slot: FAILED (replication slot 'pg_backup' doesn't exist. Please
execute 'barman receive-wal --create-slot pg_backup')
    directories: OK
    retention policy settings: OK
    backup maximum age: OK (no last_backup_maximum_age provided)
    compression settings: OK
    failed backups: OK (there are 0 failed backups)
    minimum redundancy requirements: OK (have 0 backups, expected at least 0)
    pg_basebackup: OK
    pg_basebackup compatible: OK
    pg_basebackup supports tablespaces mapping: OK
    systemid coherence: OK (no system Id stored on disk)
    pg_receivexlog: OK
    pg_receivexlog compatible: OK
    receive-wal running: FAILED (See the Barman log file for more details)
    archiver errors: OK
```

Видим, что у нас много ошибок и предупреждений. Давайте исправим их.

26. Создадим слот репликации pg_backup

```
root$ barman receive-wal --create-slot pg_backup
Creating physical replication slot 'pg_backup' on server 'pg_backup'
Replication slot 'pg_backup' created
```

27. Проверим работу barman

```
root$ barman check pg_backup
Server pg_backup:
    PostgreSQL: OK
    superuser or standard user with backup privileges: OK
    PostgreSQL streaming: OK
    wal_level: OK
    replication slot: OK
    directories: OK
    retention policy settings: OK
    backup maximum age: OK (no last_backup_maximum_age provided)
    compression settings: OK
    failed backups: OK (there are 0 failed backups)
    minimum redundancy requirements: OK (have 0 backups, expected at least 0)
    pg_basebackup: OK
    pg_basebackup compatible: OK
    pg_basebackup supports tablespaces mapping: OK
    systemid coherence: OK (no system Id stored on disk)
    pg_receivexlog: OK
    pg_receivexlog compatible: OK
    receive-wal running: OK
    archiver errors: OK
```

28. Создадим резервную копию

```
root$ sudo -u barman barman backup pg_backup
Starting backup using postgres method for server pg_backup in
/var/lib/barman/pg_backup/base/20210405T094734
Backup start at LSN: 0/3000060 (000000010000000000000003, 00000060)
Starting backup copy via pg_basebackup for 20210405T094734
Copy done (time: 3 seconds)
Finalising the backup.
```

```
This is the first backup for server pg_backup
WAL segments preceding the current backup have been found:
    000000010000000000000001 from server pg_backup has been removed
    000000010000000000000002 from server pg_backup has been removed
Backup size: 24.2 MiB
Backup end at LSN: 0/5000000 (000000010000000000000004, 00000000)
Backup completed (start time: 2021-04-05 09:47:34.216170, elapsed time: 3 seconds)
Processing xlog segments from streaming for pg_backup
    000000010000000000000003
WARNING: IMPORTANT: this backup is classified as WAITING_FOR_WALS, meaning that
Barman has not received yet all the required WAL files for the backup consistency.
This is a common behaviour in concurrent backup scenarios, and Barman automatically
set the backup as DONE once all the required WAL files have been archived.
Hint: execute the backup command with '--wait'
```

29. О чем говорит это предупреждение? Давайте исправим такую ошибку

```
root$ barman switch-wal pg_backup
The WAL file 000000010000000000000005 has been closed on server 'pg_backup'
```

30. Посмотрим список резервных копий

```
root$ barman list-backup pg_backup
pg_backup 20210405T094734 - Mon Apr  5 09:47:37 2021 - Size: 40.2 MiB - WAL Size:
1s6.0 MiB
```

31. Давайте восстановим резервную копию. Делаем «кустарно». Прошу гнилыми помидорами не закидывать. Данные скрипт приводится в качестве примера в тестовом окружении. В рабочей среде нужно делать всё через ssh.

```
root$ sudo -u postgres pg_ctl -D /var/lib/pgsql/12/main5432 stop

root$ chown -R barman:barman /var/lib/pgsql

root$ barman recover pg_backup 20210405T094734 /var/lib/pgsql/12/main5432
Starting local restore for server pg_backup using backup 20210405T094734
Destination directory: /var/lib/pgsql/12/main5432
Copying the base backup.
Copying required WAL segments.
Generating archive status files
Identify dangerous settings in destination directory.

Recovery completed (start time: 2021-04-05 10:41:58.225068, elapsed time: 1 second)

Your PostgreSQL server has been successfully prepared for recovery!

root$ chown -R postgres:postgres /var/lib/pgsql
```

32. Переходим к пользователю postgres и запускаем кластер PostgreSQL

```
root$ exit

student$ sudo -u postgres -i

postgres$ pg_ctl -D /var/lib/pgsql/12/main5432 start
waiting for server to start....2021-04-05 10:48:09.751 UTC [2776] LOG:  starting
PostgreSQL 12.6 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat
4.8.5-44), 64-bit
2021-04-05 10:48:09.752 UTC [2776] LOG:  listening on IPv4 address "127.0.0.1", port
5432
2021-04-05 10:48:09.755 UTC [2776] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
```

```
2021-04-05 10:48:09.762 UTC [2776] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5432"
2021-04-05 10:48:09.779 UTC [2776] LOG:  redirecting log output to logging collector
process
2021-04-05 10:48:09.779 UTC [2776] HINT:  Future log output will appear in directory
"log".
 done
server started
```

33. Проверяем работу кластера и данные в таблице test

```
postgres$ psql -p5433 -d wal_g

5433=> select * from test;
id |  name
----+---------
  2 | Value 2
  3 | Value 3
  1 | Value 4
(3 rows)
5433=> \q
```

5.3

Практика: Создание и восстановление резервной копии с помощью pg_probackup

Можете запустить скрипт на стенде `~/practice/lecture2/3.pg_probackup.sh`, который спикер запускал в видео.

Или пройтись по шагам текущей практики
- https://gitlab.slurm.io/postgres/slurm_course/-/blob/main/practice/lecture2/3.pg_probackup.md, также представленной ниже по тексту.

34. Проверяем, что пакеты pg_probackup установлены

```
postgres$ yum list installed | grep probackup
pg_probackup-12.x86_64          2.4.10-1.212fad4f931e1269    @pg_probackup
```

35. Создаем каталог и устанавливаем переменную окружения BACKUP_PATH

```
postgres$ mkdir /var/lib/pgsql/12/backups/probackup
postgres$ export BACKUP_PATH='/var/lib/pgsql/12/backups/probackup'
```

36. Проверяем, что она установилась

```
postgres$ echo $BACKUP_PATH
/var/lib/pgsql/12/backup
```

37. Создадим роль в PostgreSQL для выполнения бекапов и дадим ему соответствующие права

```
postgres$ createuser backup

postgres$ psql
5432=>
ALTER ROLE backup NOSUPERUSER;
ALTER ROLE backup WITH REPLICATION;
GRANT USAGE ON SCHEMA pg_catalog TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.current_setting(text) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_is_in_recovery() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_start_backup(text, boolean, boolean) TO
backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_stop_backup(boolean, boolean) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_create_restore_point(text) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_switch_wal() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_last_wal_replay_lsn() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current_snapshot() TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_snapshot_xmax(txid_snapshot) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_control_checkpoint() TO backup;
```

38. Инициализируем наш бекап

```
postgres$ pg_probackup-12 init
```

39. В нашей директории для бекапов появились следующие папки

```
postgres$ ls -l $BACKUP_PATH
total 16
drwxr-xr-x 4 postgres postgres 4096 Apr  2 10:06 ./
drwx------ 7 postgres postgres 4096 Apr  2 09:59 ../
drwx------ 2 postgres postgres 4096 Apr  2 10:06 backups/
drwx------ 2 postgres postgres 4096 Apr  2 10:06 wal/
```

40. Инициализируем инстанс main5432

```
postgres$ pg_probackup-12 add-instance --instance 'main5432' -D ~/12/main5432/
INFO: Instance 'main5432' successfully inited
```

41. Создадим новую базу данных

```
postgres$ psql
5432=> create database probackup;
CREATE DATABASE
```

42. Таблицу в этой базе данных и заполним ее тестовыми данными

```
5432=> \c probackup
5432=> create table test(id bigserial primary key, name text);
CREATE TABLE
5432=> insert into test (name) values ('Values 1'), ('Values 2'), ('Values 3');
INSERT 0 3
5432=> select * from test;
id |   name
----+----------
  1 | Values 1
```

```
  2 | Values 2
  3 | Values 3
(3 rows)
```

43. Создадим резервную копию. Команда backup принимает три параметра:

- `-b` - тип создания резервной копии. Для первого запуска нужно создать полную копию кластера PostgreSQL, поэтому команда `FULL`
- параметр `--stream` указывает на то, что нужно вместе с созданием резервной копии, параллельно передавать wal по слоту репликации. Запуск потоковой передачи wal.
- параметр `--temp-slot` указывает на то, что потоковая передача wal-ов будет использовать временный слот репликации

```
postgres$ pg_probackup-12 backup --instance 'main5432' -b FULL --stream --temp-slot
INFO: Backup start, pg_probackup version: 2.4.10, instance: main5432, backup ID:
QQXNPP, backup mode: FULL, wal mode: STREAM, remote: false, compress-algorithm: none,
compress-level: 1
WARNING: This PostgreSQL instance was initialized without data block checksums.
pg_probackup have no way to detect data block corruption without them. Reinitialize
PGDATA with option '--data-checksums'.
WARNING: Current PostgreSQL role is superuser. It is not recommended to run backup or
checkdb as superuser.
INFO: PGDATA size: 47MB
INFO: Start transferring data files
WARNING: File: "/var/lib/pgsql/12/main5432/base/16384/16442", invalid file size 8202
INFO: Data files are transferred, time elapsed: 0
INFO: wait for pg_stop_backup()
INFO: pg_stop backup() successfully executed
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 10s
INFO: Validating backup QQXNPP
INFO: Backup QQXNPP data files are valid
INFO: Backup QQXNPP resident size: 63MB
INFO: Backup QQXNPP completed
```

Видим, что наш бекап успешно создался. Однако есть два предупреждения. О чем они говорят?

44. Первая ошибка указывает на то, что у нас не включена контрольная сумма. Мы уже знаем, как это исправить. Делаем

```
postgres$ psql
5432=> drop database if exists test_checksum;
DROP DATABASE
postgres$ pg_ctl -D /var/lib/pgsql/12/main5432 stop
waiting for server to shut down.... done
server stopped

postgres$ pg_checksums -D /var/lib/pgsql/12/main5432 -e
Checksum operation completed
Files scanned:  1266
Blocks scanned: 4111
pg_checksums: syncing data directory
pg_checksums: updating control file
Checksums enabled in cluster

postgres$ pg_ctl -D /var/lib/pgsql/12/main5432 start
```

```
waiting for server to start....2021-04-02 11:55:13.139 UTC [502] LOG:  starting
PostgreSQL 12.6 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat
4.8.5-44), 64-bit
2021-04-02 11:55:13.139 UTC [502] LOG:  listening on IPv4 address "127.0.0.1", port
5432
2021-04-02 11:55:13.143 UTC [502] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
2021-04-02 11:55:13.150 UTC [502] LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432"
2021-04-02 11:55:13.163 UTC [502] LOG:  redirecting log output to logging collector
process
2021-04-02 11:55:13.163 UTC [502] HINT:  Future log output will appear in directory
"log".
 done
server started
```

Вторая предупреждением указывает, что нам нужно делать резервные копии не под суперпользователем, а под созданным специально для этого пользователем. Чуть выше мы уже это сделали, создав юзера `backup`.

45. Теперь вернемся обратно к резервным копиям и посмотрим список созданных нами бекапов

```
postgres$ pg_probackup-12 show
============================================================================================
============================================
 Instance  Version  ID      Recovery Time           Mode   WAL Mode  TLI  Time   Data
WAL  Zratio  Start LSN   Stop LSN    Status
============================================================================================
============================================
 main5432  12       QQXRQO  2021-04-02 12:23:13+00  FULL   STREAM    1/0   20s   32MB
16MB    1.00  0/11000028  0/11000168  OK
```

46. Давайте теперь в нашу таблицу test внесем дополнительные данные

```
postgres$ psql -d probackup
5432=> insert into test (name) values ('Values 4');
INSERT 0 1
```

47. И создадим инкрементальную копию

```
postgres$ pg_probackup-12 backup --instance 'main5432' -b DELTA --stream --temp-slot
-U backup
INFO: Backup start, pg_probackup version: 2.4.10, instance: main5432, backup ID:
QQXQWE, backup mode: DELTA, wal mode: STREAM, remote: false, compress-algorithm:
none, compress-level: 1
INFO: Parent backup: QQXQBF
INFO: PGDATA size: 32MB
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 0
INFO: wait for pg_stop_backup()
INFO: pg_stop backup() successfully executed
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 3s
INFO: Validating backup QQXQWE
INFO: Backup QQXQWE data files are valid
INFO: Backup QQXQWE resident size: 21MB
INFO: Backup QQXQWE completed
```

48. Посмотрим список наших резервных копий

```
postgres$ pg_probackup-12 show
```

```
========================================================================
==============================================
 Instance  Version  ID      Recovery Time           Mode   WAL Mode  TLI  Time   Data
WAL  Zratio  Start LSN   Stop LSN    Status
========================================================================
==============================================
 main5432  12       QQXRRU  2021-04-02 12:23:55+00  DELTA  STREAM    1/1   11s  159kB
32MB    1.00  0/13000028  0/13000168  OK
 main5432  12       QQXRQO  2021-04-02 12:23:13+00  FULL   STREAM    1/0   20s   32MB
16MB    1.00  0/11000028  0/11000168  OK
```

Резервные копии успешно создались

49. Давайте теперь восстановим нашу копию QQXQWE. Останавливаем базу данных на порту 5433, удаляем каталог данных main5433 и запускаем команду восстановления

```
postgres$ pg_ctl -D /var/lib/pgsql/12/main5433 stop
waiting for server to shut down.... done
server stopped

postgres$ rm -rf /var/lib/pgsql/12/main5433

postgres$ pg_probackup-12 restore --instance 'main5432' -i 'QQXQWE' -D
/var/lib/pgsql/12/main5433
INFO: Validating parents for backup QQXQWE
INFO: Validating backup QQXNPP
INFO: Restoring the database from backup at 2021-04-02 12:05:02+00
INFO: Start restoring backup files. PGDATA size: 48MB
INFO: Backup files are restored. Transfered bytes: 48MB, time elapsed: 0
INFO: Restore incremental ratio (less is better): 100% (48MB/48MB)
INFO: Syncing restored files to disk
INFO: Restored backup files are synced, time elapsed: 10s
INFO: Restore of backup QQXQWE completed.

postgres$ sed -i 's/#port = 5432/port = 5433/'
/var/lib/pgsql/12/main5433/postgresql.conf

postgres$ pg_ctl -D /var/lib/pgsql/12/main5433 start
waiting for server to start....2021-04-02 12:13:07.623 UTC [1624] LOG:  starting
PostgreSQL 12.6 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat
4.8.5-44), 64-bit
2021-04-02 12:13:07.624 UTC [1624] LOG:  listening on IPv4 address "127.0.0.1", port
5433
2021-04-02 12:13:07.627 UTC [1624] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5433"
2021-04-02 12:13:07.634 UTC [1624] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5433"
2021-04-02 12:13:07.647 UTC [1624] LOG:  redirecting log output to logging collector
process
2021-04-02 12:13:07.647 UTC [1624] HINT:  Future log output will appear in directory
"log".
 done
server started
```

50. Проверяем, что данные восстановились

```
postgres$ psql -d probackup

5432=> select * from test;
 id |   name
```

```
----+----------
  1 | Values 1
  2 | Values 2
  3 | Values 3
  4 | Values 4
(4 rows)
```

51. Утилита pg_probackup позволяет поддерживать политику хранения резервных копии. Установим хранение одной полной копии базы данных. Давайте это рассмотрим

```
postgres$ pg_probackup-12 backup --instance 'main5432' -b FULL --stream
INFO: Backup start, pg_probackup version: 2.4.10, instance: main5432, backup ID:
QQXS47, backup mode: FULL, wal mode: STREAM, remote: false, compress-algorithm: none,
compress-level: 1
WARNING: Current PostgreSQL role is superuser. It is not recommended to run backup or
checkdb as superuser.
INFO: PGDATA size: 32MB
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 0
INFO: wait for pg_stop_backup()
INFO: pg_stop backup() successfully executed
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 10s
INFO: Validating backup QQXS47
INFO: Backup QQXS47 data files are valid
INFO: Backup QQXS47 resident size: 64MB
INFO: Backup QQXS47 completed

postgres$ pg_probackup-12 show
BACKUP INSTANCE 'main5432'
================================================================================
================================================
 Instance  Version  ID      Recovery Time           Mode   WAL Mode  TLI  Time   Data
WAL  Zratio  Start LSN   Stop LSN    Status
================================================================================
================================================
 main5432  12       QQXS47  2021-04-02 12:31:20+00  FULL   STREAM    1/0   15s   32MB
32MB    1.00  0/15000028  0/15000168  OK
 main5432  12       QQXRRU  2021-04-02 12:23:55+00  DELTA  STREAM    1/1   11s  159kB
32MB    1.00  0/13000028  0/13000168  OK
 main5432  12       QQXRQO  2021-04-02 12:23:13+00  FULL   STREAM    1/0   20s   32MB
16MB    1.00  0/11000028  0/11000168  OK

postgres$ pg_probackup-12 set-config --instance  'main5432' --retention-redundancy=1

postgres$ pg_probackup-12 delete --instance  'main5432' --delete-expired --delete-wal
INFO: Evaluate backups by retention
INFO: Backup QQXS47, mode: FULL, status: OK. Redundancy: 1/1, Time Window: 0d/0d.
Active
INFO: Backup QQXRRU, mode: DELTA, status: OK. Redundancy: 2/1, Time Window: 0d/0d.
Expired
INFO: Backup QQXRQO, mode: FULL, status: OK. Redundancy: 2/1, Time Window: 0d/0d.
Expired
INFO: Delete: QQXRRU 2021-04-02 12:23:55+00
INFO: Delete: QQXRQO 2021-04-02 12:23:13+00
INFO: There are no backups to merge by retention policy
INFO: Purging finished
INFO: There is no WAL to purge by retention policy

postgres$ pg_probackup-12 show
```

```
BACKUP INSTANCE 'main5432'
================================================================================
==============================================
 Instance  Version  ID      Recovery Time           Mode  WAL Mode  TLI  Time  Data
WAL  Zratio  Start LSN   Stop LSN    Status
================================================================================
==============================================
 main5432  12       QQXS47  2021-04-02 12:31:20+00  FULL  STREAM    1/0   15s  32MB
32MB    1.00  0/15000028  0/15000168  OK
```

52. На прошлой лекции мы рассматривали способом проверки целостности каталога данных БД. Этот механизм можно сделать и с помощью использования сторонних утилит, например pg_probackup. Давайте воспользуемся этой утилитой. Останавливаем сервер PostgreSQL. Вносить изменения в файл базы данных и заново запускать сервер PostgreSQL. Ниже не будет комментариев по шагам. Они все есть в лекции 1.

```
postgres$ psql -d probackup -c "select pg_relation_filepath('test')";
pg_relation_filepath
----------------------
 base/16384/16442
(1 row)

postgres$ pg_ctl -D /var/lib/pgsql/12/main5432 stop
waiting for server to shut down.... done
server stopped

postgres$ pg_checksums -D /var/lib/pgsql/12/main5432 -d
pg_checksums: syncing data directory
pg_checksums: updating control file
Checksums disabled in cluster

postgres$ nano /var/lib/pgsql/12/main5432/base/16384/16442

postgres$ pg_ctl -D /var/lib/pgsql/12/main5432 start
waiting for server to start....2021-03-30 20:33:46.882 UTC [25498] LOG:  starting
PostgreSQL 12.6 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat
4.8.5-44), 64-bit
2021-03-30 20:33:46.882 UTC [25498] LOG:  listening on IPv4 address "127.0.0.1", port
5432
2021-03-30 20:33:46.886 UTC [25498] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
2021-03-30 20:33:46.893 UTC [25498] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5432"
2021-03-30 20:33:46.906 UTC [25498] LOG:  redirecting log output to logging collector
process
2021-03-30 20:33:46.906 UTC [25498] HINT:  Future log output will appear in directory
"log".
 done
server started
```

53. Утилита pg_probackup позволяет не только проверяем целостность базы данных, но и проводить логическую согласованность структуры индексов PostgreSQL, но для этого ей нужен дополнительное расширение amcheck. Установим его.

```
postgres$ psql -d probackup -c "CREATE EXTENSION amcheck"
CREATE EXTENSION
```

54. Запускаем проверки целостности базы данных

```
postgres$ pg_probackup-12 checkdb -D /var/lib/pgsql/12/main5432
WARNING: This PostgreSQL instance was initialized without data block checksums.
pg_probackup have no way to detect data block corruption without them. Reinitialize
PGDATA with option '--data-checksums'.
WARNING: Current PostgreSQL role is superuser. It is not recommended to run backup or
checkdb as superuser.
INFO: Start checking data files
WARNING: File: "/var/lib/pgsql/12/main5432/base/16384/16442", invalid file size 8202
INFO: Data files are valid
```

Видно, что pg_probackup смог найти ошибку в файле базы данных.

55. Проверяем, какие данные есть в таблице test

```
postgres$ psql -d probackup -c "select * from test"
    bid     |  bbalance  | filler
------------+------------+--------
 1685258241 | 1717859174 |
(1 row)
```

5.3

Домашнее задание.
Проверка резервной копии

1. Создать новую базу данных и создать там несколько таблиц с данными
2. Воспользоваться утилитой `pg_probackup` и создать полную резервную копию базы данных
3. Восстановить с помощью `pg_probackup` только что созданную нами базу данных

Чек-лист

1. Создана новая база данных и создана там несколько таблиц с данными
2. Использована утилита `pg_probackup` и создана полная резервная копия базы

данных
3. Восстановлена с помощью `pg_probackup` только, что созданная резервная копия

Решение:

1. Создать новую базу данных и создать там несколько таблиц с данными

```
postgres$ psql
psql (12.6)
Type "help" for help.
5432=> create database restore_one_db;
CREATE DATABASE
5432=> \c restore_one_db
You are now connected to database "restore_one_db" as user "postgres".
5432=> create table test (id int);
CREATE TABLE
5432=> insert into test select id from generate_series(1,1000) id;
INSERT 0 1000
```

2. Воспользоваться утилитой `pg_probackup` и создать полную резервную копию базы данных

```
postgres$ pg_probackup-12 backup --instance 'main5432' -b FULL --stream --
temp-slot --backup-path='/var/lib/pgsql/12/backups/probackup'
INFO: Backup start, pg_probackup version: 2.4.10, instance: main5432, backup
ID: QR3HV7, backup mode: FULL, wal mode: STREAM, remote: false, compress-
algorithm: none, compress-level: 1
WARNING: This PostgreSQL instance was initialized without data block
checksums. pg_probackup have no way to detect data block corruption without
them. Reinitialize PGDATA with option '--data-checksums'.
WARNING: Current PostgreSQL role is superuser. It is not recommended to run
backup or checkdb as superuser.
WARNING: Skip hidden file: '/var/lib/pgsql/12/main5432/.barman-recover.info'
INFO: PGDATA size: 40MB
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 0
INFO: wait for pg_stop_backup()
INFO: pg_stop backup() successfully executed
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 13s
INFO: Validating backup QR3HV7
INFO: Backup QR3HV7 data files are valid
INFO: Backup QR3HV7 resident size: 72MB
INFO: Backup QR3HV7 completed

postgres$ pg_probackup-12 show
BACKUP INSTANCE 'main5432'
================================================================================
================================================
 Instance  Version  ID      Recovery Time           Mode  WAL Mode  TLI  Time
Data   WAL  Zratio  Start LSN   Stop LSN    Status
================================================================================
================================================
 main5432  12       QR3HV7  2021-04-05 14:35:31+00  FULL  STREAM    1/0   18s
40MB  32MB    1.00  0/15000028  0/15000168  OK
```

3. Восстановить с помощью `pg_probackup` только что созданную нами базу данных

```
postgres$ pg_probackup-12 restore --instance 'main5432' -i 'QR3HV7' -D
/var/lib/pgsql/12/main5433 --db-include=restore_one_db --db-include=postgres

INFO: Validating backup QR3HV7
INFO: Backup QR3HV7 data files are valid
INFO: Backup QR3HV7 WAL segments are valid
INFO: Backup QR3HV7 is valid.
INFO: Restoring the database from backup at 2021-04-05 14:35:31+00
INFO: Start restoring backup files. PGDATA size: 72MB
INFO: Backup files are restored. Transfered bytes: 64MB, time elapsed: 0
INFO: Restore incremental ratio (less is better): 89% (64MB/72MB)
INFO: Syncing restored files to disk
INFO: Restored backup files are synced, time elapsed: 11s
INFO: Restore of backup QR3HV7 completed

postgres$ echo port=5433 > /var/lib/pgsql/12/main5433/postgresql.auto.conf

postgres$ pg_ctl -D /var/lib/pgsql/12/main5433 start
waiting for server to start....2021-04-05 14:46:13.380 UTC [6953] LOG:
starting PostgreSQL 12.6 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5
20150623 (Red Hat 4.8.5-44), 64-bit
2021-04-05 14:46:13.381 UTC [6953] LOG:  listening on IPv4 address
"127.0.0.1", port 5433
2021-04-05 14:46:13.384 UTC [6953] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5433"
2021-04-05 14:46:13.392 UTC [6953] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5433"
2021-04-05 14:46:13.406 UTC [6953] LOG:  redirecting log output to logging
collector process
2021-04-05 14:46:13.406 UTC [6953] HINT:  Future log output will appear in
directory "log".
 done
server started

postgres$ psql -p5433
psql (12.6)
Type "help" for help.
5433=> \c dbcopy
FATAL:  "base/24576" is not a valid data directory
DETAIL:  File "base/24576/PG_VERSION" does not contain valid data.
HINT:  You might need to initdb.
Previous connection kept
5433=> drop database dbcopy;
DROP DATABASE
5433=> \c restore_one_db
You are now connected to database "restore_one_db" as user "postgres".
5433=> select * from test limit 10;
 id
----
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
(10 rows)
```