6.2

Создаем стенд

Текущий стенд состоит из 3-ех узлов: `vs01.sXXXXXX`, `vs02.sXXXXXX`, `vs03.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 - **6. Физическая репликация** , 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: Репликации: апгрейд кластера и отказоустойчивые решения

Можете запустить скрипт на стенде `~/practice/lecture3/1.a_sync_replica.sh`, который спикер запускал в видео.

Или пройтись по шагам текущей практики
- https://gitlab.slurm.io/postgres/slurm_course/-/blob/main/practice/lecture3/1.a_sync_replica.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              12.5-1PGDG.rhel7            @pgdg12
postgresql12-contrib.x86_64      12.5-1PGDG.rhel7            @pgdg12
postgresql12-docs.x86_64         12.5-1PGDG.rhel7            @pgdg12
postgresql12-libs.x86_64         12.5-1PGDG.rhel7            @pgdg12
postgresql12-pglogical.x86_64    2.3.3-1.el7                 @barman
postgresql12-server.x86_64       12.5-1PGDG.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.

```
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 12.5 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"
```

Физическая потоковая репликация в синхронном режиме

6. Создаем базу данных, переключаемся на неё и создаем таблицу `test`

```
postgres$ psql
psql (12.5)
Type "help" for help.

5432=> create database synch_replica;
CREATE DATABASE

5432=> \c synch_replica
You are now connected to database "synch_replica" as user "postgres".

5432=> create table test(id int, name text);
CREATE TABLE
```

7. Создаем физический слот репликации synch_slot5433 и synch_slot5434

```
5432=> SELECT pg_create_physical_replication_slot('synch_slot5433');
 pg_create_physical_replication_slot
-------------------------------------
 (synch_slot,)
(1 row)
```

```
5432=> SELECT pg_create_physical_replication_slot('synch_slot5434');
 pg_create_physical_replication_slot
-------------------------------------
 (synch_slot,)
(1 row)
```

8. Проверяем статус репликации

```
5432=> \x
Expanded display is on.

5432=> SELECT * FROM pg_replication_slots;
postgres=# SELECT * FROM pg_replication_slots;
-[ RECORD 1 ]-------+---------------
slot_name           | synch_slot5434
plugin              |
slot_type           | physical
datoid              |
database            |
temporary           | f
active              | f
active_pid          |
xmin                |
catalog_xmin        |
restart_lsn         |
confirmed_flush_lsn |
-[ RECORD 2 ]-------+---------------
slot_name           | synch_slot5433
plugin              |
slot_type           | physical
datoid              |
database            |
temporary           | f
active              | f
active_pid          |
xmin                |
catalog_xmin        |
restart_lsn         |
confirmed_flush_lsn |

5432=> \q
```

9. Создаем реплику на порту 5433

```
postgres$ pg_basebackup -D /var/lib/pgsql/12/main5433 -R -S synch_slot5433 -v
pg_basebackup: initiating base backup, waiting for checkpoint to complete
pg_basebackup: checkpoint completed
pg_basebackup: write-ahead log start point: 0/4000060 on timeline 1
pg_basebackup: starting background WAL receiver
pg_basebackup: write-ahead log end point: 0/4000138
pg_basebackup: waiting for background process to finish streaming ...
pg_basebackup: syncing data to disk ...
pg_basebackup: base backup completed
```

Флаг `-R` позволяет создать потоковую репликацию по слоту `synch_slot5433`

10. Посмотрим на нашу созданную копию

```
postgres$ ls -l /var/lib/pgsql/12/main5433
total 136
drwx------ 20 postgres postgres  4096 Apr 12 06:50 ./
drwx------  6 postgres postgres  4096 Apr 12 06:50 ../
```

```
-rw-------  1 postgres postgres   224 Apr 12 06:50 backup_label
drwx------  6 postgres postgres  4096 Apr 12 06:50 base/
-rw-------  1 postgres postgres    30 Apr 12 06:50 current_logfiles
drwx------  2 postgres postgres  4096 Apr 12 06:50 global/
drwx------  2 postgres postgres  4096 Apr 12 06:50 log/
drwx------  2 postgres postgres  4096 Apr 12 06:50 pg_commit_ts/
drwx------  2 postgres postgres  4096 Apr 12 06:50 pg_dynshmem/
-rw-------  1 postgres postgres  4760 Apr 12 06:50 pg_hba.conf
-rw-------  1 postgres postgres  1636 Apr 12 06:50 pg_ident.conf
drwx------  4 postgres postgres  4096 Apr 12 06:50 pg_logical/
drwx------  4 postgres postgres  4096 Apr 12 06:50 pg_multixact/
drwx------  2 postgres postgres  4096 Apr 12 06:50 pg_notify/
drwx------  2 postgres postgres  4096 Apr 12 06:50 pg_replslot/
drwx------  2 postgres postgres  4096 Apr 12 06:50 pg_serial/
drwx------  2 postgres postgres  4096 Apr 12 06:50 pg_snapshots/
drwx------  2 postgres postgres  4096 Apr 12 06:50 pg_stat/
drwx------  2 postgres postgres  4096 Apr 12 06:50 pg_stat_tmp/
drwx------  2 postgres postgres  4096 Apr 12 06:50 pg_subtrans/
drwx------  2 postgres postgres  4096 Apr 12 06:50 pg_tblspc/
drwx------  2 postgres postgres  4096 Apr 12 06:50 pg_twophase/
-rw-------  1 postgres postgres     3 Apr 12 06:50 PG_VERSION
drwx------  3 postgres postgres  4096 Apr 12 06:50 pg_wal/
drwx------  2 postgres postgres  4096 Apr 12 06:50 pg_xact/
-rw-------  1 postgres postgres   297 Apr 12 06:50 postgresql.auto.conf
-rw-------  1 postgres postgres 26642 Apr 12 06:50 postgresql.conf
-rw-------  1 postgres postgres     0 Apr 12 06:50 standby.signal
```

11. Раздел "standby" перешёл в основной файл конфигурации

```
postgres$ cat /var/lib/pgsql/12/main5433/postgresql.conf | grep -A27 "Standby
Servers"
# - Standby Servers -

# These settings are ignored on a master server.

#primary_conninfo = ''             # connection string to sending server
                    # (change requires restart)
#primary_slot_name = ''             # replication slot on sending server
                    # (change requires restart)
#promote_trigger_file = ''         # file name whose presence ends recovery
#hot_standby = on             # "off" disallows queries during recovery
                    # (change requires restart)
#max_standby_archive_delay = 30s    # max delay before canceling queries
                    # when reading WAL from archive;
                    # -1 allows indefinite delay
#max_standby_streaming_delay = 30s    # max delay before canceling queries
                    # when reading streaming WAL;
                    # -1 allows indefinite delay
#wal_receiver_status_interval = 10s    # send replies at least this often
                    # 0 disables
#hot_standby_feedback = off        # send info from standby to prevent
                    # query conflicts
#wal_receiver_timeout = 60s        # time that receiver waits for
                    # communication from master
                    # in milliseconds; 0 disables
#wal_retrieve_retry_interval = 5s    # time to wait before retrying to
                    # retrieve WAL after a failed attempt
#recovery_min_apply_delay = 0        # minimum delay for applying changes during
recovery
```

12. Для промоута реплики используется файл `standby.signal`. При его удалении реплика переходит в режим лидера:

```
postgres$ ls -l /var/lib/pgsql/12/main5433/standby.signal
-rw------- 1 postgres postgres 0 Apr 12 06:50
/var/lib/pgsql/12/main5433/standby.signal
postgres$ cat /var/lib/pgsql/12/main5433/standby.signal
```

13. Добавим строку с портом 5433 в конфигурационный файл и стартанем сервер PostgreSQL

```
postgres$ echo "port=5433" >> /var/lib/pgsql/12/main5433/postgresql.conf
postgres$ pg_ctl -D /var/lib/pgsql/12/main5433 start
waiting for server to start....2021-04-12 07:09:06.992 UTC [2690] LOG:  starting
PostgreSQL 12.5 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat
4.8.5-44), 64-bit
2021-04-12 07:09:06.992 UTC [2690] LOG:  listening on IPv4 address "127.0.0.1", port
5433
2021-04-12 07:09:06.996 UTC [2690] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5433"
2021-04-12 07:09:07.004 UTC [2690] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5433"
2021-04-12 07:09:07.018 UTC [2690] LOG:  redirecting log output to logging collector
process
2021-04-12 07:09:07.018 UTC [2690] HINT:  Future log output will appear in directory
"log".
 done
server started
```

14. Все то же самое делаем для порта 5434

```
postgres$ pg_basebackup -D /var/lib/pgsql/12/main5434 -R -S synch_slot5434 -v; echo
"port=5434" >> /var/lib/pgsql/12/main5434/postgresql.conf; pg_ctl -D
/var/lib/pgsql/12/main5434 start
pg_basebackup: initiating base backup, waiting for checkpoint to complete
pg_basebackup: checkpoint completed
pg_basebackup: write-ahead log start point: 0/8000028 on timeline 1
pg_basebackup: starting background WAL receiver
pg_basebackup: write-ahead log end point: 0/8000100
pg_basebackup: waiting for background process to finish streaming ...
pg_basebackup: syncing data to disk ...
pg_basebackup: base backup completed
waiting for server to start....2021-04-12 07:14:56.355 UTC [2772] LOG:  starting
PostgreSQL 12.5 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat
4.8.5-44), 64-bit
2021-04-12 07:14:56.355 UTC [2772] LOG:  listening on IPv4 address "127.0.0.1", port
5434
2021-04-12 07:14:56.359 UTC [2772] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5434"
2021-04-12 07:14:56.367 UTC [2772] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5434"
2021-04-12 07:14:56.383 UTC [2772] LOG:  redirecting log output to logging collector
process
2021-04-12 07:14:56.383 UTC [2772] HINT:  Future log output will appear in directory
"log".
 done
server started
```

15. Посмотрим, в каком режиме находится репликация

```
postgres$ psql -d synch_replica
psql (12.5)
Type "help" for help.

5432=> \x
Expanded display is on.
```

```
5432=> select * from pg_stat_replication;
Expanded display is on.
-[ RECORD 1 ]----+------------------------------
pid              | 2769
usesysid         | 10
usename          | postgres
application_name | walreceiver
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2021-04-12 07:14:53.370332+00
backend_xmin     |
state            | streaming
sent_lsn         | 0/E000060
write_lsn        | 0/E000060
flush_lsn        | 0/E000060
replay_lsn       | 0/E000060
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 0
sync_state       | async
reply_time       | 2021-04-12 07:29:16.900444+00
-[ RECORD 2 ]----+------------------------------
pid              | 3239
usesysid         | 10
usename          | postgres
application_name | walreceiver
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2021-04-12 07:27:53.389583+00
backend_xmin     |
state            | streaming
sent_lsn         | 0/E000060
write_lsn        | 0/E000060
flush_lsn        | 0/E000060
replay_lsn       | 0/E000060
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 0
sync_state       | async
reply_time       | 2021-04-12 07:29:53.681008+00

5432=> \q
```

16. Посмотрим статус репликации у реплики 5433

```
postgres$ psql -p5433
5433=> \x
Expanded display is on.

5433=> select * from pg_stat_wal_receiver;
-[ RECORD 1 ]---------+-----------------------------------------------------------------------------------------------------------------------------------
pid                   | 2768
status                | streaming
receive_start_lsn     | 0/B000000
receive_start_tli     | 1
```

```
received_lsn            | 0/E000148
received_tli            | 1
last_msg_send_time      | 2021-04-12 07:41:34.056071+00
last_msg_receipt_time | 2021-04-12 07:41:34.056248+00
latest_end_lsn          | 0/E000148
latest_end_time         | 2021-04-12 07:32:32.78997+00
slot_name               | synch_slot5433
sender_host             | /var/run/postgresql
sender_port             | 5432
conninfo                | user=postgres passfile=/var/lib/pgsql/.pgpass
dbname=replication port=5432 fallback_application_name=walreceiver sslmode=prefer
sslcompression=0 gssencmode=prefer krbsrvname=postgres target_session_attrs=any

5433=> \q
```

17. Давайте теперь поменяем режим репликации с асинхронного на синхронный

```
postgres$ psql -d synch_replica

5432=> ALTER SYSTEM SET synchronous_commit = on;
ALTER SYSTEM
5432=> ALTER SYSTEM SET synchronous_standby_names='FIRST 1 (walreceiver)';
5432=> \q
```

18. Посмотрим наш конфигурационный файл `postgresql.auto.conf`

```
postgres$ cat /var/lib/pgsql/12/main5432/postgresql.auto.conf
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
synchronous_commit = 'on'
synchronous_standby_names = 'FIRST 1 (walreceiver)'
```

19. Перечитаем конфигурацию лидера

```
postgres$ pg_ctl -D /var/lib/pgsql/12/main5432 reload
server signaled
```

20. Проверяем работу репликации

```
postgres$ psql -d synch_replica

5432=> \x
Expanded display is on.

5432=> select * from pg_stat_replication;
-[ RECORD 1 ]----+------------------------------
pid              | 3931
usesysid         | 10
usename          | postgres
application_name | walreceiver
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2021-04-12 08:36:39.629684+00
backend_xmin     |
state            | streaming
sent_lsn         | 0/E000528
write_lsn        | 0/E000528
flush_lsn        | 0/E000528
replay_lsn       | 0/E000528
write_lag        |
flush_lag        |
replay_lag       |
```

```
sync_priority    | 1
sync_state       | sync
reply_time       | 2021-04-12 08:43:29.770396+00
-[ RECORD 2 ]----+------------------------------
pid              | 3932
usesysid         | 10
usename          | postgres
application_name | walreceiver
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2021-04-12 08:36:39.631369+00
backend_xmin     |
state            | streaming
sent_lsn         | 0/E000528
write_lsn        | 0/E000528
flush_lsn        | 0/E000528
replay_lsn       | 0/E000528
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 1
sync_state       | potential
reply_time       | 2021-04-12 08:43:29.770497+00

5432=> \q
```

21. Давайте теперь поставим другой тип применения изменений на репликах

```
5432=> ALTER SYSTEM SET synchronous_standby_names='ANY 1 (walreceiver)';
ALTER SYSTEM
```

22. Перечитаем конфигурацию

```
5432=>  select pg_reload_conf();
-[ RECORD 1 ]--+--
pg_reload_conf | t

5432=> select * from pg_stat_replication;
-[ RECORD 1 ]----+------------------------------
pid              | 3931
usesysid         | 10
usename          | postgres
application_name | walreceiver
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2021-04-12 08:36:39.629684+00
backend_xmin     |
state            | streaming
sent_lsn         | 0/E000528
write_lsn        | 0/E000528
flush_lsn        | 0/E000528
replay_lsn       | 0/E000528
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 1
sync_state       | quorum
reply_time       | 2021-04-12 08:44:49.923201+00
-[ RECORD 2 ]----+------------------------------
pid              | 3932
usesysid         | 10
```

```
usename          | postgres
application_name | walreceiver
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2021-04-12 08:36:39.631369+00
backend_xmin     |
state            | streaming
sent_lsn         | 0/E000528
write_lsn        | 0/E000528
flush_lsn        | 0/E000528
replay_lsn       | 0/E000528
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 1
sync_state       | quorum
reply_time       | 2021-04-12 08:44:49.923109+00

5432=> \q
```

23. Но можно ещё повысить отказоустойчивость

```
5432=> ALTER SYSTEM SET synchronous_standby_names='FIRST 2 (walreceiver)';
ALTER SYSTEM
5432=> select pg_reload_conf();
-[ RECORD 1 ]--+--
pg_reload_conf | t

5432=> select * from pg_stat_replication;
-[ RECORD 1 ]----+------------------------------
pid              | 3931
usesysid         | 10
usename          | postgres
application_name | walreceiver
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2021-04-12 08:36:39.629684+00
backend_xmin     |
state            | streaming
sent_lsn         | 0/E000528
write_lsn        | 0/E000528
flush_lsn        | 0/E000528
replay_lsn       | 0/E000528
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 1
sync_state       | sync
reply_time       | 2021-04-12 08:50:30.603736+00
-[ RECORD 2 ]----+------------------------------
pid              | 3932
usesysid         | 10
usename          | postgres
application_name | walreceiver
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2021-04-12 08:36:39.631369+00
backend_xmin     |
state            | streaming
```

```
sent_lsn          | 0/E000528
write_lsn         | 0/E000528
flush_lsn         | 0/E000528
replay_lsn        | 0/E000528
write_lag         |
flush_lag         |
replay_lag        |
sync_priority     | 1
sync_state        | sync
reply_time        | 2021-04-12 08:50:30.603825+00

5432=> \q
```

24. Давайте теперь протестируем наш отказоустойчивый кластер. Остановим кластер на порту 5433

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

25. Подключимся к серверу на порту 5432 и попробуем вставить данные в таблицу

```
postgres$ psql -d synch_replica
5432=> insert into test(id,name) select 1, 'name1';
```

25. Видим, что вставка не может закончиться. Срубим операцию

```
^CCancel request sent
WARNING:  canceling wait for synchronous replication due to user request
DETAIL:  The transaction has already committed locally, but might not have been
replicated to the standby.
INSERT 0 1
```

26. Посмотрим нашу таблицу test

```
5433=> select * from test;
 id | name
----+------
  1 | name1
(1 rows)
```

Данные в таблице есть. Но есть и предупреждение, что изменения не применились на всех репликах

27. Давайте теперь понизим статус отказоустойчивости кластера

```
5432=> ALTER SYSTEM SET synchronous_standby_names='ANY 1 (walreceiver)';
ALTER SYSTEM
5432=> select pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 row)
```

28. И проверим работу синхронной репликации

```sql
5432=> insert into test(id,name) select 2, 'name2';
INSERT 0 1
5432 => select * from test ;
 id | name
----+-------
```

```
  1 | name1
  2 | name2
(2 rows)
```

29. Запустим сервер на порту 5433

```
5432=> \q

postgres$ pg_ctl -D /var/lib/pgsql/12/main5433 start
waiting for server to start....2021-04-12 08:57:25.410 UTC [4166] LOG:  starting
PostgreSQL 12.5 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat
4.8.5-44), 64-bit
2021-04-12 08:57:25.410 UTC [4166] LOG:  listening on IPv4 address "127.0.0.1", port
5433
2021-04-12 08:57:25.414 UTC [4166] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5433"
2021-04-12 08:57:25.422 UTC [4166] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5433"
2021-04-12 08:57:25.437 UTC [4166] LOG:  redirecting log output to logging collector
process
2021-04-12 08:57:25.437 UTC [4166] HINT:  Future log output will appear in directory
"log".
 done
server started
postgres$ psql -p5433 -d synch_replica
psql (12.5)
Type "help" for help.

5433=> select * from test ;
 id | name
----+-------
  1 | name1
  2 | name2
(2 rows)

5432=> \q
```

Все записи пришли на сервер 5433.

6.3

Домашнее задание. Синхронная реплика

1. Создать синхронную репликацию на 5 серверов PostgreSQL
2. И с помощью тестов pgbench (запустить тесты на 2-3 минуты) замерить скорость работы репликации при указании в параметре synchronous_standby_names фиксации на любом одном сервере PostgreSQL.
3. И с помощью тестов pgbench (запустить тесты на 2-3 минуты) замерить скорость работы репликации при фиксации изменений на всех репликах

Какие выводы можно сделать из данного теста?

Чек-лист

1. Создана синхронная репликация на 5 серверов PostgreSQL.
2. И с помощью тестов pgbench измерена скорость работы репликации при фиксации изменений на всех серверах и при фиксации изменений на одном из серверов

Решение по ссылке в конце файла

Решение:

1. Создать синхронную репликацию на 5 серверов PostgreSQL

```
postgres$ pg_ctl -D /var/lib/pgsql/12/main5434 stop

waiting for server to shut down.... done
server stopped
postgres$ pg_ctl -D /var/lib/pgsql/12/main5433 stop
waiting for server to shut down.... done
server stopped
postgres$ pg_ctl -D /var/lib/pgsql/12/main5432 stop
waiting for server to shut down.... done
server stopped
postgres$ rm -rf /var/lib/pgsql/12/main5432

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

postgres$ rm -rf /var/lib/pgsql/12/main5434

postgres$ rm -rf /var/lib/pgsql/12/main5435

postgres$ ls -la ~/12/

total 16
drwx------ 3 postgres postgres 4096 Apr 24 09:15 ./
drwx------ 4 postgres postgres 4096 Apr 24 08:59 ../
```

```
drwx------ 2 postgres postgres 4096 Feb 11 01:16 backups/
-rw------- 1 postgres postgres  911 Apr 24 07:34 initdb.log
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

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

waiting for server to start....2021-04-24 09:15:56.458 UTC [23657] 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
2021-04-24 09:15:56.458 UTC [23657] LOG:  listening on IPv4 address
"127.0.0.1", port 5432
2021-04-24 09:15:56.462 UTC [23657] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
2021-04-24 09:15:56.471 UTC [23657] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5432"
2021-04-24 09:15:56.494 UTC [23657] LOG:  redirecting log output to logging
collector process
2021-04-24 09:15:56.494 UTC [23657] HINT:  Future log output will appear in
directory "log".
 done
server started
postgres$ psql

psql (12.6)
Type "help" for help.

5432=> create database synch_replica;

CREATE DATABASE

5432=> \c synch_replica

You are now connected to database "synch_replica" as user "postgres".

5432=> create table test(id int, name text);
```

```
CREATE TABLE


5432=> SELECT pg_create_physical_replication_slot('synch_slot5433');

 pg_create_physical_replication_slot
-------------------------------------
 (synch_slot5433,)
(1 row)

5432=> SELECT pg_create_physical_replication_slot('synch_slot5434');

 pg_create_physical_replication_slot
-------------------------------------
 (synch_slot5434,)
(1 row)

5432=> SELECT pg_create_physical_replication_slot('synch_slot5435');

 pg_create_physical_replication_slot
-------------------------------------
 (synch_slot5435,)
(1 row)

5432=> SELECT pg_create_physical_replication_slot('synch_slot5436');

 pg_create_physical_replication_slot
-------------------------------------
 (synch_slot5436,)
(1 row)


synch_replica=# \q
postgres$ pg_basebackup -D /var/lib/pgsql/12/main5434 -R -S synch_slot5434 -
v; echo "port=5434" >> /var/lib/pgsql/12/main5434/postgresql.conf; pg_ctl -D
/var/lib/pgsql/12/main5434 start

pg_basebackup: initiating base backup, waiting for checkpoint to complete
pg_basebackup: checkpoint completed
pg_basebackup: write-ahead log start point: 0/2000028 on timeline 1
pg_basebackup: starting background WAL receiver
pg_basebackup: write-ahead log end point: 0/2000138
pg_basebackup: waiting for background process to finish streaming ...
pg_basebackup: syncing data to disk ...
pg_basebackup: base backup completed
waiting for server to start....2021-04-24 09:16:57.046 UTC [23681] 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
2021-04-24 09:16:57.046 UTC [23681] LOG:  listening on IPv4 address
"127.0.0.1", port 5434
2021-04-24 09:16:57.050 UTC [23681] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5434"
2021-04-24 09:16:57.058 UTC [23681] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5434"
2021-04-24 09:16:57.079 UTC [23681] LOG:  redirecting log output to logging
collector process
2021-04-24 09:16:57.079 UTC [23681] HINT:  Future log output will appear in
directory "log".
 done
server started
```

```
postgres$ pg_basebackup -D /var/lib/pgsql/12/main5433 -R -S synch_slot5433 -
v; echo "port=5433" >> /var/lib/pgsql/12/main5433/postgresql.conf; pg_ctl -D
/var/lib/pgsql/12/main5433 start

pg_basebackup: initiating base backup, waiting for checkpoint to complete
pg_basebackup: checkpoint completed
pg_basebackup: write-ahead log start point: 0/4000028 on timeline 1
pg_basebackup: starting background WAL receiver
pg_basebackup: write-ahead log end point: 0/4000100
pg_basebackup: waiting for background process to finish streaming ...
pg_basebackup: syncing data to disk ...
pg_basebackup: base backup completed
waiting for server to start....2021-04-24 09:17:31.688 UTC [23701] 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
2021-04-24 09:17:31.688 UTC [23701] LOG:  listening on IPv4 address
"127.0.0.1", port 5433
2021-04-24 09:17:31.692 UTC [23701] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5433"
2021-04-24 09:17:31.707 UTC [23701] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5433"
2021-04-24 09:17:31.733 UTC [23701] LOG:  redirecting log output to logging
collector process
2021-04-24 09:17:31.733 UTC [23701] HINT:  Future log output will appear in
directory "log".
 done
server started



postgres$ pg_basebackup -D /var/lib/pgsql/12/main5435 -R -S synch_slot5435 -
v; echo "port=5435" >> /var/lib/pgsql/12/main5435/postgresql.conf; pg_ctl -D
/var/lib/pgsql/12/main5435 start

pg_basebackup: initiating base backup, waiting for checkpoint to complete
pg_basebackup: checkpoint completed
pg_basebackup: write-ahead log start point: 0/6000028 on timeline 1
pg_basebackup: starting background WAL receiver
pg_basebackup: write-ahead log end point: 0/6000100
pg_basebackup: waiting for background process to finish streaming ...
pg_basebackup: syncing data to disk ...
pg_basebackup: base backup completed
waiting for server to start....2021-04-24 09:18:00.707 UTC [23716] 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
2021-04-24 09:18:00.707 UTC [23716] LOG:  listening on IPv4 address
"127.0.0.1", port 5435
2021-04-24 09:18:00.727 UTC [23716] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5435"
2021-04-24 09:18:00.768 UTC [23716] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5435"
2021-04-24 09:18:00.817 UTC [23716] LOG:  redirecting log output to logging
collector process
2021-04-24 09:18:00.817 UTC [23716] HINT:  Future log output will appear in
directory "log".
 done
server started



postgres$ pg_basebackup -D /var/lib/pgsql/12/main5436 -R -S synch_slot5436 -
v; echo "port=5436" >> /var/lib/pgsql/12/main5436/postgresql.conf; pg_ctl -D
/var/lib/pgsql/12/main5436 start

pg_basebackup: initiating base backup, waiting for checkpoint to complete
```

```
pg_basebackup: checkpoint completed
pg_basebackup: write-ahead log start point: 0/8000028 on timeline 1
pg_basebackup: starting background WAL receiver
pg_basebackup: write-ahead log end point: 0/8000100
pg_basebackup: waiting for background process to finish streaming ...
pg_basebackup: syncing data to disk ...
pg_basebackup: base backup completed
waiting for server to start....2021-04-24 09:18:24.915 UTC [23735] 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
2021-04-24 09:18:24.915 UTC [23735] LOG:  listening on IPv4 address
"127.0.0.1", port 5436
2021-04-24 09:18:24.942 UTC [23735] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5436"
2021-04-24 09:18:24.998 UTC [23735] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5436"
2021-04-24 09:18:25.041 UTC [23735] LOG:  redirecting log output to logging
collector process
2021-04-24 09:18:25.041 UTC [23735] HINT:  Future log output will appear in
directory "log".
 done
server started
postgres$ ps -ef | grep postgres

root     17520 29852  0 08:57 pts/0    00:00:00 sudo -u postgres -i
postgres 17528 17520  0 08:57 pts/0    00:00:00 -bash
postgres 23657     1  0 09:15 ?        00:00:00 /usr/pgsql-12/bin/postgres -D
/var/lib/pgsql/12/main5432
postgres 23658 23657  0 09:15 ?        00:00:00 postgres: logger
postgres 23660 23657  0 09:15 ?        00:00:00 postgres: checkpointer
postgres 23661 23657  0 09:15 ?        00:00:00 postgres: background writer
postgres 23662 23657  0 09:15 ?        00:00:00 postgres: walwriter
postgres 23663 23657  0 09:15 ?        00:00:00 postgres: autovacuum launcher
postgres 23664 23657  0 09:15 ?        00:00:00 postgres: stats collector
postgres 23665 23657  0 09:15 ?        00:00:00 postgres: logical replication
launcher
postgres 23681     1  0 09:16 ?        00:00:00 /usr/pgsql-12/bin/postgres -D
/var/lib/pgsql/12/main5434
postgres 23682 23681  0 09:16 ?        00:00:00 postgres: logger
postgres 23683 23681  0 09:16 ?        00:00:00 postgres: startup
recovering 000000010000000000000009
postgres 23684 23681  0 09:16 ?        00:00:00 postgres: checkpointer
postgres 23685 23681  0 09:16 ?        00:00:00 postgres: background writer
postgres 23686 23681  0 09:16 ?        00:00:00 postgres: stats collector
postgres 23687 23681  0 09:16 ?        00:00:00 postgres: walreceiver
streaming 0/9000060
postgres 23688 23657  0 09:16 ?        00:00:00 postgres: walsender postgres
[local] streaming 0/9000060
postgres 23701     1  0 09:17 ?        00:00:00 /usr/pgsql-12/bin/postgres -D
/var/lib/pgsql/12/main5433
postgres 23702 23701  0 09:17 ?        00:00:00 postgres: logger
postgres 23703 23701  0 09:17 ?        00:00:00 postgres: startup
recovering 000000010000000000000009
postgres 23704 23701  0 09:17 ?        00:00:00 postgres: checkpointer
postgres 23705 23701  0 09:17 ?        00:00:00 postgres: background writer
postgres 23706 23701  0 09:17 ?        00:00:00 postgres: stats collector
postgres 23707 23701  0 09:17 ?        00:00:00 postgres: walreceiver
streaming 0/9000060
postgres 23708 23657  0 09:17 ?        00:00:00 postgres: walsender postgres
[local] streaming 0/9000060
postgres 23716     1  0 09:17 ?        00:00:00 /usr/pgsql-12/bin/postgres -D
/var/lib/pgsql/12/main5435
postgres 23717 23716  0 09:17 ?        00:00:00 postgres: logger
```

```
postgres 23718 23716  0 09:18 ?        00:00:00 postgres: startup
recovering 000000010000000000000009
postgres 23719 23716  0 09:18 ?        00:00:00 postgres: checkpointer
postgres 23720 23716  0 09:18 ?        00:00:00 postgres: background writer
postgres 23721 23716  0 09:18 ?        00:00:00 postgres: stats collector
postgres 23722 23716  0 09:18 ?        00:00:00 postgres: walreceiver
streaming 0/9000060
postgres 23723 23657  0 09:18 ?        00:00:00 postgres: walsender postgres
[local] streaming 0/9000060
postgres 23735     1  0 09:18 ?        00:00:00 /usr/pgsql-12/bin/postgres -D
/var/lib/pgsql/12/main5436
postgres 23736 23735  0 09:18 ?        00:00:00 postgres: logger
postgres 23737 23735  0 09:18 ?        00:00:00 postgres: startup
recovering 000000010000000000000009
postgres 23738 23735  0 09:18 ?        00:00:00 postgres: checkpointer
postgres 23739 23735  0 09:18 ?        00:00:00 postgres: background writer
postgres 23740 23735  0 09:18 ?        00:00:00 postgres: stats collector
postgres 23741 23735  0 09:18 ?        00:00:00 postgres: walreceiver
streaming 0/9000060
postgres 23742 23657  0 09:18 ?        00:00:00 postgres: walsender postgres
[local] streaming 0/9000060
postgres 23744 17528  0 09:18 pts/0    00:00:00 ps -ef
postgres 23745 17528  0 09:18 pts/0    00:00:00 grep --color=auto postgres
```

2. И с помощью тестов `pgbench` (запустить тесты на 2-3 минуты) замерить скорость работы репликации при указании в параметре `synchronous_standby_names` фиксации на любом одном сервере PostgreSQL.

```
postgres$ psql -d synch_replica

psql (12.6)
Type "help" for help.

5432=> ALTER SYSTEM SET synchronous_commit = on;
ALTER SYSTEM
5432=> ALTER SYSTEM SET synchronous_standby_names='ANY 1 (walreceiver)';
ALTER SYSTEM
5432=> select pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 row)
5432=> \q
postgres$ pgbench -i -d synch_replica

dropping old tables...
NOTICE:  table "pgbench_accounts" does not exist, skipping
NOTICE:  table "pgbench_branches" does not exist, skipping
NOTICE:  table "pgbench_history" does not exist, skipping
NOTICE:  table "pgbench_tellers" does not exist, skipping
creating tables...
generating data...
100000 of 100000 tuples (100%) done (elapsed 0.14 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done.


[postgres@vs01.s000000.slurm.io ~]$ pgbench -c 5 -j 4 -T 120 synch_replica

starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 1
```

```
query mode: simple
number of clients: 5
number of threads: 4
duration: 120 s
number of transactions actually processed: 7407
latency average = 81.053 ms
tps = 61.688351 (including connections establishing)
tps = 61.693920 (excluding connections establishing)
```

3. И с помощью тестов `pgbench` (запустить тесты на 2-3 минуты) замерить скорость работы репликации при фиксации изменений на всех репликах

```
postgres$ psql

psql (12.6)
Type "help" for help.

5432=> ALTER SYSTEM SET synchronous_standby_names='FIRST 4 (walreceiver)';

ALTER SYSTEM
5432=> select pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 row)
5432=> \q
postgres$ pgbench -i -d synch_replica

dropping old tables...
creating tables...
generating data...
100000 of 100000 tuples (100%) done (elapsed 0.15 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done.
postgres$ pgbench -c 5 -j 4 -T 120 synch_replica
starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 1
query mode: simple
number of clients: 5
number of threads: 4
duration: 120 s
number of transactions actually processed: 7564
latency average = 79.366 ms
tps = 62.998953 (including connections establishing)
tps = 63.003562 (excluding connections establishing)
```