Запись с потоковой репликацией

Сначала надо настроить синхронную кворумную репликацию (см. документ)

Далее:

На мастере была создана тестовая таблица users\_posts:

CREATE TABLE IF NOT EXISTS public.users\_posts

(

id bigint NOT NULL DEFAULT nextval('users\_posts\_id\_seq'::regclass),

id\_user bigint NOT NULL,

post character varying COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT users\_posts\_pkey PRIMARY KEY (id, id\_user, post)

)

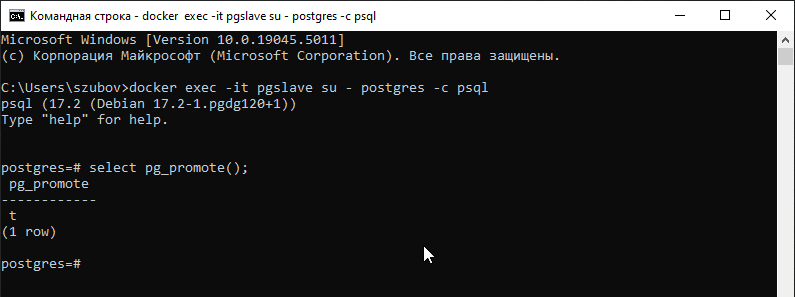
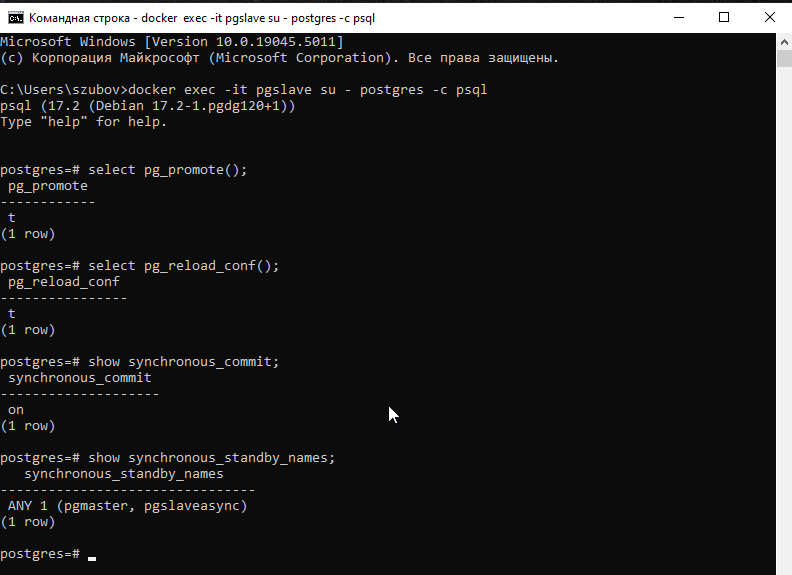
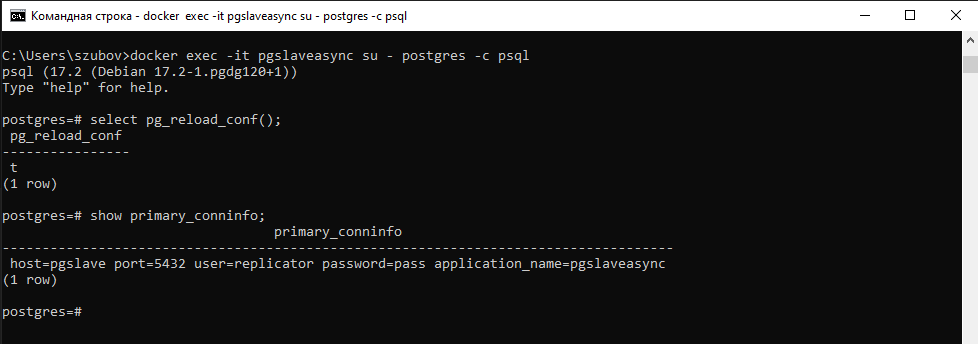
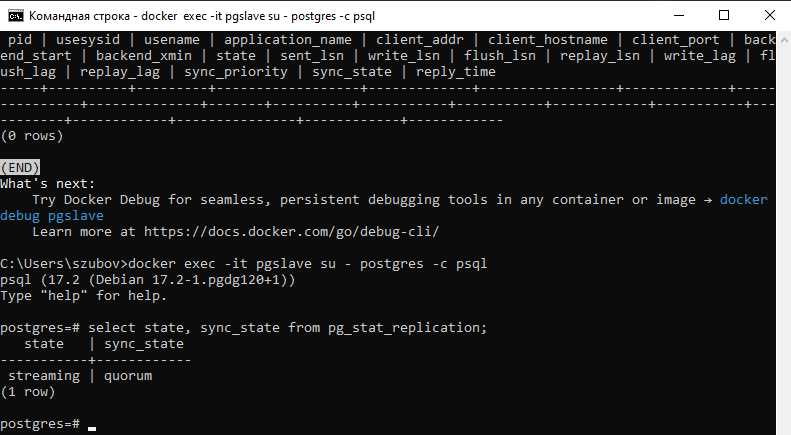
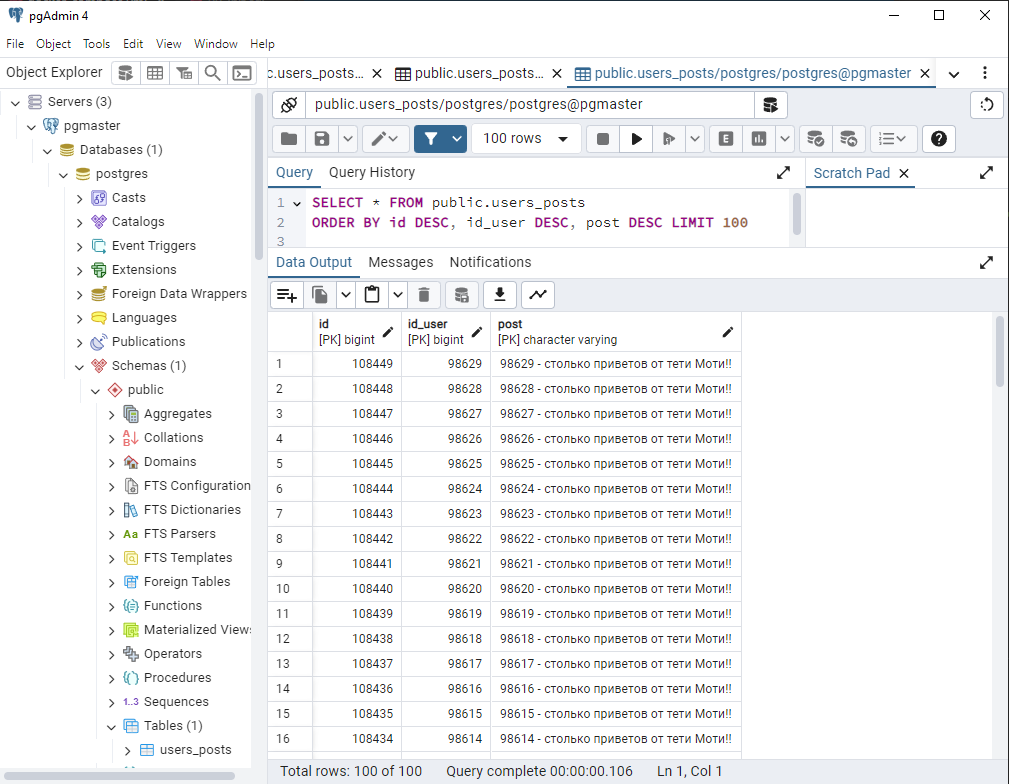
Таблица тут же появилась на репликах.

Далее с помощью скрипта Python на мастер (pgmaster) была создана нагрузка на запись 100000 строк постов в таблицу users\_posts и где-то на записи 56000 поста была остановлена реплика pgslaveasync (docker stop pgslaveasync), при этом запись в мастер продолжалась до 100000 постов.

Проверкой количества записей в таблице (select count(\*) from users\_posts) на серверах pgmaster и pgslave было установлено, что количество записей единакого и равно 100000.

После этого была запущена реплика pgslaveasync (docker pgslaveasync start) и проверено количество записей в таблице users\_posts, которое также оказалось равным 100000 и пробелов или перепутывания строк не наблюдалось.

Далее для проверки переключения мастера на слейв были проделаны следующие действия:

* остановлен мастер (docker stop pgmaster)
* выполнена команда
  + docker exec -it pgslave su – postgres -c psql
  + select pg\_promote();
* перезаписан файл postgresql.conf прямо в редакторе из volumes:
  + synchronous\_commit = on
  + synchronous\_standbynames = ‘select pg\_reload\_conf();’
* перечитан конфиг:
  + docker exec -it pgslave su – postgres -c psql
  + select pg\_reload\_conf();
* проверена настройка на pgslave:
  + show synchronous\_commit
  + show synchronous\_standbynames 
* далее переключаем реплику pgslaveasync к новому мастеру pgslave:
  + изменяем файл postgresql.conf прямо в редакторе из volumes
  + перечитываем конфиг на pgslaveasync
    - docker exec -it pgslaveasync su – postgres -c psql
    - select pg\_reload\_conf();
    - Проверка настроек – show primary\_conninfo;
  + почему-то настройки не применяются, остается тот же мастер – pgmaster
  + поэтому перезапускаем контейнер pgslaveasync
  + убеждаемся, что настройки применились: 
  + проверяем, что pgslaveasync подключился как реплика к новому мастеру pgslave: 
* теперь надо переключить pgmaster, т.е. сделать его репликой pgslave:
  + создаем в редакторе файл standby.signal без данных и сохраняем его в каталоге volumes для бывшего мастера
  + изменяем файл postgresql.conf прямо в редакторе из volumes: primary\_conninfo = ‘host=pgslave port=5432 user=replicator password=pass application\_name=pgmaster’
  + запускаем pgmaster docker start pgmaster
  + убеждаемся, что данные в таблице users\_posts не потерялись
* запускаем скрипт нагрузки, но уже на новый мастер – pgslave
  + проверяем, что новые записи в новый мастер pgslave добавились
  + проверяем, что новые записи в новую реплику pgmaster добавились
  + проверяем, что новые записи в старую реплику pgslaveasync добавились
* Делаем вывод – переключения прошли успешно и данные сохранились! 

Выводы: настройка синхронной репликации в PostgreSQL и переключение на новый мастер при отказе старого - нетривиальная задача, если ее делать вручную.

Также переключение на новый мастер должно происходить практически мгновенно для того, чтобы не потерять данные (тут можно использовать очереди). И нужно перенастраивать приложение на новый мастер (балансировщик?), но, вроде, все работает. Нужно использовать дополнительные инструменты типа Patroni для автоматического failover.