# ДЗ «Полусинхронная репликация»

1. Настраиваем ассинхронную репликацию:

а) Создан новый docker-compose файл на основе текущего для проекта социальной сети

б) В файл добавлены секции для еще двух контейнеров pg (с пустой БД) (tordb, lokidb)

Получили конфигурацию из трех СУБД:

pg\_odin – существующая БД, будет первичным мастером

pg\_tor, pg\_loki – новые пустые БД, будут слейвами

в) В настройках pg\_odin

I) Добавили пользователя для репликации:

create role replicator with login replication password 'pass';

II) В файле postgresql.conf раскоментировал параметры ведения wal:  
 ssl = off

wal\_level = replica

max\_wal\_senders = 10

III) Разрешили любую подсеть для репликации в конфиге pg\_hba.conf (т.к. каждый рестар стенда может создать новую подсеть):

host replication replicator all md5

IV) Создали backup данных для переноса на слевы:

pg\_basebackup -h pg\_odin -D ./pgslave\_bcp -U replicator -v -P --wal-method=stream

IIV) В созданном бекапе:

Создали сигнальный файл: standby.signal

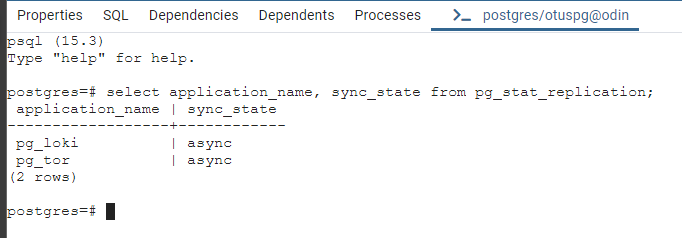
В конфиге прописали строку подключения для каждого слева (будем раскоменчивать необходимый):

#primary\_conninfo = 'host=pg\_odin port=5432 user=replicator password=pass application\_name=pg\_tor'

#primary\_conninfo = 'host=pg\_odin port=5432 user=replicator password=pass application\_name=pg\_loki'

г) Остановили все PG, перенесли бекап данных в слейвы и раскоментили необходимые строки подключения к мастеру

д) Убедились, что все слевый в асинхронном режиме:



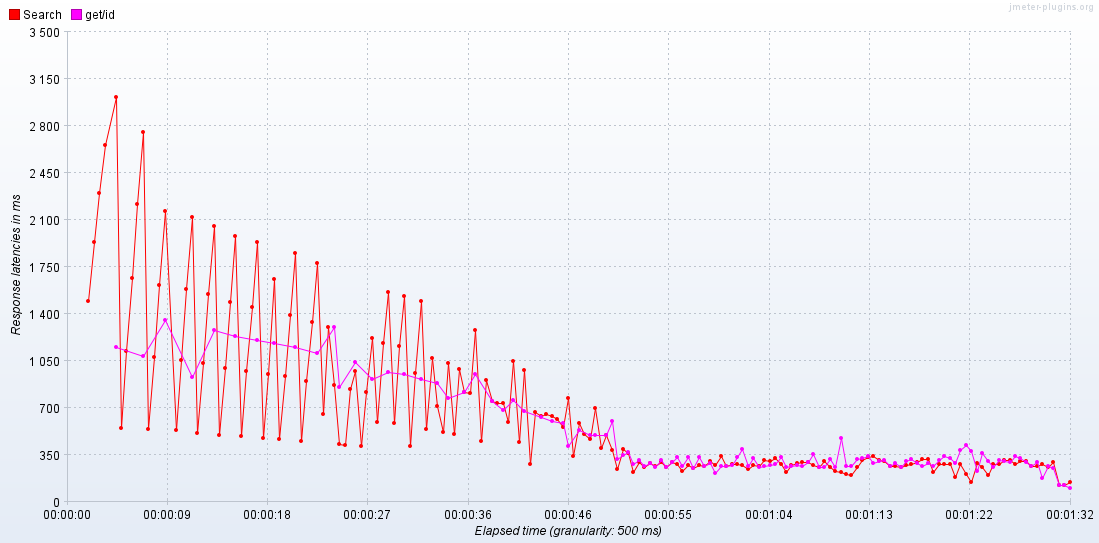
1. Переключаем методы get и search на один из слейвов (pg\_tor):

Для переключение будут использоваться pg\_odin как master на котором работают все методы сервиса. pg\_tor – слейв, на который будем переключать методы.

Выполняем нагрузочный тест методов get и search анкет до переключения на слейв:

Jmeter: по 1 запросу каждого метода, 5 (параллельных потоков), 100 повторений.

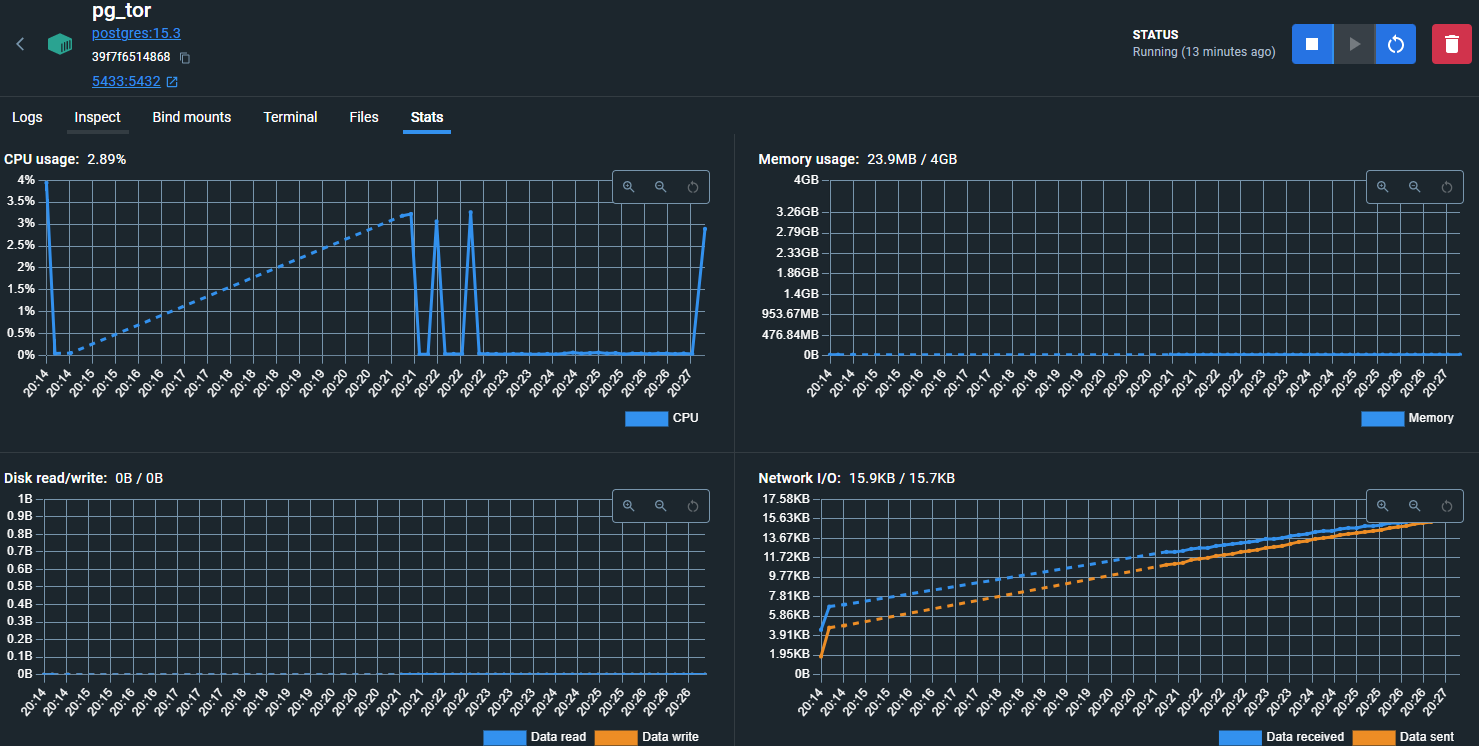
Latency:



pg\_odin (docker stats):



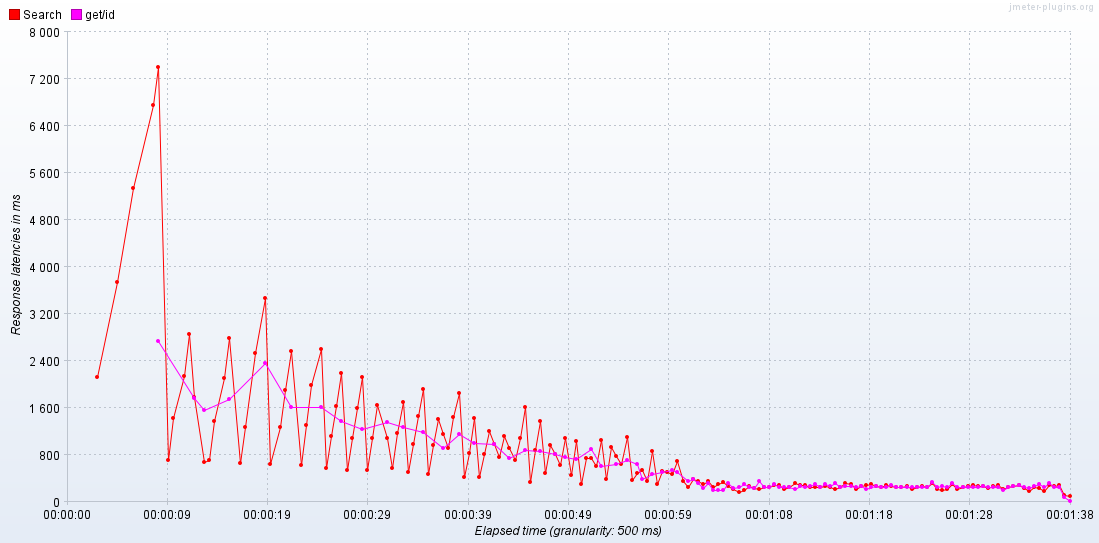
pg\_tor (docker stats):



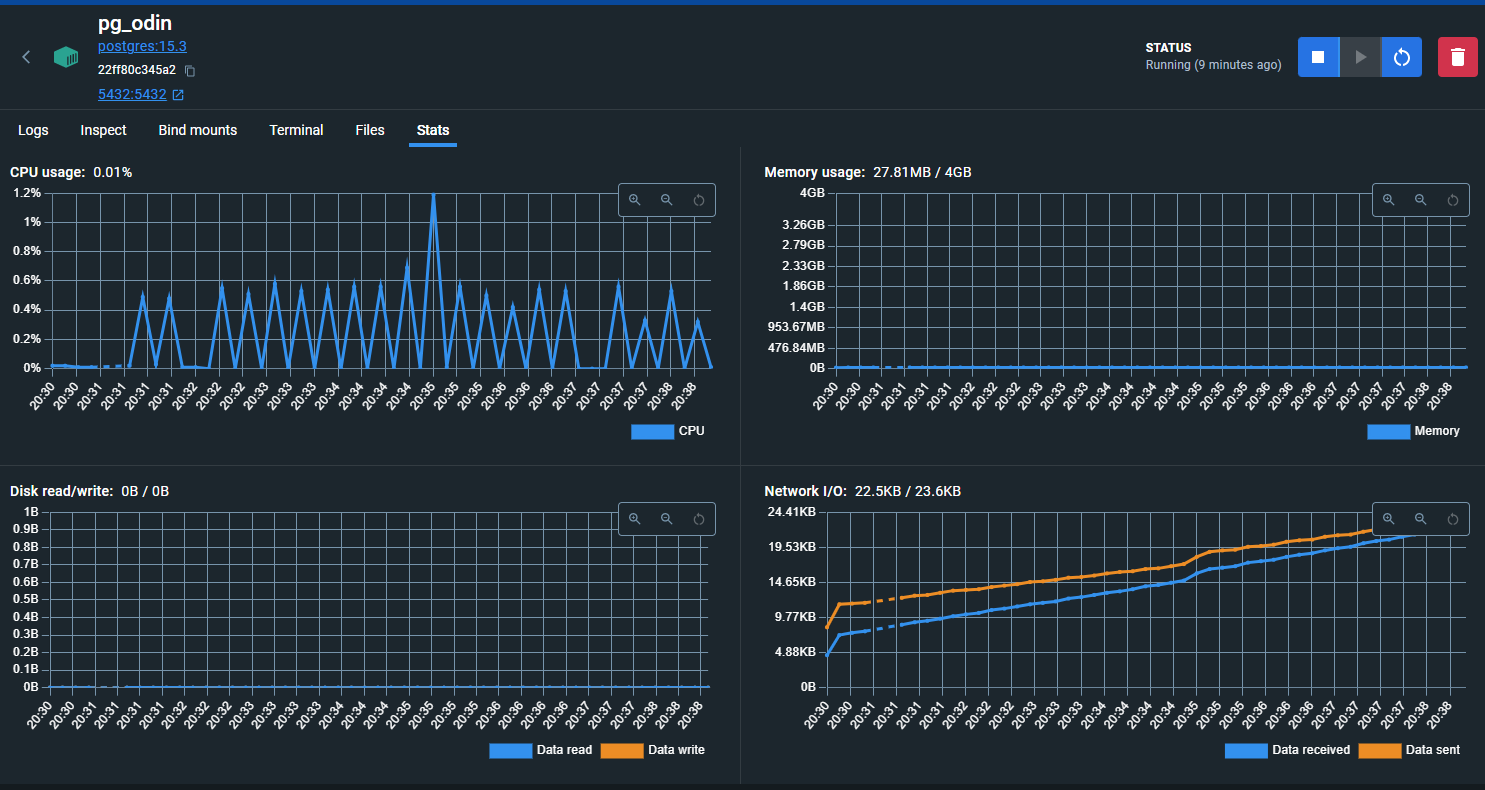
Б) В основном модуле проекта, добавляем второе подключение (объект класса SocialDB) db\_slave и в обработке запросов get и search меняем вызов методов из объекта db\_master на объект db\_slave

Перезапускаем сервисы и проводим такую же нагрузку:

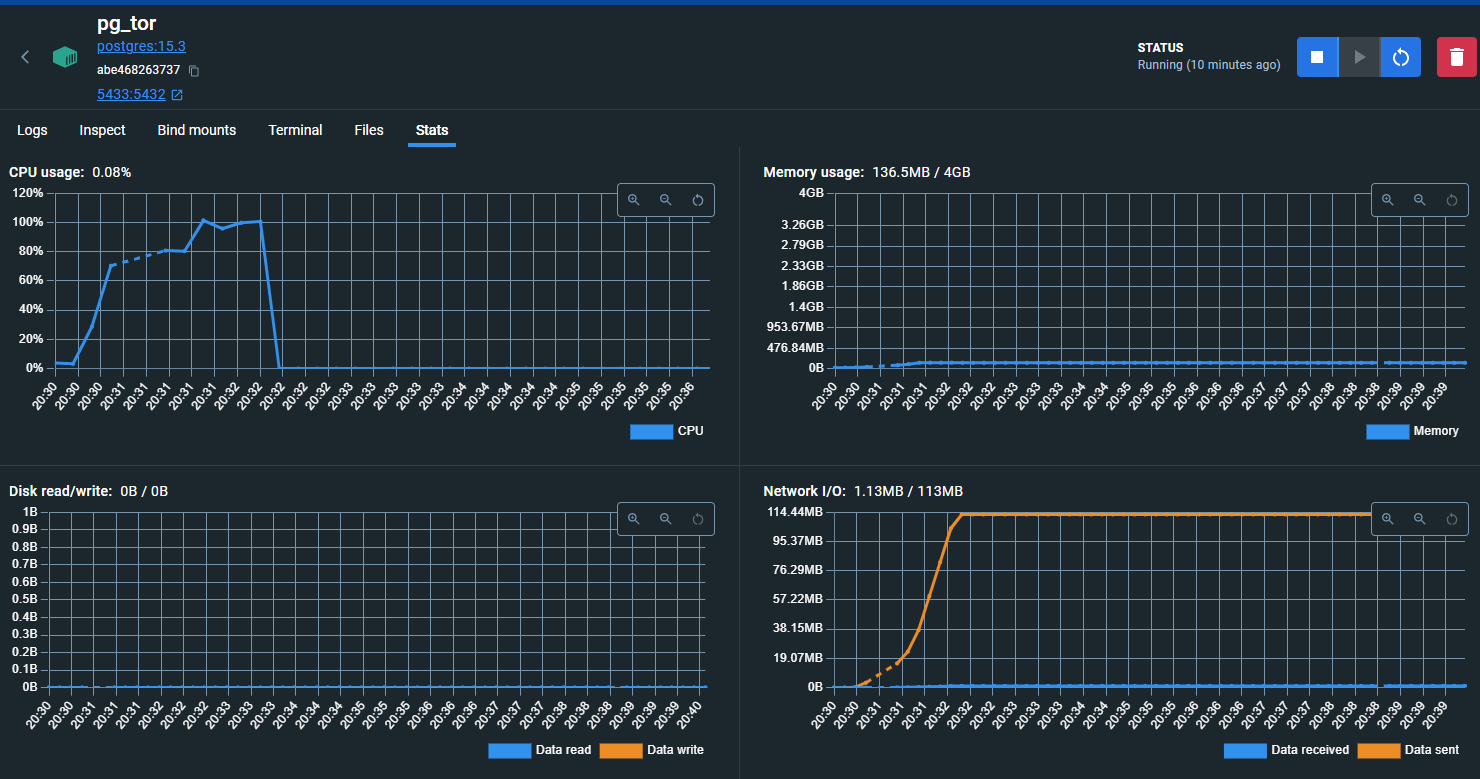
Latency:



pg\_odin:



pg\_tor:



Статистика БД показывается, что нагрузка на метода перешла на слейв.

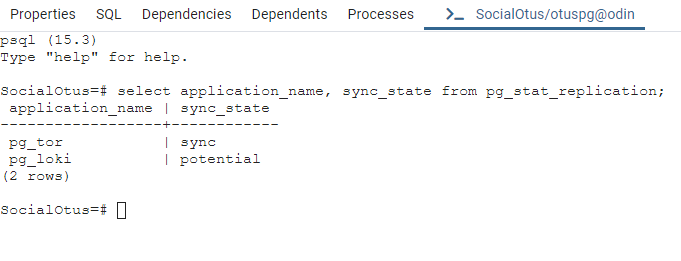
3) Включаем потоковую синхронную репликацию.

- добавляем в конфигурацию мастера параметры синхронной репликации:

synchronous\_commit = on

synchronous\_standby\_names = 'FIRST 1 (pg\_tor, pg\_loki)'

- перезапускаем сервисы и проверяем статус:



Включена синхронная репликация.

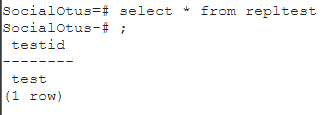
- создаем таблицу для тестирования и проверяем репликацию записи:

CREATE TABLE repltest (testid text NOT NULL UNIQUE);

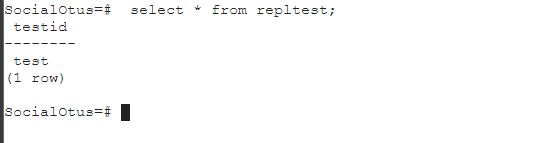
insert into repltest (testid) VALUES ('test');

select \* from repltest

pg\_odin:



pg\_tor:



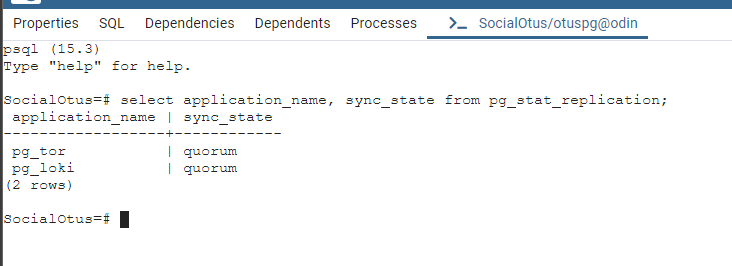
Данные реплицированы.

1. Настраиваем кворумную репликацию:

На мастере изменим в конфиге строку:

synchronous\_standby\_names = 'ANY 1 (pg\_tor, pg\_loki)'

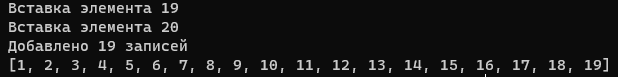
* Перезапускаем и проверяем работу синхронизации:



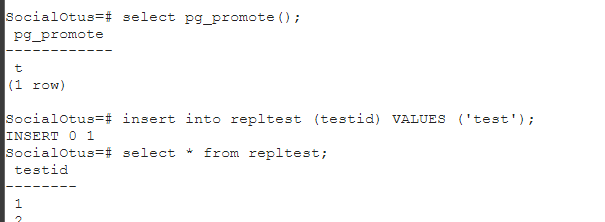
Для создания нагрузки на запись в тестовую таблицу написан python скрипт, который подключается к мастеру и каждые 0,5 секунд записывает в таблицу очередное значение, при обрыве связи в консоль выводит информацию о количестве записанных элементов и массив с успешно записанными элементами. Код сохранен в файле replication\_test.py.

- Запускаем нагрузку на запись и подождав 5-10 сек убиваем мастер (docker stop):

Вывод скрипта после потери связи до мастера:



- промоутим pg\_tor до мастера и проверяем что запись на нем работает:



И включаем репликацию, для этого правим в конфиге:

synchronous\_commit = on

synchronous\_standby\_names = 'ANY 1 (pg\_loki, pg\_odin)'

перечитываем конфиг^

psql select pg\_reload\_conf();

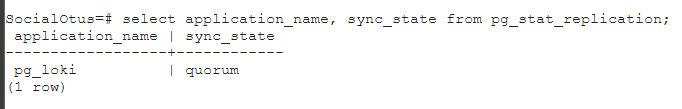
- переключаем pg\_loki на новый мастер pg\_tor, для этого изменяем в конфиге строку:

primary\_conninfo = 'host=pg\_tor port=5432 user=replicator password=pass application\_name=pg\_loki'

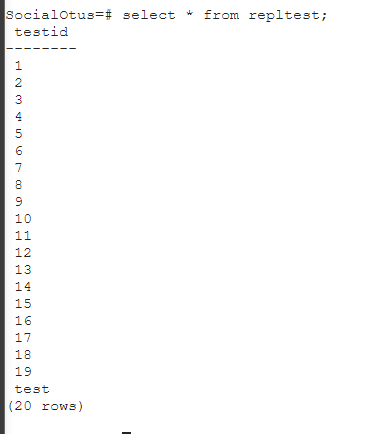
перечитываем конфиг^

psql select pg\_reload\_conf();

- проверяем, что репликация работает:



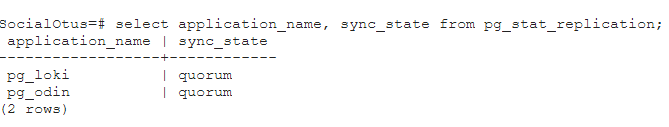
- проверяем потерю транзакция запросом на pg\_loki:



Потерь транзакций при потери мастера не произошло за счет использования синхронной репликации.

* Востанавливаем pg\_odin как слейв в репликации, для этого:
  + Добавим пустой файл в каталог pgdata standby.signal
  + Добавляем строчку подключения к новому мастеру в конфиг:

primary\_conninfo = 'host=pg\_tor port=5432 user=replicator password=pass application\_name=pg\_odin'



Кворумная репликация 1 мастер – 2 слейва полностью востановлена.