Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ОТЧЕТ

по практической работе
«HA Postgres Cluster»

по дисциплине «**Администрирование компьютерных сетей**»

Автор: Кахикало К.Р.

Факультет: ИКТ

Группа: К3340

Преподаватель: Самохин Н. Ю.

Санкт-Петербург 2025г.

Установил docker на виртуальную машину:

```
Preparing to unpack .../4-docker-ce_5%3a27.5.1-1~ubuntu.24.04~noble_amd64.deb ...
Unpacking docker-ce (5:27.5.1-1~ubuntu.24.04~noble) ...
Selecting previously unselected package docker-ce-rootless-extras.
Preparing to unpack .../5-docker-ce-rootless-extras_5%3a27.5.1-1~ubuntu.24.04~noble_amd64.deb ...
Unpacking docker-ce-rootless-extras (5:27.5.1-1~ubuntu.24.04~noble) ...
Selecting previously unselected package docker-compose-plugin.
Preparing to unpack .../6-docker-compose-plugin_2.32.4-1~ubuntu.24.04~noble_amd64.deb ...
Unpacking docker-compose-plugin (2.32.4-1~ubuntu.24.04~noble) ...
Selecting previously unselected package libltdl7:amd64.
Preparing to unpack .../7-libltdl7_2.4.7-7build1_amd64.deb ...
Unpacking libltdl7:amd64 (2.4.7-7build1) ...
Selecting previously unselected package libslirp0:amd64.
Preparing to unpack .../8-libslirp0_4.7.0-1ubuntu3_amd64.deb ...
Unpacking libslirp0:amd64 (4.7.0-1ubuntu3) ...
Selecting previously unselected package slirp4netns.
Preparing to unpack .../9-slirp4netns_1.2.1-1build2_amd64.deb ...
Unpacking slirp4netns (1.2.1-1build2) ...
Setting up docker-buildx-plugin (0.20.0-1~ubuntu.24.04~noble) ...
Setting up containerd.io (1.7.25-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
Setting up docker-compose-plugin (2.32.4-1~ubuntu.24.04~noble) ...
Setting up libltdl7:amd64 (2.4.7-7build1) ...
Setting up docker-ce-cli (5:27.5.1-1~ubuntu.24.04~noble) ...
Setting up libslirp0:amd64 (4.7.0-1ubuntu3) ...
Setting up pigz (2.8-1) ...
Setting up docker-ce-rootless-extras (5:27.5.1-1~ubuntu.24.04~noble) ...
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.5.1-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.3) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@lab-3:~# sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:d715f14f9eca81473d9112df50457893aa4d099adeb4729f679006bf5ea12407
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Настраиваю конфиг для postgres. Избавляюсь от master и slave и заменяю на номера, чтобы не путаться.

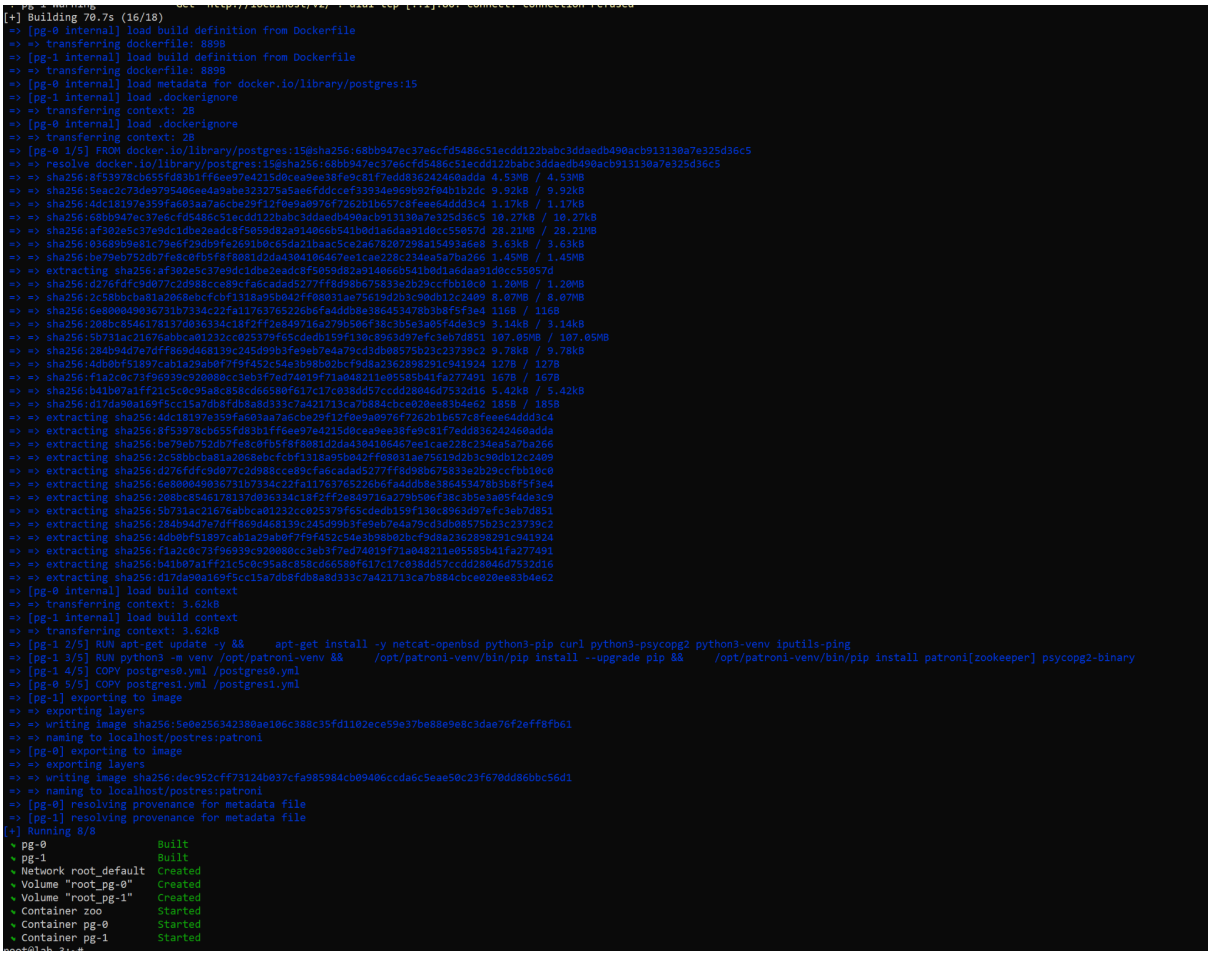
```yaml
scope: my_cluster # Имя нашего кластера
name: postgresql0 # Имя первой ноды

restapi: # Адреса первой ноды
  listen: pg-0:8008
  connect_address: pg-0:8008

zookeeper:
    hosts:
      - zoo:2181 # Адрес Zookeeper

bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 10485760
    master_start_timeout: 300
    synchronous_mode: true
    postgresql:
      use_pg_rewind: true
      use_slots: true
      parameters:
        wal_level: replica
        hot_standby: "on"
        wal_keep_segments: 8
        max_wal_senders: 10
        max_replication_slots: 10
        wal_log_hints: "on"
        archive_mode: "always"
        archive_timeout: 1800s
        archive_command: mkdir -p /tmp/wal_archive && test ! -f /tmp/wal_archive/%f && cp %p /tmp/wal_archive/%f

  pg_hba:
  - host replication replicator 0.0.0.0/0 md5
  - host all all 0.0.0.0/0 md5

postgresql:
  listen: 0.0.0.0:5432
  connect_address: pg-0:5432 # Адрес первой ноды
  data_dir: /var/lib/postgresql/data/postgresql0 # Место хранения данных первой ноды
  bin_dir: /usr/lib/postgresql/15/bin
  pgpass: /tmp/pgpass0
  authentication:
    replication: # логопасс для репликаци, при желании можно поменять
      username: replicator
      password: rep-pass
    superuser: # администский логопасс, при желании можно поменять (в том числе в файле compose)
      username: postgres
      password: postgres
  parameters:
    unix_socket_directories: '.'

watchdog:
  mode: off

tags:
    nofailover: false
    noloadbalance: false
    clonefrom: false
    nosync: false
```

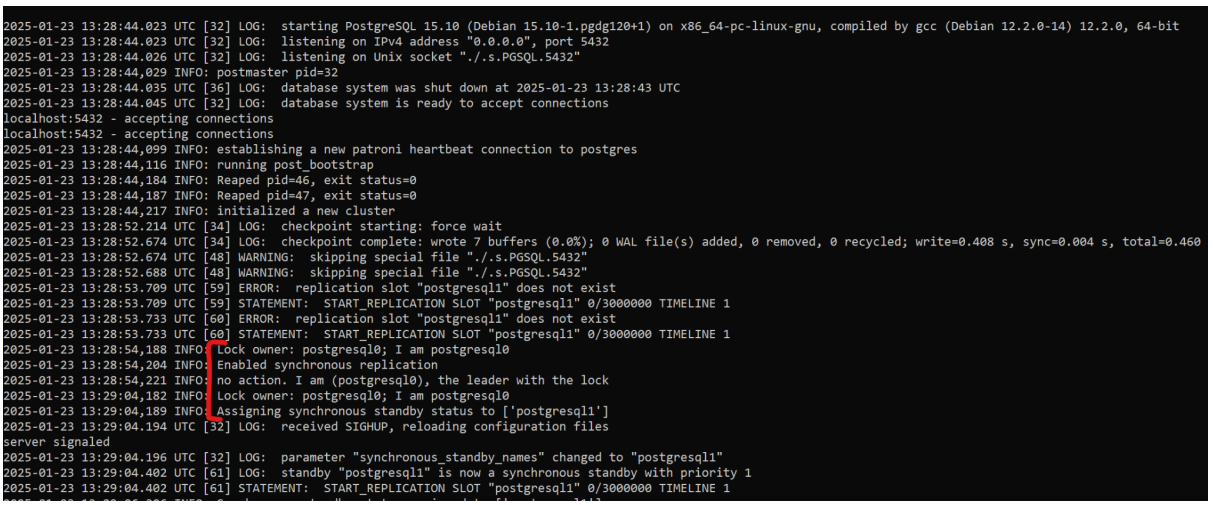
```yaml
scope: my_cluster # Имя нашего кластера
name: postgresql1 # Имя первой ноды

restapi: # Адреса первой ноды
  listen: pg-1:8008
  connect_address: pg-1:8008

zookeeper:
    hosts:
      - zoo:2181 # Адрес Zookeeper

bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 10485760
    master_start_timeout: 300
    synchronous_mode: true
    postgresql:
      use_pg_rewind: true
      use_slots: true
      parameters:
        wal_level: replica
        hot_standby: "on"
        wal_keep_segments: 8
        max_wal_senders: 10
        max_replication_slots: 10
        wal_log_hints: "on"
        archive_mode: "always"
        archive_timeout: 1800s
        archive_command: mkdir -p /tmp/wal_archive && test ! -f /tmp/wal_archive/%f && cp %p /tmp/wal_archive/%f

  pg_hba:
  - host replication replicator 0.0.0.0/0 md5
  - host all all 0.0.0.0/0 md5

postgresql:
  listen: 0.0.0.0:5432
  connect_address: pg-1:5432 # Адрес первой ноды
  data_dir: /var/lib/postgresql/data/postgresql1 # Место хранения данных первой ноды
  bin_dir: /usr/lib/postgresql/15/bin
  pgpass: /tmp/pgpass0
  authentication:
    replication: # логопасс для репликаци, при желании можно поменять
      username: replicator
      password: rep-pass
    superuser: # адмнинский логопасс, при желании можно поменять (в том числе в файле compose)
      username: postgres
      password: postgres
  parameters:
    unix_socket_directories: '.'

watchdog:
  mode: off

tags:
    nofailover: false
    noloadbalance: false
    clonefrom: false
    nosync: false
```

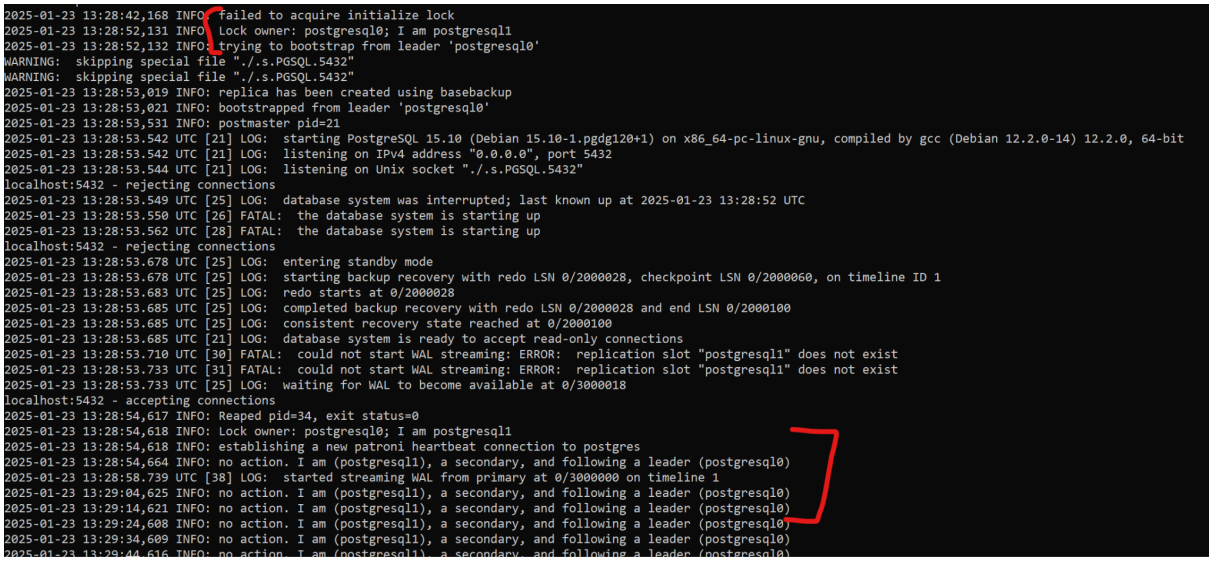
В docker-compose.yml так же избавляюсь от master и slave.

```yaml
services:
  pg-0:
    build: .
    image: localhost/postres:patroni # имя для кастомного образа из Dockerfile, можно задать любое
    container_name: pg-0 # Будущий адрес первой ноды
    restart: always
    hostname: pg-0
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      PGDATA: '/var/lib/postgresql/data/pgdata'
    expose:
      - 8008
    ports:
      - 5433:5432
    volumes:
      - pg-0:/var/lib/postgresql/data
    command: patroni /postgres0.yml

  pg-1:
    build: .
    image: localhost/postres:patroni # имя для кастомного образа из Dockerfile, можно задать любое
    container_name: pg-1 # Будущий адрес второй ноды
    restart: always
    hostname: pg-1
    expose:
      - 8008
    ports:
      - 5434:5432
    volumes:
     - pg-1:/var/lib/postgresql/data
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      PGDATA: '/var/lib/postgresql/data/pgdata'
    command: patroni /postgres1.yml

  zoo:
    image: confluentinc/cp-zookeeper:7.7.1
    container_name: zoo # Будущий адрес зукипера
    restart: always
    hostname: zoo
    ports:
      - 2181:2181
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000

volumes:
  pg-0:
  pg-1:
```

Запускаю compose и проверяю что ноды postgres договорились о том, кто главный.







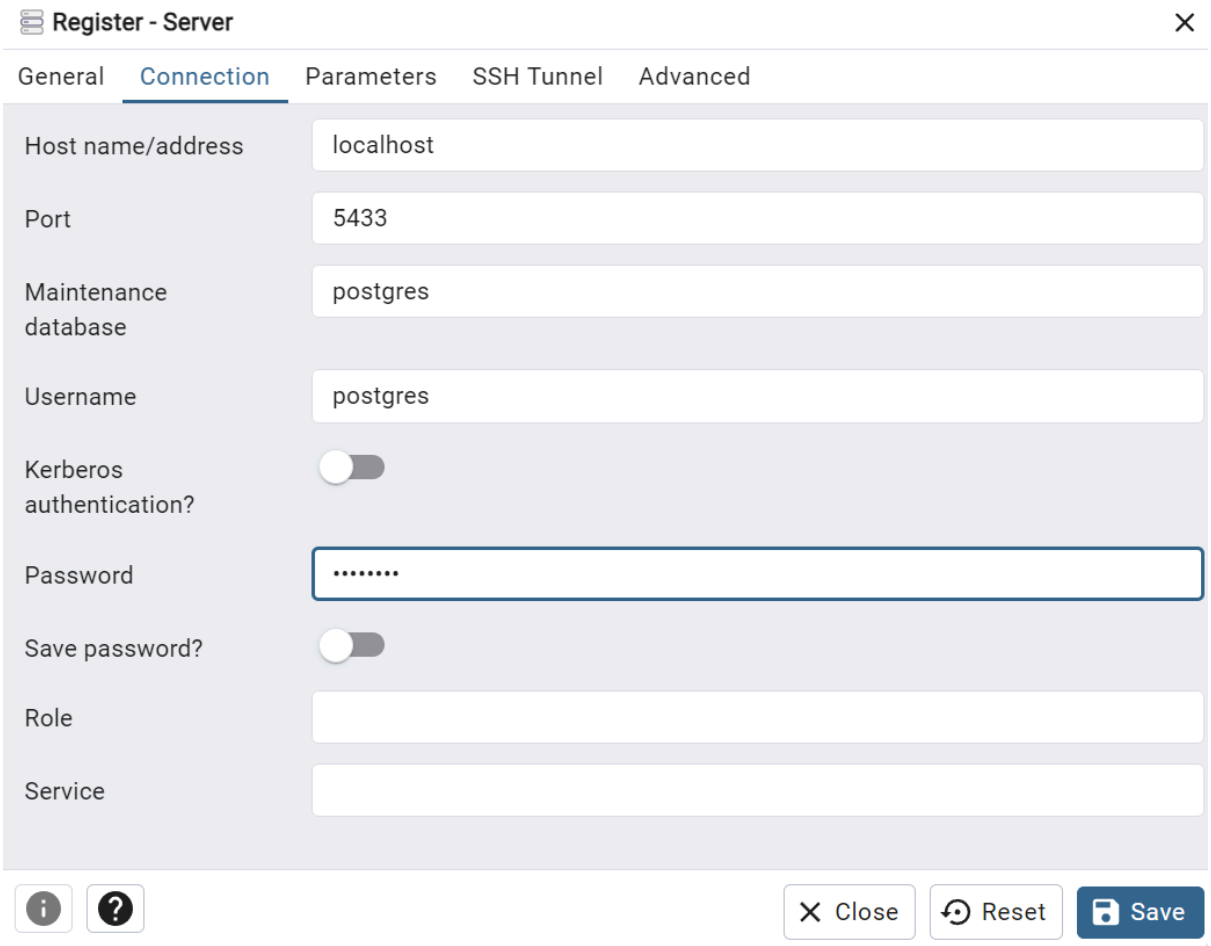
Порты 8008 и 5432 вынесены в разные директивы, expose и ports. По сути, если записать 8008 в ports, то он тоже станет exposed. В чем разница?
Expose даёт доступ к обращениям к порту другим контейнерам, но не хосту. ports сопоставляет порт хоста и порт контейнера так, чтобы при обращении к порту хоста можно было бы подключиться к определённому порту контейнера. При этом в ports можно еще и указать интерфейс, по которому порт будет доступен на хосте.

При обычном перезапуске композ-проекта, будет ли сбилден заново образ? А если предварительно отредактировать файлы postgresX.yml? А если содержимое самого Dockerfile? Почему?
При обычной перезагрузке не будет. Если предварительно отредактировать конфиг файл тоже. Если поменяется содержимое Dockerfile, то новое изображение будет создано. Потому что docker смотрит только на содержимое файла для сборки изображения, а конфиги, которые указаны там, он не может проверить.

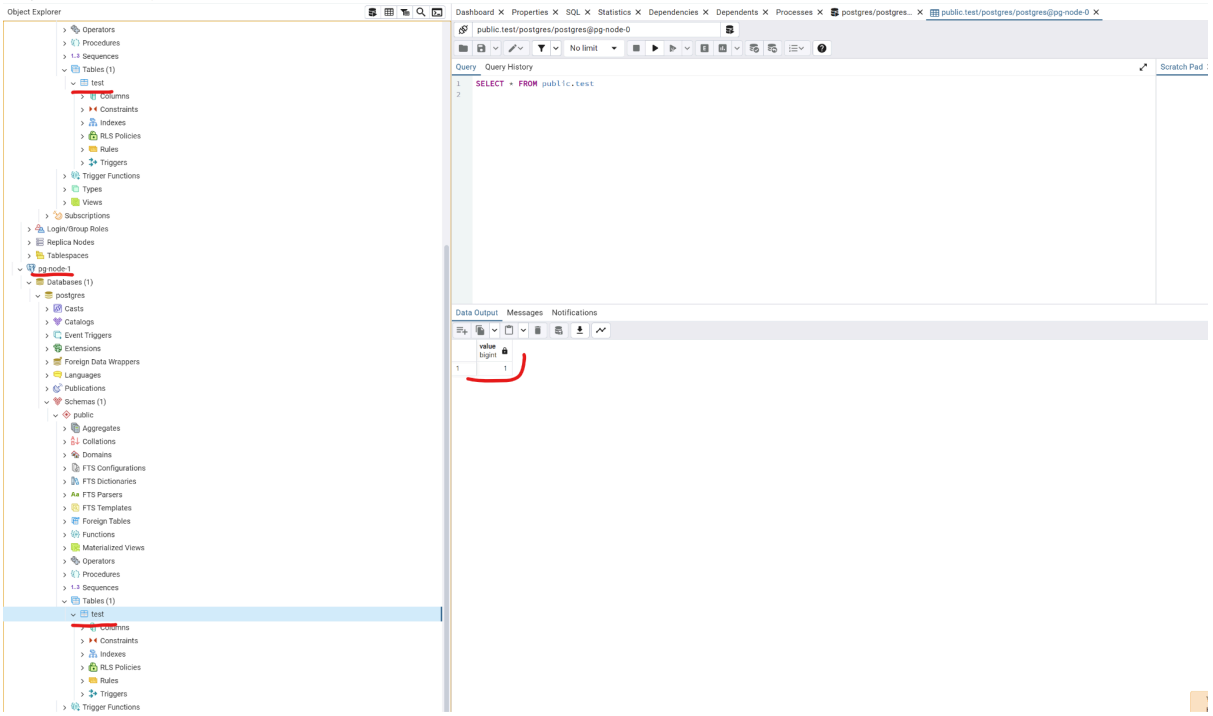
Получаю доступ к портам баз данных из локальной машины:

	Listen interface	Port	Destination host	Port	Comment
☑	127.0.0.1	8080	localhost	8080	
☑	127.0.0.1	8082	localhost	8082	
☑	127.0.0.1	3000	localhost	3000	
☑	127.0.0.1	5433	localhost	5433	
☑	127.0.0.1	5434	localhost	5434	

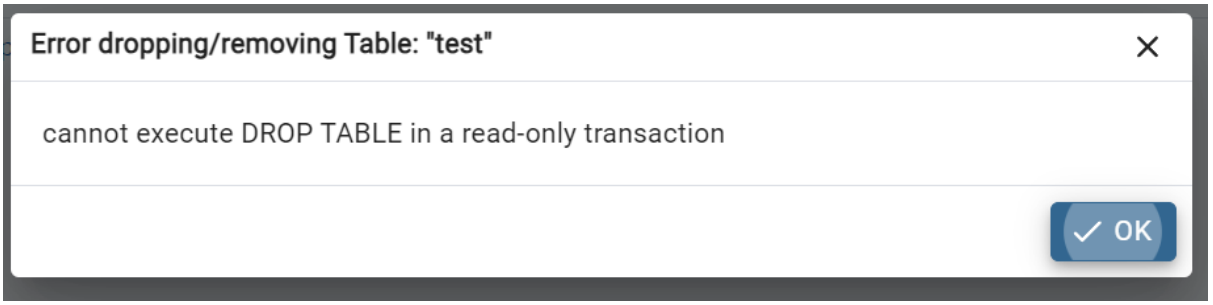
Добавляю подключение к нодам postgres:



Создал таблицу в мастере, добавил туда значение, таблица автоматически создалась в slave.



Удалить таблицу из slave не получилось.



Добавляю haproxy в docker-compose:

```yaml
services:
  pg-0:
    build: .
    image: localhost/postres:patroni # имя для кастомного образа из Dockerfile, можно задать любое
    container_name: pg-0 # Будущий адрес первой ноды
    restart: always
    hostname: pg-0
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      PGDATA: '/var/lib/postgresql/data/pgdata'
    expose:
      - 8008
    ports:
      - 5433:5432
    volumes:
      - pg-0:/var/lib/postgresql/data
    command: patroni /postgres0.yml

  pg-1:
    build: .
    image: localhost/postres:patroni # имя для кастомного образа из Dockerfile, можно задать любое
    container_name: pg-1 # Будущий адрес второй ноды
    restart: always
    hostname: pg-1
    expose:
      - 8008
    ports:
      - 5434:5432
    volumes:
     - pg-1:/var/lib/postgresql/data
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      PGDATA: '/var/lib/postgresql/data/pgdata'
    command: patroni /postgres1.yml

  zoo:
    image: confluentinc/cp-zookeeper:7.7.1
    container_name: zoo # Будущий адрес зукипера
    restart: always
    hostname: zoo
    ports:
      - 2181:2181
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000

  haproxy:
    image: haproxy:3.0
    container_name: postgres_entrypoint # Это будет адрес подключения к БД, можно выбрать любой
    ports:
      - 5432:5432 # Это будет порт подключения к БД, можно выбрать любой
      - 7000:7000
    depends_on: # Не забываем убедиться, что сначала все корректно поднялось
      - pg-0
      - pg-1
      - zoo
    volumes:
      - ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg

volumes:
  pg-0:
  pg-1:
```

Добавляю конфиг для haproxy и немного меняю названия нод:

```
global
    maxconn 100

defaults
    log global
    mode tcp
    retries 3
    timeout client 30m
    timeout connect 4s
    timeout server 30m
    timeout check 5s

listen stats
    mode http
    bind *:7000
    stats enable
    stats uri /

listen postgres
    bind *:5432 # Выбранный порт из docker-compose.yml
    option httpchk
    http-check expect status 200 # Описываем нашу проверку доступности (в данном случае обычный HTTP-пинг)
    default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
    server postgresql_pg_0_5432 pg-0:5432 maxconn 100 check port 8008 # Адрес первой ноды постгреса
    server postgresql_pg_1_5432 pg-1:5432 maxconn 100 check port 8008 # Адрес второй ноды постгреса
```

Перезапускаю compose и проверяю, что всё работает:

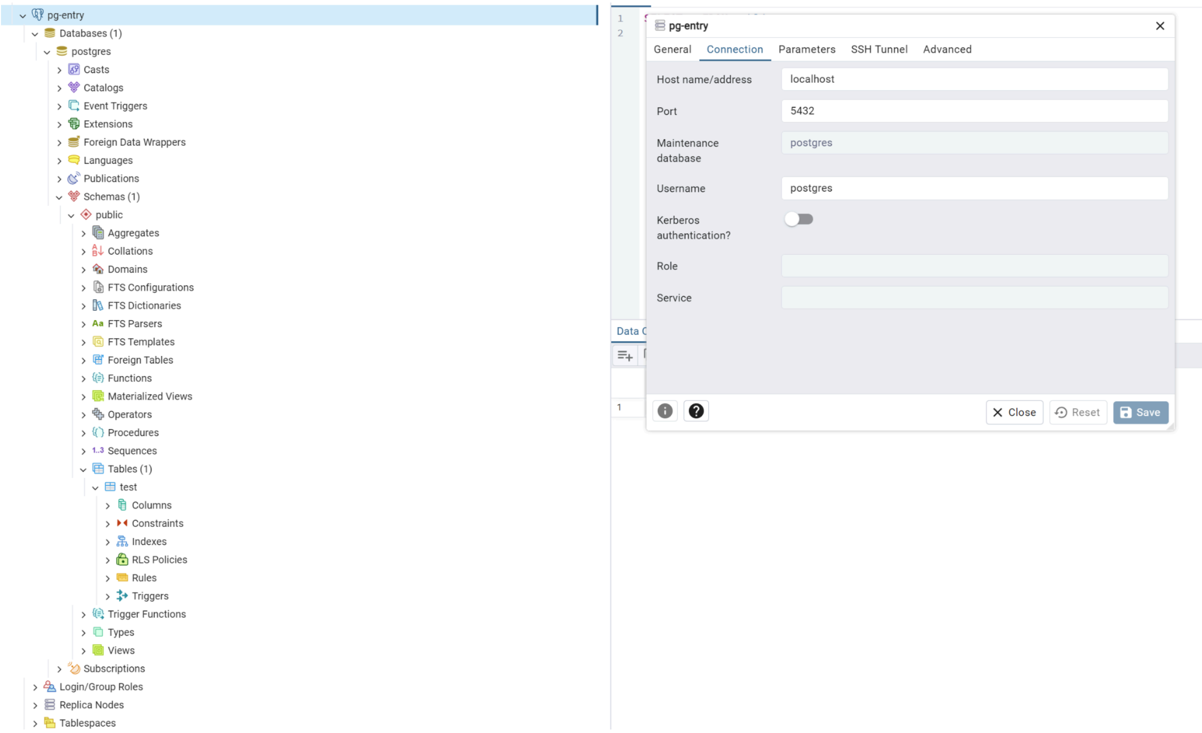
```
root@lab-3:~# docker compose up -d
[+] Running 7/7
 ✔ haproxy Pulled
   ✔ af302e5c37e9 Already exists
   ✔ 89d67b1065a9 Pull complete
   ✔ 8855209f7173 Pull complete
   ✔ 444fc48b92cc Pull complete
   ✔ 4252e78e7e4a Pull complete
   ✔ 4f4fb700ef54 Pull complete
[+] Running 4/4
 ✔ Container zoo                  Running
 ✔ Container pg-0                 Running
 ✔ Container pg-1                 Running
 ✔ Container postgres_entrypoint  Started
```

```
2025-01-23 15:44:17,882 INFO: Lock owner: None; I am postgresql0
2025-01-23 15:44:17,896 INFO: Reaped pid=18, exit status=0
2025-01-23 15:44:17,896 INFO: Lock owner: None; I am postgresql0
2025-01-23 15:44:17,897 INFO: starting as a secondary
2025-01-23 15:44:18.492 UTC [24] LOG:  starting PostgreSQL 15.10 (Debian 15.10-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.
2025-01-23 15:44:18.494 UTC [24] LOG:  listening on IPv4 address "0.0.0.0", port 5432
2025-01-23 15:44:18.499 UTC [24] LOG:  listening on Unix socket "./.s.PGSQL.5432"
2025-01-23 15:44:18.509 UTC [27] LOG:  database system was interrupted while in recovery at log time 2025-01-23 15:44:03 UTC
2025-01-23 15:44:18.509 UTC [27] HINT:  If this has occurred more than once some data might be corrupted and you might need to choose an earlie
2025-01-23 15:44:18,544 INFO: postmaster pid=24
2025-01-23 15:44:18.562 UTC [29] FATAL:  the database system is starting up
localhost:5432 - rejecting connections
2025-01-23 15:44:18.578 UTC [31] FATAL:  the database system is starting up
localhost:5432 - rejecting connections
2025-01-23 15:44:18.665 UTC [27] WARNING:  specified neither primary_conninfo nor restore_command
2025-01-23 15:44:18.665 UTC [27] HINT:  The database server will regularly poll the pg_wal subdirectory to check for files placed there.
2025-01-23 15:44:18.665 UTC [27] LOG:  entering standby mode
2025-01-23 15:44:18.671 UTC [27] LOG:  consistent recovery state reached at 0/50000A0
2025-01-23 15:44:18.671 UTC [27] LOG:  invalid record length at 0/50000A0: wanted 24, got 0
2025-01-23 15:44:18.671 UTC [27] LOG:  waiting for WAL to become available at 0/50000B8
2025-01-23 15:44:18.672 UTC [24] LOG:  database system is ready to accept read-only connections
localhost:5432 - accepting connections
2025-01-23 15:44:19,652 INFO: establishing a new patroni heartbeat connection to postgres
2025-01-23 15:44:19,734 INFO: Got response from postgresql1 http://pg-1:8008/patroni: {"state": "starting", "postmaster_start_time": "2025-01-2
d_timestamp": null, "paused": false}, "timeline": 1, "cluster_unlocked": true, "dcs_last_seen": 1737647058, "database_system_identifier": "7463
2025-01-23 15:44:19,793 INFO: promoted self to leader by acquiring session lock
2025-01-23 15:44:19.796 UTC [27] LOG:  received promote request
2025-01-23 15:44:19.796 UTC [27] LOG:  redo is not required
server promoting
2025-01-23 15:44:19.800 UTC [27] LOG:  selected new timeline ID: 2
2025-01-23 15:44:19.842 UTC [27] LOG:  archive recovery complete
2025-01-23 15:44:19.850 UTC [25] LOG:  checkpoint starting: force
2025-01-23 15:44:19.859 UTC [24] LOG:  database system is ready to accept connections
2025-01-23 15:44:19.863 UTC [25] LOG:  checkpoint complete: wrote 3 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.003 s,
2025-01-23 15:44:20,051 INFO: establishing a new patroni restapi connection to postgres
2025-01-23 15:44:20,837 INFO: Lock owner: postgresql0; I am postgresql0
2025-01-23 15:44:20,853 INFO: Reaped pid=58, exit status=0
2025-01-23 15:44:20,854 INFO: Assigning synchronous standby status to ['postgresql1']
2025-01-23 15:44:20.858 UTC [24] LOG:  received SIGHUP, reloading configuration files
server signaled
2025-01-23 15:44:20.859 UTC [24] LOG:  parameter "synchronous_standby_names" changed to "postgresql1"
2025-01-23 15:44:21.066 UTC [54] LOG:  standby "postgresql1" is now a synchronous standby with priority 1
2025-01-23 15:44:21.066 UTC [54] STATEMENT:  START_REPLICATION SLOT "postgresql1" 0/5000000 TIMELINE 2
2025-01-23 15:44:22,977 INFO: Synchronous standby status assigned to ['postgresql1']
2025-01-23 15:44:22,995 INFO: no action. I am (postgresql0), the leader with the lock
2025-01-23 15:44:30,831 INFO: no action. I am (postgresql0), the leader with the lock
root@lab-3:~#
```

Добавляю порт для подключения через haproxy на локальную машину:

	Listen interface	Port	Destination host	Port	Comment
☑	127.0.0.1	8080	localhost	8080	
☑	127.0.0.1	8082	localhost	8082	
☑	127.0.0.1	3000	localhost	3000	
☑	127.0.0.1	5433	localhost	5433	
☑	127.0.0.1	5434	localhost	5434	
☑	127.0.0.1	5432	localhost	5432	

Добавил подключение через haproxy и убедился, что там отображается ранее созданная таблица:



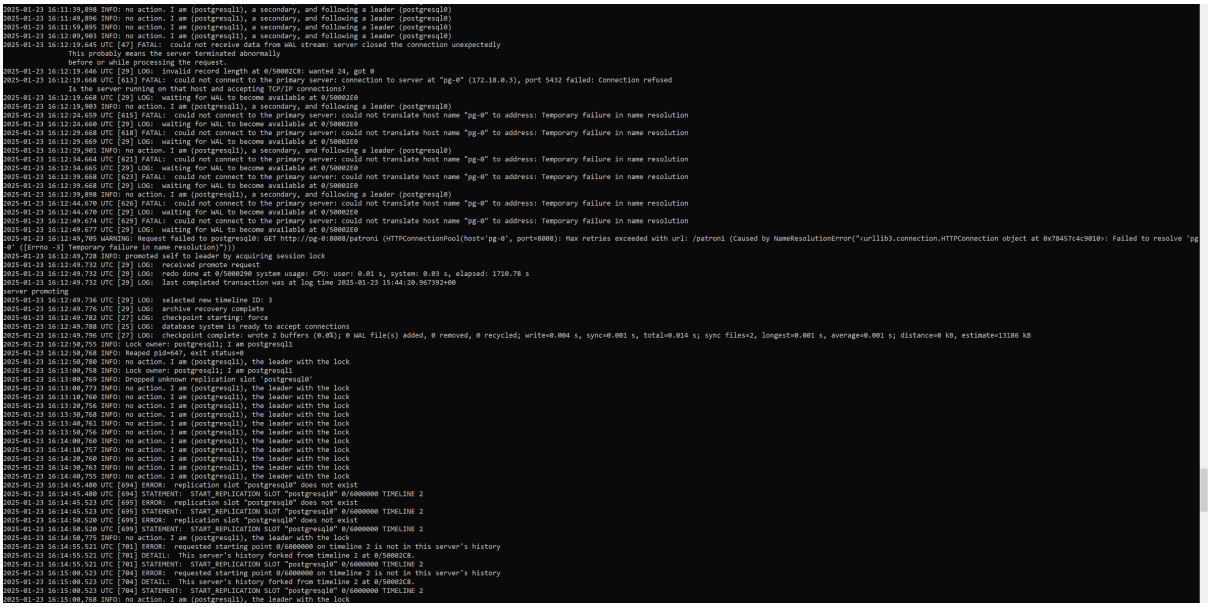
Если остановить первую ноду, через haproxy внести изменения на вторую ноду, которая стала мастером, а потом поднять первую ноду, то:

1. Нода 1 станет ведомой
2. Она не запустится

```
2025-01-23 16:17:01,255 INFO: Lock owner: postgresql1; I am postgresql0
2025-01-23 16:17:01,256 INFO: starting as a secondary
2025-01-23 16:17:01.768 UTC [69] LOG:  starting PostgreSQL 15.10 (Debian 15.10-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
2025-01-23 16:17:01.768 UTC [69] LOG:  listening on IPv4 address "0.0.0.0", port 5432
2025-01-23 16:17:01.770 UTC [69] LOG:  listening on Unix socket "./.s.PGSQL.5432"
2025-01-23 16:17:01,776 INFO: postmaster pid=69
2025-01-23 16:17:01.779 UTC [73] LOG:  database system was shut down at 2025-01-23 16:16:40 UTC
2025-01-23 16:17:01.780 UTC [73] LOG:  entering standby mode
2025-01-23 16:17:01.780 UTC [73] FATAL:  requested timeline 3 is not a child of this server's history
2025-01-23 16:17:01.780 UTC [73] DETAIL:  Latest checkpoint is at 0/7000028 on timeline 2, but in the history of the requested timeline, the server forked off from that timeline at 0/50002C8.
2025-01-23 16:17:01.784 UTC [69] LOG:  startup process (PID 73) exited with exit code 1
2025-01-23 16:17:01.784 UTC [69] LOG:  aborting startup due to startup process failure
2025-01-23 16:17:01.785 UTC [69] LOG:  database system is shut down
localhost:5432 - no response
2025-01-23 16:17:02,792 ERROR: postmaster is not running
2025-01-23 16:17:11,242 INFO: Lock owner: postgresql1; I am postgresql0
2025-01-23 16:17:11,242 INFO: failed to start postgres
root@lab-3:~# docker ps
CONTAINER ID   IMAGE                              COMMAND                  CREATED          STATUS          PORTS                                                                          NAMES
a7ee13ead119   haproxy:3.0                        "docker-entrypoint.s…"   33 minutes ago   Up 33 minutes   0.0.0.0:5432->5432/tcp, :::5432->5432/tcp, 0.0.0.0:7000->7000/tcp, :::7000->7000/tcp   postgres_entrypoint
e9e2cd5c67f1   confluentinc/cp-zookeeper:7.7.1   "/etc/confluent/dock…"   33 minutes ago   Up 33 minutes   2888/tcp, 0.0.0.0:2181->2181/tcp, :::2181->2181/tcp, 3888/tcp                  zoo
146ef294bfa1   localhost/postres:patroni          "docker-entrypoint.s…"   33 minutes ago   Up 36 seconds   8008/tcp, 0.0.0.0:5433->5432/tcp, [::]:5433->5432/tcp                          pg-0
667af4f6fa1f   localhost/postres:patroni          "docker-entrypoint.s…"   33 minutes ago   Up 33 minutes   8008/tcp, 0.0.0.0:5434->5432/tcp, [::]:5434->5432/tcp                          pg-1
```

Data Output Messages Notifications

```
connection failed: connection to server at "127.0.0.1", port 5433 failed: server closed the connection unexpectedly
        This probably means the server terminated abnormally
        before or while processing the request.
```



В начале ведомая нода пытается получить актуальный WAL, потом хотя бы подключиться к мастеру, но у неё не выходит. Через время она становится мастером. Во время внесения изменений в новый мастер, создаётся новый таймлайн. Изменения вносятся через entrypoint успешно и к ним, через этот же entrypoint можно иметь доступ. Затем старая нода поднимается. Она работала на втором таймлайне, но этого второго таймлайна нет на новом сервере. Из-за этого старая нода не может успешно стартовать.