Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №3

«HA Postgres Cluster»

по дисциплине «Администрирование компьютерных сетей»

Автор: Полякова Д.И.

Факультет: ПИН

Группа: K3342

Преподаватель: Самохин Н.Ю.

Санкт-Петербург 2024

Часть 1. Поднимаем Postgres

1. Подготовка окружения

1) Использовала команду ssh, чтобы подключиться к ВМ. В этом случае у меня есть логин root, пароль 2nXhc YBqSfaH6, и IP-адрес ВМ 45.12.229.80.

2)Ввела команду для подключения, затем ввела пароль:

ssh root@45.12.229.80

После успешного подключения видим командную строку:

root@ooo:~#

3)Установила Docker и Docker Compose на мою виртуальную машину.
Для установки Docker:

```
root@ooo:~# sudo apt update
sudo apt install -y docker.io
sudo systemctl start docker
sudo systemctl enable docker
Hit:1 http://mirror.selectel.ru/ubuntu jammy InRelease
Get:2 http://mirror.selectel.ru/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://mirror.selectel.ru/ubuntu jammy-backports InRelease [127 kB]
Hit:4 https://mirror.selectel.ru/3rd-party/cloud-init-deb/jammy jammy InRelease
Get:5 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:6 http://mirror.selectel.ru/ubuntu jammy-updates/main Sources [522 kB]
Get:7 http://mirror.selectel.ru/ubuntu jammy-updates/main amd64 Packages [2149 kB]
Get:8 http://mirror.selectel.ru/ubuntu jammy-updates/universe amd64 Packages [1134 kB]
Get:9 https://dl.cloudsmith.io/public/caddy/stable/deb/debian any-version InRelease [8266 B]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/main Sources [301 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1932 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [913 kB]
```

Для установки Docker Compose:

```
root@ooo:~# sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 12.1M  100 12.1M    0     0  10.4M      0  0:00:01  0:00:01 --:--:-- 40.3M
docker-compose version 1.29.2, build 5becea4c
root@ooo:~#
```

2. Создание Dockerfile

4)На виртуальной машине создала директорию для проекта postgres_cluster и перешла в неё:

```
root@ooo:~# mkdir postgres_cluster
cd postgres_cluster
root@ooo:~/postgres_cluster#
```

5)Создала Dockerfile командой nano Dockerfile и добавила туда код из лабы.

3. Создание конфигурации Patroni

6)Создала первый конфигурационный файл postgres0.yml, который будет использоваться для первой ноды (pg-master) командой nano postgres0.yml и вставила код из задания лабы
7) Создала второй конфигурационный файл postgres1.yml, который будет использоваться для второй ноды (pg-slave)
8) Я скопировала первый файл для удобства и переименовала его:
cp postgres0.yml postgres1.yml
9) Открыла его для редактирования и немного изменила:
Изменила строки name: postgresql0 на name: postgresql1
Изменила строки listen: pg-master:8008 и connect_address: pg-master:8008 на listen: pg-slave:8008 и connect_address: pg-slave:8008
Изменила строки connect_address: pg-master:5432 на connect_address: pg-slave:5432
Изменила строки data_dir: /var/lib/postgresql/data/postgresql0 на data_dir: /var/lib/postgresql/data/postgresql1

Теперь в папке postgres_cluster лежат два файла:

- postgres0.yml
- postgres1.yml

```
Last login: Wed Nov 13 11:45:59 2024 from 9
root@ooo:~# cd postgres_cluster/
root@ooo:~/postgres_cluster# ls
postgres0.yml  postgres1.yml
root@ooo:~/postgres_cluster#
```

4. Создание docker-compose.yml
10)Создала файл docker-compose.yml и вставила туда код из лабы
5. Развертывание кластера и проверка его состояния
После настройки всех конфигурационных файлов запустила кластер:
docker-compose up -d

```
root@ooo:~/postgres_cluster# docker-compose up -d
Building pg-master
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  8.192kB
Step 1/7 : FROM postgres:15
15: Pulling from library/postgres
2d429b9e73a6: Pull complete
8264392a8c63: Pull complete
10dc7a46d285: Pull complete
a3e832e126d8: Pull complete
a57591b52775: Pull complete
8bd747615de8: Pull complete
e96a70b56e29: Pull complete
3db3c2765775: Pull complete
1635531b3f1f: Pull complete
cccb4da64aa2: Pull complete
a1cc9f823061: Pull complete
dbb2f6dde8bc: Pull complete
e1a3b3584c6c: Pull complete
92a5f4d34f14: Pull complete
```

Использовала команду docker logs для проверки журналов контейнеров:

docker logs pg-master
docker logs pg-slave
docker logs zoo

Часть 2. Проверяем репликацию

Проверила, что кластер PostgreSQL запущен и работает

docker-compose ps

```
Creating network "postgres_cluster_default" with the default driver
Creating pg-master ... done
Creating pg-slave  ... done
Creating zoo       ... done
root@ooo:~/postgres_cluster# docker-compose ps
  Name                 Command               State                                  Ports
-------------------------------------------------------------------------------------------------------------------
pg-master   docker-entrypoint.sh patro ...   Up      0.0.0.0:5433->5432/tcp,:::5433->5432/tcp, 8008/tcp
pg-slave    docker-entrypoint.sh patro ...   Up      0.0.0.0:5434->5432/tcp,:::5434->5432/tcp, 8008/tcp
zoo         /etc/confluent/docker/run        Up      0.0.0.0:2181->2181/tcp,:::2181->2181/tcp, 2888/tcp, 3888/tcp
root@ooo:~/postgres_cluster# Connection to 45.12.229.80 closed by remote host.
Connection to 45.12.229.80 closed.
dariapolyakova@MacBook-Air-Dasa-2 ~ %
```

1. Подключение к нодам кластера PostgreSQL через psql

2)Подключилась к pg-master и ввела пароль postgres:

psql -h 127.0.0.1 -p 5433 -U postgres -d postgres

Подключилась к pg-slave и ввела пароль postgres:

Для этого открыла новый терминал и подключилась ко второму контейнеру pg-slave

psql -h 127.0.0.1 -p 5434 -U postgres -d postgres

2.Создание таблицы и вставка данных на мастер-ноде

Создала таблицу на pg-master:

В терминале с подключением к pg-master выполнила команду для создания таблицы.

```
CREATE TABLE test_table (
  id SERIAL PRIMARY KEY,
```

```
  name VARCHAR(50),
  value INT
);
```

Эта команда создала таблицу test_table с тремя полями: id, name, и value.

Вставляем данные в таблицу на pg-master:

Вставила несколько строк в таблицу:
```
INSERT INTO test_table (name, value) VALUES ('Example1', 123);
```

```
INSERT INTO test_table (name, value) VALUES ('Example2', 456);
```

Эти команды добавили две записи в таблицу test_table.

```
postgres=# CREATE TABLE test_table (
  id SERIAL PRIMARY KEY,
  name VARCHAR(50),
  value INT
);
CREATE TABLE
postgres=# INSERT INTO test_table (name, value) VALUES ('Example1', 123);
INSERT INTO test_table (name, value) VALUES ('Example2', 456);
INSERT 0 1
INSERT 0 1
```

3. Проверка репликации на slave-ноде

Проверка данных на pg-slave:

Переключилась на терминал с подключением к pg-slave.

Выполнила команду для проверки, создалась ли таблица и скопировались ли данные на ноде-реплике:
```
SELECT * FROM test_table;
```

Я увидела те же записи, которые добавила в таблицу на pg-master. Это означает, что репликация работает корректно.

```
postgres=# SELECT * FROM test_table;
 id |   name    | value
----+-----------+-------
  1 | Example1 |   123
  2 | Example2 |   456
(2 rows)
```

4.Попытка редактирования данных на pg-slave:

Я попробовала выполнить операцию вставки на pg-slave:
INSERT INTO test_table (name, value) VALUES ('Test', 789);

Поскольку pg-slave работает в режиме только для чтения, я получила ошибку о невозможности записи данных.

```
postgres=# INSERT INTO test_table (name, value) VALUES ('Test', 789);
ERROR:  cannot execute INSERT in a read-only transaction
```

Часть 3. Делаем среднего роста высокую доступность

1.Добавление HAProxy в docker-compose.yml

1)Открыла файл docker-compose.yml и добавила сервис haproxy:

```
...

haproxy:

  image: haproxy:3.0

  container_name: postgres_entrypoint

  ports:

    - 5432:5432

    - 7000:7000

  depends_on:

    - pg-master

    - pg-slave

    - zoo

  volumes:

    - ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg

...
```

Этот блок описывает контейнер HAProxy, который будет балансировать подключение к PostgreSQL между мастер- и слейв-нодами.

2.Создание конфигурационного файла haproxy.cfg

2)В той же директории, где находится docker-compose.yml, создала файл haproxy.cfg.

3.Перезапуск проекта

После добавления haproxy в docker-compose.yml и создания haproxy.cfg, перезапустила проект:

Остановила текущие контейнеры
docker-compose down

Запустила все контейнеры с новой конфигурацией:
docker-compose up -d

4.Проверка состояния контейнеров

Проверка Patroni: Надо убедиться, что ноды PostgreSQL распределили роли между собой. Использовала команду для проверки логов:

docker logs pg-master

docker logs pg-slave

Проверка ролей мастер и реплика в Patroni:

В логах pg-master и pg-slave мне нужно найти строки, которые указывают на то, что одна нода назначена как мастер, а другая как реплика. Это можно определить по следующим строкам:

На мастере (pg-master) лог должен содержать:
promoted self to leader by acquiring session lock

Это подтверждает, что нода pg-master взяла на себя роль лидера.

```
6", "patroni": {"version": "4.0.3", "scope": "my_cluster", "name": "postgresql1"}}
2024-11-13 15:28:31,819 INFO: promoted self to leader by acquiring session lock
2024-11-13 15:28:31.821 UTC [33] LOG:  received promote request
2024-11-13 15:28:31.821 UTC [33] LOG:  redo is not required
```

На реплике (pg-slave) можно найти строку:

no action. I am (postgresql1), a secondary, and following a leader (postgresql0)

Это указывает на то, что pg-slave успешно работает в роли реплики и синхронизируется с мастером.

```
2024-11-13 15:58:11.130 UTC [24] LOG:  parameter "primary_slot_name" changed to "postgresql1"
server signaled
2024-11-13 15:58:11,143 INFO: Reaped pid=45, exit status=0
2024-11-13 15:58:11,150 INFO: no action. I am (postgresql1), a secondary, and following a leader (postgresql0)
2024-11-13 15:58:11.154 UTC [43] LOG:  fetching timeline history file for timeline 3 from primary server
2024-11-13 15:58:11.160 UTC [43] LOG:  started streaming WAL from primary at 0/4000000 on timeline 2
```

Проверка подключения Zookeeper к кластеру Patroni:

Выполнила команду для просмотра логов Zookeeper:

docker logs zoo

Проверила наличие строк, которые подтверждают успешное подключение к Zookeeper и отсутствие ошибок:

INFO binding to port 0.0.0.0/0.0.0.0:2181

Также нет сообщений о Connection dropped или socket connection error.

```
[2024-11-13 15:58:08,615] WARN maxCnxns is not configured, using default value 0. (org.apache.zookeeper.server.ServerCnxnFactory)
[2024-11-13 15:58:08,617] INFO Configuring NIO connection handler with 10s sessionless connection timeout, 1 selector thread(s), 4 work
[2024-11-13 15:58:08,618] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2024-11-13 15:58:08,637] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.watch
[2024-11-13 15:58:08,637] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.watch
```

Проверка работы HAProxy:

Запустила команду для просмотра логов HAProxy:

docker logs postgres_entrypoint

```
root@ooo:~/postgres_cluster# docker logs postgres_entrypoint
[NOTICE]   (1) : New worker (8) forked
[NOTICE]   (1) : Loading success.
[WARNING]  (8) : Server postgres/postgresql_pg_master_5432 is DOWN, reason: Layer4 connection problem, info: "Connection refused", check duration: 0ms. 1 active and 0 backup servers left. 0 sessions active, 0 requeued, 0 remaining in queue.
[WARNING]  (8) : Server postgres/postgresql_pg_slave_5432 is DOWN, reason: Layer4 connection problem, info: "Connection refused", check duration: 0ms. 0 active and 0 backup servers left. 0 sessions active, 0 requeued, 0 remaining in queue.
[ALERT]    (8) : proxy 'postgres' has no server available!
[WARNING]  (8) : Server postgres/postgresql_pg_master_5432 is UP, reason: Layer7 check passed, code: 200, check duration: 3ms. 1 active and 0 backup servers online. 0 sessions requeued, 0 total in queue.
root@ooo:~/postgres_cluster#
```

Судя по логам postgres_entrypoint, HAProxy сначала не смог установить соединение с мастером и репликой PostgreSQL, но затем мастер-нода была успешно поднята.

5.Подключение к базе данных через HAProxy

1)Получила IP-адрес контейнера HAProxy (postgres_entrypoint):

docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' postgres_entrypoint

2)Подключилась с использованием IP-адреса:

psql -h 172.25.0.5 -p 5432 -U postgres -d postgres

```
psql: error: could not translate host name "postgres_entrypoint" to address: Name or service not known
root@ooo:~/postgres_cluster# docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' postgres_entrypoint
172.25.0.5
root@ooo:~/postgres_cluster# psql -h 172.25.0.5 -p 5432 -U postgres -d postgres
```

6.Проверка перенаправления на мастер-ноду

После успешного подключения выполнила SQL-запрос, чтобы проверить, на какую ноду перенаправляет HAProxy:

SELECT pg_is_in_recovery();

Если результат false - я подключена к мастер-ноде. Если результат true, это указывает на слейв-ноду.

```
postgres=# SELECT pg_is_in_recovery();
 pg_is_in_recovery
-------------------
 f
(1 row)
```

Команда вернула значение f, что означает, что сервер, к которому я подключилась, **н**е находится в режиме восстановления и является мастером.

Этот значит, что соединение через HAProxy было успешно направлено к мастер-ноде кластера PostgreSQL, как и должно происходить. HAProxy автоматически направляет запросы на мастер, а не на реплику.

ЗАДАНИЕ

Любым способом выключаем доступ до ноды, которая сейчас является мастером (например, через docker stop). Некоторое время ждем, после этого анализируем логи и так же пытаемся считать/записать что-то в БД через entrypoint подключение. Затем необходимо расписать, получилось или нет, а так же объяснить, что в итоге произошло после принудительного выключения мастера (со скриншотами).

1.Остановка мастер-ноду PostgreSQL

Так как текущая мастер-нода определена, я остановила ее с помощью команды docker stop.

```
docker stop pg-master
```

Подождала несколько секунд, чтобы дать HAProxy и Patroni время на переключение

3. Анализ логов

Проверила логи Patroni и HAProxy, чтобы увидеть, как система реагирует на отключение мастер-ноды.

Логи Patroni на реплике (которая станет новым мастером):

```
docker logs pg-slave
```

Логи HAProxy:

```
docker logs postgres_entrypoint
```

В логах Patroni я увидела процесс переключения, когда реплика автоматически получает роль мастера.

Я сделала выводы на основе следующих строк логов:
Patroni (лог контейнера pg-slave)
Потеря связи с мастер-нодой pg-master:
Лог:
2024-11-13 16:47:19.553 UTC [55] FATAL: could not receive data from WAL stream: server closed the connection unexpectedly
2024-11-13 16:47:19.577 UTC [1043] FATAL: could not connect to the primary server: connection to server at "pg-master" (172.25.0.3), port 5432 failed: Connection refused

Эти строки показывают, что Patroni на pg-slave потерял соединение с pg-master, указывая на отказ pg-master.

Ошибка подключения к мастеру:

Лог:
2024-11-13 16:47:24.603 UTC [1046] FATAL: could not connect to the primary server: could not translate host name "pg-master" to address: Name or service not known

Это сообщение подтверждает ошибки подключения к pg-master, показывая, что Patroni проверяет доступность мастера и не может восстановить связь.

Переключение ролей:

Лог:
2024-11-13 16:47:49,819 INFO: promoted self to leader by acquiring session lock

Эта строка говороит о том, что pg-slave инициирует процесс перехода в статус мастера, взяв на себя управление кластером.

Завершение восстановления и готовность к подключениям:

Лог:
2024-11-13 16:47:49.903 UTC [24] LOG: database system is ready to accept connections

Эта строка указывает на завершение процесса восстановления и подтверждает, что pg-slave стал новым мастером, доступным для подключений.

HAProxy (лог контейнера postgres_entrypoint)

Обнаружение отказа pg-master:

Лог:
[WARNING] (8) : Server postgres/postgresql_pg_master_5432 is DOWN, reason: Layer4 connection problem, info: "Connection refused"

Этот лог HAProxy показывает, что балансировщик не может связаться с pg-master, сигнализируя об отказе.

Переключение на новый мастер:

Лог:
[WARNING] (8) : Server postgres/postgresql_pg_slave_5432 is UP, reason: Layer7 check passed, code: 200

Здесь видно, что HAProxy переключился на pg-slave, который теперь работает как мастер. Этот лог подтверждает успешное переключение HAProxy на новую мастер-ноду.

4.Попытка подключиться к БД и выполнить операции чтения и записи
После переключения на новую мастер-ноду (текущий pg-slave), подключилась к HAProxy и попробовала выполнить команды для проверки работоспособности:
psql -h 172.25.0.5 -p 5432 -U postgres -d postgres

После подключения выполнила команды, чтобы проверить доступность и права записи:

Проверила, что я подключена к мастеру
SELECT pg_is_in_recovery();

Если результат f, то я подключена к новому мастеру.

```
postgres=# SELECT pg_is_in_recovery();
 pg_is_in_recovery
-------------------
 f
(1 row)
```

Выполнила тестовую запись (создание таблицы):

```
postgres=# CREATE TABLE test_failover (id SERIAL PRIMARY KEY, name VARCHAR(50));
INSERT INTO test_failover (name) VALUES ('Test after failover');
CREATE TABLE
INSERT 0 1
```

Итоги

Что произошло:

- После отключения оригинальной мастер-ноды Patroni обнаружил недоступность мастера и автоматически переключил роль на реплику.

- HAProxy также перенастроился, направляя входящие запросы на новый мастер.
- В итоге новый мастер поддерживает операции записи, а база данных остается доступной.

Выводы:

Система успешно справилась с отказом узла, сохраняя доступность базы данных и обеспечивая высокую доступность.

Ответы на вопросы:
1)Порты 8008 и 5432 вынесены в разные директивы, expose и ports. По сути, если записать 8008 в ports, то он тоже станет exposed. В чем разница?

Разница между директивами expose и ports в Docker Compose заключается в том, как они открывают порты и на что это влияет:

1. expose: открывает порт только для внутренних сетей Docker. Этот порт становится доступен для других контейнеров внутри одной сети Docker Compose, но не доступен для внешних запросов.
2. ports: пробрасывает порт из контейнера на хост-машину. Это значит, что порт будет доступен для внешних запросов.

У меня:

Порт 5432 (через ports) пробрасывается на внешние порты 5433 и 5434 хоста, что позволяет подключаться к базе данных PostgreSQL извне.

Порт 8008 (через expose) открыт только для внутренней сети Docker и используется для связи между контейнерами Patroni (pg-master и pg-slave) для определения лидера в кластере.

2) При обычном перезапуске композ-проекта, будет ли сбилден заново образ? А если предварительно отредактировать файлы postgresX.yml? А если содержимое самого Dockerfile? Почему?

При перезапуске проекта с помощью Docker Compose поведение будет разным в зависимости от того, что именно изменено:

1. Обычный перезапуск (docker-compose up -d) без изменений:
    - При перезапуске Docker Compose проверяет, существуют ли образы и контейнеры, описанные в файле docker-compose.yml. Если они существуют и не изменены, он просто перезапустит их, не пересобирая образы и не пересоздавая контейнеры.
    - Итог: Образ не будет пересобран, так как Docker считает его актуальным и не требует изменений.
2. Перезапуск с изменениями в файлах postgresX.yml:

- Docker Compose не отслеживает изменения в файлах, которые монтируются как тома или копируются в контейнер, таких как postgres0.yml и postgres1.yml, если они не включены в процесс сборки.
- В данном случае, если postgresX.yml монтированы как тома, изменения вступят в силу сразу, так как Docker просто подхватит обновленные файлы при запуске контейнера.
- Итог: При обычном перезапуске проект не пересоберется автоматически из-за изменений в файлах postgresX.yml.

3. Перезапуск с изменениями в Dockerfile:
    - Изменения в Dockerfile заставят Docker Compose пересобрать образ. Docker отслеживает изменения в Dockerfile, и при наличии изменений в нём при следующей сборке он создаст новый образ.
    - Итог: Образ будет сбилден заново, изменения в Dockerfile потребуют пересборки для вступления в силу (например, командой docker-compose up --build).