

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНЫХ
ТЕХНОЛОГИЙ

Лабораторная работа №4

по дисциплине

«Распределенные системы хранения данных»

Вариант 32185

Выполнил:

Векшин Арсений Иванович Р3316

Трошкин Александр Евгеньевич Р3316

Преподаватель:

Николаев Владимир Вячеславович

Санкт-Петербург

~ 2025 ~

Задание	3
Этап 0. Настройка ВМ	5
Этап 1. Конфигурация	6
Задание	6
Конфигурация узла А	6
Конфигурация узла В	7
Конфигурация узла С	9
Настройка узла pgpool	11
Демонстрация репликации	17
Этап 2.1. Подготовка к сбою	19
Задание	19
Выполнение	19
Этап 2.2. Сбой	21
Задание	21
Выполнение	21
Логи	22
Этап 2.3. Обработка сбоя	23
Задание	23
Выполнение	23
Логи	25
Этап 3. Восстановление	26
Задание	26
Выполнение	26
Подготовка	26
Узел А	26
Узел В	27
Узел С	28
Восстановление	28
Вывод	30

Задание

Лабораторная работа №4

Введите вариант: 32185

Внимание! У разных вариантов разный текст задания!

Цель работы - ознакомиться с методами и средствами построения отказоустойчивых решений на базе СУБД Postgres; получить практические навыки восстановления работы системы после отказа.

Работа рассчитана на двух человек и выполняется в три этапа: настройка, симуляция и обработка сбоя, восстановление.

Требования к выполнению работы

- В качестве хостов использовать одинаковые виртуальные машины.
- В первую очередь необходимо обеспечить сетевую связность между VM.
- 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.

Этап 1. Конфигурация

Настроить репликацию postgres на трёх узлах: А - основной, В и С - резервные. Для управления использовать pgpool-II. Репликация с А на В синхронная. Репликация с А на С асинхронная. Продемонстрировать, что новые данные реплицируются на В в синхронном режиме, а на С с задержкой.

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 переполнение дискового пространства на основном узле - заполнить всё свободное пространство раздела с PGDATA "мусорными" файлами.

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.

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.

Этап 0. Настройка ВМ

ip-адрес виртуальных машин: 184.134.94.41

Порты для подключения: 2223 - 2226

Имя пользователя: user

Пароль: user

Параметры локальной сети ВМ:

узел pgpool: 192.168.100.10

узел A: 192.168.100.11

узел B: 192.168.100.12

узел C: 192.168.100.13

узел проху: 192.168.100.14

Порты, на которых работают postgresql

Узел A: 5432

Узел B: 5433

Узел C: 5434

```
sudo apt update
sudo apt install -y wget gnupg lsb-release

wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc |
sudo gpg --dearmor -o /usr/share/keyrings/postgresql.gpg

echo "deb [signed-by=/usr/share/keyrings/postgresql.gpg]
http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -cs)-pgdg main" |
\
sudo tee /etc/apt/sources.list.d/pgdg.list

sudo apt update
sudo apt install -y postgresql postgresql-contrib

sudo systemctl status postgresql
```

Настройка сетевого взаимодействия:

```
> sudo apt update && sudo apt install nano iputils-ping -y
> sudo nano /etc/netplan/01-netcfg.yaml
network:
  version: 2
  ethernets:
    enp0s3:                # NAT-интерфейс
      dhcp4: true          # Оставляем по DHCP
    enp0s8:                # Внутренняя сеть
      dhcp4: no
      addresses:
        - 192.168.100.11/24
> sudo netplan apply
```

Этап 1. Конфигурация

Задание

Настроить репликацию postgres на трёх узлах: А - основной, В и С - резервные. Для управления использовать pgpool-II. Репликация с А на В синхронная. Репликация с А на С асинхронная. Продемонстрировать, что новые данные реплицируются на В в синхронном режиме, а на С с задержкой.

Конфигурация узла А

/etc/postgresql/17/main/postgresql.conf

```
listen_addresses = '*'
port = 5432
wal_level = replica
max_wal_senders = 10
wal_keep_size = 256MB
archive_mode = off
archive_command = ''
hot_standby = on

# Для синхронной репликации с В
synchronous_commit = on
synchronous_standby_names = 'standby_b'
```

/etc/postgresql/17/main/pg_hba.conf

#	TYPE	DATABASE	USER	ADDRESS	METHOD
	local	all	all		trust
	host	all	all	0.0.0.0/0	md5
	host	all	all	::/0	md5
	host	replication	all	0.0.0.0/0	md5
	host	replication	all	0.0.0.0/0	md5

Создадим роли для репликации и тестов

```
psql -U postgres -c "
CREATE ROLE replica_user WITH REPLICATION LOGIN PASSWORD 'strong_password';
CREATE USER test_user WITH PASSWORD 'test_password';
ALTER USER test_user CREATEDB;
GRANT CREATE ON DATABASE postgres TO test_user;
GRANT USAGE ON ALL SCHEMAS IN DATABASE postgres TO test_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO
test_user;
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO test_user;
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT, INSERT, UPDATE,
DELETE ON TABLES TO test_user;
```

```
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT USAGE, SELECT ON SEQUENCES
TO test_user;
GRANT ALL PRIVILEGES ON SCHEMA public TO test_user;
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL ON TABLES TO test_user;

"
```

Перезапустим postgresql

```
sudo systemctl restart postgresql
ss -tulnp | grep 5432
```

```
user@rshd1:~$ ss -tulnp | grep 5432
tcp    LISTEN 0      200      0.0.0.0:5432      0.0.0.0:*
tcp    LISTEN 0      200      [::]:5432        [::]:*
user@rshd1:~$ sudo systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/usr/lib/systemd/system/postgresql.service; enabled; preset: enabled)
   Active: active (exited) since Sun 2025-06-08 17:51:30 UTC; 15s ago
     Process: 1851 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
    Main PID: 1851 (code=exited, status=0/SUCCESS)
      CPU: 3ms

Jun 08 17:51:30 rshd1 systemd[1]: Starting postgresql.service - PostgreSQL RDBMS...
Jun 08 17:51:30 rshd1 systemd[1]: Finished postgresql.service - PostgreSQL RDBMS.
```

```
psql -U replica_user -d postgres -c "SELECT pg_is_in_recovery();"

```

```
user@rshd1:~$ psql -U replica_user -d postgres -c "SELECT pg_is_in_recovery();"
 pg_is_in_recovery
-----
 f
(1 row)
```

Добавим права на исполнение для pgpool

```
> sudo visudo
user ALL=(postgres) NOPASSWD: /usr/lib/postgresql/17/bin/pg_ctl
```

Конфигурация узла В

Остановим PostgreSQL и очистим каталог данных PostgreSQL

```
sudo systemctl stop postgresql
sudo rm -rf /var/lib/postgresql/17/main/*
```

Создадим базовую резервную копию с узла А на узел В

```
sudo -u postgres pg_basebackup -h 192.168.100.11 -p 5432 -U replica_user
-D /var/lib/postgresql/17/main -Fp -Xs -P -R -v
```

Внесем изменения в /etc/postgresql/17/main/postgresql.conf

```
listen_addresses = '*'
port = 5433
hot_standby = on      # Разрешить запросы на чтение на реплике
```

Внесем изменения в /etc/postgresql/17/main/pg_hba.conf

```
local    all             all                                     trust
host     all             all             0.0.0.0/0                      md5
host     all             all             ::/0                          md5
host     replication     all             0.0.0.0/0                      md5
host     replication     all             0.0.0.0/0                      md5
```

Проверим параметры репликации

/var/lib/postgresql/17/main/postgresql.auto.conf

```
primary_conninfo = '
    user=replica_user
    password=strong_password
    channel_binding=prefer
    host=192.168.100.11
    port=5432
    sslmode=prefer
    sslnegotiation=postgres
    sslcompression=0
    sslcertmode=allow
    sslsni=1
    ssl_min_protocol_version=TLSv1.2
    gssencmode=prefer
    krbsrvname=postgres
    gssdelegation=0
    target_session_attrs=any
    load_balance_hosts=disable
    application_name=standby_b'
```

Создадим сигнальный файл, если он не был создан автоматически

```
sudo touch /var/lib/postgresql/17/main/standby.signal
```

Перезапускаем сервер на узле B

```
sudo systemctl start postgresql
```

```
user@rshd2:/var/lib/postgresql/17$ sudo tail -f /var/log/postgresql/postgresql-17-main.log
2025-06-05 13:12:40.504 UTC [2357] LOG:  listening on IPv6 address "::", port 5433
2025-06-05 13:12:40.509 UTC [2357] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5433"
2025-06-05 13:12:40.519 UTC [2360] LOG:  database system was interrupted; last known up at 2025-06-05 12:54:52 UTC
2025-06-05 13:12:41.351 UTC [2360] LOG:  starting backup recovery with redo LSN 0/2000028, checkpoint LSN 0/2000080, on timeline ID 1
2025-06-05 13:12:41.352 UTC [2360] LOG:  entering standby mode
2025-06-05 13:12:41.363 UTC [2360] LOG:  redo starts at 0/2000028
2025-06-05 13:12:41.367 UTC [2360] LOG:  completed backup recovery with redo LSN 0/2000028 and end LSN 0/2000120
2025-06-05 13:12:41.367 UTC [2360] LOG:  consistent recovery state reached at 0/2000120
2025-06-05 13:12:41.367 UTC [2357] LOG:  database system is ready to accept read-only connections
2025-06-05 13:12:41.400 UTC [2361] LOG:  started streaming WAL from primary at 0/3000000 on timeline 1
```

Лог содержит:



```
consistent recovery state reached at 0/2000120
database system is ready to accept read-only connections
started streaming WAL from primary at 0/3000000 on timeline 1
```

Значит всё отработало корректно

Проверим состояние репликации на узле А:

```
> sudo psql -u postgres -c "SELECT * FROM pg_stat_replication;"

 application_name | sync_state |
-----+-----
 standby_b       | sync      |
```

Полный вывод команды → 

```
psql -U replica_user -d postgres -c "SELECT pg_is_in_recovery();"
```

```
user@rshd2:~$ psql -U replica_user -d postgres -c "SELECT pg_is_in_recovery();"
pg_is_in_recovery
-----
t
(1 row)
```

Значит мы всё настроили корректно

Добавим права на исполнение для pgpool

```
> sudo visudo
user ALL=(postgres) NOPASSWD: /usr/lib/postgresql/17/bin/pg_ctl
```

Конфигурация узла С

Остановим PostgreSQL и очистим каталог данных PostgreSQL

```
sudo systemctl stop postgresql
sudo rm -rf /var/lib/postgresql/17/main/*
```

Создадим базовую резервную копию с узла А на узел С

```
sudo -u postgres pg_basebackup -h 192.168.100.11 -p 5432 -U replica_user
-D /var/lib/postgresql/17/main -Fp -Xs -P -R -v
```

Внесем изменения в postgresql.conf

```
sudo nano /etc/postgresql/17/main/postgresql.conf

listen_addresses = '*'
port = 5434
hot_standby = on      # Разрешить запросы на чтение на реплике
```

Внесем изменения в /etc/postgresql/17/main/pg_hba.conf

#	TYPE	DATABASE	USER	ADDRESS	METHOD
	local	all	all		trust
	host	all	all	0.0.0.0/0	md5
	host	all	all	::/0	md5
	host	replication	all	0.0.0.0/0	md5
	host	replication	all	0.0.0.0/0	md5

Проверим параметры репликации

/var/lib/postgresql/17/main/postgresql.auto.conf

```
primary_conninfo = '
    user=replica_user
    password=strong_password
    channel_binding=prefer
    host=192.168.100.11
    port=5432
    sslmode=prefer
    sslnegotiation=postgres
    sslcompression=0
    sslcertmode=allow
    sslsni=1
    ssl_min_protocol_version=TLSv1.2
    gssencmode=prefer
    krbsrvname=postgres
    gssdelegation=0
    target_session_attrs=any
    load_balance_hosts=disable
    application_name=standby_c'
```

Создадим сигнальный файл, если он не был создан автоматически

```
sudo touch /var/lib/postgresql/17/main/standby.signal
```

Перезапускаем сервер на узле C

```
sudo systemctl start postgresql
```

```
user@rshd3:/$ sudo systemctl start postgresql
user@rshd3:/$ sudo tail -f /var/log/postgresql/postgresql-17-main.log
2025-06-05 13:35:23.391 UTC [1741] LOG:  listening on IPv6 address "::", port 5434
2025-06-05 13:35:23.398 UTC [1741] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5434"
2025-06-05 13:35:23.411 UTC [1744] LOG:  database system was interrupted; last known up at 2025-06-05 13:30:18 UTC
2025-06-05 13:35:24.451 UTC [1744] LOG:  starting backup recovery with redo LSN 0/4000028, checkpoint LSN 0/4000080, on timeline ID 1
2025-06-05 13:35:24.451 UTC [1744] LOG:  entering standby mode
2025-06-05 13:35:24.463 UTC [1744] LOG:  redo starts at 0/4000028
2025-06-05 13:35:24.468 UTC [1744] LOG:  completed backup recovery with redo LSN 0/4000028 and end LSN 0/4000120
2025-06-05 13:35:24.468 UTC [1744] LOG:  consistent recovery state reached at 0/4000120
2025-06-05 13:35:24.468 UTC [1741] LOG:  database system is ready to accept read-only connections
2025-06-05 13:35:24.501 UTC [1745] LOG:  started streaming WAL from primary at 0/5000000 on timeline 1
```

Лог содержит:

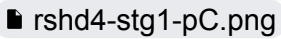
```
consistent recovery state reached at 0/4000120
database system is ready to accept read-only connections
started streaming WAL from primary at 0/5000000 on timeline 1
```

Значит всё отработало корректно

Проверим состояние репликации **на узле А:**

```
> sudo psql -u postgres -c "SELECT * FROM pg_stat_replication;"
```

application_name	sync_state
17/main	async

Полный вывод команды → 

```
psql -U replica_user -d postgres -c "SELECT pg_is_in_recovery();"

```

```
user@rshd3:~$ psql -U replica_user -d postgres -c "SELECT pg_is_in_recovery();"
pg_is_in_recovery
-----
t
(1 row)

```

Значит мы всё настроили корректно

Для будущей демонстрации, что данные действительно отправляются с задержкой, настроим задержку репликации и применим настройки без перезапуска

```
ALTER SYSTEM SET recovery_min_apply_delay = '30s';
SELECT pg_reload_conf();

```

Добавим права на исполнение для pgpool

```
> sudo visudo
user ALL=(postgres) NOPASSWD: /usr/lib/postgresql/17/bin/pg_ctl

```

Настройка узла pgpool

```
sudo apt update && sudo apt install pgpool2 postgresql-17-pgpool2 -y

```

Настроим параметры для pcp-команд

```
> sudo pg_md5 strong_password
replica_user:56b8e2260668d16fbbb98fa7038069dc
> sudo pg_md5 pcp_password
pcp_user:ce15daeecd251aee8278b12737f3fa70

```

Вывод команды поместим в /etc/pgpool2/pcp.conf

В файл `/root/.pcppass` с правами 600 поместим

```
localhost:9898:pcp_user:pcp_password

```

Создадим директории для работы pgpool

(Надо повторять после каждой перезагрузки ВМ)

```
sudo mkdir -p /var/run/pgpool
sudo chown postgres:postgres /var/run/pgpool
sudo chmod 755 /var/run/pgpool
sudo mkdir -p /var/log/pgpool
sudo chown postgres:postgres /var/log/pgpool
```

Внесем правки в /etc/pgpool2/pgpool.conf

```

listen_addresses = '*'
port = 9999
pcp_listen_addresses = '*'
pcp_port = 9898
pcp_socket_dir = '/var/run/pgpool'

# Узел A (Primary)
backend_hostname0 = '192.168.100.11'
backend_port0 = 5432
backend_weight0 = 1
backend_data_directory0 = '/var/lib/postgresql/17/main'
backend_flag0 = 'ALLOW_TO_FAILOVER'
backend_application_name0 = 'primary_a'

# Узел B (Synchronous Standby)
backend_hostname1 = '192.168.100.12'
backend_port1 = 5433
backend_weight1 = 1
backend_data_directory1 = '/var/lib/postgresql/17/main'
backend_flag1 = 'ALLOW_TO_FAILOVER'
backend_application_name1 = 'standby_b'

# Узел C (Asynchronous Standby)
backend_hostname2 = '192.168.100.13'
backend_port2 = 5434
backend_weight2 = 1
backend_data_directory2 = '/var/lib/postgresql/17/main'
backend_flag2 = 'ALLOW_TO_FAILOVER'
backend_application_name2 = 'standby_c'

sr_check_period = 10
health_check_period = 10
failover_command = '/etc/pgpool2/failover.sh'

health_check_user = 'replica_user'
health_check_password = 'strong_password'
sr_check_user = 'replica_user'
sr_check_password = 'strong_password'
search_primary_node_timeout = 1min

disable_load_balance_on_write="transaction"

```

Внесем правки в /etc/pgpool2/pool_hba.conf

#	TYPE	DATABASE	USER	CIDR-ADDRESS	METHOD
	local	all	all		trust
	host	all	all	0.0.0.0/0	trust
	host	all	all	:::1/128	trust

Добавим данные для подключения в /var/lib/postgresql/.pgpass

```
sudo -u postgres touch /var/lib/postgresql/.pgpass
sudo -u postgres chmod 0600 /var/lib/postgresql/.pgpass
sudo -u postgres nano /var/lib/postgresql/.pgpass
```

Добавим туда строки :

```
192.168.100.11:5432*:replica_user:strong_password
192.168.100.12:5433*:replica_user:strong_password
192.168.100.13:5434*:replica_user:strong_password
```

Создадим скрипт /etc/pgpool2/failover.sh

```
sudo chmod +x /etc/pgpool2/failover.sh
```

```
#!/bin/bash
set -o errexit -o pipefail

# --- Аргументы от pgpool ---
FALLEN_NODE_ID=$1 # %d: ID упавшего узла (0-based)

# --- Конфигурация ---
PCP_HOST="localhost"
PCP_PORT=9898
PCP_USER="pcp_user"
export PCPPASSFILE=/etc/pgpool2/.pcppass

# Настройка SSH
SSH_USER="user"
SSH_KEY="/var/lib/pgpool/.ssh/rshd-lab4-connection"
SSH_OPTS="-o ConnectTimeout=10 -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null"

# Журналы
LOG_DIR="/var/log/pgpool"
LOG_FILE="$LOG_DIR/failover.log"

# Сопоставление узлов: IP, порт SSH, PGDATA
declare -A NODE_HOSTS=( [0]="192.168.100.11" [1]="192.168.100.12"
[2]="192.168.100.13" )
declare -A NODE_PORTS=( [0]="22" [1]="22" [2]="22" )
declare -A NODE_PGDATA=( [0]="/var/lib/postgresql/17/main"
[1]="/var/lib/postgresql/17/main" [2]="/var/lib/postgresql/17/main" )

declare -a NODE_CHECK_ORDER=(1 2 0)

# --- Логирование ---
```

```

mkdir -p "$LOG_DIR"
touch "$LOG_FILE"
chown postgres:postgres "$LOG_FILE"
exec >> "$LOG_FILE" 2>&1
echo "-----"
echo "$(date +%Y-%m-%d %T): Failover initiated for node:
$FALLEN_NODE_ID"

validate_standby_node() {
    local node_id=$1
    echo "INFO: Checking node $node_id..."

    local node_info
    if ! node_info=$(pcp_node_info -h "$PCP_HOST" -p "$PCP_PORT" -U
"$PCP_USER" -w -n "$node_id"); then
        echo "WARN: pcp_node_info failed for node $node_id"
        return 1
    fi

    IFS=' ' read -ra INFO <<< "$node_info"
    echo "DEBUG: Raw info: $node_info"
    for i in "${INFO[@]}"; do echo "DEBUG: Parsed info: $i"; done

    local pgpool_status="${INFO[2]}"      # 3-й элемент
    local pg_status="${INFO[4]}"          # 5-й элемент
    local pg_role="${INFO[6]}"            # 7-й элемент

    echo "INFO: Node $node_id | pgpool_status=$pgpool_status,
pg_status=$pg_status, pg_role=$pg_role"

    if [[ "$pgpool_status" -eq 2 && "$pg_status" == "up" && "$pg_role"
== "standby" ]]; then
        echo "SUCCESS: Valid standby candidate"
        return 0
    else
        echo "INFO: Not suitable (pgpool_status:$pgpool_status,
pg_status:$pg_status, role:$pg_role)"
        return 1
    fi
}

promote_standby() {
    local node_id=$1
    local host="${NODE_HOSTS[$node_id]}"
    local port="${NODE_PORTS[$node_id]}"
    local pgdata="${NODE_PGDATA[$node_id]}"

```

```

    echo "INFO: Promoting node $node_id ($host:$port)..."

    if ! ssh $SSH_OPTS -i "$SSH_KEY" -p "$port" "$SSH_USER@$host" \
        "sudo -u postgres /usr/lib/postgresql/17/bin/pg_ctl promote -D
'$pgdata'"; then
        echo "ERROR: pg_ctl promote failed"
        return 1
    fi

    echo "INFO: Waiting 10s for promotion..."
    sleep 10

    if ! pcp_promote_node -h "$PCP_HOST" -p "$PCP_PORT" -U "$PCP_USER"
-w -n "$node_id"; then
        echo "ERROR: pcp_promote_node failed"
        return 1
    fi

    echo "SUCCESS: Node $node_id promoted"
    return 0
}

main() {
    for node_id in "${NODE_CHECK_ORDER[@]}; do
        [[ "$node_id" -eq "$FALLEN_NODE_ID" ]] && continue

        if validate_standby_node "$node_id"; then
            if promote_standby "$node_id"; then
                echo "$(date +"%Y-%m-%d %T"): Failover successful. New
primary: $node_id"
                exit 0
            fi
        fi
    done

    echo "FATAL: No suitable standby found"
    exit 1
}

main

```

Занык pgpool

```
sudo systemctl start pgpool2
```



```
sudo systemctl enable pgpool2 # добавим в автозапуск
```

Проверим статус pgpool

```
sudo systemctl status pgpool2
ps aux | grep pgpool
```

Проверим статус узлов

```
> sudo pcp_node_info -h localhost -p 9898 -U pcp_user -w
```

```
root@rshd-main:~# pcp_node_info -h localhost -p 9898 -U pcp_user -w
192.168.100.11 5432 2 0.333333 up up primary primary 0 none none 2025-06-11 21:30:02
192.168.100.12 5433 2 0.333333 up up standby standby 0 none none 2025-06-11 21:30:02
192.168.100.13 5434 2 0.333333 up up standby standby 0 none none 2025-06-11 21:30:02
```

Демонстрация репликации

Конфигурация системы

```
SELECT
    application_name,
    sync_state,
    replay_lag,
    pg_current_wal_lsn() - replay_lsn AS lag_bytes
FROM pg_stat_replication;
```

```
postgres=# FROM pg_stat_replication;
 application_name | sync_state | replay_lag | lag_bytes
-----+-----+-----+-----
 standby_c       | async     | 00:00:00.004936 | 0
 standby_b       | sync      | 00:00:00.004681 | 0
(2 rows)
```

Синхронная репликация

узел A

```
postgres=# CREATE TABLE test_sync (id SERIAL PRIMARY KEY, data TEXT);
CREATE TABLE
postgres=# INSERT INTO test_sync(data) VALUES ('synced_immediately');
INSERT 0 1
```

узел B

```
postgres=# SELECT * FROM test_sync;
 id | data
----+-----
  1 | synced_immediately
(1 row)
```

Асинхронная репликация

узел A

```
postgres=# INSERT INTO test_sync(data) VALUES ('async_with_delay3');
INSERT 0 1
```

узел C

```
postgres=# SELECT * FROM test_sync;
 id |      data
-----+-----
  1 | synced_immediately
  2 | async_with_delay
  3 | async_with_delay2
(3 rows)

(задержка в 30 секунд)

postgres=# SELECT * FROM test_sync;
 id |      data
-----+-----
  1 | synced_immediately
  2 | async_with_delay
  3 | async_with_delay2
  4 | async_with_delay3
(4 rows)
```

Данные появились с задержкой - profit

Этап 2.1. Подготовка к сбою

Задание

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

Выполнение

Для подключения с клиентского узла, на узле с pgpool II внесли следующие изменения

1. В /etc/pgpool2/pool_hba.conf добавили запись

```
host    postgres    test_user    all                    md5
```

2. В pgpool.conf внесли следующие изменения

```
enable_pool_hba = on
pool_passwd = '/etc/pgpool2/pool_passwd'
```

3. В /etc/pgpool2/pool_passwd добавили следующую строку

```
test_user:test_password
```

После чего подключение со стороннего узла отработало корректно

```
user@rshd-proxy:~$ psql -h 192.168.100.10 -p 9999 -U test_user -d postgres
Password for user test_user:
psql (17.5 (Ubuntu 17.5-1.pgdg24.04+1))
Type "help" for help.

postgres=> \dt
               List of relations
Schema | Name      | Type  | Owner
-----+-----+-----+-----
public | test_sync | table | postgres
(1 row)
```

Для возможности создавать таблицы от имени test_user в схеме public базы данных postgres (поскольку мы находимся в тестовой среде), раздадим пользователю test_user все привелегии

```
postgres=# GRANT ALL PRIVILEGES ON SCHEMA public TO test_user;
GRANT
postgres=# ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL ON TABLES
TO test_user;
ALTER DEFAULT PRIVILEGES
```

Теперь создадим таблицу test_table2 и посмотрим, появилась ли она на primary узле и на репликах

```
> psql -h 192.168.100.10 -p 9999 -U test_user -d postgres
postgres=> CREATE TABLE test_table1 (
```

```
id SERIAL PRIMARY KEY,  
name TEXT  
);  
CREATE TABLE
```

Узел А

```
postgres=# \dt  
List of relations  
Schema | Name | Type | Owner  
-----+-----+-----+-----  
public | test_sync | table | postgres  
public | test_table | table | test_user  
public | test_table1 | table | test_user  
(3 rows)
```

Узел В

```
postgres=# \dt  
List of relations  
Schema | Name | Type | Owner  
-----+-----+-----+-----  
public | test_sync | table | postgres  
public | test_table | table | test_user  
public | test_table1 | table | test_user  
(3 rows)
```

Узел С

```
postgres=# \dt  
List of relations  
Schema | Name | Type | Owner  
-----+-----+-----+-----  
public | test_sync | table | postgres  
public | test_table | table | test_user  
(2 rows)  
  
postgres=# \dt  
List of relations  
Schema | Name | Type | Owner  
-----+-----+-----+-----  
public | test_sync | table | postgres  
public | test_table | table | test_user  
public | test_table1 | table | test_user  
(3 rows)
```

На узле С, согласно нашим настройкам асинхронной репликации, таблица появилась не сразу, а только спустя 30 секунд. На остальных узлах таблица появилась сразу

Также на втором клиенте создали ещё одну таблицу test_table2, и всё отработало корректно.

Этап 2.2. Сбой

Задание

- Симулировать переполнение дискового пространства на основном узле - заполнить всё свободное пространство раздела с PGDATA “мусорными” файлами.

Выполнение

Скрипт для заполнения диска мусорными файлами

```
#!/bin/bash
# fill_disk.sh -- Заполняет свободное место мусорными файлами

TARGET_DIR="/var/lib/postgresql/17/main/filldisk_trash"
BLOCK_SIZE_MB=100

mkdir -p "$TARGET_DIR"
cd "$TARGET_DIR" || exit 1
echo "Свободного места до заполнения: "
df --output=avail / | tail -n 1

echo "Заполнение диска мусорными файлами в $TARGET_DIR..."
i=0
while :; do
    dd if=/dev/urandom of="trash_${i}.bin" bs=1M count=$BLOCK_SIZE_MB
    status=none || break
    ((i++))
    # Проверим, есть ли ещё свободное место
    FREE_SPACE=$(df --output=avail "$TARGET_DIR" | tail -n 1)
    if (( FREE_SPACE < BLOCK_SIZE_MB )); then
        echo "Свободное место закончилось."
        break
    fi
done

echo "Создано $i файлов, каждый по $BLOCK_SIZE_MB MB."
echo "Свободного места после заполнения: "
df --output=avail / | tail -n 1
```

Скрипт для восстановления состояния

```
#!/bin/bash
# cleanup_disk.sh -- Удаляет созданные мусорные файлы

TARGET_DIR="/var/lib/postgresql/17/main/filldisk_trash"

echo "Свободного места до очистки: "
df --output=avail / | tail -n 1

if [ -d "$TARGET_DIR" ]; then
    echo "Удаление мусорных файлов из $TARGET_DIR..."
    rm -rf "$TARGET_DIR"
    echo "Очистка завершена."
else
    echo "Папка $TARGET_DIR не найдена. Нечего удалять."
fi
echo "Свободного места после очистки: "
df --output=avail / | tail -n 1
```

Логи

```
user@rshd1:~$ sudo bash fill_disk.sh
Свободного места до заполнения:
520316
Заполнение диска мусорными файлами в /var/lib/postgresql/17/main/filldisk_trash...
Свободное место закончилось.
Создано 6 файлов, каждый по 100 MB.
Свободного места после заполнения:
0
user@rshd1:~$ sudo bash cleanup_disk.sh
Свободного места до очистки:
0
Удаление мусорных файлов из /var/lib/postgresql/17/main/filldisk_trash...
Очистка завершена.
Свободного места после очистки:
520408
```

Этап 2.3. Обработка сбоя

Задание

- 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.

Выполнение

Добавим запись в таблицу (с узла rshd-proxy)

```
postgres=> insert into test_table values(10, 'deede');  
ERROR: could not extend file "base/5/16402": No space left on device  
HINT: Check free disk space.
```

Проверим состояние узлов сразу после ошибки

```
user@rshd-main:~$ sudo pcp_node_info -h localhost -p 9898 -U pcp_user -w  
192.168.100.11 5432 3 0.333333 down down primary unknown 0 none none 2025-06-12 12:21:52  
192.168.100.12 5433 2 0.333333 up up standby standby 0 none none 2025-06-12 11:58:28  
192.168.100.13 5434 2 0.333333 up up standby standby 0 none none 2025-06-12 11:58:28
```

Посмотрим логи pgpool

```
journalctl -u pgpool2 -n 100
```

pgpool запустил автоматический поиск нового primary узла при помощи failover.sh

```
=== Starting degeneration. shutdown host 192.168.100.11(5432) ===  
LOG: Restart all children  
LOG: execute command: /etc/pgpool2/failover.sh 0 192.168.100.12 5432  
/var/lib/postgresql/17/main 1 0  
LOG: forked new pcp worker, pid=1207 socket=7  
LOG: PCP process with pid: 1207 exit with SUCCESS.  
LOG: PCP process with pid: 1207 exits with status 0  
ERROR: Failed to check replication time lag  
DETAIL: No persistent db connection for the node 0  
HINT: check sr_check_user and sr_check_password  
CONTEXT: while checking replication time lag  
LOG: forked new pcp worker, pid=1211 socket=7  
LOG: pcp_promote_node: promote option: n  
LOG: received promote backend request for node_id: 1 from pid [1211]  
LOG: PCP process with pid: 1211 exit with SUCCESS.  
LOG: PCP process with pid: 1211 exits with status 0  
LOG: find_primary_node_repeatedly: waiting for finding a primary node  
LOG: find_primary_node: primary node is 1  
LOG: find_primary_node: standby node is 2
```

```
LOG: failover: set new primary node: 1
LOG: failover: set new main node: 1
LOG: === Failover done. shutdown host 192.168.100.11(5432) ===
LOG: starting promotion. promote host 192.168.100.12(5433)
LOG: Restart all children
LOG: starting follow degeneration. shutdown host 192.168.100.11(5432)
LOG: starting follow degeneration. shutdown host 192.168.100.13(5434)
LOG: failover: 2 follow backends have been degenerated
LOG: failover: set new primary node: 1
LOG: failover: set new main node: 1
LOG: === Promotion done. promoted host 192.168.100.12(5433) ===
```

После ошибки узла А узел С автоматически отключается (каскадная репликация)

```
root@rshd-main:/home/user# pcp_node_info -h localhost -p 9898 -U pcp_user -w
192.168.100.11 5432 3 0.333333 down down standby unknown 0 none none 2025-06-12 21:15:16
192.168.100.12 5433 2 0.333333 up up primary primary 0 none none 2025-06-12 21:15:16
192.168.100.13 5434 3 0.333333 down up standby standby 0 none none 2025-06-12 21:15:16
```

Укажем pgroll что необходимо включить узел С обратно

```
sudo pcp_attach_node -h localhost -p 9898 -U pcp_user -n 2 -w
```

pgpool пытается восстановить состояние узла С

```
LOG: === Starting fail back. reconnect host 192.168.100.13(5434) ===
LOG: Node 1 is not down (status: 2)
LOG: PCP process with pid: 1287 exit with SUCCESS.
LOG: PCP process with pid: 1287 exits with status 0
LOG: Do not restart children because we are failing back node id 2
host: 192.168.100.13 port: 5434 and we are in streaming replication mode
and not all backends were down
LOG: find_primary_node_repeatedly: waiting for finding a primary node
LOG: find_primary_node: primary node is 1
LOG: find_primary_node: standby node is 2
LOG: failover: set new primary node: 1
LOG: failover: set new main node: 1
LOG: === Failback done. reconnect host 192.168.100.13(5434) ===
LOG: worker process received restart request
LOG: restart request received in pcp child process
LOG: PCP child 1277 exits with status 0 in failover()
LOG: fork a new PCP child pid 1288 in failover()
LOG: PCP process: 1288 started
```

Перезапустим pgpool

```
sudo systemctl restart pgpool2
```

pgpool сменив primary узел А -> В, восстановил состояние узла С

```
root@rshd-main:/home/user# pcp_node_info -h localhost -p 9898 -U pcp_user -w
192.168.100.11 5432 3 0.333333 down down standby unknown 0 none none 2025-06-12 21:15:16
192.168.100.12 5433 2 0.333333 up up primary primary 0 none none 2025-06-12 21:15:16
192.168.100.13 5434 2 0.333333 up up standby standby 1264 none none 2025-06-12 21:22:05
```


Проверим чтение\запись в таблицы

```
user@rshd-proxy:~$ psql -h 192.168.100.10 -p 9999 -U test_user -d postgres
Password for user test_user:
psql (17.5 (Ubuntu 17.5-1.pgdg24.04+1))
Type "help" for help.

postgres=> select * from test_table2;
 id | name
----+-----
  1 | hello
  2 | hello
  3 | hello
(3 rows)

postgres=> insert into test_table2 values (4, 'hello');
INSERT 0 1
postgres=> select * from test_table2;
 id | name
----+-----
  1 | hello
  2 | hello
  3 | hello
  4 | hello
(4 rows)

postgres=> |
```

Логи

Логи работы failover.sh

```
2025-06-12 21:15:06: Failover initiated for node: 0
INFO: Checking node 1...
DEBUG: Raw info: 192.168.100.12 5433 2 0.333333 up up standby standby 0 none none 2025-06-12 21:13:44
DEBUG: Parsed info: 192.168.100.12
DEBUG: Parsed info: 5433
DEBUG: Parsed info: 2
DEBUG: Parsed info: 0.333333
DEBUG: Parsed info: up
DEBUG: Parsed info: up
DEBUG: Parsed info: standby
DEBUG: Parsed info: standby
DEBUG: Parsed info: 0
DEBUG: Parsed info: none
DEBUG: Parsed info: none
DEBUG: Parsed info: 2025-06-12
DEBUG: Parsed info: 21:13:44
INFO: Node 1 | pgpool_status=2, pg_status=up, pg_role=standby
SUCCESS: Valid standby candidate
INFO: Promoting node 1 (192.168.100.12:22)...
Warning: Permanently added '192.168.100.12' (ED25519) to the list of known hosts.
waiting for server to promote.... done
server promoted
INFO: Waiting 10s for promotion...
pcp_promote_node -- Command Successful
SUCCESS: Node 1 promoted
2025-06-12 21:15:16: Failover successful. New primary: 1
```

Этап 3. Восстановление

Задание

- 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.

Выполнение

Подготовка

Перед восстановлением остановим pgpool

```
sudo systemctl stop pgpool2
```

Узел А

Очистим мусорные данные

```
user@rshd1:~$ sudo bash cleanup_disk.sh
[sudo] password for user:
Свободного места до очистки:
0
Удаление мусорных файлов из /var/lib/postgresql/17/main/filldisk_trash...
Очистка завершена.
Свободного места после очистки:
519300
```

Остановить PostgreSQL и удалить все данные бд.

```
sudo systemctl stop postgresql
sudo rm -rf /var/lib/postgresql/17/main/*
```

Клонируем данные с текущего primary узла

```
sudo -u postgres pg_basebackup \
-h 192.168.100.12 \
-p 5433 \
-U replica_user \
-D /var/lib/postgresql/17/main \
-Fp -Xs -R -v
```

Запускаем PostgreSQL

```
sudo systemctl start postgresql
```

Узел В

Остановить PostgreSQL и удалить все данные бд.

```
sudo systemctl stop postgresql  
sudo rm -rf /var/lib/postgresql/17/main/*
```

Клонируем данные с текущего primary узла

```
sudo -u postgres pg_basebackup \  
-h 192.168.100.11 \  
-p 5432 \  
-U replica_user \  
-D /var/lib/postgresql/17/main \  
-Fp -Xs -R -v
```

Проверим состояние /var/lib/postgresql/17/main/postgresql.auto.conf

```
primary_conninfo = '  
    user=replica_user  
    password=strong_password  
    channel_binding=prefer  
    host=192.168.100.11  
    port=5432  
    sslmode=prefer  
    sslnegotiation=postgres  
    sslcompression=0  
    sslcertmode=allow  
    sslsni=1  
    ssl_min_protocol_version=TLSv1.2  
    gssencmode=prefer  
    krbsrvname=postgres  
    gssdelegation=0  
    target_session_attrs=any  
    load_balance_hosts=disable  
    application_name=standby_b'
```

Создадим сигнальный файл, если он не был создан автоматически

```
sudo touch /var/lib/postgresql/17/main/standby.signal
```

Запускаем PostgreSQL

```
sudo systemctl start postgresql
```

Узел C

Остановить PostgreSQL и удалить все данные бд.

```
sudo systemctl stop postgresql  
sudo rm -rf /var/lib/postgresql/17/main/*
```

Клонируем данные с текущего primary узла

```
sudo -u postgres pg_basebackup \  
-h 192.168.100.11 \  
-p 5432 \  
-U replica_user \  
-D /var/lib/postgresql/17/main \  
-Fp -Xs -R -v
```

Проверим состояние /var/lib/postgresql/17/main/postgresql.auto.conf

```
primary_conninfo = '  
    user=replica_user  
    password=strong_password  
    channel_binding=prefer  
    host=192.168.100.11  
    port=5432  
    sslmode=prefer  
    sslnegotiation=postgres  
    sslcompression=0  
    sslcertmode=allow  
    sslsni=1  
    ssl_min_protocol_version=TLSv1.2  
    gssencmode=prefer  
    krbsrvname=postgres  
    gssdelegation=0  
    target_session_attrs=any  
    load_balance_hosts=disable  
    application_name=standby_c'
```

Создадим сигнальный файл, если он не был создан автоматически

```
sudo touch /var/lib/postgresql/17/main/standby.signal
```

Запускаем PostgreSQL

```
sudo systemctl start postgresql
```

Восстановление

Запускаем pgpool

```
sudo systemctl start pgpool2
```

Проверим состояние узлов

```
user@rshd-main:~$ sudo pcp_node_info -h localhost -p 9898 -U pcp_user -w
192.168.100.11 5432 2 0.333333 up up primary primary 0 none none 2025-06-13 00:59:31
192.168.100.12 5433 2 0.333333 up up standby standby 0 none none 2025-06-13 00:59:31
192.168.100.13 5434 2 0.333333 up up standby standby 0 none none 2025-06-13 00:59:31
```

Проверим состояние данных

```
postgres=> select * from test_table;
 id | name
----+-----
  1 | hello
  2 | hello
  3 | hello
  4 | hello
  5 | hello
  6 | hello
  7 | hello
  8 | hello
  9 | hello
(9 rows)

postgres=> insert into test_table values(10, 'hello');
INSERT 0 1
postgres=> select * from test_table;
 id | name
----+-----
  1 | hello
  2 | hello
  3 | hello
  4 | hello
  5 | hello
  6 | hello
  7 | hello
  8 | hello
  9 | hello
 10 | hello
(10 rows)
```

Данные, добавленные во время сбоя успешно восстановлены!

Запись новых данных - доступна

Вывод

