Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Дисциплина: Распределённые системы хранения данных

Лабораторная работа 4

Вариант 25

Выполнили:
Кривоносов Егор Дмитриевич
Иманзаде Фахри Рашид Оглы

Группа: P33111

Преподаватель:
Николаев Владимир Вячеславович

2022 г.

Санкт-Петербург

Текст задания

Работа рассчитана на двух человек. Для выполнения и демонстрации лабораторной работы разрешено (и рекомендуется) использование своих компьютеров. В случае отсутствия возможности использования своего компьютера для выполнения лабораторной работы обратитесь к преподавателю для корректировки варианта и получения доступа к узлам.

В качестве хостов использовать одинаковые виртуальные машины.

В первую очередь настроить сеть виртуальных машин:

- Если ВМ запускаются на одном хосте, рекомендуется использовать NAT сеть.
- Если ВМ запускаются на различных хостах, рекомендуется использовать сетевые интерфейсы в режиме “Bridge”; для связи рекомендуется использовать проводное соединение.
- Проверить сетевую связность между всеми узлами (ping, ssh).

Для подключения к СУБД (например, через psql), использовать отдельную виртуальную или физическую машину.

Перед тем как “сломать” узел на этапе 2, рекомендуется выполнить снапшот виртуальной машины.

Для демонстрации наполнения базы, а также доступа на запись (см. задание ниже) использовать не меньше двух таблиц, трёх столбцов, пяти строк, двух транзакций, двух клиентских сессий. Данные не обязаны быть осмысленными, но должны быть легко отличимы - повторяющиеся строки запрещены.

Этап 1 Настройка:

Развернуть postgres на двух узлах в режиме потоковой репликации. Не использовать дополнительные пакеты. Продемонстрировать доступ в режиме чтение/запись на основном сервере, а также что новые данные синхронизируются на резервный.

Этап 2.1 Подготовка:

1. Установить несколько клиентских подключений к СУБД.
2. Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

Этап 2.2 Сбой:

Симулировать недоступность основного узла - отключить сетевой интерфейс виртуальной машины, переключить его в изолированную подсеть и т.п.

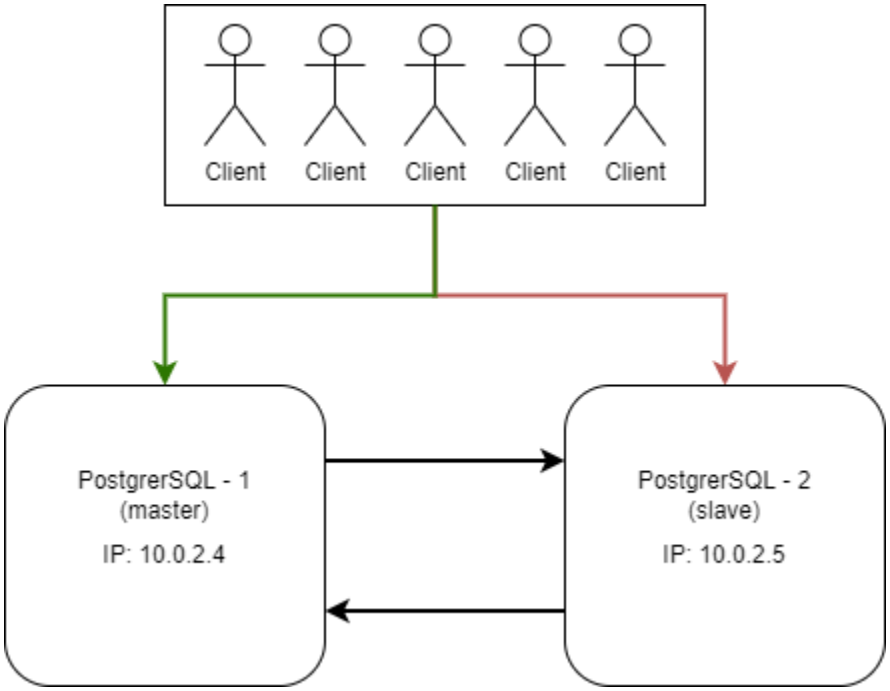
Этап 2.3 Отработка:

1. Найти продемонстрировать в логах релевантные сообщения об ошибках.
2. Выполнить фейловер на резервный сервер.
3. Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

Этап 3 Восстановление:

1. Восстановить работу основного узла - откатить действие, выполненное с виртуальной машиной на этапе 2.2.
2. Актуализировать состояние базы на основном узле - накатить все изменения данных, выполненные на этапе 2.3.
3. Восстановить работу узлов в исходной конфигурации (в соответствии с этапом 1).
4. Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

Выполнение



Этап 1. Настройка

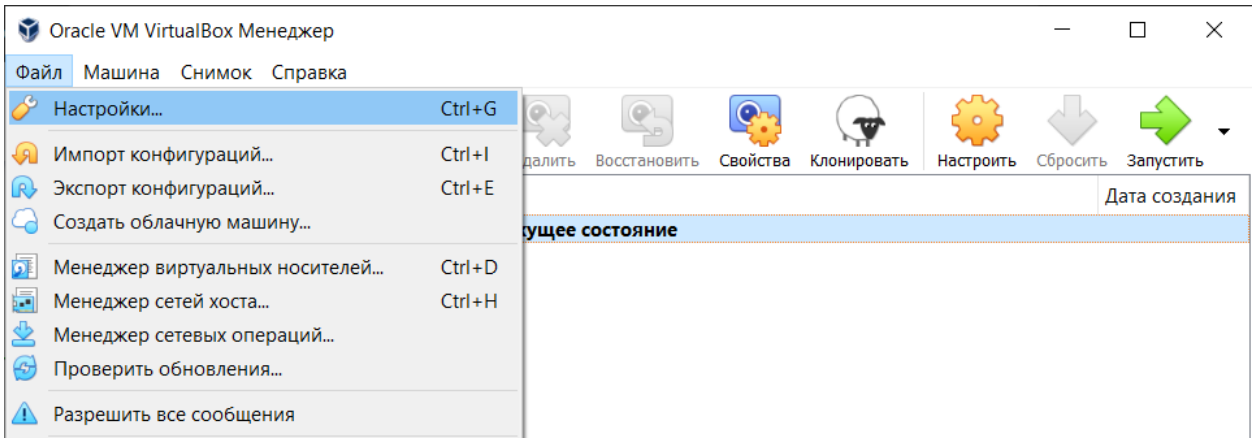
Создадим и настроим 3 виртуальные машины:

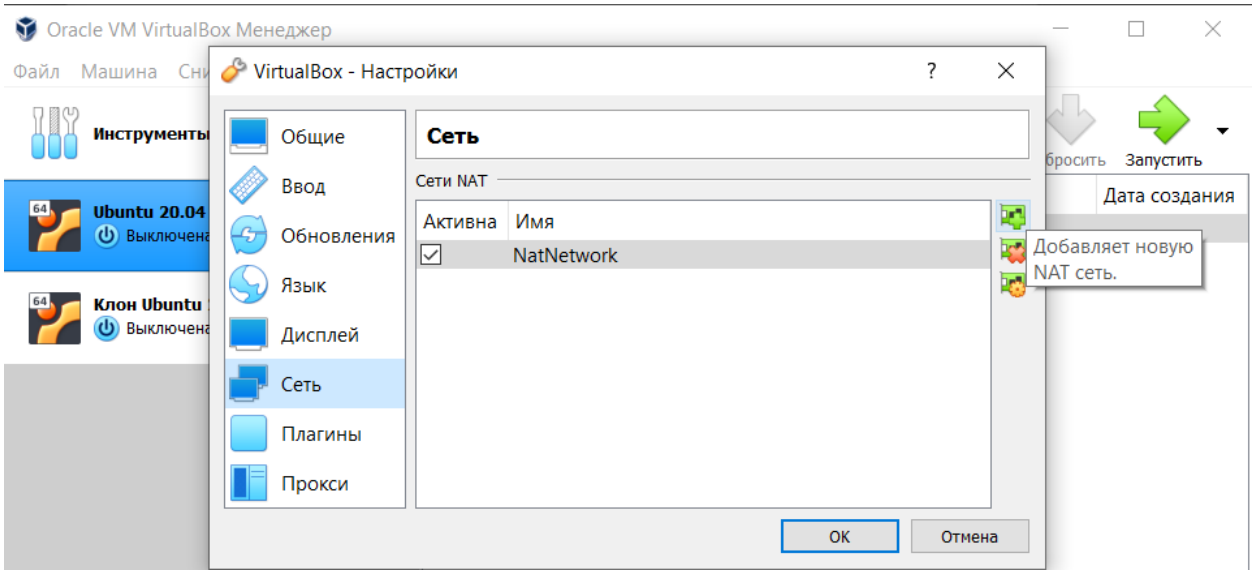
- master (A - основной)
- slave (B - резервный)
- desktop (для подключения через **psql**)

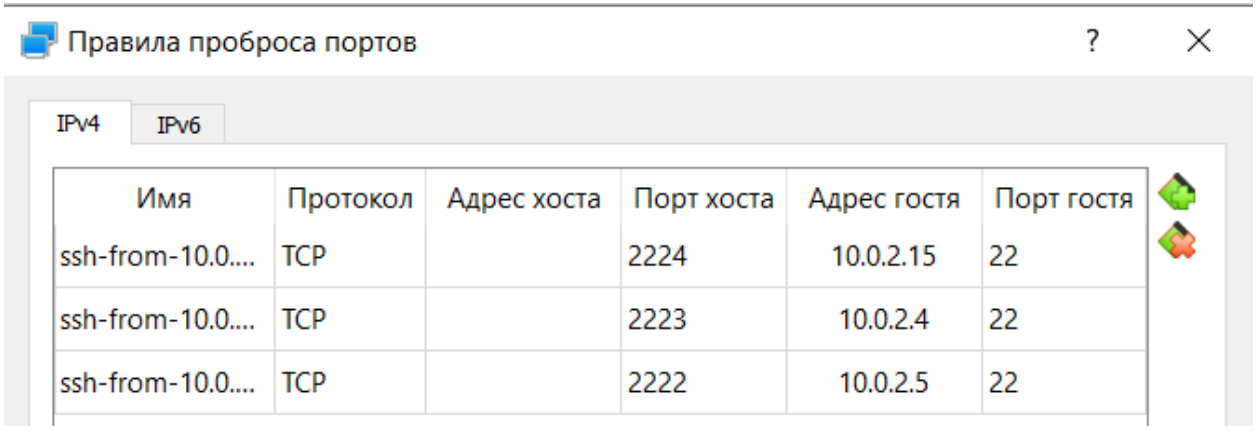
На первые две установим postgresql и конфигурируем доступ по **ssh** с **rsa key**.

Настройка VirtualBox

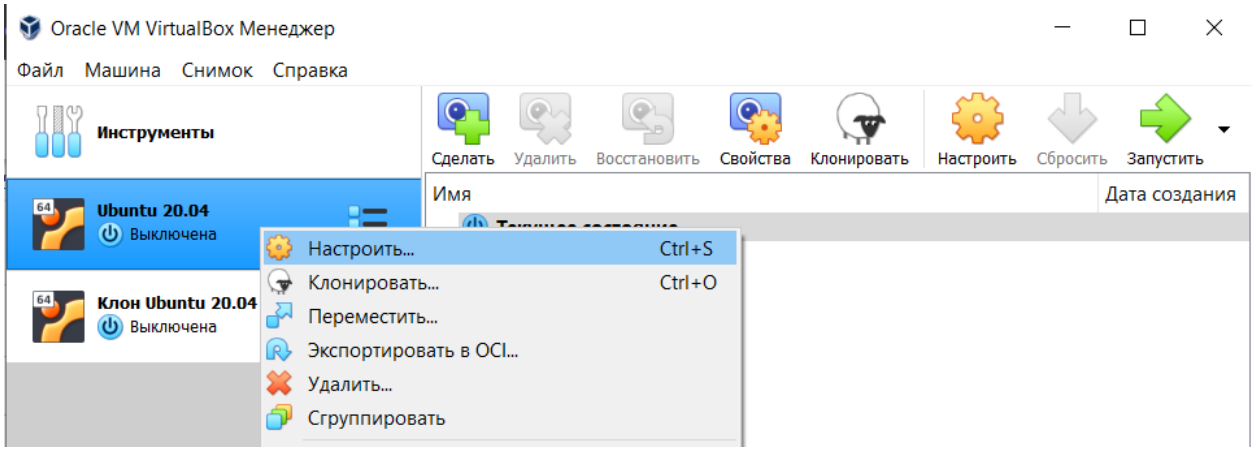
Настроим сначала **“NAT сеть”** для виртуальных машин:

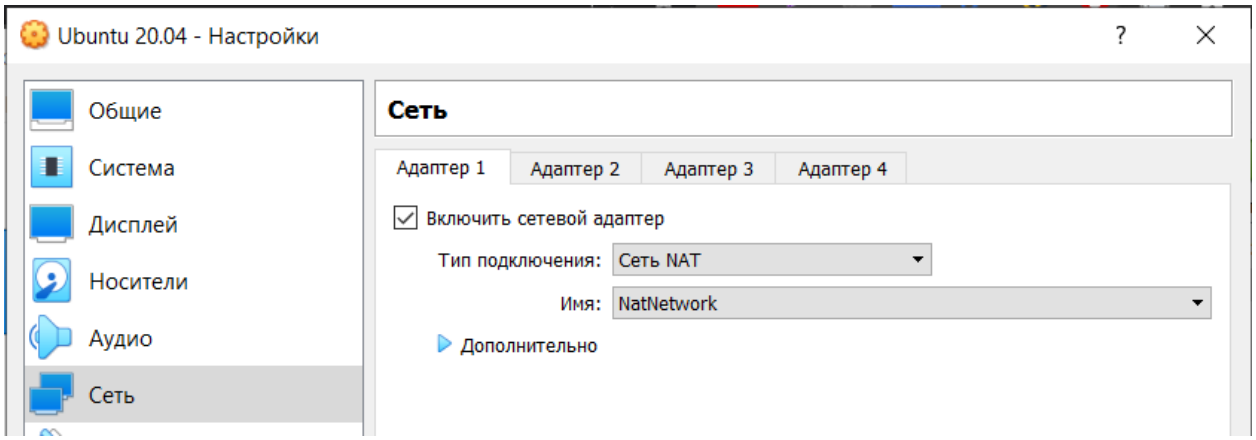




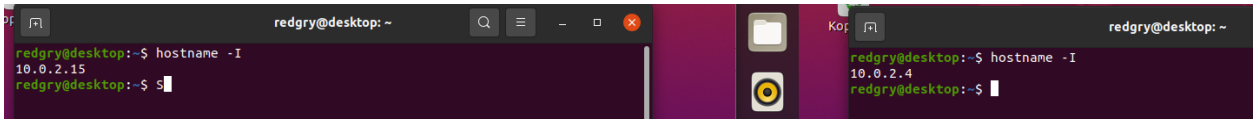


Для каждой виртуальной машины повторим нижние два шага



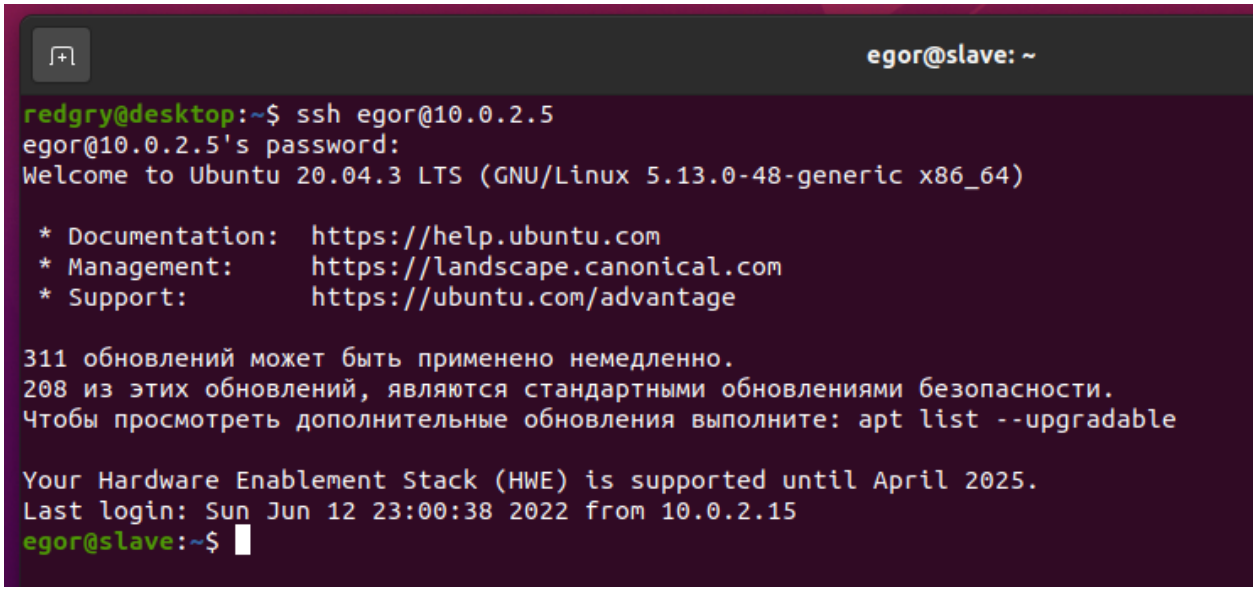


Обязательно, если мы делаем клон нашей одной из виртуальных машин нужно не забыть поменять **MAC-адрес**, иначе всегда будет у обеих машин одинаковый **IP**.
После этого перезагрузим наши VM и проверим, что **IP-адрес** они имеют разный.



Дальше, чтобы был доступ по **ssh** нужно будет установить **ssh-server** командой:
sudo apt-get install openssh-server

После этого проверим, что мы можем подключиться по **ssh**:



Настройка PostgreSQL

Смена пользователя на postgres:

sudo su - postgresql

Настройка Master

В конце файла **/etc/postgresql/12/main/pg_hba.conf** мы добавим 1 строчку, которая разрешить пользователю **postgres** подключаться к этому серверу из **slave**.

```
# IPv4 local connections:S
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            md5
host    replication     all             ::1/128                 md5

# for stanby nodes
host    replication     postgres        10.0.2.5/32             md5
```

Если бы наши машины находились бы в разных сетях или на нужно было бы, чтобы они общались друг с другом по публичным адресам - скорее всего нам пришлось бы настроить Firewall ещё.

Дальше нам нужно настроить репликации. Откроем конфигурационный файл /etc/**postgresql/12/main/postgresql.conf** найдем нужные нам параметры и изменим их:

```
listen_addresses = 'localhost, 10.0.2.4'          # 10.0.2.4 - IP-адрес master
wal_level = hot_standby
archive_mode = on
archive_command = 'cd .'
max_wal_senders = 2
hot_standby = on
```

Все, мы настроили Master. Теперь перезагрузим сервер, чтобы настройки применились:
service postgresql restart

Настройка Slave

Перед настройкой остановить сервер postgresql:
service postgresql stop

Аналогично как мы делали на Master отредактируем **pg_hba.conf** файл. Разница будет заключаться в том, что мы теперь укажем IP-адрес master’a.

```
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            md5
host    replication     all             ::1/128                 md5

# for master
host    replication     postgres        10.0.2.4/32             md5
```

Теперь правим файл postgresql.conf. Настройки аналогичные, только меняем IP-адрес.

```
listen_addresses = 'localhost, 10.0.2.5'          # 10.0.2.4 - IP-адрес slave
wal_level = hot_standby
archive_mode = on
archive_command = 'cd .'
max_wal_senders = 2
hot_standby = on
```

Настройка серверов одинаковая, отличается только IP-адрес. Это потому, что при необходимости реплика может стать мастером, а вся разница будет в наличии одного лишь файла.

Теперь удаляем все через пользователя **postgres** в каталоге ~/12/
И снова создаем папку БД, но уже пустую.
rm -rf main; mkdir main; chmod go-rwx main

```
postgres@slave:~/12$ pwd
/var/lib/postgresql/12
postgres@slave:~/12$ rm -rf main; mkdir main; chmod go-rwx main
postgres@slave:~/12$
```

Теперь выгрузим БД с **Master**.
pg_basebackup -P -R -X stream -c fast -h 10.0.2.4 -U postgres -D ./main

```
postgres@slave:~/12$ pg_basebackup -P -R -X stream -c fast -h 10.0.2.4 -U postgr
es -D ./main
Password:
24626/24626 kB (100%), 1/1 tablespace
```

```
postgres@slave:~/12$ ls -l main
итого 80
-rw------- 1 postgres postgres  224 июн 13 01:11 backup_label
drwx------ 5 postgres postgres 4096 июн 13 01:11 base
drwx------ 2 postgres postgres 4096 июн 13 01:11 global
drwx------ 2 postgres postgres 4096 июн 13 01:11 pg_commit_ts
drwx------ 2 postgres postgres 4096 июн 13 01:11 pg_dynshmem
drwx------ 4 postgres postgres 4096 июн 13 01:11 pg_logical
drwx------ 4 postgres postgres 4096 июн 13 01:11 pg_multixact
drwx------ 2 postgres postgres 4096 июн 13 01:11 pg_notify
drwx------ 2 postgres postgres 4096 июн 13 01:11 pg_replslot
drwx------ 2 postgres postgres 4096 июн 13 01:11 pg_serial
drwx------ 2 postgres postgres 4096 июн 13 01:11 pg_snapshots
drwx------ 2 postgres postgres 4096 июн 13 01:11 pg_stat
drwx------ 2 postgres postgres 4096 июн 13 01:11 pg_stat_tmp
drwx------ 2 postgres postgres 4096 июн 13 01:11 pg_subtrans
drwx------ 2 postgres postgres 4096 июн 13 01:11 pg_tblspc
drwx------ 2 postgres postgres 4096 июн 13 01:11 pg_twophase
-rw------- 1 postgres postgres    3 июн 13 01:11 PG_VERSION
drwx------ 3 postgres postgres 4096 июн 13 01:11 pg_wal
drwx------ 2 postgres postgres 4096 июн 13 01:11 pg_xact
-rw------- 1 postgres postgres  257 июн 13 01:11 postgresql.auto.conf
-rw------- 1 postgres postgres    0 июн 13 01:11 standby.signal
```

В команде **pg_basebackup** есть параметр **-R**. Он означает, что PostgreSQL-сервер также создаст пустой файл **standby.signal**. Данный файл означает, что этот сервер - реплика.

После этого запустим наш сервер снова.
service postgresql start

Этап 2.1 Подготовка

На **Master** узле:

```
postgres=# select client_addr, state from pg_stat_replication ;
 client_addr |   state
-------------+-----------
 10.0.2.5    | streaming
(1 row)
```

На **Slave** узле:

```
postgres=# select sender_host, status from pg_stat_wal_receiver;
 sender_host |  status
-------------+-----------
 10.0.2.4    | streaming
(1 row)
```

Создадим несколько таблиц и заполним их:

psql -c "CREATE TABLE test_table1 (id SERIAL PRIMARY KEY, name text NOT NULL, age INT);"
psql -c "CREATE TABLE test_table2 (id SERIAL PRIMARY KEY, name text NOT NULL, age INT);"
psql -c "INSERT INTO test_table1 (name, age) VALUES ('Egor', 17), ('Faxri', 18), ('Diana', 17), ('Petr', 0), ('Vladimir', 27);"
psql -c "INSERT INTO test_table1 (name, age) VALUES ('kkk', 3), ('aaa', 76), ('bbb', 23), ('dd', 12), ('Egor', 2);"
psql -c "INSERT INTO test_table2 (name, age) VALUES ('kkk', 3), ('aaa', 76), ('bbb', 23), ('dd', 12), ('Egor', 2);"

```
postgres@master:~$ psql -c "INSERT INTO test_table2 (name, age) VALUES ('kkk',
3), ('aaa', 76), ('bbb', 23), ('dd', 12), ('Egor', 2);"
INSERT 0 5
```

Проверим теперь, что на резервном узле у нас тоже данные изменились:

```
redgry@desktop:~$ ssh postgres@10.0.2.5
postgres@10.0.2.5's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.13.0-48-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

306 обновлений может быть применено немедленно.
203 из этих обновлений, являются стандартными обновлениями безопасности.
Чтобы просмотреть дополнительные обновления выполните: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Mon Jun 13 00:22:51 2022 from 10.0.2.4
postgres@slave:~$ psql -c "select * from test_table1"
 id |   name   | age
----+----------+-----
  1 | Egor     |  17
  2 | Faxri    |  18
  3 | Diana    |  17
  4 | Petr     |   0
  5 | Vladimir |  27
  6 | kkk      |   3
  7 | aaa      |  76
  8 | bbb      |  23
  9 | dd       |  12
 10 | Egor     |   2
(10 rows)

postgres@slave:~$ psql -c "select * from test_table2"
 id | name | age
----+------+-----
(0 rows)

postgres@slave:~$ psql -c "select * from test_table2"
 id | name | age
----+------+-----
  1 | kkk  |   3
  2 | aaa  |  76
  3 | bbb  |  23
  4 | dd   |  12
  5 | Egor |   2
(5 rows)
```

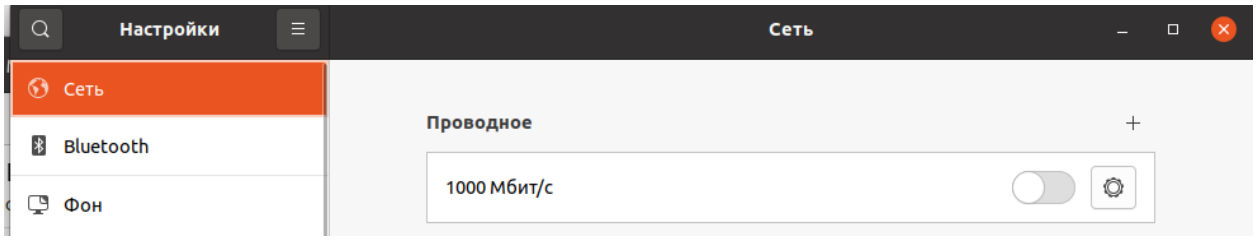
Теперь контрольная проверка - попробуем создать новую таблицу на сервере **Slave**. Если мы сделали все правильно, то сервер не должен позволить нам этого сделать. Все потому что он настроен только на репликацию с основной БД.

```
postgres@slave:~$ psql -c "CREATE TABLE test (id int, name text);"
ERROR:  cannot execute CREATE TABLE in a read-only transaction
postgres@slave:~$
```

Значит репликация настроена правильно.

Этап 2.2 Сбой

Просто отключим интернет XD



Этап 2.3 Отработка

Посмотрим логи об ошибках на slave узел:

```
2022-06-13 03:02:15.949 MSK [15998] LOG:  invalid record length at 0/3029528: wanted 24, got 0
2022-06-13 03:02:18.994 MSK [16669] LOG:  started streaming WAL from primary at 0/3000000 on timeline 1
2022-06-13 03:05:49.283 MSK [16669] FATAL:  terminating walreceiver due to timeout
2022-06-13 03:05:59.656 MSK [16681] FATAL:  could not connect to the primary server: could not connect to server: Нет маршрута до узла
                Is the server running on host "10.0.2.4" and accepting
                TCP/IP connections on port 5432?
2022-06-13 03:06:02.728 MSK [16682] FATAL:  could not connect to the primary server: could not connect to server: Нет маршрута до узла
                Is the server running on host "10.0.2.4" and accepting
                TCP/IP connections on port 5432?
2022-06-13 03:06:05.800 MSK [16695] FATAL:  could not connect to the primary server: could not connect to server: Нет маршрута до узла
                Is the server running on host "10.0.2.4" and accepting
                TCP/IP connections on port 5432?
2022-06-13 03:06:12.744 MSK [16696] FATAL:  could not connect to the primary server: could not connect to server: Нет маршрута до узла
                Is the server running on host "10.0.2.4" and accepting
```

Выполним failover с помощью команды:e

/usr/lib/postgresql/12/bin/pg_ctl promote -D /var/lib/postgresql/12/main

```
postgres@slave:~$ /usr/lib/postgresql/12/bin/pg_ctl promote -D ~/12/main
waiting for server to promote.... done
server promoted
postgres@slave:~$
```

Посмотрим логи на репликанте:

```
2022-06-13 03:14:11.421 MSK [15998] LOG:  received promote request
2022-06-13 03:14:11.421 MSK [17115] FATAL:  terminating walreceiver process due to administrator command
2022-06-13 03:14:11.422 MSK [15998] LOG:  redo done at 0/30294F0
2022-06-13 03:14:11.422 MSK [15998] LOG:  last completed transaction was at log time 2022-06-13 01:36:21.572261+03
2022-06-13 03:14:11.437 MSK [15998] LOG:  selected new timeline ID: 2
2022-06-13 03:14:11.483 MSK [15998] LOG:  archive recovery complete
2022-06-13 03:14:11.498 MSK [15997] LOG:  database system is ready to accept connections
egor@slave:~$
```

Попробуем добавить что-то в таблицу **test_table2** и посмотрим изменения:

```
egor@slave: ~
postgres@slave:~$ psql -c "select * from test_table2;"
 id | name | age
----+------+-----
  1 | kkk  |   3
  2 | aaa  |  76
  3 | bbb  |  23
  4 | dd   |  12
  5 | Egor |   2
(5 rows)

postgres@slave:~$ psql -c "insert into test_table2 (name, age) values ('NEW_TEST', 1337);"
INSERT 0 1
postgres@slave:~$ psql -c "select * from test_table2;"
 id |   name   | age
----+----------+------
  1 | kkk      |    3
  2 | aaa      |   76
  3 | bbb      |   23
  4 | dd       |   12
  5 | Egor     |    2
 34 | NEW_TEST | 1337
(6 rows)

postgres@slave:~$
```

Как мы можем наблюдать наш **Slave**-сервер теперь является мастером и работает на запись и чтение.

Напишем скрипт, который будет автоматически отрабатывать и менять **Slave**-сервер на **Master** если он не доступен (failover)

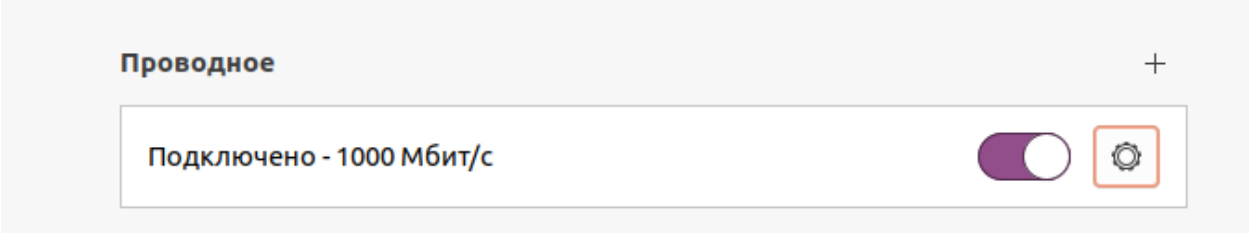
```
#!/bin/sh

while ping -c1 $1 >/dev/null
do
        :
done
touch $HOME/12/main/failover
```

Также нужно выставить параметр **promote_trigger_file**, чтобы когда создавался наш **Slave** переходит в режим чтения и запись.

```
promote_trigger_file = 'failover'
```

Этап 3. Восстановление

Восстанавливаем работу основного узла просто включив снова интернет XD



Теперь нужно синхронизировать Slave с Master (бывшего). Для того чтобы были обновлены изменения после пункта 2.3
Для этого удаляем на **Master** в каталоге ~/12 директорию main с пересозданием её
rm -rf main; mkdir main; chmod go-rwx main

```
postgres@master:~/12$ pwd
/var/lib/postgresql/12
postgres@master:~/12$ rm -rf main; mkdir main; chmod go-rwx main
postgres@master:~/12$
```

Теперь выгрузим БД со **Slave** при помощи синхронизации двух узлов.
Команда выполняется на **Master'е**
rsync -avv postgres@10.0.2.5:~/12/* ~/12/

```
postgres@master:~/12$ rsync -avv postgres@10.0.2.5:~/12/* ~/12/
opening connection using: ssh -l postgres 10.0.2.5 rsync --server --sender -vvlogDtpre.iLsfxC . "~/12/*"  (10 args)
The authenticity of host '10.0.2.5 (10.0.2.5)' can't be established.
ECDSA key fingerprint is SHA256:nDEORRHpRuIEcWgfDP0XMFU/tNwEm1uPESelCDR0VI4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.5' (ECDSA) to the list of known hosts.
postgres@10.0.2.5's password:
receiving incremental file list
delta-transmission enabled
main/
main/PG_VERSION
main/backup_label.old
main/postgresql.auto.conf
main/postmaster.opts
main/postmaster.pid
main/base/
main/base/1/
main/base/1/112
main/base/1/113
main/base/1/1247
```

А также создаём на **Slave standby.signal**, чтобы просигнализировать ему о том, что он снова репликант и перезапускаем кластер.
touch ~/12/main/standby.signal
service postgresql restart

```
-rw-rw-r-- 1 postgres postgres    0 июн 14 12:20 standby.signal
postgres@slave:~/12/main$ service postgresql restart
postgres@slave:~/12/main$ service postgresql status
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
     Active: active (exited) since Tue 2022-06-14 12:20:21 MSK; 6s ago
    Process: 17908 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 17908 (code=exited, status=0/SUCCESS)
postgres@slave:~/12/main$
```

Возвращаемся на **Master**, запускаем БД и видим, что данные успешно обновились.

```
postgres@master:~/12$ service postgresql start
postgres@master:~/12$ service postgresql status
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
     Active: active (exited) since Tue 2022-06-14 12:21:28 MSK; 2s ago
    Process: 11723 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 11723 (code=exited, status=0/SUCCESS)
postgres@master:~/12$ S
```

```
postgres@master:~/12$ psql
psql (12.11 (Ubuntu 12.11-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# select * from test_table2;
 id |   name    | age
----+-----------+------
  1 | kkk       |    3
  2 | aaa       |   76
  3 | bbb       |   23
  4 | dd        |   12
  5 | Egor      |    2
 34 | NEW_TEST  | 1337
(6 rows)

postgres=#
```

Попробуем добавить данные в **Master**:
psql -c "insert into test_table2 (name, age) valuser ('rest', 777)"

```
postgres=# insert into test_table2 (name, age) values ('rest', 777);
INSERT 0 1
postgres=# select * from test_table2;
 id |   name    | age
----+-----------+------
  1 | kkk       |    3
  2 | aaa       |   76
  3 | bbb       |   23
  4 | dd        |   12
  5 | Egor      |    2
 34 | NEW_TEST  | 1337
 35 | rest      |  777
(7 rows)

postgres=#
```

Проверим, изменились ли данные на **Slave**:
psql -c "select * from test_table2;"

```
postgres@slave:~$ psql -c "select * from test_table2;"
 id |   name    | age
----+-----------+-----
  1 | kkk       |   3
  2 | aaa       |  76
  3 | bbb       |  23
  4 | dd        |  12
  5 | Egor      |   2
 34 | NEW_TEST  | 1337
 35 | rest      |  777
(7 rows)

postgres@slave:~$
```

Все данные перенеслись корректно :)

Проверим возможно ли изменить данные со стороны **Slave**:
psql -c "insert into test_table2 (name, age) valuser ('test', 1)"

```
postgres@slave:~$ psql -c "insert into test_table2 (name, age) values ('test', 1);"
ERROR:  cannot execute INSERT in a read-only transaction
postgres@slave:~$
```

Не получилось из-за того, что у нас **Slave** только на чтение. Значит конфигурация была полностью восстановлена и все работает корректно.

Вывод

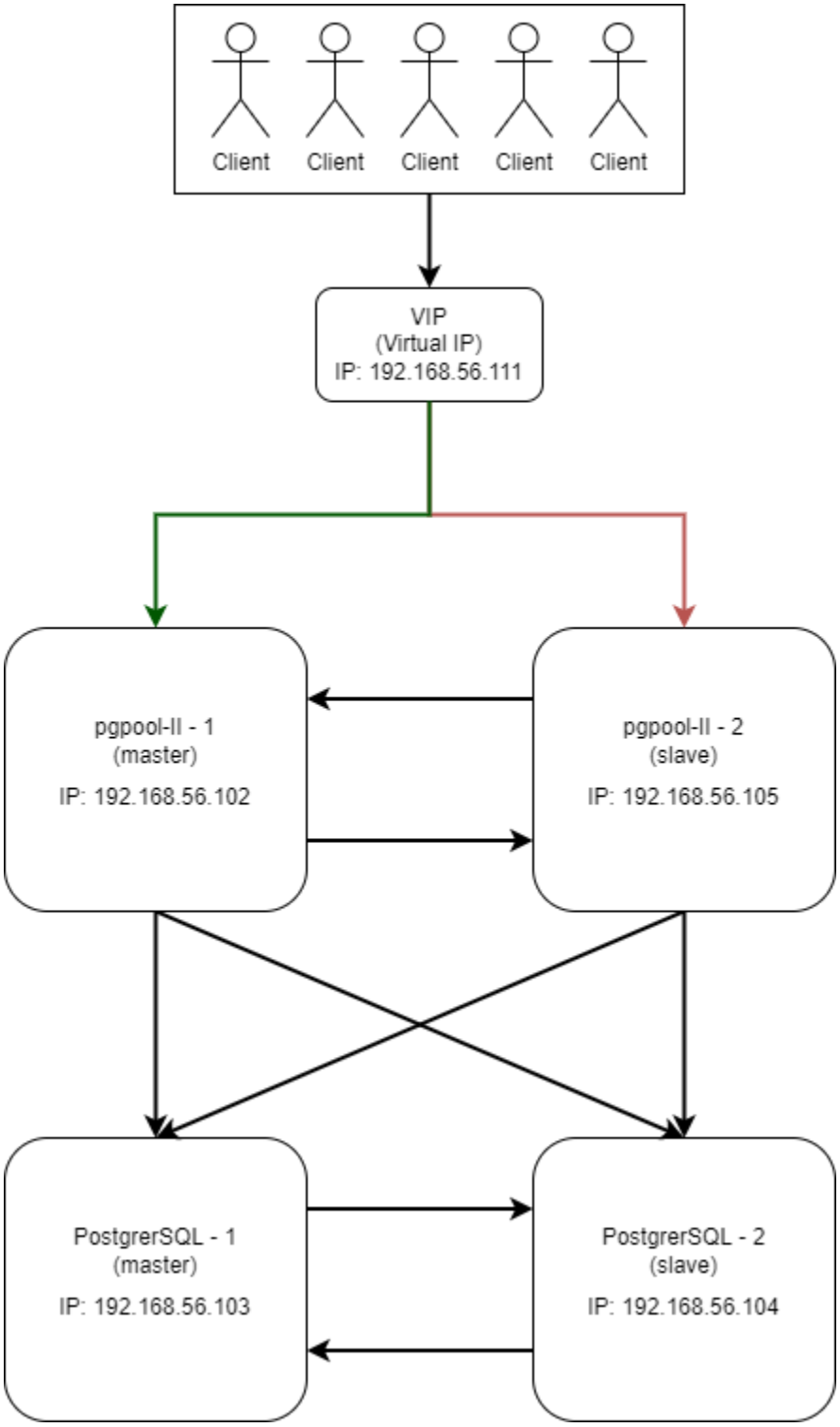
В данной лабораторной работе была проведена настройка виртуальных машин, объединение их в NAT сеть, настройка балансировки и работа с отказоустойчивостью.

НУЖНО СДЕЛАТЬ:
1. IP поправить для подключения
2. Переполнение для мастера

Доп задание на 100%

А теперь выполним дополнительное задание. Наша система не является корректной с той точки зрения, что пользователю при падении одной из **БД** приходится менять **IP**, чтобы подключиться к резервной. Это не хорошо и нужно как-то выйти из ситуации. Поэтому мы решили добавить **PGPOOL**. Но pgpool так как находится тоже на отдельном сервере, он может случайно упасть. Поэтому мы добавим **2 pgpool** на разные сервера, чтобы при выходе основного, все запросы шли на второй, при этом **IP-адрес один** и переподключиться не нужно. Ещё pgpool позволит упростить механизм **авто-фейловера** и восстановления, так как в настройках у него есть специальные возможности следить за БД и при выходе из строя выполнять нужную нам команду. В нашем случае мы напишем **скрипт для авто-фейловера**.

Концепция:


Client
Client
Client
Client
Client
VIP
(Virtual IP)
IP: 192.168.56.111
pgpool-II - 1
(master)
IP: 192.168.56.102
pgpool-II - 2
(slave)
IP: 192.168.56.105
PostgrerSQL - 1
(master)
IP: 192.168.56.103
PostgrerSQL - 2
(slave)
IP: 192.168.56.104

Описание как делал:

Для выполнения этого задания советую воспользоваться хорошими гайдами в интернете:
Статья 1 Статья 2 Статья 3

Также в **Virtual Box** нужно будет либо добавить новый адаптер для создания новой сети или заменить **“NAT сеть”** из прошлого задания. Я делал первый вариант и менял на сеть: **“Виртуальный адаптер хоста”**. Это нужно, чтобы создать VIP, который будет виден во всей сети нашей. Нигде в интернете не нашел нормально пояснения как это делать и разбирался с этим несколько часиков, так как в первый раз с этим столкнулся.

После того как поменяли сеть, заходим в правильном порядке на наши виртуалке, а то IP будут по-другому раз переделены.
Мой вариант: Client -> pgpool-1 -> db-1 -> db-2 -> pgpool-2

На всех виртуальных машинах должен стоять postgresql-client. На postgresql ставим дополнительные пакеты для работы с pgpool. А на машины pgpool только pgpool.

Дальше делаем все по гайдам в статьях.
Файлы для моей системы находятся у меня на диске в папке РСХД (вранье, мне лень их было перекидывать с ноута :/).

Главное:

DB: pg_hba.conf + postgresql.conf + /var/log/postgresql
Запуск, чтобы оно работало: service postgresql start (restart, status)

Pgpool: pgpool.conf + /var/log/postgresql/pgpool_status (это файл очень важен при определении primary и standby)
Запуск, чтобы удобно читать логи: pgpool -n (на ctrl+c - выключается)

Дать права пользователю postgres на выполнение команд от root (или настроить по-человечески) и ещё ему при добавлении новых файлов (в нашем случае это были скрипты) дать на них права.

Вывод

Pgpool прикольная вещь, но так как нормальных гайдов нет - ****** все это доп задание. Да оно крутое и очень полезное. Но это ******. Из простой достаточно лабы сделать полностью новую с 0 почти лабу одним допом :/
******. Главное я это сделал и сдал, оно работает и точка!