

Федеральное государственное автономное
образовательное учреждение высшего образования
«Научно-образовательная корпорация ИТМО»

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия

Отчёт по лабораторной работе №3

По дисциплине «Распределённые системы хранения данных» (семестр 6)

Студент:

Дениченко Александр Р3312

Практик:

Осипов Святослав

Санкт-Петербург
2025 г.

Задание

Цель работы - настроить процедуру периодического резервного копирования базы данных, сконфигурированной в ходе выполнения лабораторной работы №2, а также разработать и отладить сценарии восстановления в случае сбоев.

Узел из предыдущей лабораторной работы используется в качестве основного. Новый узел используется в качестве резервного. Учётные данные для подключения к новому узлу выдаёт преподаватель. В сценариях восстановления необходимо использовать копию данных, полученную на первом этапе данной лабораторной работы.

Этап 1. Резервное копирование

Настроить резервное копирование с основного узла на резервный следующим образом: Периодические полные копии с помощью SQL Dump. По расписанию (cron) раз в сутки, методом SQL Dump с сжатием. Созданные архивы должны сразу перемещаться на резервный хост, они не должны храниться на основной системе. Срок хранения архивов на резервной системе - 4 недели. По истечении срока хранения, старые архивы должны автоматически уничтожаться.

Изначально сделаем донастройку конфигураций прежней базы данных.

Листинг 1: kitty

```
1 docker create --name postgres-cont-1 -e POSTGRES_PASSWORD=root -p 9193:9193 postgres
```

Листинг 2: kitty

```
1 docker start postgres-cont-1
2 docker exec -it postgres-cont-1 /bin/bash
```

Изменение некоторых настроек разрешений

Листинг 3: postgresql.conf

```
1 listen_addresses = '*'
```

Листинг 4: pg_hba.conf

```
1 host      all      all      0.0.0.0/0      scram-sha-256
```

Подключение к первому узлу теперь выглядит следующим образом.

Листинг 5: kitty

```
1 psql -h 127.0.0.1 -p 9193 -U postgres -d postgres
2 psql -h 127.0.0.1 -p 9193 -U postgres -d fatrednews
3 psql -h 127.0.0.1 -p 9193 -U fatreduser -d fatrednews
```

postgres - root
fatreduser - changeMe

Табличные пространства из прошлой лабы

Листинг 6: kitty

| List of tablespaces | | | | | | |
|---------------------|----------|---------------------------|-------------------|---------|-------|--|
| Name | Owner | Location | Access privileges | Options | Size | |
| Description | | | | | | |
| pg_default | postgres | | | | 22 MB | |
| pg_global | postgres | | | | 565 | |
| sgk31 | postgres | /var/lib/postgresql/sgk31 | | | 0 | |
| bytes | | | | | | |

```

7   yrp30      | postgres | /var/lib/postgresql/yrp30 |      | 0
   bytes |
8   yva58      | postgres | /var/lib/postgresql/yva58 |      | 0
   bytes |
9   (5 rows)

```

Создан узел для хранения бэкапов

Листинг 7: kitty

```

1   docker create --name postgres-backup -e POSTGRES_PASSWORD=root -p 9194:9193 postgres
2   docker network connect pg_backup_network postgres-backup
3   docker restart postgres-backup

```

Добавлены утилиты на резервном узле и на основном

Листинг 8: kitty

```

1   apt-get install -y cron openssh-client openssh-server gzip

```

Была сделана сеть докер для обмена

Листинг 9: kitty

```

1   docker network create --driver bridge postgres-backup-net
2   docker network connect postgres-backup-net postgres-cont-1 postgres-backup
3   docker restart postgres-cont-1 postgres-backup

```

Добавим конфиг в сервер

Листинг 10: kitty

```

1   echo "PermitRootLogin yes" >> /etc/ssh/sshd_config
2   chmod 700 /root/.ssh
3   mkdir -p /run/ssh
4   chmod 755 /run/ssh
5   service ssh start

```

Добавлены ключи для упрощения обмена данными на основном сервере

Листинг 11: kitty

```

1   ssh-keygen -t rsa -b 4096
2   cat ~/.ssh/id_rsa.pub

```

И добавили ключи в резервный узел

Листинг 12: kitty

```

1   vim ~/.ssh/authorized_keys ....

```

Настройка скрипта для копирования

Листинг 13: kitty

```

1   #!/bin/bash
2
3   PG_USER="postgres"
4   BACKUP_DIR="/tmp/backups"
5   REMOTE_HOST="postgres-backup"
6   REMOTE_DIR="/backups"
7   TIMESTAMP=$(date +%Y%m%d_%H%M%S)
8   BACKUP_FILE="full_backup-$TIMESTAMP.tar.gz"
9   LOG_FILE="/tmp/backup_log.txt"
10  PGDATA="/var/lib/postgresql/data"

```

```

11 echo "Starting backup: $TIMESTAMP" >> $LOG_FILE
12
13
14 mkdir -p $BACKUP_DIR
15 BACKUP_TEMP_DIR="$BACKUP_DIR/backup_$TIMESTAMP"
16 mkdir -p "$BACKUP_TEMP_DIR"
17
18 if pg_dumpall -p 9193 -U $PG_USER > "$BACKUP_TEMP_DIR/full_backup.sql"; then
19     echo "Database dump created successfully" >> $LOG_FILE
20 else
21     echo "Database dump failed" >> $LOG_FILE
22     exit 1
23 fi
24
25 cp "$PGDATA/postgresql.conf" "$BACKUP_TEMP_DIR/" || echo "Failed to copy postgresql.conf"
26 >> $LOG_FILE
27 cp "$PGDATA/pg_hba.conf" "$BACKUP_TEMP_DIR/" || echo "Failed to copy pg_hba.conf" >>
28 $LOG_FILE
29
30 mkdir -p "$BACKUP_TEMP_DIR/pg_tblspc"
31 cp -r "$PGDATA/pg_tblspc/"* "$BACKUP_TEMP_DIR/pg_tblspc/" 2>/dev/null || echo "No
32 tablespaces found or failed to copy" >> $LOG_FILE
33
34 for tblspc in "$PGDATA/pg_tblspc/"*; do
35     if [ -L "$tblspc" ]; then
36         REAL_PATH=$(readlink -f "$tblspc")
37         mkdir -p "$BACKUP_TEMP_DIR/tblspc_data/$(basename "$tblspc")"
38         cp -r "$REAL_PATH/"* "$BACKUP_TEMP_DIR/tblspc_data/$(basename "$tblspc")/"
39     fi
40 done
41
42 tar -czf "$BACKUP_DIR/$BACKUP_FILE" -C "$BACKUP_TEMP_DIR" .
43
44 if scp "$BACKUP_DIR/$BACKUP_FILE" "$REMOTE_HOST:$REMOTE_DIR/"; then
45     echo "Backup transferred successfully" >> $LOG_FILE
46 else
47     echo "Transfer failed" >> $LOG_FILE
48     exit 1
49 fi
50
51 rm -rf "$BACKUP_TEMP_DIR"
52 rm "$BACKUP_DIR/$BACKUP_FILE"
53
54 echo "Backup script completed" >> $LOG_FILE

```

Сама настройка для автоматизации на основном хосте

Листинг 14: kitty

```

1 crontab -e
2 */1 * * * * /backup.sh
3 cron

```

Настройка для автоматизации на доп хосте

Листинг 15: kitty

```

1 crontab -e

```

```

2      */2 * * * * /cleanup_dumps.sh
3      stop

```

Расчет объема резервных копий через месяц

Исходные данные:

- Средний объем новых данных в БД за сутки: 950 МБ.
- Средний объем измененных данных за сутки: 150 МБ.
- Период хранения резервных копий: 4 недели (28 дней).

Предположения:

1. Бэкап делается полностью (включает все данные, а не только новые или измененные).
2. Измененные данные не увеличивают общий объем бэкапа, так как они уже входят в полную копию.
3. Объем БД увеличивается со временем за счет новых данных.

Каждый день создается новая полная резервная копия. Объем бэкапа на n -й день равен всему объему базы данных на тот момент.

Объем базы через n дней можно выразить как:

$$V(n) = V_0 + 950 \times n$$

Где:

- V_0 — начальный объем базы (пусть 0 для расчета за 1 месяц),
- 950 — рост базы в день.

Общий объем всех бэкапов за 28 дней:

$$V_{\text{total}} = \sum_{n=1}^{28} (950 \times n)$$

—

Рассчитаем сумму:

$$V_{\text{total}} = 950 \times (1 + 2 + \dots + 28)$$

Сумма арифметической прогрессии:

$$S = \frac{n(n+1)}{2}$$

Где $n = 28$:

$$S = \frac{28 \times 29}{2} = 406$$

Подставляем:

$$V_{\text{total}} = 950 \times 406 = 385700 \text{ МБ} = 385.7 \text{ ГБ}$$

Потеря основного узла

Листинг 16: kitty

```

1  #!/bin/bash
2
3  PG_USER="postgres"
4  BACKUP_DIR="/backups"
5  RESTORE_TEMP_DIR="/tmp/restore_temp"
6  PGDATA="/var/lib/postgresql/data"
7  LOG_FILE="/tmp/restore_log.txt"
8  BACKUP_FILE="$1"
9
10 if [ -z "$BACKUP_FILE" ]; then

```

```

11     echo "Error: Backup file not specified. Usage: $0
12         /backups/full_backup-YYYYMMDD_HHMMSS.tar.gz" >> $LOG_FILE
13     exit 1
14 fi
15 echo "Starting restore: $(date +%Y%m%d_%H%M%S)" >> $LOG_FILE
16
17 if pg_ctl -D "$PGDATA" status > /dev/null 2>&1; then
18     pg_ctl -D "$PGDATA" stop -m fast || {
19         echo "Failed to stop PostgreSQL" >> $LOG_FILE
20         exit 1
21     }
22 fi
23
24 rm -rf "$PGDATA"/* || {
25     echo "Failed to clean PGDATA" >> $LOG_FILE
26     exit 1
27 }
28
29 mkdir -p "$RESTORE_TEMP_DIR"
30 tar -xzf "$BACKUP_FILE" -C "$RESTORE_TEMP_DIR" || {
31     echo "Failed to extract backup" >> $LOG_FILE
32     exit 1
33 }
34
35 cp "$RESTORE_TEMP_DIR/postgresql.conf" "$PGDATA/" || echo "Failed to restore
36     postgresql.conf" >> $LOG_FILE
37 cp "$RESTORE_TEMP_DIR/pg_hba.conf" "$PGDATA/" || echo "Failed to restore pg_hba.conf" >>
38     $LOG_FILE
39
40 for tblspc in "$RESTORE_TEMP_DIR/pg_tblspc/"*; do
41     if [ -f "$tblspc" ]; then
42         TBL_ID=$(basename "$tblspc")
43         TBL_PATH=$(readlink "$tblspc")
44         mkdir -p "$TBL_PATH" || echo "Failed to create tablespace dir $TBL_PATH" >>
45             $LOG_FILE
46         ln -s "$TBL_PATH" "$PGDATA/pg_tblspc/$TBL_ID" || echo "Failed to link tablespace
47             $TBL_ID" >> $LOG_FILE
48     fi
49 done
50
51 if [ -d "$RESTORE_TEMP_DIR/tblspc_data" ]; then
52     for tblspc in "$RESTORE_TEMP_DIR/tblspc_data/"*; do
53         TBL_ID=$(basename "$tblspc")
54         TBL_PATH=$(readlink "$RESTORE_TEMP_DIR/pg_tblspc/$TBL_ID")
55         cp -r "$tblspc/*" "$TBL_PATH/" || echo "Failed to restore tablespace data for
56             $TBL_ID" >> $LOG_FILE
57     done
58 fi
59
60 pg_ctl -D "$PGDATA" start || {
61     echo "Failed to start PostgreSQL" >> $LOG_FILE
62     exit 1
63 }
64
65 psql -U $PG_USER -f "$RESTORE_TEMP_DIR/full_backup.sql" || {

```

```
61     echo "Failed to restore database" >> $LOG_FILE
62     exit 1
63 }
64
65 rm -rf "$RESTORE_TEMP_DIR"
66
67 echo "Restore completed successfully" >> $LOG_FILE
```

Пример восстановления базы данных

Запуск резервного узла

Листинг 17: kitty

```
1 docker exec -it postgres-backup /bin/bash
```

Должно быть включено ssh и общая есть у двух контейнеров.

Листинг 18: kitty

```
1 service ssh start
```

Делаем типо последний бэкап с основного узла перед его выходом из строя

Листинг 19: kitty

```
1 root@7f898d3520a8:/# ./backup.sh
2 full_backup-20250401_190056.tar.gz          100%   14KB   30.0MB/s   00:00
```