Санкт-Петербургский национально исследовательский университет

информационных технологий, механики и оптики

Факультет программной инженерии и компьютерной техники



**Лабораторная работа № 4 по дисциплине**

**«РСХД»**

Вариант 11.

Выполнили:

Студенты группы P33131

Ибадуллаев Алибаба

Осипов Василий

Преподаватель: Шешуков Д. М.

Санкт-Петербург

2023

**Задание.**

Этап 1 Настройка:

    Настроить репликацию postgres на трёх узлах: A - основной, B и C - резервные. Для управления использовать pgpool-II. Репликация с A на B синхронная. Репликация с A на C асинхронная. Продемонстрировать, что новые данные реплицируются на B в синхронном режиме, а на C с задержкой.

Этап 2.1 Подготовка:

    a. Установить несколько клиентских подключений к СУБД.

    b. Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

Этап 2.2 Сбой:

    Симулировать отказ основного узла - выполнить жесткое выключение виртуальной машины.

Этап 2.3 Отработка:

    a. Найти продемонстрировать в логах релевантные сообщения об ошибках.

    b. Выполнить фейловер на резервный сервер.

    c. Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

Этап 3 Восстановление:

    a. Восстановить работу основного узла - откатить действие, выполненное с виртуальной машиной на этапе 2.2.

    b. Актуализировать состояние базы на основном узле - накатить все изменения данных, выполненные на этапе 2.3.

    c. Восстановить работу узлов в исходной конфигурации (в соответствии с этапом 1).

    d. Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

**Выполнение.**

**Этап 1. Настройка**

Для настройки использовались Docker-контейнеры. Основной узел pg0 и два резервных: pg1, pg2. Для управления использовался pgpool-II.

Docker compose file:

networks:

my-network:

driver: bridge

services:

pgpool:

container\_name:

pgpool

image: bitnami/pgpool:4

ports:

- 5432:5432

environment:

- PGPOOL\_BACKEND\_NODES=0:pg0:5432,1:pg1:5432,2:pg2:5432

- PGPOOL\_SR\_CHECK\_USER=postgres

- PGPOOL\_SR\_CHECK\_PASSWORD=postgres

- PGPOOL\_ENABLE\_LDAP=no

- PGPOOL\_POSTGRES\_USERNAME=postgres

- PGPOOL\_POSTGRES\_PASSWORD=postgres

- PGPOOL\_ADMIN\_USERNAME=postgres

- PGPOOL\_ADMIN\_PASSWORD=postgres

- PGPOOL\_ENABLE\_LOG\_PER\_NODE\_STATEMENT=yes

- PGPOOL\_ENABLE\_LOG\_CONNECTIONS=yes

- PGPOOL\_ENABLE\_LOG\_HOSTNAME=yes

healthcheck:

test: ["CMD", "/opt/bitnami/scripts/pgpool/healthcheck.sh"]

interval: 10s

timeout: 5s

retries: 5

pg0:

container\_name:

pg0

image: postgres:14

ports:

- 5432

volumes:

- /Users/alibaba/Documents/GitHub/Distributed-Storage-Systems/lab4/pg0:/var/lib/postgresql/data

environment:

- POSTGRES\_USER=postgres

- POSTGRES\_PASSWORD=postgres

depends\_on:

- pgpool

pg1:

container\_name:

pg1

image: postgres:14

ports:

- 5432

volumes:

- /Users/alibaba/Documents/GitHub/Distributed-Storage-Systems/lab4/pg1:/var/lib/postgresql/data

environment:

- POSTGRES\_USER=postgres

- POSTGRES\_PASSWORD=postgres

depends\_on:

- pg0

- pgpool

pg2:

container\_name:

pg2

image: postgres:14

ports:

- 5432

volumes:

- /Users/alibaba/Documents/GitHub/Distributed-Storage-Systems/lab4/pg2:/var/lib/postgresql/data

environment:

- POSTGRES\_USER=postgres

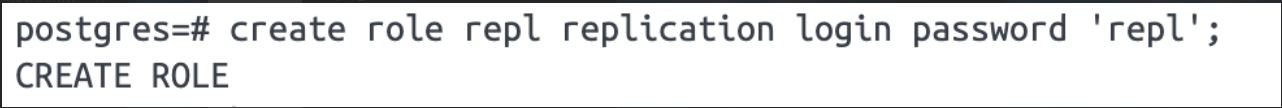
- POSTGRES\_PASSWORD=postgres

depends\_on:

- pg0

- pgpool

Подключимся к master-узлу и создадим пользователя для репликации:



Изменим под него настройки аутентификации:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Проделаем на мастер-узле следующие изменения конфигурации. Настроим реплицирование и укажем, что на узел pg1 репликация должна производится синхронно (по заданию):

############# pg0/postgresql.conf #############

wal\_level = replica

archive\_mode = on

max\_wal\_senders = 10

wal\_keep\_segments = 10

hot\_standby = on

archive\_command = 'test ! -f /var/lib/postgresql/data/archiver/%f && cp %p /var/lib/postgresql/data/archiver/%f'

port = 5432

wal\_log\_hints = on

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Настроим standby (резервные) узлы. Дадим им знать про наш primary-кластер:

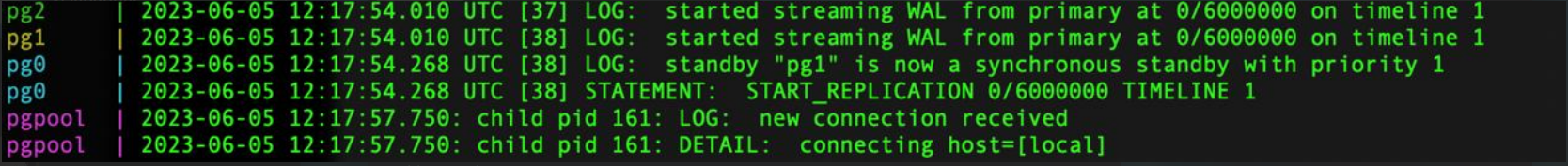
Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Когда всё настроено, создаем реплики, от имени нашего специального пользователя:

**pg\_basebackup -h pg0 -p 5432 -U repl -D /var/lib/postgresql/data/backup -Fp -Xs -R**

Проверяем, что всё успешно перенеслось и перезапускаем все кластеры для применения изменений конфигурации:

****

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

Как видим, всё хорошо, pg0 является master-узлом. Pg1, pg2 standby-кластеры. Репликация на pg1 осуществляется синхронно

**Этап 2.1. Подготовка**

Устанавливаем несколько клиентских соединений и проверяем работу на чтение и на запись:

****

**Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

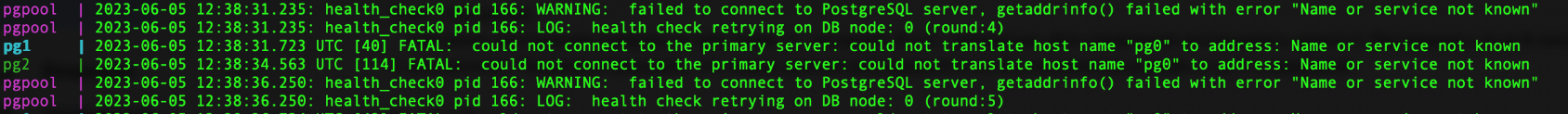
Как видим, репликация работает успешно.

**Этап 2.2. Сбой**

Принудительно выключаем контейнер с primary-узлом.

**Этап 2.3. Отработка**

Как видим, теперь standby-узлы не могут к нему подключиться:



Теперь произведём фейловер, с помощью команды:

pg\_ctl promote -D $PGDATA

**Изображение выглядит как снимок экрана, текст, Шрифт

Автоматически созданное описание**

Как видим, pg1 received promote request. Теперь резервный узел выступает в роли основного. Теперь он работает в режиме записи тоже, поэтому изменим наши данные. Вставим ещё две строки:  
Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

**Этап 3. Восстановление**

Нужно снова запустить pg0 в standby-режиме и синхронизировать все изменения, проделанные на предыдущем шаге. Для этого изменим конфигурацию:

############# pg0/postgresql.conf ############# (after making old master as standby)

recovery\_target\_timeline = 'latest'

archive\_cleanup\_command = 'pg\_archivecleanup /var/lib/postgresql/data/archiver %r'

restore\_command = 'cp /var/lib/postgresql/data/archiver/%f %p'

primary\_conninfo = 'user=repl host=pg1 port=5432'

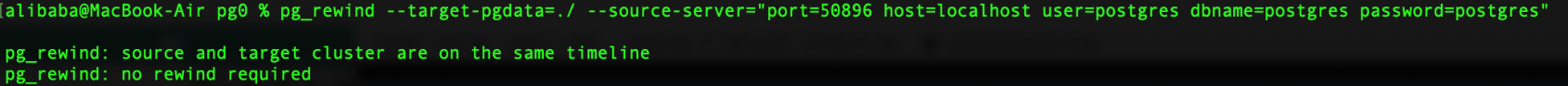
Еще добавили файл standby.signal

Теперь запускаем pg0:

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

С помощью команды pg\_rewind убеждаемся в том, что данные на двух кластерах находятся в актуальном, синхронизированном состоянии:



Удаляем standby.signal, теперь pg0 в исходном состоянии primary кластера.

Проверим наши данные после восстановления:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Как видим, внесенные в этапе 2 изменения здесь присутствуют, а также при добавлении новой строки она появляется на обоих кластерах, значит синхронизация работает, как и прежде

**Вывод.**

В ходе выполнения данной лабораторной работы мы познакомились с инструментами обеспечения высокой доступности баз данных, в частности с репликацией, балансировкой нагрузки и восстановлением из реплики. Так же настроили сеть для связи кластеров в БД внутри контейнеров.