Университет ИТМО
Факультет программной инженерии и компьютерной техники

Учебно-исследовательская работа №4 (УИР 4)

По дисциплине «Распределённые системы хранения данных»

Выполнили:
Нодири Хисравхон
Бахилин Никита Витальевич
Вариант: 3023

Преподаватель:
Максимов Андрей Николаевич

г. Санкт-Петербург
2025г.

Содержание

1 Условие

1.1 Цель работы

Цель работы. Ознакомление с методами и средствами построения отказоустойчивых решений на базе СУБД PostgreSQL; получение практических навыков восстановления работы системы после отказа. Работа рассчитана на двух человек и включает три этапа: настройка, симуляция и обработка сбоя, восстановление.

1.2 Требования к выполнению работы

- В качестве хостов использовать одинаковые виртуальные машины.
- В первую очередь необходимо обеспечить сетевую связность между ВМ.
- Для подключения к СУБД (например, через `psql`), использовать отдельную виртуальную или физическую машину.
- Демонстрировать наполнение базы и доступ на запись на примере `не менее`, чем двух таблиц, столбцов, строк, транзакций и клиентских сессий.

1.3 Этап 1. Конфигурация

Развернуть postgres на двух узлах в режиме балансировки нагрузки. Использовать pgbouncer. Продемонстрировать обработку транзакций обоими серверами.

1.4 Этап 2. Симуляция и обработка сбоя

`2.1 Подготовка:`

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

`2.2 Сбой:`

- Симулировать ошибку диска на основном узле — удалить директорию `PGDATA` со всем содержимым.

`2.3 Обработка:`

- Найти и продемонстрировать в логах релевантные сообщения об ошибках.
- Выполнить переключение (failover) на резервный сервер.
- Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

1.5 Восстановление

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

2 Этап 1 — Конфигурация кластера и балансировщика

Инфраструктура

- **pg1** — `147.45.40.78`
- **pg2** — `87.120.36.97`
- **client** — Отдельная машина/ПК с `psql`

2.1 /etc/hosts для упрощения адресации

```
echo "147.45.40.78 pg1"  | sudo tee -a /etc/hosts
echo "87.120.36.97 pg2"  | sudo tee -a /etc/hosts
```

2.2 PostgreSQL 16: настройка master (pg1)

```
# /etc/postgresql/16/main/postgresql.conf (fragment)
listen_addresses = '*'
wal_level        = replica
max_wal_senders  = 10
wal_keep_size    = 256MB
archive_mode     = on
archive_command  = 'test ! -f /var/lib/postgresql/archive/%f && cp %p /var/lib/
    postgresql/archive/%f'
hot_standby      = on
```

Создаём архив каталог и перезапускаем:

```
sudo mkdir -p /var/lib/postgresql/archive
sudo chown postgres:postgres /var/lib/postgresql/archive
sudo systemctl restart postgresql
```

Создаём роли и БД:

```
CREATE ROLE replicator WITH REPLICATION LOGIN PASSWORD 'replpass';
CREATE ROLE labuser    WITH LOGIN PASSWORD 'labpass';
CREATE DATABASE labdb OWNER labuser;

GRANT ALL ON TABLE streams TO labuser;
GRANT ALL ON TABLE idols TO labuser;
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO labuser;
```

Правила доступа (`pg_hba.conf`):

```
# Clients
host labdb      labuser     0.0.0.0/0      md5
# Replication from pg2
host replication replicator 87.120.36.97/32 md5
```

2.3 Поднимаем standby (pg2)

```
# Off old cluster
sudo systemctl stop postgresql
sudo -u postgres rm -rf /var/lib/postgresql/16/main/*

# Base backup + slot
sudo -u postgres pg_basebackup \
  -h pg1 -D /var/lib/postgresql/16/main \
  -U replicator -Fp -Xs -P -R -C -S pg2_slot

sudo systemctl start postgresql
```

pg2 pg_hba.conf:

```
host    replication    replicator    147.45.40.78/32    md5
host    all    all    0.0.0.0/0    md5
```

Файл `postgresql.auto.conf` на pg2 содержит:

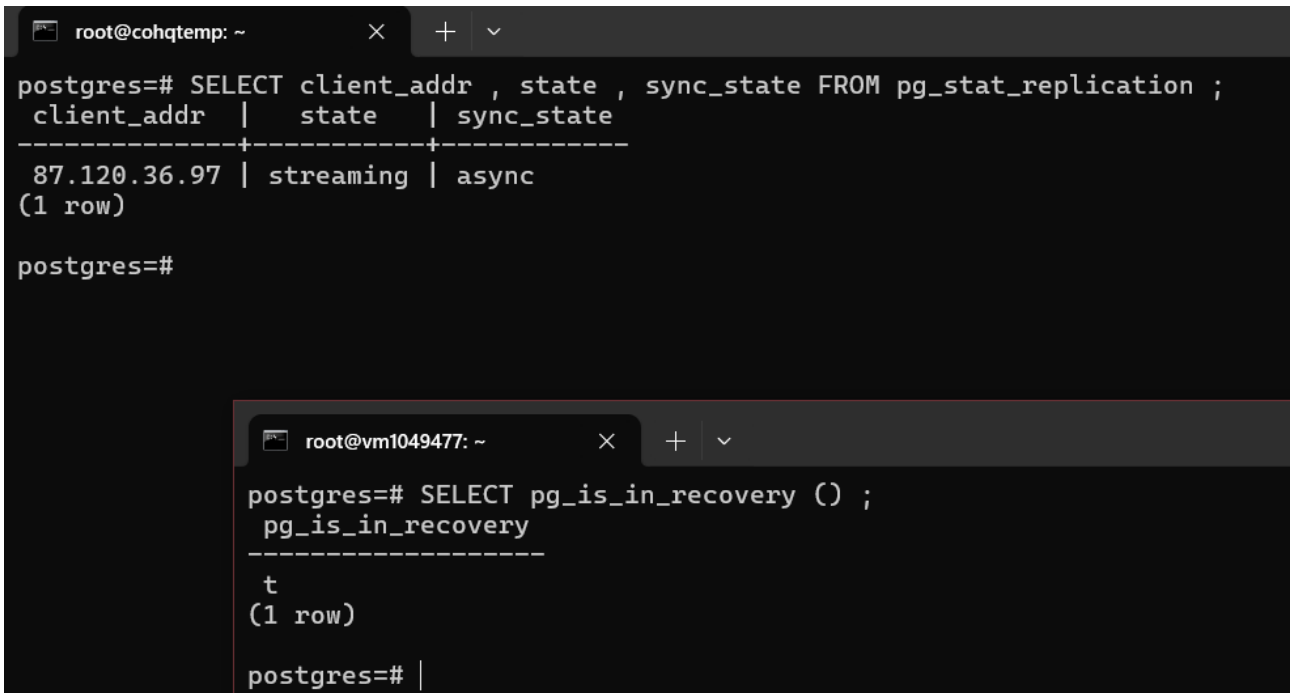
```
primary_conninfo = 'user=replicator password=replpass host=pg1 port=5432'
primary_slot_name = 'pg2_slot'
```

Проверяем:

```
-- pg1
SELECT client_addr,state,sync_state FROM pg_stat_replication;
-- pg2
SELECT pg_is_in_recovery();
```

2.4 Заполняем тестовыми данными

```
CREATE TABLE IF NOT EXISTS idols (
    id          serial PRIMARY KEY,
    name_en     text   NOT NULL,
    name_jp     text   NOT NULL,
    gen         text   NOT NULL,
    debut_date  date
);

CREATE TABLE IF NOT EXISTS streams (
    id          serial PRIMARY KEY,
    talent_id   int    REFERENCES idols(id),
    title       text,
    viewers_k   int,
    stream_date date
);

BEGIN;

INSERT INTO idols (name_en, name_jp, gen, debut_date) VALUES
  ('Tokino Sora',      '               ',           '0  gen  ',      '2017-09-07'),
  ('Shirakami Fubuki', '               ',           '1  gen  ',      '2018-06-01'),
  ('Minato Aqua',      '            ',            'GAMERS',     '2018-08-08'),
  ('Usada Pekora',     '               ',           '3  gen  ',      '2019-07-17'),
  ('Houshou Marine',   '               ',           '3  gen  ',      '2019-07-17'),
  ('Gawr Gura',        '                  ',        ' E N Myth ',    '2020-09-13'),
  ('Mori Calliope',    '               ',           ' E N Myth ',    '2020-09-12');

WITH ids AS (
  SELECT id FROM idols WHERE name_en IN ('Usada Pekora','Gawr Gura')
)
INSERT INTO streams (talent_id, title, viewers_k, stream_date) VALUES
  ((SELECT id FROM ids OFFSET 0 LIMIT 1),
   'Pekora peko-peko', 180, '2025-05-17'),
  ((SELECT id FROM ids OFFSET 1 LIMIT 1),
   'Guraaaaa', 194, '2025-07-1');

COMMIT;
```



2.5 Установка и настройка PgBouncer (pg1)

```
sudo apt-get install pgbouncer -y
```

/etc/pgbouncer/pgbouncer.ini

```
[databases]
labdb = host=147.45.40.78,87.120.36.97 port=5432 dbname=labdb

[pgbouncer]
listen_addr        = 0.0.0.0
listen_port        = 6432
auth_type          = md5
auth_file          = /etc/pgbouncer/userlist.txt
pool_mode          = transaction
server_round_robin = 1
```

/etc/pgbouncer/userlist.txt

```
"labuser" "labpass"
```

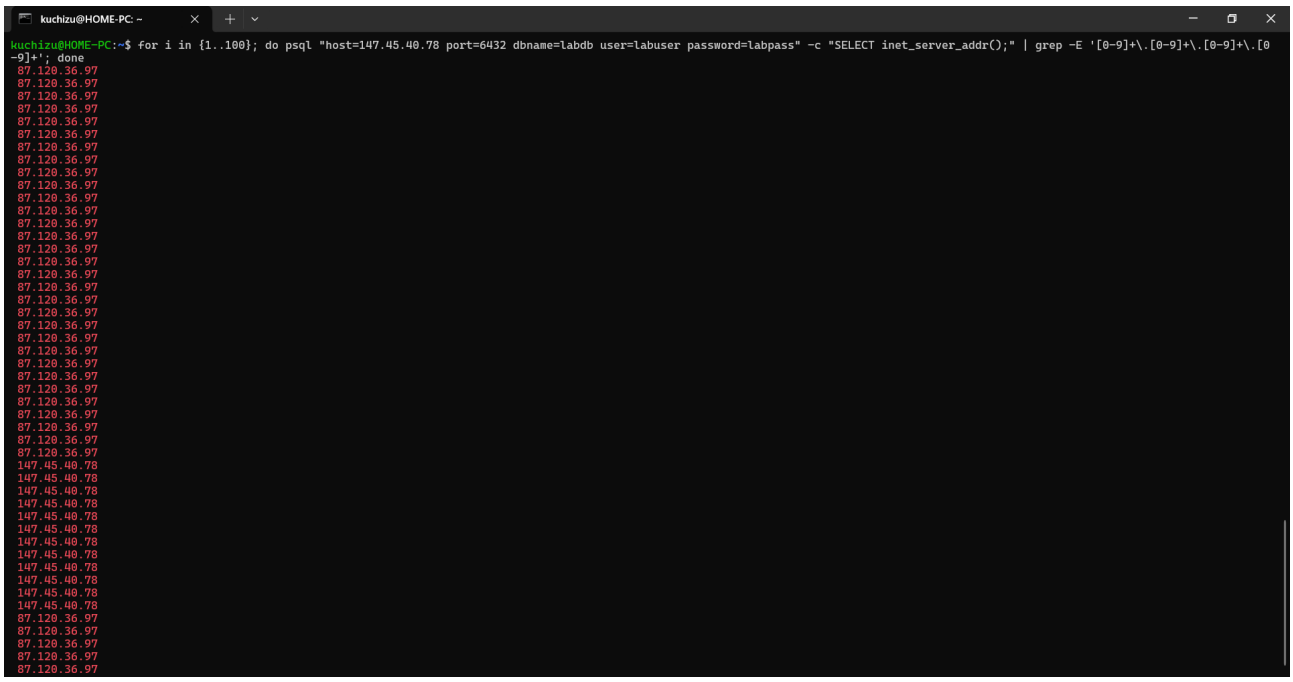
Запуск и автозапуск:

```
sudo systemctl enable --now pgbouncer
```

2.6 Проверка балансировки

Client:

```
for i in {1..100}; do
  psql "host=147.45.40.78 port=6432 dbname=labdb user=labuser password=labpass" \
       -c "SELECT inet_server_addr();" |
       grep -E '[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+'
done
```



3 Этап 2. Симуляция и обработка сбоя

3.1 Client

```
#!/usr/bin/env bash
# set -euo pipefail

PG1_IP=147.45.40.78
PG2_IP=87.120.36.97
PGHOST=$PG1_IP
PGPORT=6432
DB=labdb
USER=labuser
PASS=labpass
DSN="host=$PGHOST port=$PGPORT dbname=$DB user=$USER password=$PASS"

c_reset="\033[0m"
c_red="\033[31m"
c_green="\033[32m"
c_cyan="\033[36m"
c_yellow="\033[33m"

log_ok()   { printf '%(%F %T)T \033[32m[+]\033[0m %s\n' -1 "$*"; }
log_fail() { printf '%(%F %T)T \033[31m[  ]\033[0m %s\n' -1 "$*"; }

color_ip() {
  case "$1" in
    $PG1_IP) printf "${c_cyan}%s${c_reset}"   "$1" ;;
    $PG2_IP) printf "${c_yellow}%s${c_reset}" "$1" ;;
    *)       printf "%s" "$1" ;;
  esac
}

get_primary_ip() {
  for ip in "$PG1_IP" "$PG2_IP"; do
    if [[ $(PGPASSWORD=$PASS psql -h "$ip" -U "$USER" -d "$DB" -At -c "SELECT
        pg_is_in_recovery();") == "f" ]]; then
      echo "$ip"
      return
    fi
  done
```

```
  log_fail "No primary detected"
  exit 1
}

writer() {
  while true; do
    primary_ip=$(get_primary_ip)
    dsn="host=$primary_ip port=5432 dbname=$DB user=$USER password=$PASS"

    if out=$(psql "$dsn" -AtF',' \
        -c "INSERT INTO streams(talent_id,title,viewers_k,stream_date)
              VALUES (1,'live workload',123,now())
            RETURNING id, inet_server_addr();" 2>&1); then
      IFS=',' read -r id ip <<<"$out"
      log_ok "writer : inserted id=$id  backend=$(color_ip "$ip")"
    else
      log_fail "writer : $out"
    fi
    sleep 5
  done
}

reader() {
  tag=$1
  while true; do
    if out=$(psql "$DSN" -AtF',' \
         -c "SELECT count(*), inet_server_addr() FROM streams;" 2>&1); then
      IFS=',' read -r cnt ip <<<"$out"
      log_ok "$tag: rows=$cnt  backend=$(color_ip "$ip")"
    else
      log_fail "$tag: $out"
    fi
    sleep 1
  done
}

schema_inspector() {
  log_ok "inspector : inspecting schema and privileges "
  psql "$DSN" -c "\z" 2>&1 | sed 's/^/inspector : /'
}

schema_inspector

writer &
sleep 1
for n in {1..10}; do
  reader "r$n" &
  sleep 1
done

wait
```

Workload.sh

3.2 Crash & Recover

```
#!/usr/bin/env bash
set -euo pipefail

PG1=147.45.40.78
PG2=87.120.36.97
PGPORT=5432
BOUNCER_PORT=6432
DB=labdb
APPUSER=labuser
```

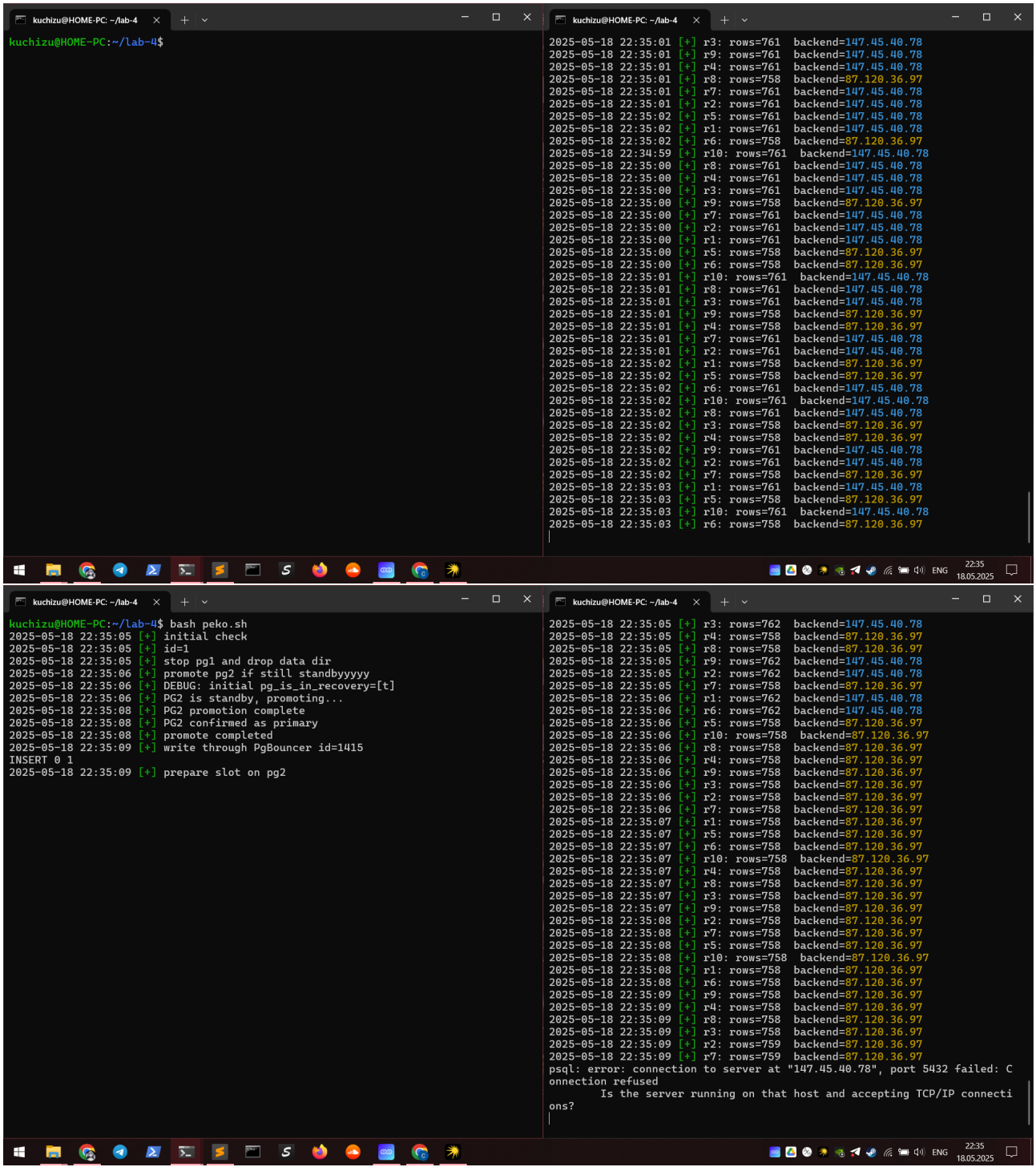
```
APPPASS=labpass
REPLUSER=replicator
REPLPASS=replpass
DATA_DIR=/var/lib/postgresql/16/main
PATH_ADDITION="/usr/bin:/usr/lib/postgresql/16/bin"
slot="pg1_slot_$(date +%s)"

g="\033[32m[+]\033[0m"
r="\033[31m[  ]\033[0m"
ok()   { printf '%(%F %T)T %b %s\n' -1 "$g" "$*"; }
fail() { printf '%(%F %T)T %b %s\n' -1 "$r" "$*"; }

# psql_b() {
#   PGPASSWORD="$APPPASS" psql "host=$PG1 port=$BOUNCER_PORT dbname=$DB user=$APPUSER" -qAt -c "$1"
# }

# psql_p2() {
#   PGPASSWORD="$APPPASS" psql -h "$PG2" -p "$PGPORT" -U "$APPUSER" -d "$DB" -qAt -c "$1"
# }

psql_b(){ PGPASSWORD=$APPPASS psql "host=$PG1 port=$BOUNCER_PORT dbname=$DB user=$APPUSER" -At -c "$1"; }

psql_p2(){
  out=$(PGPASSWORD=$APPPASS psql -h "$PG2" -p "$PGPORT" -U "$APPUSER" -d "$DB" -At -c "$1") || {
    fail "psql_p2 error while running: $1"
    exit 1
  }
  ok "$out"
}

psql_p2_bool() {
  PGPASSWORD=$APPPASS psql -h "$PG2" -p "$PGPORT" \
    -U "$APPUSER" -d "$DB" -At -c "SELECT pg_is_in_recovery();" \
  | awk '{print $1}'
}

ssh1(){ ssh root@"$PG1" "export PATH=$PATH_ADDITION:\$PATH; $*"; }
ssh2(){ ssh root@"$PG2" "export PATH=$PATH_ADDITION:\$PATH; $*"; }

ok "initial check"
row1=$(psql_b "SELECT id from streams limit 1") || { fail "write failed"; exit 1; }
ok "id=$row1"

# exit 0;

ok "stop pg1 and drop data dir"
ssh1 "systemctl stop postgresql && rm -rf $DATA_DIR"

ok "promote pg2 if still standbyyyyy"

# if [[ $(psql_p2 "SELECT pg_is_in_recovery();") == "t" ]]; then
#   ssh2 "rm -f $DATA_DIR/standby.signal && sudo -u postgres pg_ctl promote -D $DATA_DIR"
#   sleep 3
# fi

# [[ $(psql_p2 "SELECT pg_is_in_recovery()") == "f" ]] || { fail "pg2 promotion failed"; exit 1; }

status=$(psql_p2_bool)
ok "DEBUG: initial pg_is_in_recovery=[$status]" >&2

```

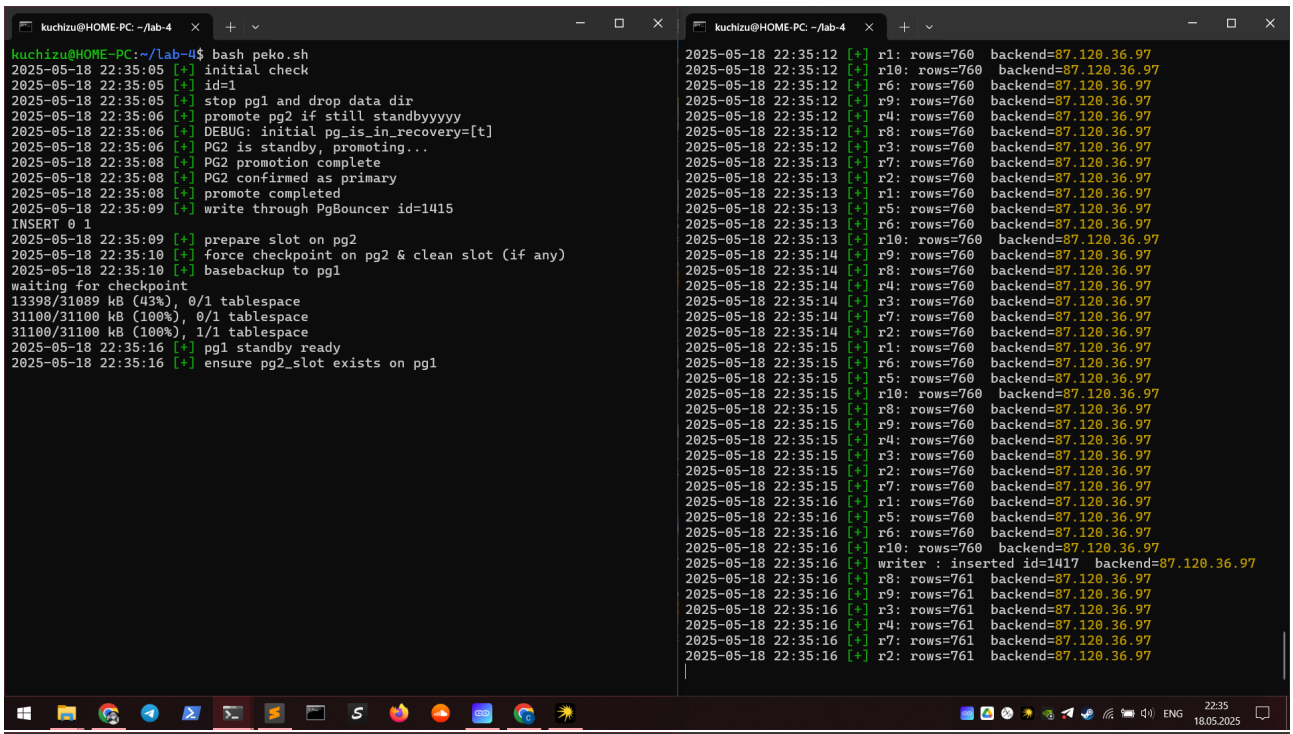
```
if [[ "$status" == "t" ]]; then
  ok "PG2 is standby, promoting..."
  ssh2 "pg_ctlcluster 16 main promote"
  until [[ "$(psql_p2_bool)" == "f" ]]; do
    sleep 1
  done
  ok "PG2 promotion complete"
fi

status=$(psql_p2_bool)
if [[ "$status" != "f" ]]; then
  fail "PG2 promotion failed (still in recovery)"
  exit 1
fi
ok "PG2 confirmed as primary"

ok "promote completed"

row1=$(psql_b "INSERT INTO streams(talent_id,title,viewers_k,stream_date) VALUES (1,'row1',111,now()) RETURNING id;")
ok "write through PgBouncer id=$row1"

ok "prepare slot on pg2"
ssh2 "sudo -u postgres psql -qAt 2>/dev/null -c \"SET client_min_messages=error; SELECT pg_drop_replication_slot('pg1_slot') WHERE EXISTS (SELECT 1 FROM pg_replication_slots WHERE slot_name='pg1_slot');\""

ok "force checkpoint on pg2 & clean slot (if any)"
ssh2 "
  sudo -u postgres psql -d postgres -qAt -c \"
      SELECT pg_drop_replication_slot('$slot')
      WHERE EXISTS (SELECT 1 FROM pg_replication_slots WHERE slot_name = '$slot');
      CHECKPOINT;
  \"
"

ok "basebackup to pg1"
ssh1 "
  rm -rf $DATA_DIR &&
  mkdir -p $(dirname $DATA_DIR) &&
  chown -R postgres:postgres $(dirname $DATA_DIR)

  sudo -u postgres env PGPASSWORD=$REPLPASS pg_basebackup \
       -h $PG2 \
       -D $DATA_DIR \
       -U $REPLUSER \
       -Fp -Xs -P -R \
       -C -S $slot -w

  systemctl start postgresql &&
  for i in {1..60}; do sudo -u postgres pg_isready -q && exit 0; sleep 1; done; exit 111
"

[ $? -eq 0 ] || { fail "pg1 failed to start"; exit 1; }

ok "pg1 standby ready"

ok "ensure pg2_slot exists on pg1"
ssh1 'sudo -u postgres psql -qAt -c "
  DO \$\$
  BEGIN
    IF NOT EXISTS (SELECT 1 FROM pg_replication_slots WHERE slot_name = ''pg2_slot'') THEN
      PERFORM pg_create_physical_replication_slot(''pg2_slot'');
    END IF;
```

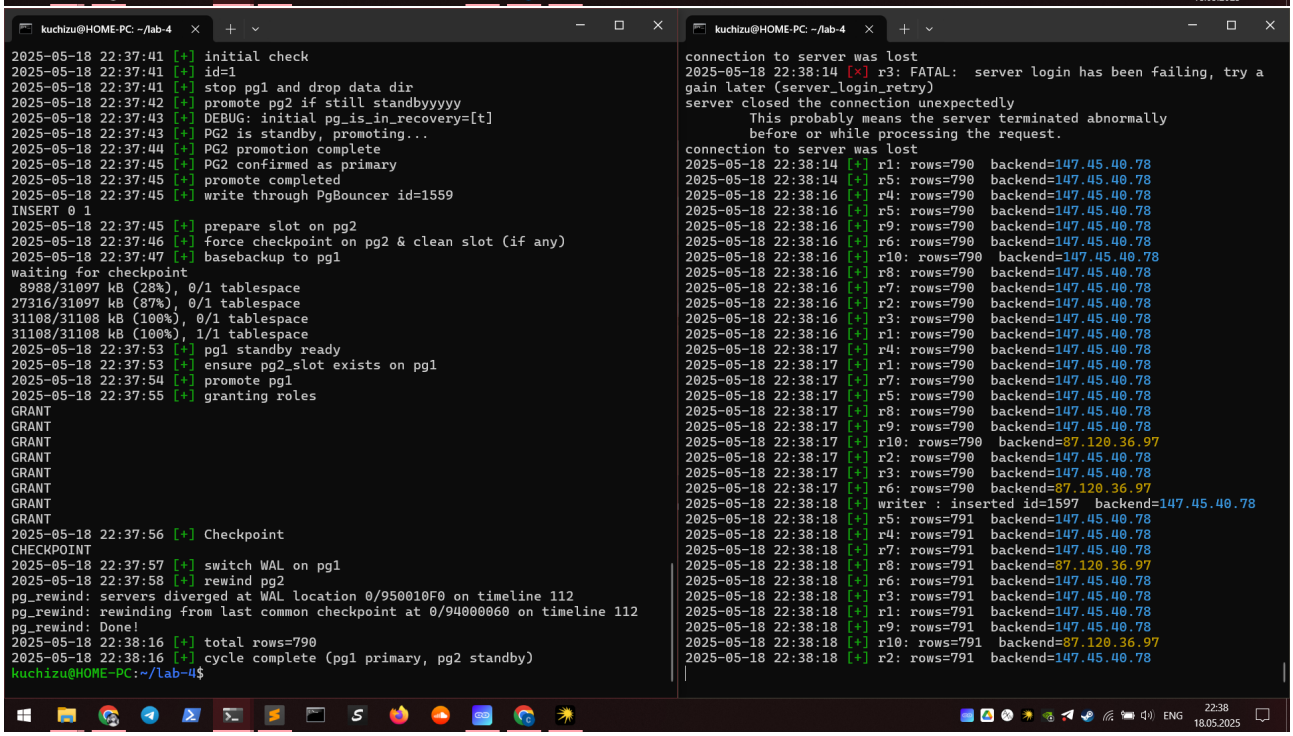
```
  END;
  \$\$;
"'

ok "promote pg1"
ssh1 "pg_ctlcluster 16 main promote"
# sleep 3

ok "granting roles"
ssh1 "
  sudo -u postgres psql -d postgres -c \"
    GRANT EXECUTE ON FUNCTION pg_read_binary_file(text)                           TO $REPLUSER;
    GRANT EXECUTE ON FUNCTION pg_read_binary_file(text, boolean)                  TO $REPLUSER;
    GRANT EXECUTE ON FUNCTION pg_read_binary_file(text, bigint, bigint)           TO $REPLUSER;
    GRANT EXECUTE ON FUNCTION pg_read_binary_file(text, bigint, bigint, boolean)  TO $REPLUSER;
    GRANT EXECUTE ON FUNCTION pg_ls_dir(text)                                     TO $REPLUSER;
    GRANT EXECUTE ON FUNCTION pg_ls_dir(text, boolean, boolean)                   TO $REPLUSER;
    GRANT EXECUTE ON FUNCTION pg_stat_file(text)                                  TO $REPLUSER;
    GRANT EXECUTE ON FUNCTION pg_stat_file(text, boolean)                         TO $REPLUSER;
  \"
"

ok "Checkpoint"
ssh1 "sudo -u postgres psql -c 'CHECKPOINT;'"

ok "switch WAL on pg1"
ssh1 "sudo -u postgres psql -q -c 'CHECKPOINT; SELECT pg_switch_wal();' > /dev/null 2>&1"

ok "rewind pg2"
ssh2 "
  systemctl stop postgresql
  sudo -u postgres env PGPASSWORD=$REPLPASS /usr/lib/postgresql/16/bin/pg_rewind \
    --target-pgdata=$DATA_DIR \
    --source-server=\"host=$PG1 port=$PGPORT user=$REPLUSER password=$REPLPASS dbname=postgres\"
  echo \"primary_conninfo = 'host=$PG1 port=$PGPORT user=$REPLUSER password=$REPLPASS'\" > $DATA_DIR/postgresql.auto.conf
  echo \"primary_slot_name = 'pg2_slot'\" >> $DATA_DIR/postgresql.auto.conf
  touch $DATA_DIR/standby.signal
  systemctl start postgresql
"

sleep 7

# row3=$(psql_b "INSERT INTO streams(talent_id,title,viewers_k,stream_date) VALUES (1,'row3',3,now()) RETURNING id") || { fail "final write failed"; exit 1; }
total=$(psql_b "SELECT count(*) FROM streams")

# ok "final write id=$row3"
ok "total rows=$total"
ok "cycle complete (pg1 primary, pg2 standby)"
```

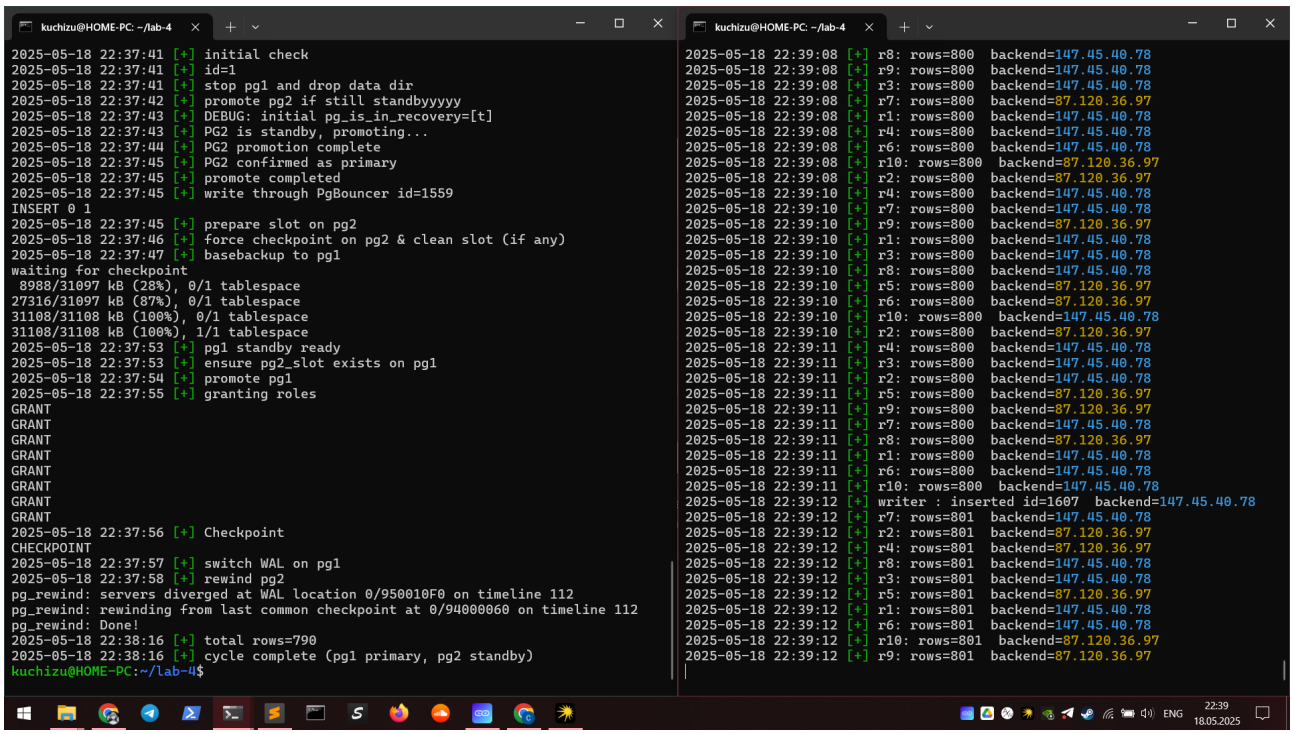
Peko.sh

3.3 Демонстрация работы




kuchizu@HOME-PC: ~/lab-4
kuchizu@HOME-PC:~/lab-4$ bash peko.sh
2025-05-18 22:35:05 [+] initial check
2025-05-18 22:35:05 [+] id=1
2025-05-18 22:35:05 [+] stop pg1 and drop data dir
2025-05-18 22:35:06 [+] promote pg2 if still standbyyyyy
2025-05-18 22:35:06 [+] DEBUG: initial pg_is_in_recovery=[t]
2025-05-18 22:35:06 [+] PG2 is standby, promoting...
2025-05-18 22:35:08 [+] PG2 promotion complete
2025-05-18 22:35:08 [+] PG2 confirmed as primary
2025-05-18 22:35:08 [+] promote completed
2025-05-18 22:35:09 [+] write through PgBouncer id=1415
INSERT 0 1
2025-05-18 22:35:09 [+] prepare slot on pg2
2025-05-18 22:35:10 [+] force checkpoint on pg2 & clean slot (if any)
2025-05-18 22:35:10 [+] basebackup to pg1
waiting for checkpoint
13398/31089 kB (43%), 0/1 tablespace
31100/31100 kB (100%), 0/1 tablespace
31100/31100 kB (100%), 1/1 tablespace
2025-05-18 22:35:16 [+] pg1 standby ready
2025-05-18 22:35:16 [+] ensure pg2_slot exists on pg1
kuchizu@HOME-PC: ~/lab-4
2025-05-18 22:35:12 [+] r1: rows=760 backend=87.120.36.97
2025-05-18 22:35:12 [+] r10: rows=760 backend=87.120.36.97
2025-05-18 22:35:12 [+] r6: rows=760 backend=87.120.36.97
2025-05-18 22:35:12 [+] r9: rows=760 backend=87.120.36.97
2025-05-18 22:35:12 [+] r4: rows=760 backend=87.120.36.97
2025-05-18 22:35:12 [+] r8: rows=760 backend=87.120.36.97
2025-05-18 22:35:12 [+] r3: rows=760 backend=87.120.36.97
2025-05-18 22:35:13 [+] r7: rows=760 backend=87.120.36.97
2025-05-18 22:35:13 [+] r2: rows=760 backend=87.120.36.97
2025-05-18 22:35:13 [+] r1: rows=760 backend=87.120.36.97
2025-05-18 22:35:13 [+] r5: rows=760 backend=87.120.36.97
2025-05-18 22:35:13 [+] r6: rows=760 backend=87.120.36.97
2025-05-18 22:35:13 [+] r10: rows=760 backend=87.120.36.97
2025-05-18 22:35:14 [+] r9: rows=760 backend=87.120.36.97
2025-05-18 22:35:14 [+] r8: rows=760 backend=87.120.36.97
2025-05-18 22:35:14 [+] r4: rows=760 backend=87.120.36.97
2025-05-18 22:35:14 [+] r3: rows=760 backend=87.120.36.97
2025-05-18 22:35:14 [+] r7: rows=760 backend=87.120.36.97
2025-05-18 22:35:14 [+] r2: rows=760 backend=87.120.36.97
2025-05-18 22:35:15 [+] r1: rows=760 backend=87.120.36.97
2025-05-18 22:35:15 [+] r6: rows=760 backend=87.120.36.97
2025-05-18 22:35:15 [+] r5: rows=760 backend=87.120.36.97
2025-05-18 22:35:15 [+] r10: rows=760 backend=87.120.36.97
2025-05-18 22:35:15 [+] r8: rows=760 backend=87.120.36.97
2025-05-18 22:35:15 [+] r9: rows=760 backend=87.120.36.97
2025-05-18 22:35:15 [+] r4: rows=760 backend=87.120.36.97
2025-05-18 22:35:15 [+] r3: rows=760 backend=87.120.36.97
2025-05-18 22:35:15 [+] r2: rows=760 backend=87.120.36.97
2025-05-18 22:35:15 [+] r7: rows=760 backend=87.120.36.97
2025-05-18 22:35:16 [+] r1: rows=760 backend=87.120.36.97
2025-05-18 22:35:16 [+] r5: rows=760 backend=87.120.36.97
2025-05-18 22:35:16 [+] r6: rows=760 backend=87.120.36.97
2025-05-18 22:35:16 [+] r10: rows=760 backend=87.120.36.97
2025-05-18 22:35:16 [+] writer : inserted id=1417 backend=87.120.36.97
2025-05-18 22:35:16 [+] r8: rows=761 backend=87.120.36.97
2025-05-18 22:35:16 [+] r9: rows=761 backend=87.120.36.97
2025-05-18 22:35:16 [+] r3: rows=761 backend=87.120.36.97
2025-05-18 22:35:16 [+] r4: rows=761 backend=87.120.36.97
2025-05-18 22:35:16 [+] r7: rows=761 backend=87.120.36.97
2025-05-18 22:35:16 [+] r2: rows=761 backend=87.120.36.97
ENG
22:35
18.05.2025


kuchizu@HOME-PC: ~/lab-4
2025-05-18 22:37:41 [+] initial check
2025-05-18 22:37:41 [+] id=1
2025-05-18 22:37:41 [+] stop pg1 and drop data dir
2025-05-18 22:37:42 [+] promote pg2 if still standbyyyyy
2025-05-18 22:37:43 [+] DEBUG: initial pg_is_in_recovery=[t]
2025-05-18 22:37:43 [+] PG2 is standby, promoting...
2025-05-18 22:37:44 [+] PG2 promotion complete
2025-05-18 22:37:45 [+] PG2 confirmed as primary
2025-05-18 22:37:45 [+] promote completed
2025-05-18 22:37:45 [+] write through PgBouncer id=1559
INSERT 0 1
2025-05-18 22:37:45 [+] prepare slot on pg2
2025-05-18 22:37:46 [+] force checkpoint on pg2 & clean slot (if any)
2025-05-18 22:37:47 [+] basebackup to pg1
waiting for checkpoint
8988/31097 kB (28%), 0/1 tablespace
27316/31097 kB (87%), 0/1 tablespace
31108/31108 kB (100%), 0/1 tablespace
31108/31108 kB (100%), 1/1 tablespace
2025-05-18 22:37:53 [+] pg1 standby ready
2025-05-18 22:37:53 [+] ensure pg2_slot exists on pg1
2025-05-18 22:37:54 [+] promote pg1
2025-05-18 22:37:55 [+] granting roles
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
2025-05-18 22:37:56 [+] Checkpoint
CHECKPOINT
2025-05-18 22:37:57 [+] switch WAL on pg1
2025-05-18 22:37:58 [+] rewind pg2
pg_rewind: servers diverged at WAL location 0/950010F0 on timeline 112
pg_rewind: rewinding from last common checkpoint at 0/94000060 on timeline 112
pg_rewind: Done!
2025-05-18 22:38:16 [+] total rows=790
2025-05-18 22:38:16 [+] cycle complete (pg1 primary, pg2 standby)
kuchizu@HOME-PC:~/lab-4$
kuchizu@HOME-PC: ~/lab-4
connection to server was lost
2025-05-18 22:38:14 [x] r3: FATAL: server login has been failing, try a
gain later (server_login_retry)
server closed the connection unexpectedly
This probably means the server terminated abnormally
before or while processing the request.
connection to server was lost
2025-05-18 22:38:14 [+] r1: rows=790 backend=147.45.40.78
2025-05-18 22:38:14 [+] r5: rows=790 backend=147.45.40.78
2025-05-18 22:38:16 [+] r4: rows=790 backend=147.45.40.78
2025-05-18 22:38:16 [+] r5: rows=790 backend=147.45.40.78
2025-05-18 22:38:16 [+] r9: rows=790 backend=147.45.40.78
2025-05-18 22:38:16 [+] r6: rows=790 backend=147.45.40.78
2025-05-18 22:38:16 [+] r10: rows=790 backend=147.45.40.78
2025-05-18 22:38:16 [+] r8: rows=790 backend=147.45.40.78
2025-05-18 22:38:16 [+] r7: rows=790 backend=147.45.40.78
2025-05-18 22:38:16 [+] r2: rows=790 backend=147.45.40.78
2025-05-18 22:38:16 [+] r3: rows=790 backend=147.45.40.78
2025-05-18 22:38:16 [+] r1: rows=790 backend=147.45.40.78
2025-05-18 22:38:17 [+] r4: rows=790 backend=147.45.40.78
2025-05-18 22:38:17 [+] r1: rows=790 backend=147.45.40.78
2025-05-18 22:38:17 [+] r7: rows=790 backend=147.45.40.78
2025-05-18 22:38:17 [+] r5: rows=790 backend=147.45.40.78
2025-05-18 22:38:17 [+] r8: rows=790 backend=147.45.40.78
2025-05-18 22:38:17 [+] r9: rows=790 backend=147.45.40.78
2025-05-18 22:38:17 [+] r10: rows=790 backend=87.120.36.97
2025-05-18 22:38:17 [+] r2: rows=790 backend=147.45.40.78
2025-05-18 22:38:17 [+] r3: rows=790 backend=147.45.40.78
2025-05-18 22:38:17 [+] r6: rows=790 backend=87.120.36.97
2025-05-18 22:38:18 [+] writer : inserted id=1597 backend=147.45.40.78
2025-05-18 22:38:18 [+] r5: rows=791 backend=147.45.40.78
2025-05-18 22:38:18 [+] r4: rows=791 backend=147.45.40.78
2025-05-18 22:38:18 [+] r7: rows=791 backend=147.45.40.78
2025-05-18 22:38:18 [+] r8: rows=791 backend=87.120.36.97
2025-05-18 22:38:18 [+] r6: rows=791 backend=147.45.40.78
2025-05-18 22:38:18 [+] r3: rows=791 backend=147.45.40.78
2025-05-18 22:38:18 [+] r1: rows=791 backend=147.45.40.78
2025-05-18 22:38:18 [+] r9: rows=791 backend=147.45.40.78
2025-05-18 22:38:18 [+] r10: rows=791 backend=87.120.36.97
2025-05-18 22:38:18 [+] r2: rows=791 backend=147.45.40.78
ENG
22:38
18.05.2025



4 Вывод

В ходе выполнения работы была реализована автоматизированная система отказоустойчивости PostgreSQL с использованием двух узлов (primary/standby), PgBouncer и скрипта, обеспечивающего полный цикл переключения ролей и восстановления репликации.(((((