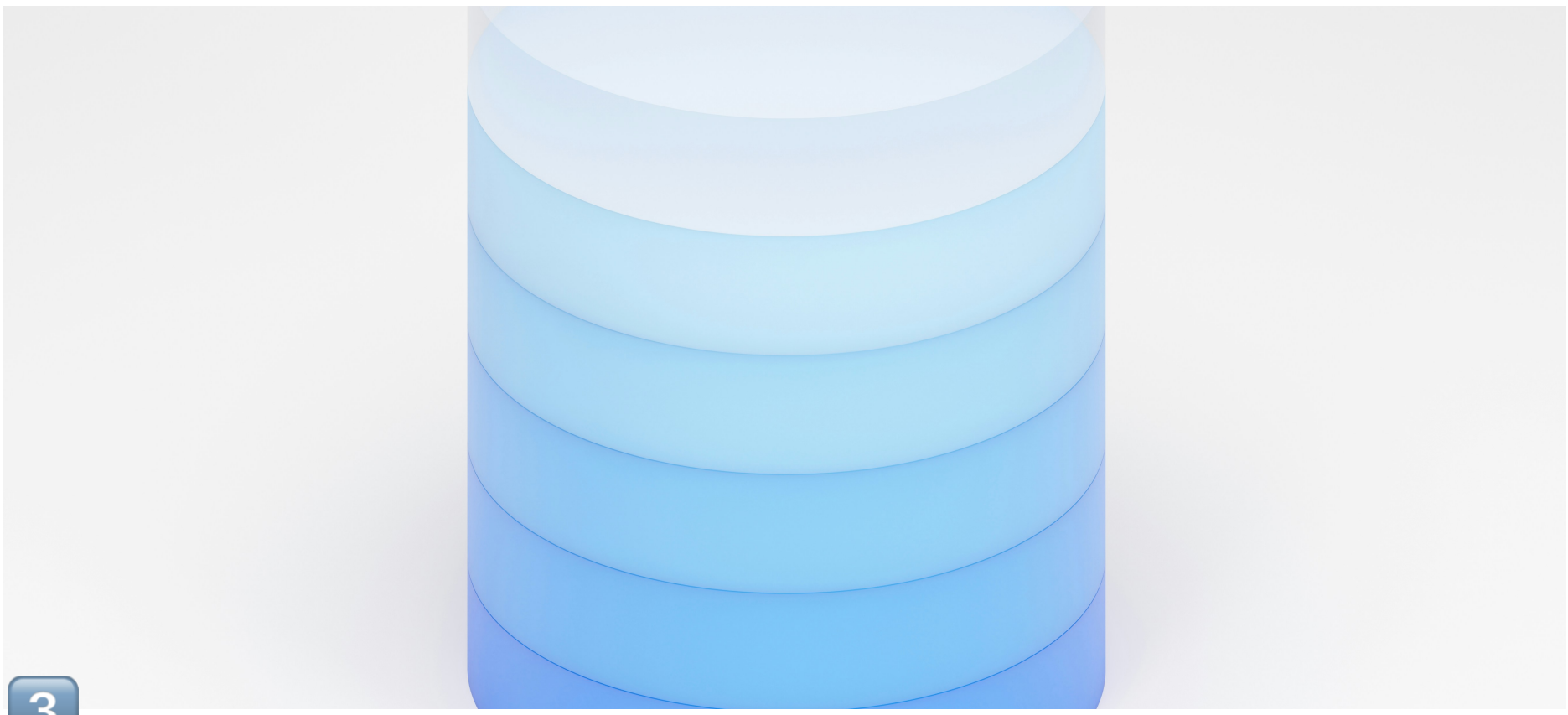
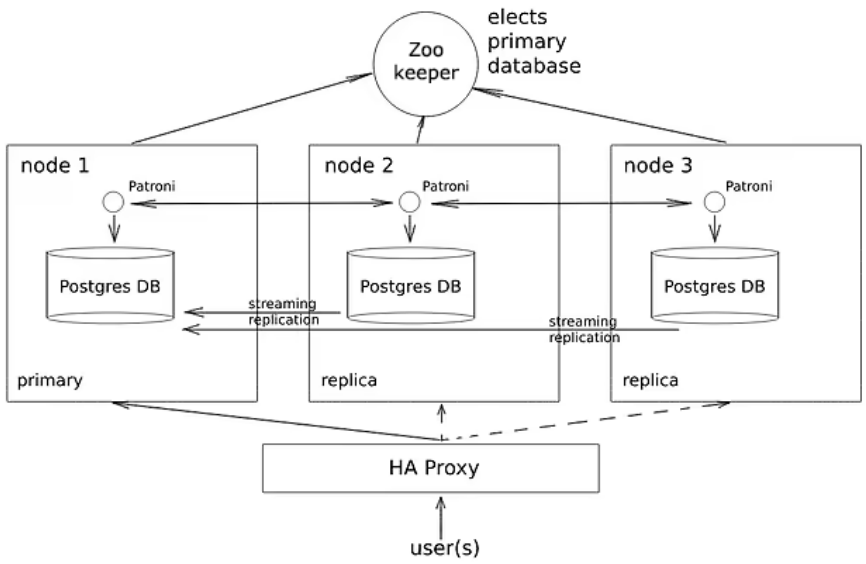


ЛР 3. HA Postgres Cluster

▼ **Задача**

Развернуть и настроить высокодоступный кластер Postgres



💡 Работу можно выполнять локально или на выданной ВМ (возможно, придется доустановить Docker)

▼ **Гайд**

▼ Часть 1. Поднимаем Postgres

1. Подготавливаем Dockerfile для нашего постгреса. Кластеризацию будем делать с помощью Patroni, а ему необходим доступ к бинарникам самого постгреса. Поэтому будем билдить образ, который сразу содержит в себе Postgres + Patroni

▼ `Dockerfile`

```
FROM postgres:15

# Ставим нужные для Patroni зависимости
RUN apt-get update -y && \
    apt-get install -y netcat-openbsd python3-pip curl python3-psycopg2 python3-venv iputils-ping

# Используем виртуальное окружение, доустанавливаем, собственно, Patroni
RUN python3 -m venv /opt/patroni-venv && \
    /opt/patroni-venv/bin/pip install --upgrade pip && \
    /opt/patroni-venv/bin/pip install patroni[zookeeper] psycopg2-binary

# Копируем конфигурацию для двух узлов кластера Patroni
COPY postgres0.yml /postgres0.yml
COPY postgres1.yml /postgres1.yml

ENV PATH="/opt/patroni-venv/bin:$PATH"
```

```
USER postgres

#CMD не задаем, т.к. все равно будем переопределять его далее в compose
```

2. Подготавливаем compose файл, в котором описываем наш деплой постгреса. Так же добавляем в него Zookepeer, который нужен для непосредственного управления репликацией и определением “лидера” кластера

▼ `docker-compose.yml`

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

  pg-slave:
    build: .
    image: localhost/postres:patroni # имя для кастомного образа из Dockerfile, можно задать любое
    container_name: pg-slave # Будущий адрес второй ноды
    restart: always
    hostname: pg-slave
    expose:
      - 8008
    ports:
      - 5434:5432
    volumes:
     - pg-slave:/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-master:
  pg-slave:
```

> 💡 *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?*

3. Создаем упомянутые выше `postgres0.yml` и затем на основе него — `postgres1.yml` (надо будешь лишь поменять *имя, адреса и место хранения данных* ноды с первой на вторую)

▼ `postgres0.yml`

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

restapi: # Адреса первой ноды
  listen: pg-master:8008
  connect_address: pg-master: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-master: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: # админский 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
```

4. Деплоим. Проверяем в логах, что зукипер запустился, и что одна нода постгреса из двух стала лидером/овнером/мастером (***есть вероятность, что вопреки названию это будет НЕ pg-master, это нормально!***)

   ▼ *Пример*

```
[+] Building 0.3s (14/16)                                                                                                             docker:default
 => [pg-master internal] load build definition from Dockerfile                                                                                  0.0s
 => => transferring dockerfile: 686B                                                                                                            0.0s
 => [pg-slave internal] load metadata for docker.io/library/postgres:15                                                                         0.0s
 => [pg-master internal] load .dockerignore                                                                                                     0.0s
 => => transferring context: 2B                                                                                                                 0.0s
 => [pg-slave internal] load build definition from Dockerfile                                                                                   0.0s
 => => transferring dockerfile: 686B                                                                                                            0.0s
 => [pg-slave 1/5] FROM docker.io/library/postgres:15                                                                                           0.0s
 => [pg-master internal] load build context                                                                                                     0.0s
 => => transferring context: 68B                                                                                                                0.0s
 => [pg-slave internal] load .dockerignore                                                                                                      0.0s
 => => transferring context: 2B                                                                                                                 0.0s
 => CACHED [pg-master 2/5] RUN apt-get update -y &&     apt-get install -y netcat-openbsd python3-pip curl python3-psycopg2 python3-venv iputils-ping      0.0s
 => CACHED [pg-master 3/5] RUN python3 -m venv /opt/patroni-venv &&     /opt/patroni-venv/bin/pip install --upgrade pip &&     /opt/patroni-venv/bin/pip install patroni[zookeeper] ps  0.0s
 => CACHED [pg-master 4/5] COPY postgres0.yml /postgres0.yml                                                                                    0.0s
 => CACHED [pg-slave 5/5] COPY postgres1.yml /postgres1.yml                                                                                     0.0s
 => [pg-master] exporting to image                                                                                                              0.0s
 => => exporting layers                                                                                                                         0.0s
 => => writing image sha256:945a66f0c4055b77e76975bce962ce974613b196b7344efd5bd751605e512f2a                                                    0.0s
 => => naming to localhost/postres:patroni                                                                                                      0.0s
 => [pg-slave internal] load build context                                                                                                      0.0s
 => => transferring context: 68B                                                                                                                0.0s
 => [pg-slave] exporting to image                                                                                                               0.1s
 => => exporting layers                                                                                                                         0.0s
 => => writing image sha256:ce75dc992e6a5421524a2a1c832b319d6e0d7bdd139fcf39575da229f8ab4f64                                                    0.0s
 => => naming to localhost/postres:patroni                                                                                                      0.0s
[+] Running 4/4
 ✓ Network postgres_lab_default  Created                                                                                                        0.1s
 ✓ Container pg-slave            Started                                                                                                        0.4s
 ✓ Container pg-master           Started                                                                                                        0.4s
 ✓ Container zoo                 Started                                                                                                        0.5s

llidd@winlid MINGW64 ~/postgres_lab
$ docker ps
CONTAINER ID   IMAGE                             COMMAND                  CREATED          STATUS          PORTS                                            NAMES
844c3b77412e   confluentinc/cp-zookeeper:7.7.1   "/etc/confluent/dock…"   26 seconds ago   Up 25 seconds   2888/tcp, 0.0.0.0:2181->2181/tcp, 3888/tcp   zoo
43805a19cbac   localhost/postres:patroni         "docker-entrypoint.s…"   26 seconds ago   Up 25 seconds   8008/tcp, 0.0.0.0:5433->5432/tcp                 pg-master
7ff4472b4b5c   localhost/postres:patroni         "docker-entrypoint.s…"   26 seconds ago   Up 25 seconds   8008/tcp, 0.0.0.0:5434->5432/tcp                 pg-slave
```

```
2024-11-11 21:17:43.966 UTC [24] LOG:  database system is ready to accept read-only connections
2024-11-11 21:17:43.967 UTC [28] LOG:  invalid record length at 0/40000A0: wanted 24, got 0
2024-11-11 21:17:43.967 UTC [28] LOG:  waiting for WAL to become available at 0/40000B8
localhost:5432 - accepting connections
localhost:5432 - accepting connections
2024-11-11 21:17:44,779 INFO: establishing a new patroni heartbeat connection to postgres
2024-11-11 21:17:44,801 INFO: Got response from postgresql1 http://pg-slave:8008/patroni: {"state": "starting", "postmaster_start_time": "2024-11-11 21:17:44.033271+00:00", "role": "replica
", "server_version": 150008, "xlog": {"received_location": 67109024, "replayed_location": 67109024, "replayed_timestamp": null, "paused": false}, "timeline": 1, "cluster_unlocked": true, "d
cs_last_seen": 1731359863, "database_system_identifier": "7436133407519563799", "patroni": {"version": "4.0.3", "scope": "my_cluster", "name": "postgresql1"}}
2024-11-11 21:17:44,825 INFO: promoted self to leader by acquiring session lock
server promoting
2024-11-11 21:17:44.826 UTC [28] LOG:  received promote request
2024-11-11 21:17:44.826 UTC [28] LOG:  redo done at 0/4000028 system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.86 s
2024-11-11 21:17:44.826 UTC [28] LOG:  last completed transaction was at log time 2024-11-11 21:16:49.649892+00
2024-11-11 21:17:44.832 UTC [28] LOG:  selected new timeline ID: 2
2024-11-11 21:17:44.868 UTC [28] LOG:  archive recovery complete
2024-11-11 21:17:44.881 UTC [26] LOG:  checkpoint starting: force
2024-11-11 21:17:44.883 UTC [24] LOG:  database system is ready to accept connections
2024-11-11 21:17:45,849 INFO: Lock owner: postgresql0; I am postgresql0
2024-11-11 21:17:45,856 INFO: Reaped pid=55, exit status=0
2024-11-11 21:17:45,857 INFO: Assigning synchronous standby status to ['postgresql1']
2024-11-11 21:17:45.858 UTC [24] LOG:  received SIGHUP, reloading configuration files
server signaled
2024-11-11 21:17:45.859 UTC [24] LOG:  parameter "synchronous_standby_names" changed to "postgresql1"
2024-11-11 21:17:45.868 UTC [53] LOG:  standby "postgresql1" is now a synchronous standby with priority 1
2024-11-11 21:17:45.868 UTC [53] STATEMENT:  START_REPLICATION SLOT "postgresql1" 0/4000000 TIMELINE 2
2024-11-11 21:17:45.870 UTC [26] LOG:  checkpoint complete: wrote 41 buffers (0.3%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.963 s, sync=0.007 s, total=0.990 s; sync files=11, l
ongest=0.003 s, average=0.001 s; distance=32768 kB, estimate=32768 kB
2024-11-11 21:17:45.870 UTC [26] LOG:  checkpoint starting: immediate force wait
2024-11-11 21:17:45.883 UTC [26] LOG:  checkpoint complete: wrote 0 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.001 s, sync=0.001 s, total=0.014 s; sync files=0, lon
gest=0.000 s, average=0.000 s; distance=0 kB, estimate=29491 kB
2024-11-11 21:17:47,972 INFO: Synchronous standby status assigned to ['postgresql1']
2024-11-11 21:17:47,981 INFO: no action. I am (postgresql0), the leader with the lock
2024-11-11 21:17:47,998 INFO: no action. I am (postgresql0), the leader with the lock
2024-11-11 21:17:57,994 INFO: no action. I am (postgresql0), the leader with the lock
```

```
2024-11-11 21:17:44,788 INFO: establishing a new patroni heartbeat connection to postgres
2024-11-11 21:17:44,793 INFO: establishing a new patroni restapi connection to postgres
localhost:5432 - accepting connections
2024-11-11 21:17:45,060 INFO: Lock owner: postgresql0; I am postgresql1
2024-11-11 21:17:45,063 INFO: Local timeline=1 lsn=0/40000A0
2024-11-11 21:17:45.067 UTC [24] LOG:  received SIGHUP, reloading configuration files
server signaled
2024-11-11 21:17:45.068 UTC [24] LOG:  parameter "primary_conninfo" changed to "dbname=postgres user=replicator passfile=/tmp/pgpass0 host=pg-master port=5432
e=postgresql1 gssencmode=prefer channel_binding=prefer"
2024-11-11 21:17:45.068 UTC [24] LOG:  parameter "primary_slot_name" changed to "postgresql1"
2024-11-11 21:17:45,073 INFO: Reaped pid=43, exit status=0
2024-11-11 21:17:45,076 INFO: no action. I am (postgresql1), a secondary, and following a leader (postgresql0)
2024-11-11 21:17:45.078 UTC [42] LOG:  fetching timeline history file for timeline 2 from primary server
2024-11-11 21:17:45.083 UTC [42] LOG:  started streaming WAL from primary at 0/4000000 on timeline 1
2024-11-11 21:17:45.083 UTC [42] LOG:  replication terminated by primary server
2024-11-11 21:17:45.083 UTC [42] DETAIL:  End of WAL reached on timeline 1 at 0/40000A0.
2024-11-11 21:17:45.084 UTC [42] FATAL:  terminating walreceiver process due to administrator command
2024-11-11 21:17:45.084 UTC [28] LOG:  new target timeline is 2
2024-11-11 21:17:45.095 UTC [47] LOG:  started streaming WAL from primary at 0/4000000 on timeline 2
2024-11-11 21:17:45.120 UTC [28] LOG:  redo starts at 0/40000A0
2024-11-11 21:17:55,061 INFO: Lock owner: postgresql0; I am postgresql1
2024-11-11 21:17:55,064 INFO: Local timeline=2 lsn=0/4000258
2024-11-11 21:17:55,074 INFO: primary_timeline=2
2024-11-11 21:17:55,087 INFO: no action. I am (postgresql1), a secondary, and following a leader (postgresql0)
2024-11-11 21:18:05,065 INFO: no action. I am (postgresql1), a secondary, and following a leader (postgresql0)
2024-11-11 21:18:15,062 INFO: no action. I am (postgresql1), a secondary, and following a leader (postgresql0)
2024-11-11 21:18:25,062 INFO: no action. I am (postgresql1), a secondary, and following a leader (postgresql0)
```

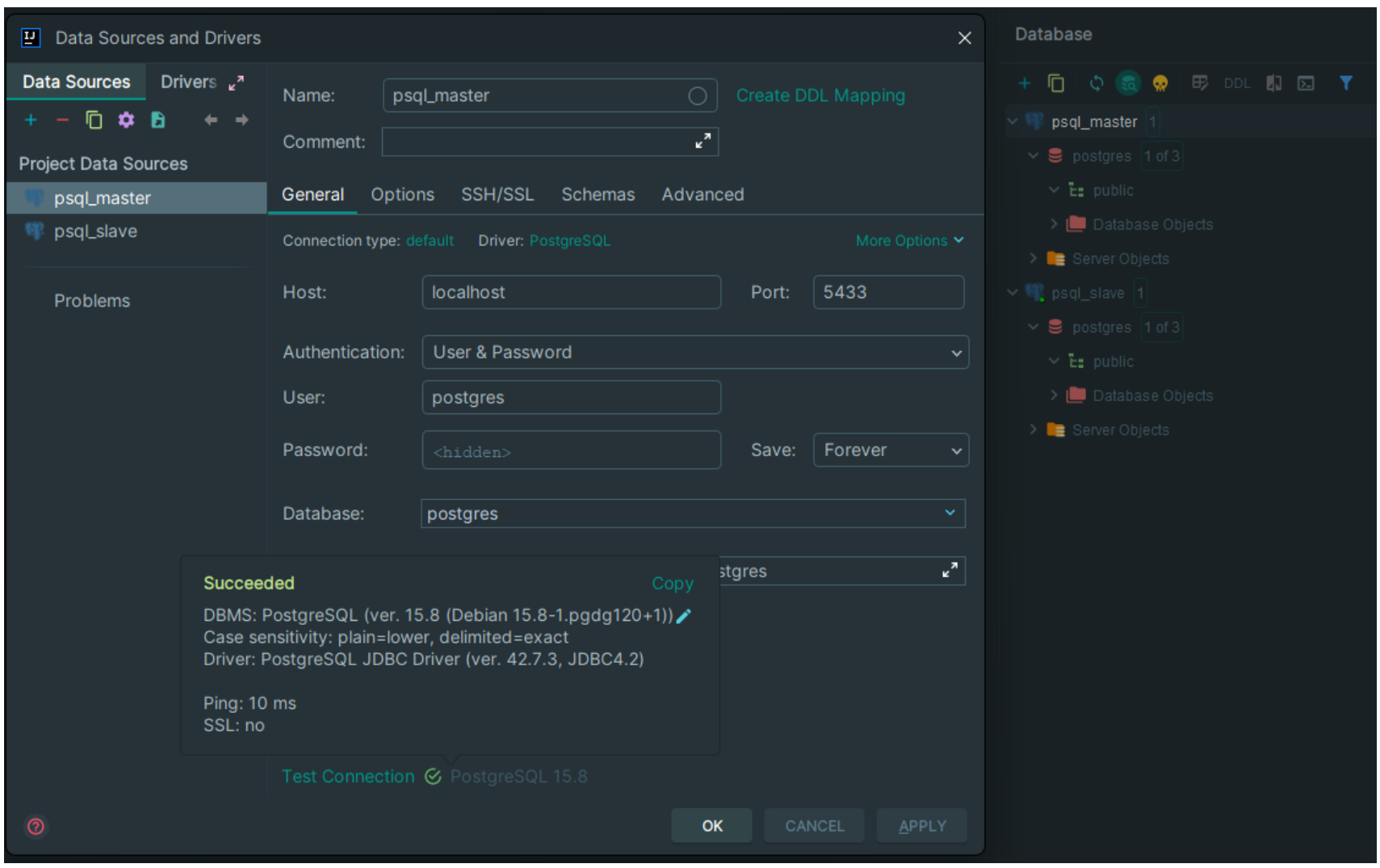
```
[2024-11-11 21:32:10,353] INFO Logging initialized @347ms to org.eclipse.jetty.util.log.Slf4jLog (org.eclipse.jetty.util.log)
[2024-11-11 21:32:10,398] WARN o.e.j.s.ServletContextHandler@229f66ed{/,null,STOPPED} contextPath ends with /* (org.eclipse.jetty.server.handler.ContextHandler)
[2024-11-11 21:32:10,398] WARN Empty contextPath (org.eclipse.jetty.server.handler.ContextHandler)
[2024-11-11 21:32:10,412] INFO jetty-9.4.54.v20240208; built: 2024-02-08T19:42:39.027Z; git: cef3fbd6d736a21e7d541a5db490381d95a2047d; jvm 17.0.12+7-LTS (org.eclipse.jetty.server.Server)
[2024-11-11 21:32:10,432] INFO DefaultSessionIdManager workerName=node0 (org.eclipse.jetty.server.session)
[2024-11-11 21:32:10,432] INFO No SessionScavenger set, using defaults (org.eclipse.jetty.server.session)
[2024-11-11 21:32:10,433] INFO node0 Scavenging every 660000ms (org.eclipse.jetty.server.session)
[2024-11-11 21:32:10,442] WARN ServletContext@o.e.j.s.ServletContextHandler@229f66ed{/,null,STARTING} has uncovered http methods for path: /* (org.eclipse.jetty.security.SecurityHandler)
[2024-11-11 21:32:10,450] INFO Started o.e.j.s.ServletContextHandler@229f66ed{/,null,AVAILABLE} (org.eclipse.jetty.server.handler.ContextHandler)
[2024-11-11 21:32:10,459] INFO Started ServerConnector@564fabc8{HTTP/1.1, (http/1.1)}{0.0.0.0:8080} (org.eclipse.jetty.server.AbstractConnector)
[2024-11-11 21:32:10,459] INFO Started @456ms (org.eclipse.jetty.server.Server)
[2024-11-11 21:32:10,459] INFO Started AdminServer on address 0.0.0.0, port 8080 and command URL /commands (org.apache.zookeeper.server.admin.JettyAdminServer)
[2024-11-11 21:32:10,462] INFO Using org.apache.zookeeper.server.NIOServerCnxnFactory as server connection factory (org.apache.zookeeper.server.ServerCnxnFactory)
[2024-11-11 21:32:10,463] WARN maxCnxns is not configured, using default value 0. (org.apache.zookeeper.server.ServerCnxnFactory)
[2024-11-11 21:32:10,464] INFO Configuring NIO connection handler with 10s sessionless connection timeout, 2 selector thread(s), 24 worker threads, and 64 kB direct buffers. (org.apache.zoo
keeper.server.NIOServerCnxnFactory)
[2024-11-11 21:32:10,464] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2024-11-11 21:32:10,474] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.watch.WatchManagerFactory)
[2024-11-11 21:32:10,474] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.watch.WatchManagerFactory)
[2024-11-11 21:32:10,474] INFO zookeeper.snapshotSizeFactor = 0.33 (org.apache.zookeeper.server.ZKDatabase)
[2024-11-11 21:32:10,474] INFO zookeeper.commitLogCount=500 (org.apache.zookeeper.server.ZKDatabase)
[2024-11-11 21:32:10,478] INFO zookeeper.snapshot.compression.method = CHECKED (org.apache.zookeeper.server.persistence.SnapStream)
[2024-11-11 21:32:10,478] INFO Snapshotting: 0x0 to /var/lib/zookeeper/data/version-2/snapshot.0 (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2024-11-11 21:32:10,480] INFO Snapshot loaded in 5 ms, highest zxid is 0x0, digest is 1371985504 (org.apache.zookeeper.server.ZKDatabase)
[2024-11-11 21:32:10,480] INFO Snapshotting: 0x0 to /var/lib/zookeeper/data/version-2/snapshot.0 (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2024-11-11 21:32:10,481] INFO Snapshot taken in 0 ms (org.apache.zookeeper.server.ZooKeeperServer)
[2024-11-11 21:32:10,486] INFO PrepRequestProcessor (sid:0) started, reconfigEnabled=false (org.apache.zookeeper.server.PrepRequestProcessor)
[2024-11-11 21:32:10,486] INFO zookeeper.request_throttler.shutdownTimeout = 10000 ms (org.apache.zookeeper.server.RequestThrottler)
[2024-11-11 21:32:10,498] INFO Using checkIntervalMs=60000 maxPerMinute=10000 maxNeverUsedIntervalMs=0 (org.apache.zookeeper.server.ContainerManager)
[2024-11-11 21:32:10,499] INFO ZooKeeper audit is disabled. (org.apache.zookeeper.audit.ZKAuditProvider)
[2024-11-11 21:32:11,021] INFO Creating new log file: log.1 (org.apache.zookeeper.server.persistence.FileTxnLog)
```

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

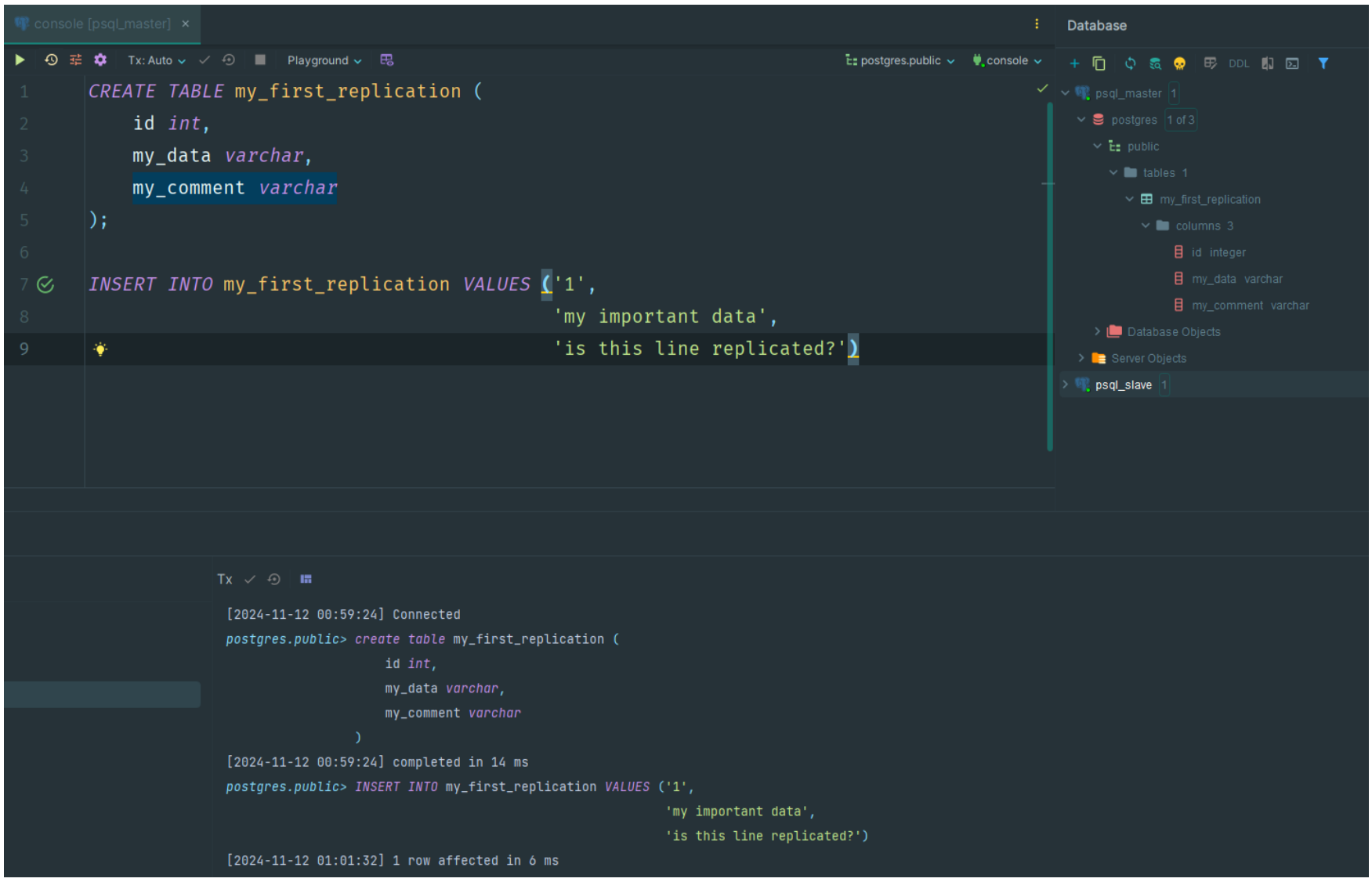
▼ Часть 2. Проверяем репликацию

1. Берем ЛЮБОЙ постгрес клиент (*голый psql, pgAdmin, DBeaver, ...*) и подключаемся к обеим нодам постгреса:
   - dbname/username/password = `postgres` (либо свой вариант, если меняли в конфигах/композе)
   - host/port = `pg-master:5433` и `pg-slave:5434`

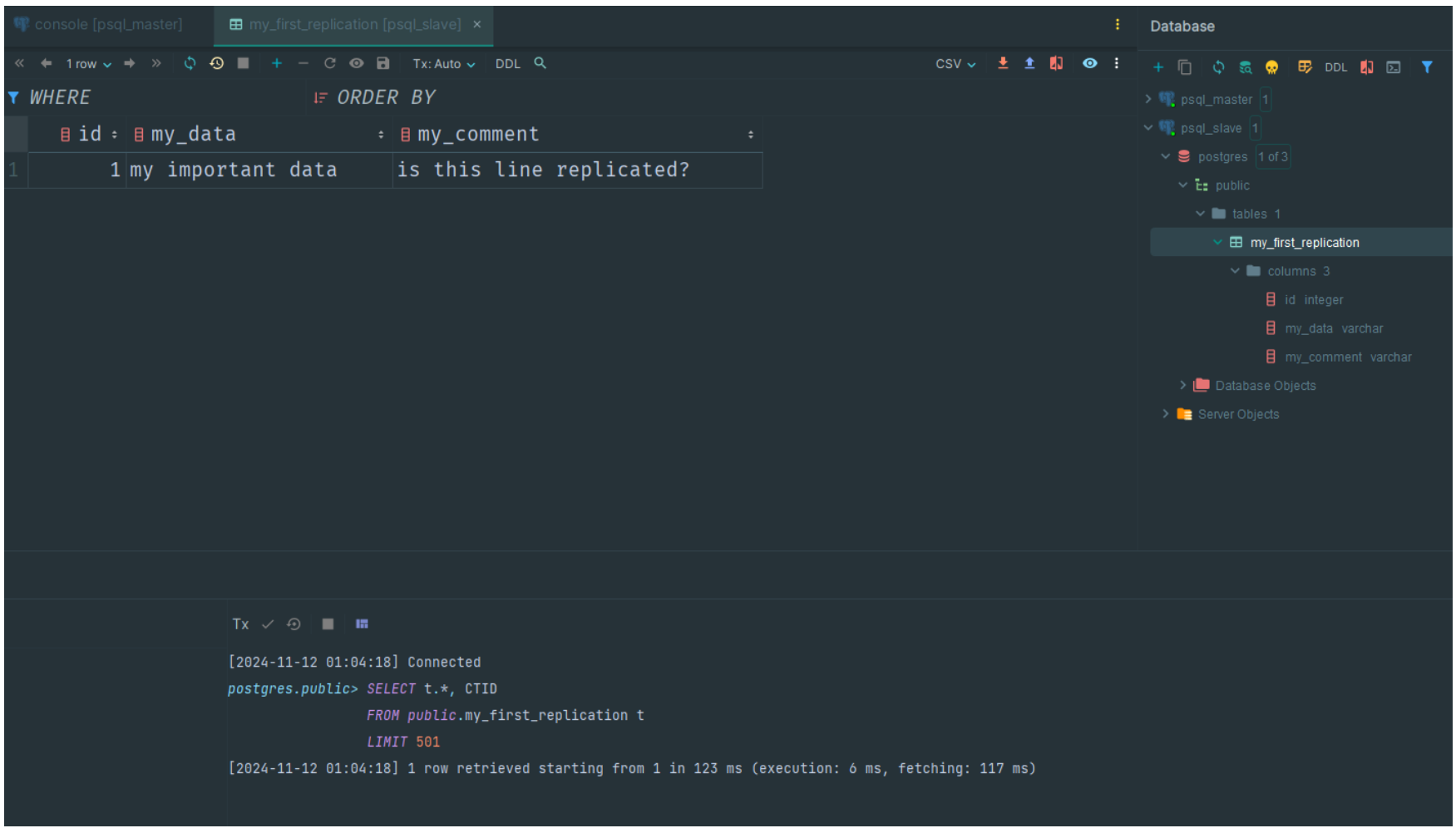
   ▼ *Пример*



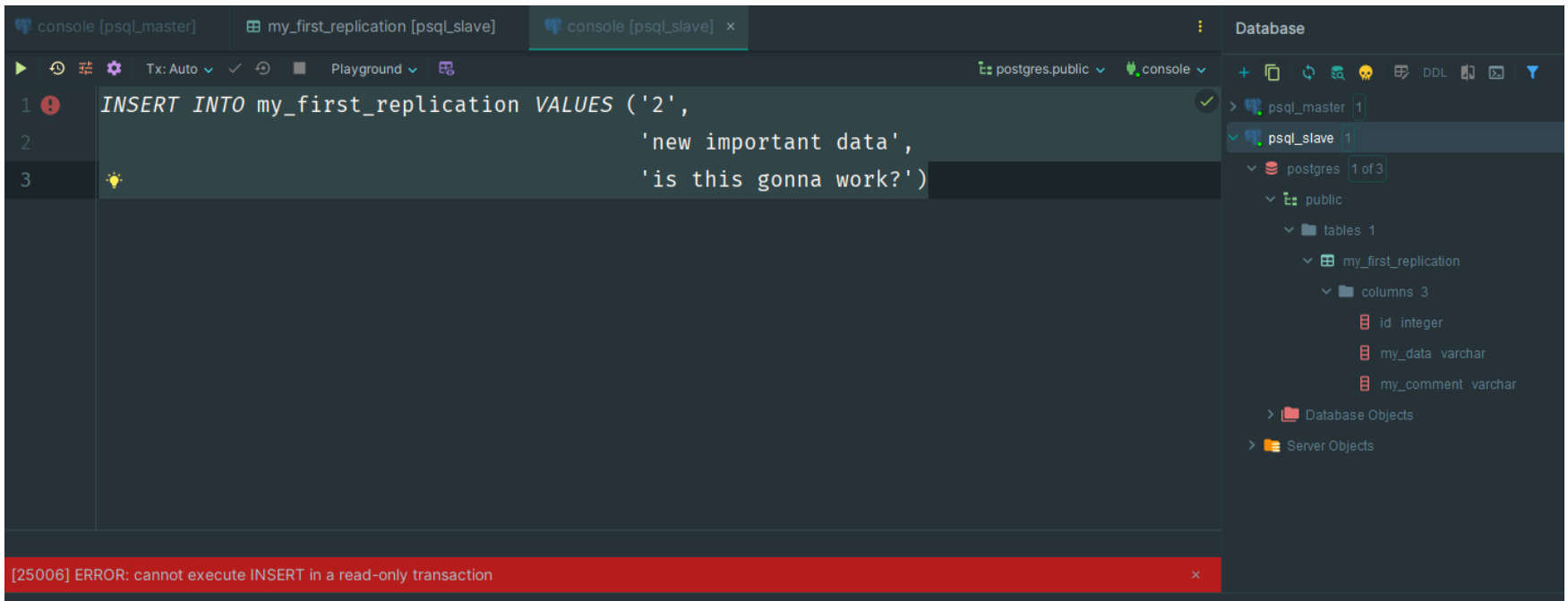
2. Из двух подключений выбираем **pg-master** (***только если, оно является Лидером этого кластера***). Создаем таблицу с ЛЮБОЙ структурой и записываем в нее ЛЮБЫЕ данные



3. Заходим в подключение **pg-slave** и наблюдаем магию: во второй базе данных автоматически создалась такая же таблица с такими же данными



4. В подключении **pg-slave** пробуем провести какую-нибудь операцию на редактирование. Например, попытаемся вставить новые данные в таблицу, или вовсе удалить ее. Получим отказ, т.к. эта нода работает в режиме *slave/readonly*



▼ Часть 3. Делаем ~~среднего роста~~ высокую доступность

1. Для балансировки трафика нам нужен специальное ПО, собственно балансировщик. Например, HAProxy — добавляем его в `docker-compose.yml`:

```
...
  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-master
      - pg-slave
      - zoo
    volumes:
      - ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg
...
```

2. Не забываем создать упомянутый выше `haproxy.cfg` со следующим содержимым:

```
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_master_5432 pg-master:5432 maxconn 100 check port 8008 # Адрес первой ноды постгреса
    server postgresql_pg_slave_5432 pg-slave:5432 maxconn 100 check port 8008 # Адрес второй ноды постгреса
```

3. Перезапускаем проект, проверяем что:
   - Обе ноды корректно поднялись и распределили между собой роли мастера и слейва (*аналогично Пункту 4 Части 1*)

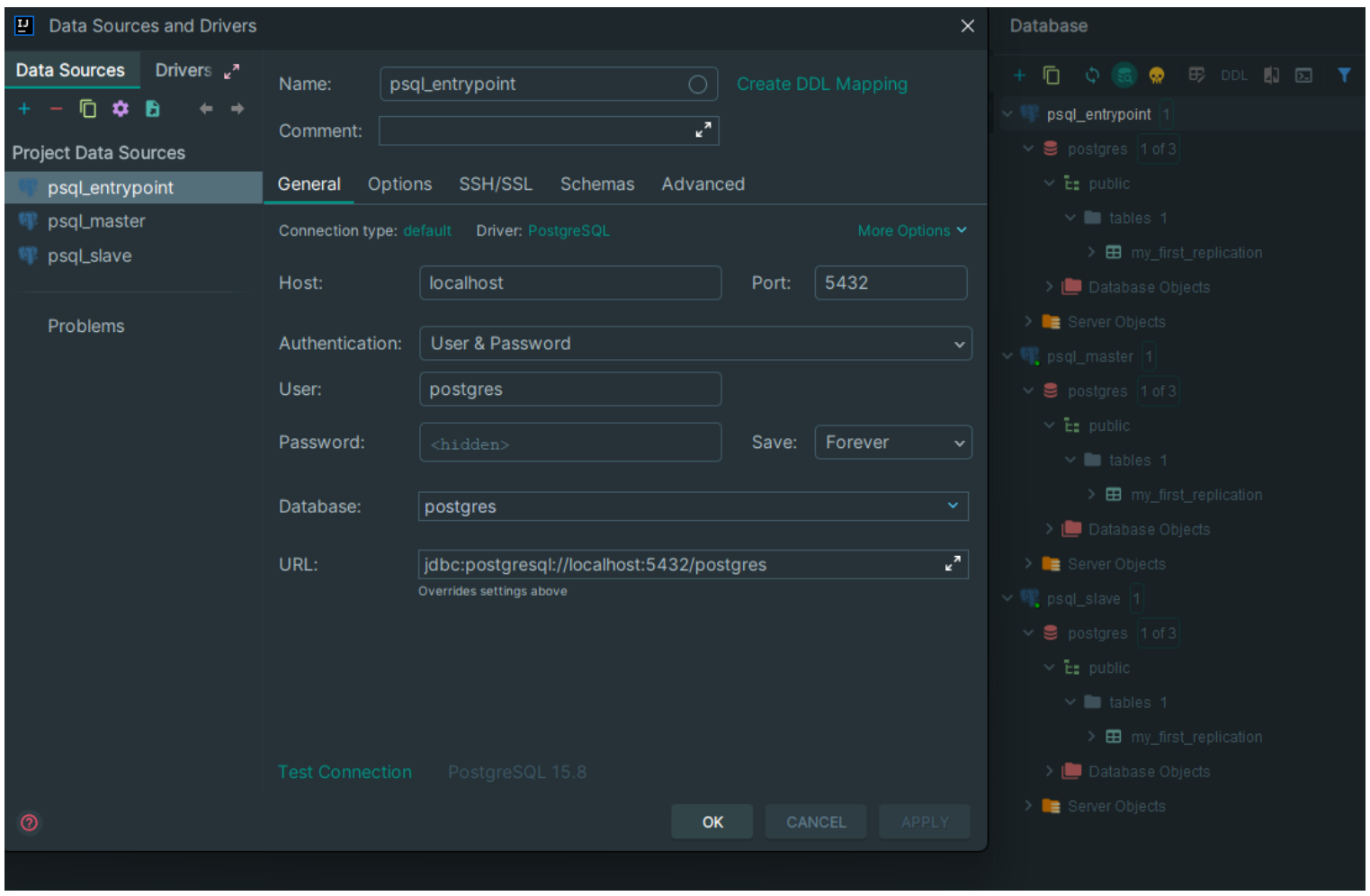
- Зукипер подцепился к кластеру без ошибок (*аналогично Пункту 4 Части 1*)
- Хапрокси поднялась без ошибок

▼ *Пример*

```
llidd@winlid MINGW64 ~/postgres_lab
$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED              STATUS              PORTS                                                NAMES
ed9d6e412993   haproxy:3.0                    "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:5432->5432/tcp, 0.0.0.0:7000->7000/tcp       postgres_entrypoint
37dee451ed4c   confluentinc/cp-zookeeper:7.7.1   "/etc/confluent/dock…"   About a minute ago   Up About a minute   2888/tcp, 0.0.0.0:2181->2181/tcp, 3888/tcp           zoo
e79a7f6c9b95   localhost/postres:patroni       "docker-entrypoint.s…"   About a minute ago   Up About a minute   8008/tcp, 0.0.0.0:5433->5432/tcp                     pg-master
84fa3a441f3c   localhost/postres:patroni       "docker-entrypoint.s…"   About a minute ago   Up About a minute   8080/tcp, 0.0.0.0:5434->5432/tcp                     pg-slave

llidd@winlid MINGW64 ~/postgres_lab
$ docker logs ed9d6e412993
[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: 1ms. 1 active and 0 backup servers online. 0 sessions requeued, 0 t
otal in queue.
```

4. Аналогично Пункту 1 Части 2, подключаемся к “новой” базе данных, в качестве адреса используя имя контейнера с HAProxy. Проверяем, что подключение успешное - по умолчанию оно всегда будет редиректить трафик на мастер-ноду



▼ **Задание**

> 💡 *За выполнение выделенных желтым цветом заданий, лаба засчитывается автоматом без ответов на доп.вопросы выше!*

- 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)

ЛИБО

- Улучшить лабораторную, настроив кластер так, чтобы после “возвращения” второй ноды в кластер она *автоматически* получала данные, которые были записаны в ее отсутствие

ЛИБО

- Сделать лабораторную на другом стеке (оставить patroni, но заменить zookeper и haproxy на что-то аналогичное)

▼ **Отчетность**

Отчет с ходом работы (аналогичный тексту текущей лабораторной) и скриншотами, а так же ответы на вопросы. Можно в документе (Word, PDF, etc) или в git репозитории