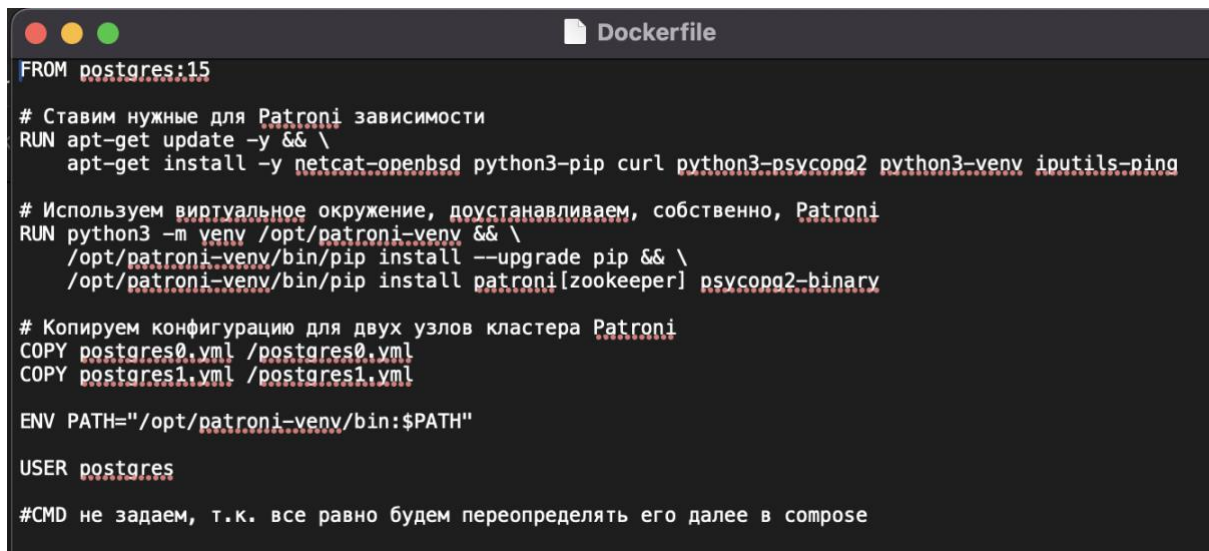


# Лабораторная работа №3

Работу выполнили: Скирляк Ярослав (К3339), Григорьев Алексей (К3340),  
Гафаров Данил (К3343)

## 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



```
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, устанавливаем, 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 [Zookeeper](#), который нужен для непосредственного управления репликацией и определением “лидера” кластера

```
docker-compose.yml

1  services:
2    pg-master:
3      build: .
4      image: localhost/postres:patroni # имя для кастомного образа из Dockerfile, можно задать любое
5      container_name: pg-master # Будущий адрес первой ноды
6      restart: always
7      hostname: pg-master
8      environment:
9        POSTGRES_USER: postgres
10       POSTGRES_PASSWORD: postgres
11       PGDATA: '/var/lib/postgresql/data/pgdata'
12     expose:
13       - 8008
14     ports:
15       - 5433:5432
16     volumes:
17       - pg-master:/var/lib/postgresql/data
18     command: patroni /postgres0.yml
19
20    pg-slave:
21      build: .
22      image: localhost/postres:patroni # имя для кастомного образа из Dockerfile, можно задать любое
23      container_name: pg-slave # Будущий адрес второй ноды
24      restart: always
25      hostname: pg-slave
26      expose:
27       - 8008
28     ports:
29       - 5434:5432
30     volumes:
31       - pg-slave:/var/lib/postgresql/data
32     environment:
33       POSTGRES_USER: postgres
34       POSTGRES_PASSWORD: postgres
35       PGDATA: '/var/lib/postgresql/data/pgdata'
36     command: patroni /postgres1.yml
37
```

```
37
38     zoo:
39       image: confluentinc/cp-zookeeper:7.7.1
40       container_name: zoo # Будущий адрес зукипера
41       restart: always
42       hostname: zoo
43       ports:
44         - 2181:2181
45       environment:
46         ZOOKEEPER_CLIENT_PORT: 2181
47         ZOOKEEPER_TICK_TIME: 2000
48
49     volumes:
50       pg-master:
51       pg-slave:
```

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)



```
1 scope: my_cluster # Имя нашего кластера
2 name: postgresql0 # Имя первой ноды
3
4 restapi: # Адреса первой ноды
5   listen: pg-master:8008
6   connect_address: pg-master:8008
7
8 zookeeper:
9   hosts:
10     - zoo:2181 # Адрес Zookeeper
11
12 bootstrap:
13   dcs:
14     ttl: 30
15     loop_wait: 10
16     retry_timeout: 10
17     maximum_lag_on_failover: 10485760
18     master_start_timeout: 300
19     synchronous_mode: true
20   postgresql:
21     use_pg_rewind: true
22     use_slots: true
23     parameters:
24       wal_level: replica
25       hot_standby: "on"
26       wal_keep_segments: 8
27       max_wal_senders: 10
28       max_replication_slots: 10
29       wal_log_hints: "on"
30       archive_mode: "always"
31       archive_timeout: 1800s
32       archive_command: mkdir -p /tmp/wal_archive && test ! -f /tmp/wal_archive/%f && cp %p /tmp/wal_archive/%f
33
34 pg_hba:
35   - host replication replicator 0.0.0.0/0 md5
36   - host all all 0.0.0.0/0 md5
37
38 postgresql:
39   listen: 0.0.0.0:5432
40   connect_address: pg-master:5432 # Адрес первой ноды
41   data_dir: /var/lib/postgresql/data/postgresql0 # Место хранения данных первой ноды
42   bin_dir: /usr/lib/postgresql/15/bin
43   pgpass: /tmp/pgpass0
44   authentication:
45     replication: # логонасс для репликации, при желании можно поменять
46       username: replicator
47       password: rep-pass
48     superuser: # админский логонасс, при желании можно поменять (в том числе в файле compose)
49       username: postgres
50       password: postgres
51   parameters:
52     unix_socket_directories: '.'
53
54 watchdog:
55   mode: off
56
57 tags:
58   nofailover: false
59   noloadbalance: false
60   clonefrom: false
61   nosync: false
```

```
postgres1.yml
postgres0.yml

1 scope: my_cluster # Имя кластера
2 name: postgresql1 # Имя второй ноды
3
4 restapi: # Адреса второй ноды
5   listen: pg-slave:8008
6   connect_address: pg-slave:8008
7
8 zookeeper:
9   hosts:
10    - zoo:2181 # Адрес Zookeeper
11
12 bootstrap:
13   dcs:
14     ttl: 30
15     loop_wait: 10
16     retry_timeout: 10
17     maximum_lag_on_failover: 10485760
18     master_start_timeout: 300
19     synchronous_mode: true
20   postgresql:
21     use_pg_rewind: true
22     use_slots: true
23     parameters:
24       wal_level: replica
25       hot_standby: "on"
26       wal_keep_segments: 8
27       max_wal_senders: 10
28       max_replication_slots: 10
29       wal_log_hints: "on"
30       archive_mode: "always"
31       archive_timeout: 1800s
32       archive_command: mkdir -p /tmp/wal_archive && test ! -f /tmp/wal_archive/%f && cp %p /tmp/wal_archive/%f
33
34 pg_hba:
35   - host replication replicator 0.0.0.0/0 md5
36   - host all all 0.0.0.0/0 md5
37
38 postgresql:
39   listen: 0.0.0.0:5432
40   connect_address: pg-slave:5432 # Адрес второй ноды
41   data_dir: /var/lib/postgresql/data/postgresql1 # Место хранения данных второй ноды
42   bin_dir: /usr/lib/postgresql/15/bin
43   pgpass: /tmp/pgpass1
44   authentication:
45     replication: # Логин и пароль для репликации
46       username: replicator
47       password: rep-pass
48     superuser: # Логин и пароль администратора
49       username: postgres
50       password: postgres
51   parameters:
52     unix_socket_directories: '.,'
53
54 watchdog:
55   mode: off
56
57 tags:
58   nofailover: false
59   noloadbalance: false
60   clonefrom: false
61   nosync: false
62
```

4. Деплоим. Проверяем в логах, что зукпер запустился, и что одна нода постгреса из двух стала лидером/овнером/мастером

<input type="checkbox"/>	lab-3	Running (3/3)	0.47%
<input type="checkbox"/>	<b>pg-master</b> 6b2e58387f91	localhost/postres:patroni	Running 0.18% <a href="#">5433:5432</a>
<input type="checkbox"/>	<b>pg-slave</b> dd434b7b3951	localhost/postres:patroni	Running 0.19% <a href="#">5434:5432</a>
<input type="checkbox"/>	<b>zoo</b> 7cdf8d810149	confluentinc/cp-zookeeper:7.7.1	Running 0.1% <a href="#">2181:2181</a>

<b>pg-slave</b> localhost/postres:patroni dd434b7b3951 <a href="#">5434:5432</a>	
Logs	Inspect Bind mounts Exec Files Stats
<pre>2024-12-10 22:21:41 2024-12-10 19:21:41,980 INFO: no action. I am (postgresql), the leader with the lock 2024-12-10 22:21:51 2024-12-10 19:21:51,980 INFO: no action. I am (postgresql), the leader with the lock 2024-12-10 22:22:01 2024-12-10 19:22:01,982 INFO: no action. I am (postgresql), the leader with the lock 2024-12-10 22:22:11 2024-12-10 19:22:11,980 INFO: no action. I am (postgresql), the leader with the lock 2024-12-10 22:22:21 2024-12-10 19:22:21,978 INFO: no action. I am (postgresql), the leader with the lock 2024-12-10 22:22:31 2024-12-10 19:22:31,977 INFO: no action. I am (postgresql), the leader with the lock</pre>	

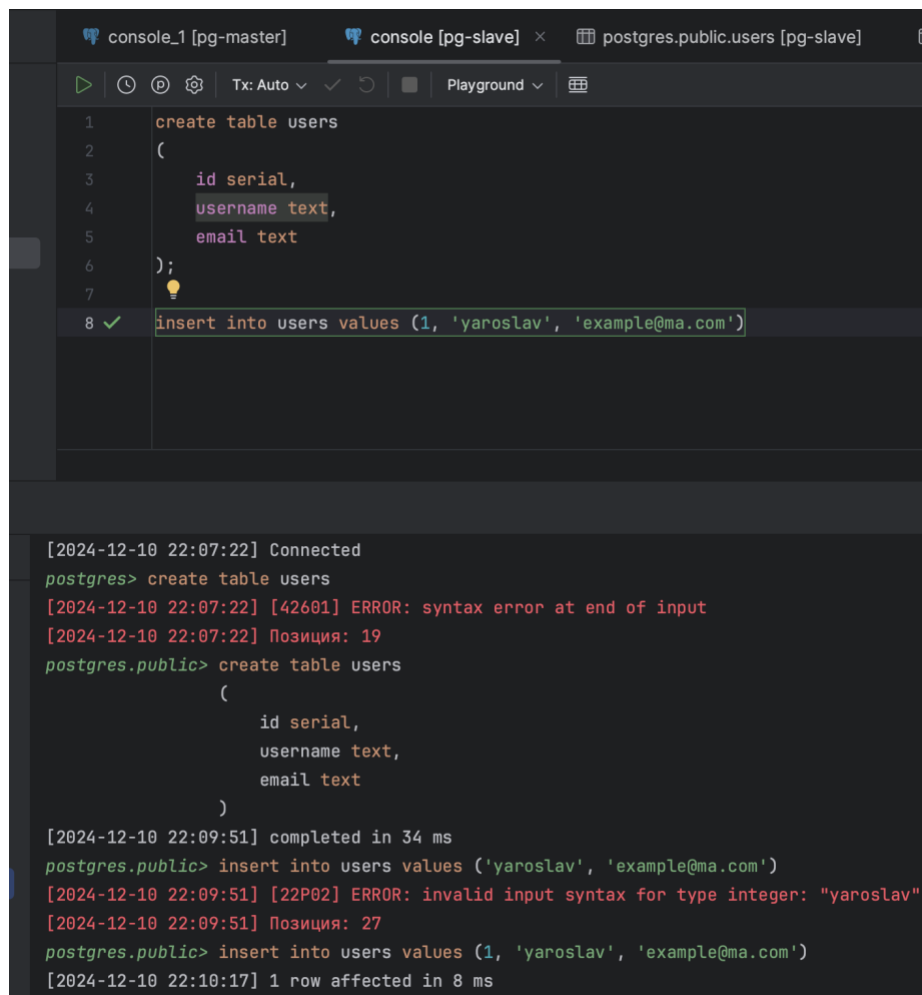
<b>pg-master</b> localhost/postres:patroni 6b2e58387f91 <a href="#">5433:5432</a>		<b>STATUS</b> Running (23 minutes)
Logs	Inspect Bind mounts Exec Files Stats	
<pre>2024-12-10 22:21:31 2024-12-10 19:21:31,639 INFO: no action. I am (postgresl0), a secondary, and following a leader (postgresql) 2024-12-10 22:21:41 2024-12-10 19:21:41,641 INFO: no action. I am (postgresl0), a secondary, and following a leader (postgresql) 2024-12-10 22:21:51 2024-12-10 19:21:51,641 INFO: no action. I am (postgresl0), a secondary, and following a leader (postgresql) 2024-12-10 22:22:01 2024-12-10 19:22:01,639 INFO: no action. I am (postgresl0), a secondary, and following a leader (postgresql) 2024-12-10 22:22:11 2024-12-10 19:22:11,649 INFO: no action. I am (postgresl0), a secondary, and following a leader (postgresql) 2024-12-10 22:22:21 2024-12-10 19:22:21,646 INFO: no action. I am (postgresl0), a secondary, and following a leader (postgresql)</pre>		

## 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

Ну надеюсь не снимут баллы за то что без скрина подключение

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

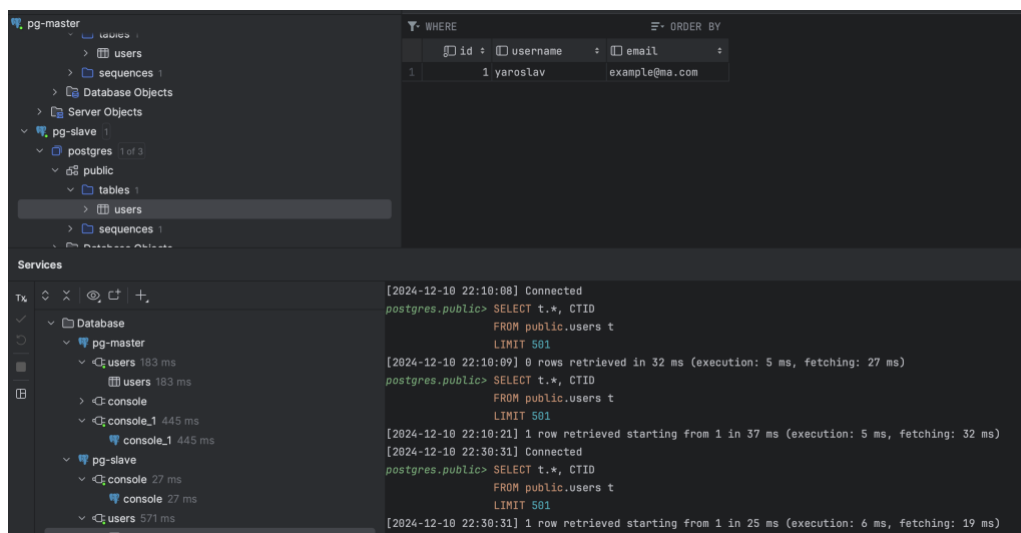


The screenshot shows a PostgreSQL playground interface with two tabs: 'console\_1 [pg-master]' and 'console [pg-slave]'. The 'pg-master' tab is active, showing a SQL script with the following commands:

```
1 create table users
2 (
3     id serial,
4     username text,
5     email text
6 );
7
8 insert into users values (1, 'yaroslav', 'example@ma.com')
```

The execution results for the 'pg-master' tab are as follows:

```
[2024-12-10 22:07:22] Connected
postgres> create table users
[2024-12-10 22:07:22] [42601] ERROR: syntax error at end of input
[2024-12-10 22:07:22] Позиция: 19
postgres.public> create table users
(
    id serial,
    username text,
    email text
)
[2024-12-10 22:09:51] completed in 34 ms
postgres.public> insert into users values ('yaroslav', 'example@ma.com')
[2024-12-10 22:09:51] [22P02] ERROR: invalid input syntax for type integer: "yaroslav"
[2024-12-10 22:09:51] Позиция: 27
postgres.public> insert into users values (1, 'yaroslav', 'example@ma.com')
[2024-12-10 22:10:17] 1 row affected in 8 ms
```



The screenshot shows a PostgreSQL database management tool interface. The left sidebar displays the database structure, including the 'users' table. The main area shows the table structure and query results.

**Table Structure:**

id	username	email
1	yaroslav	example@ma.com

**Services:**

- pg-master
  - users 183 ms
  - console 445 ms
  - console\_1 445 ms
- pg-slave
  - console 27 ms
  - console 27 ms
  - users 571 ms

**Query Results:**

```
[2024-12-10 22:10:08] Connected
postgres.public> SELECT t.*, CTID
FROM public.users t
LIMIT 501
[2024-12-10 22:10:09] 0 rows retrieved in 32 ms (execution: 5 ms, fetching: 27 ms)
postgres.public> SELECT t.*, CTID
FROM public.users t
LIMIT 501
[2024-12-10 22:10:21] 1 row retrieved starting from 1 in 37 ms (execution: 5 ms, fetching: 32 ms)
[2024-12-10 22:30:31] Connected
postgres.public> SELECT t.*, CTID
FROM public.users t
LIMIT 501
[2024-12-10 22:30:31] 1 row retrieved starting from 1 in 25 ms (execution: 6 ms, fetching: 19 ms)
```

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

The screenshot displays a database management interface. On the left, a tree view shows the database structure for 'pg-master' and 'pg-slave'. The 'pg-master' connection is selected, showing a 'public' schema with a 'users' table. The main pane shows the 'users' table with columns 'id', 'username', and 'email', containing one row: '1 yaroslav example@ma.com'. The bottom pane shows the 'Services' section with a list of connections: 'pg-master', 'pg-slave', and their respective console and users tables. The console output for 'pg-master' shows a successful query execution: 'SELECT t.\*, CTID FROM public.users t LIMIT 501'.

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*

The screenshot displays a database management interface for the 'pg-slave' connection. The top bar shows the connection name 'pg-slave' and the table 'postgres.public.users'. The main pane shows a console window with a red error message: '[25006] ERROR: cannot execute INSERT in a read-only transaction'. The console output shows a failed query execution: 'insert into users values (2, 'alex', 'example@ma.com')'.

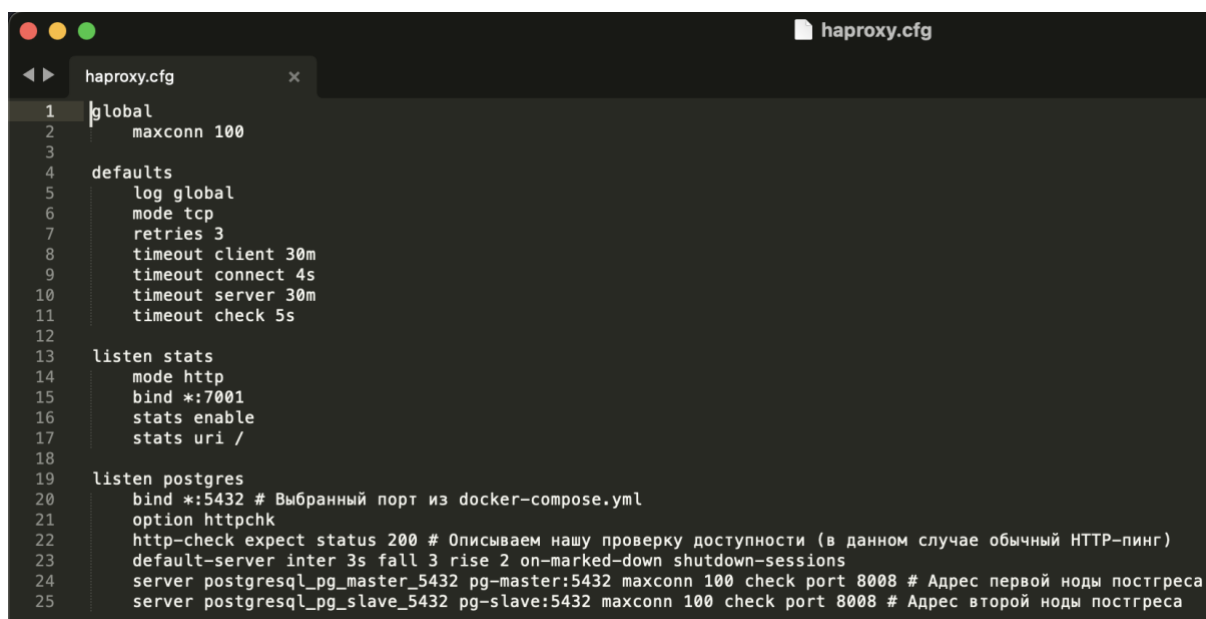


### Часть 3. Делаем среднего роста высокую доступность

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_entrpoint # Это будет адрес подключения к БД, можно выбрать любой
  ports:
    - 5432:5432 # Это будет порт подключения к БД, можно выбрать любой
    - 7001:7000
  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 содержанием:



```
1 global
2     maxconn 100
3
4 defaults
5     log global
6     mode tcp
7     retries 3
8     timeout client 30m
9     timeout connect 4s
10    timeout server 30m
11    timeout check 5s
12
13 listen stats
14     mode http
15     bind *:7001
16     stats enable
17     stats uri /
18
19 listen postgres
20     bind *:5432 # Выбранный порт из docker-compose.yml
21     option httpchk
22     http-check expect status 200 # Описываем нашу проверку доступности (в данном случае обычный HTTP-пинг)
23     default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
24     server postgresql_pg_master_5432 pg-master:5432 maxconn 100 check port 8008 # Адрес первой ноды постгреса
25     server postgresql_pg_slave_5432 pg-slave:5432 maxconn 100 check port 8008 # Адрес второй ноды постгреса
```

Я поменял порт с 7000 на 7001, потому что у меня 7000 был занят и мне было лень смотреть чем :) К счастью это ни на что не влияет и можно выдохнуть.

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



2. Зукипер подцепился к кластеру без ошибок (*аналогично*

*Пункту 4 Части 1)*

3. Хапрокси поднялась без ошибок

```
[+] Running 4/4
✓ Container pg-slave      Running      0.0s
✓ Container zoo           Running      0.0s
✓ Container pg-master     Running      0.0s
✓ Container postgres_entrypoint Recreated    0.1s
Attaching to pg-master, pg-slave, postgres_entrypoint, zoo
postgres_entrypoint | [NOTICE] (1) : haproxy version is 3.0.6-c2c0090
postgres_entrypoint | [NOTICE] (1) : path to executable is /usr/local/sbin/haproxy
postgres_entrypoint | [ALERT] (1) : config : parsing [/usr/local/etc/haproxy/haproxy.cfg:25]: Missing LF on last line, file
might have been truncated at position 132.
postgres_entrypoint | [ALERT] (1) : config : Error(s) found in configuration file : /usr/local/etc/haproxy/haproxy.cfg
postgres_entrypoint | [ALERT] (1) : config : Fatal errors found in configuration.
postgres_entrypoint exited with code 1
pg-master           | 2024-12-10 19:40:41,641 INFO: no action. I am (postgresql0), a secondary, and following a leader (postgre
sql1)
pg-slave            | 2024-12-10 19:40:43,026 INFO: no action. I am (postgresql1), the leader with the lock
```

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

### Data Sources and Drivers

▼

→

Name: entry-pointer

Create DDL Mapping

Comment:

General

Options

SSH/SSL

Schemas

Advanced

Connection type: default

Driver: PostgreSQL

More Options

Host: localhost

Port: 5432

Authentication: User & Password

User: postgres

Password: <hidden>

Save: Forever

Database: postgres

URL: jdbc:postgresql://localhost:5432/postgres

Succeeded

Copy

DBMS: PostgreSQL (ver. 15.8 (Homebrew))

Case sensitivity: plain=lower, delimited=exact

Driver: PostgreSQL JDBC Driver (ver. 42.7.3, JDBC4.2)

Ping: 8 ms

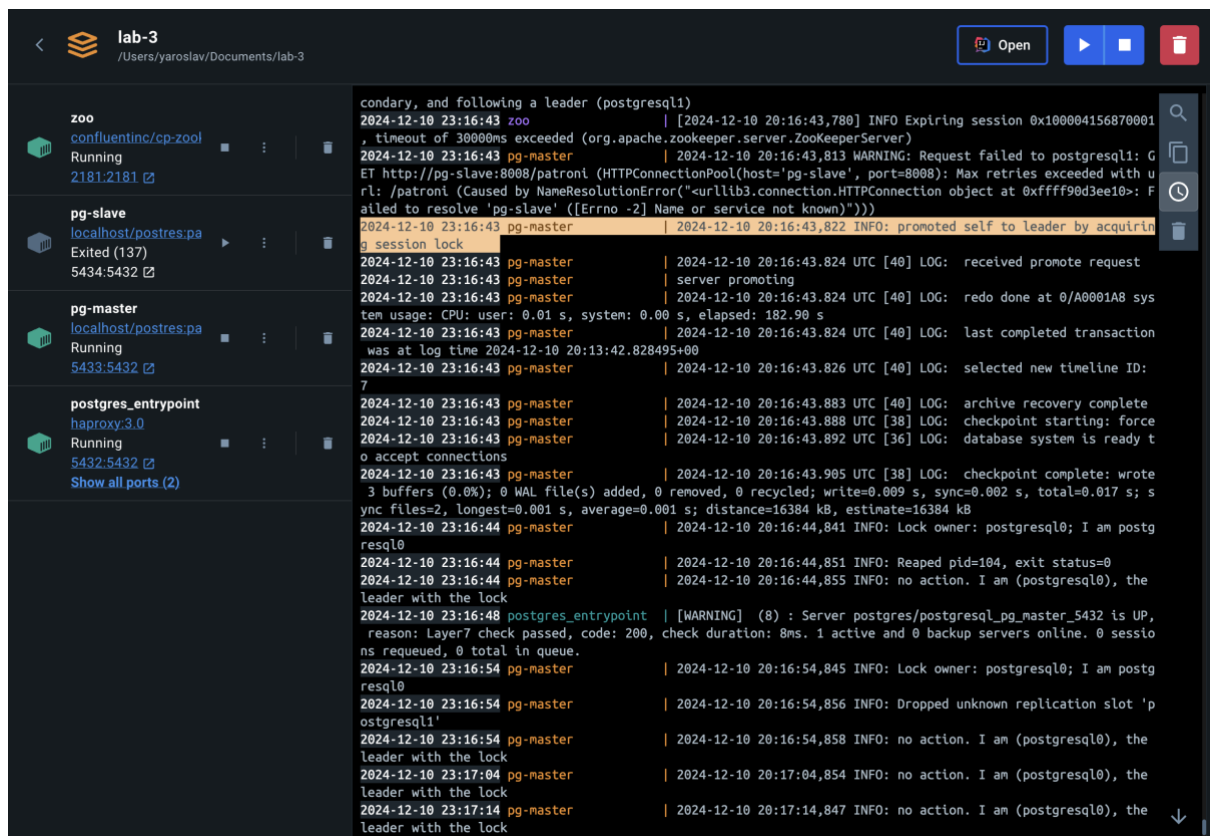
SSL: no

Test Connection

✓ PostgreSQL 15.8

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

Остановил одну из нод после чего буду вносить в базу данных новые данные. На скриншоте видно, что нода перехватила управление на себя.



В приведённой настройке Patroni уже предусмотрены параметры для хранения WAL:

- `wal_keep_segments` задаёт количество сегментов для восстановления реплики после простоя.
- `archive_mode` и `archive_command` включают архивирование WAL.

```
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
```

Replication slots обеспечивают сохранение лидером WAL-сегментов для отстающих реплик

```
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"
```

Поскольку у нас все настроено правильно, patroni теперь автоматически синхронизирует данные при возвращении узла в кластер.

Теперь проверяем:

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
ebd98cb65af7	haproxy:3.0		"docker-entrypoint.s..."	9 minutes ago	Up 9 minutes	0.0.0.0:5432->5432/tcp, :::5432->5432/tcp, 0.0.0.0:7000->7000/tcp, :::7000->7000/tcp
3eee8deed898	confluentinc/cp-zookeeper:7.7.1	postgres_entrypoint	"/etc/confluent/dock..."	23 minutes ago	Up 23 minutes	2888/tcp, 0.0.0.0:2181->2181/tcp, :::2181->2181/tcp, 3888/tcp
c82e27ef567c	localhost/postres:patroni	zoo	"docker-entrypoint.s..."	23 minutes ago	Up 23 minutes	8008/tcp, 0.0.0.0:5434->5432/tcp, [::]:5434->5432/tcp
		pg-slave				

При работе на другом компьютере у меня возникли трудности с лабораторной работой и мне пришлось создавать другие базы данных

Прямо сейчас мы создаем новую запись в нашей таблице:

The screenshot shows a database console interface with three tabs: 'console [pg-master]', 'console [pg-entrypoint]' (selected), and 'console [pg-slave]'. The 'pg-entrypoint' tab contains a single SQL query: `select * from test;`. Below the query, the 'Output' panel displays the results of the query as a table with 2 rows. The table has columns 'id', 'key', and 'value'. The first row contains '1', 'test', and 'test'. The second row contains '2', 'test2', and 'test2'.

id	key	value
1	test	test
2	test2	test2

Далее восстанавливаем реплику и проверяем селекты на ней

This screenshot is identical to the one above, showing the same database console interface with the 'pg-entrypoint' tab selected. The SQL query `select * from test;` is entered, and the 'Output' panel shows the same two rows of data: (1, test, test) and (2, test2, test2).

id	key	value
1	test	test
2	test2	test2

И нам осталось проверить репликацию на мастере:

The screenshot shows a PostgreSQL console interface with three tabs: 'console [pg-master]', 'console [pg-entrypoint]', and 'console [pg-slave]'. The 'pg-master' tab is active, displaying the command `SELECT * FROM pg_stat_replication;` and its output. The output is a single row representing a streaming replication connection. The 'Database' sidebar on the right shows the 'postgres' database selected. The 'Output' panel at the bottom shows the query results in a table format.

pid	usesysid	username	application_name	client_addr	client_hostname	client_port	backend_start	backend_xmin	state	sent_lsn	write_lsn
332	16384	replicator	postgresqls	172.18.0.4	null		47338 2024-12-07 13:26:58.289774 +09:00 null		streaming	0/9223F68	0/9223F68