PostgreSQL: восстановление состояния на заданный момент времени

Стек

- CentOS 7.4
- PostgreSQL 10

Стенд

- Четыре (4) виртуальные машины (далее ВМ), сетевые интерфейсы которых (по 1 шт. на ВМ) находятся в одном широковещательном сегменте Ethernet (например, VLAN), т. е. наличествует связность на L2. ВМ имеют следующие обозначения, IP-адреса и роли:

node1.pgcluster	`10.0.0.2/28`	ВМ кластера, собранного в прошлой статье
node2.pgcluster	`10.0.0.3/28`	
node3.pgcluster	`10.0.0.4/28`	
cluster1.pgcluster	`10.0.0.5/28`	Виртуальный IP-адрес мастера для подключения к СУБД клиентского ПО
cluster2.pgcluster	`10.0.0.6/28`	Виртуальный IP-адрес мастера для подключения реплик
archive.pgcluster	`10.0.0.7/28`	ВМ для хранения архивов

- На все ВМ установлена CentOS 7.4
  - SELinux отключён

    ```
    setenforce 0

    sed -i -r 's|(SELINUX=).*|\1disabled|g' /etc/selinux/config
    ```

  - правила IPTables очищены

    ```
    systemctl disable --now iptables.service
    ```

- Указанные выше имена нод должны совпадать с выводом **`uname -n`**
- Все ноды должны резольвить друг друга по именам.

Легенда

- метка `[cluster]` означает, что команда должна выполняться на нодах кластера
- метка `[archive]` означает, что команда должна выполняться на ноде с архивом

Архивация

Концепт

- PostgreSQL на текущем мастере через команду, задаваемую параметром **archive_command**, непрерывно архивирует "отработанные" сегменты WAL на **archive.pgcluster** (с использованием **lbzip2** и **rsync**);
    - на **archive.pgcluster** периодически производится удаление самых старых архивов во избежание переполнения диска;
- на **archive.pgcluster** периодически вызывается **pg_basebackup** для создания полных копий экземпляра СУБД;
    - старые копии удаляются перед созданием новой; хранится заданное количество копий.

Установка ПО

Подключим репозиторий и установим необходимые пакеты:

```
[archive]

yum -y install 'https://download.postgresql.org/pub/repos/yum/10/redhat/rhel-7-x86_64/pgdg-centos10-10-2.noarch.rpm'
yum -y install epel-release
yum -y install postgresql10-server postgresql10-contrib rsync lbzip2

[cluster]

yum -y install lbzip2
```

Настройка хранилища бэкапов

Создадим пользователей и расширим права доступа в каталог с сокетом postgresql (при работе в производственном окружении будет разумным предварительно создать два LV и подключить их в **/srv/walbackup** и **/srv/basebackup**, или вообще разнести функции WAL archive и basebackup по разным серверам)

```
[archive]

useradd -d /srv/walbackup walbackup
useradd -G walbackup,postgres -d /srv/basebackup basebackup
echo 'PASSWORD0' | passwd --stdin basebackup
echo 'PASSWORD1' | passwd --stdin walbackup
chmod g+rwx /var/run/postgresql
chmod g+rx /srv/walbackup
```

Затем поместим в систему скрипты для бэкапа и очистки

```
[archive]

mkdir -p /srv/southbridge/bin
cd /srv/southbridge/bin && { curl \
    -O "https://raw.githubusercontent.com/nbw74/scripts/trunk/backup/pgsql/sb-pgsql-basebackupwrapper.sh" \
    -O "https://raw.githubusercontent.com/nbw74/scripts/trunk/backup/pgsql/sb-pgsql-prunewal.sh" \
    ; chmod a+x sb-pgsql-basebackupwrapper.sh sb-pgsql-prunewal.sh \
    ; cd -; }
```

Аутентификационная информация для pg_basebackup (используется соединение для репликации):

```
[archive]

sudo -iu basebackup
echo 'cluster2.pgcluster:5432:replication:replicator:REPLICATOR_PASSWORD' >> .pgpass
chmod 0600 .pgpass
```

Затем добавим записи в cron

```
sudo -iu basebackup crontab -e

# Полный архив каждые 4 дня, храним 8 архивов
04 00 * * */4 /srv/southbridge/bin/sb-pgsql-basebackupwrapper.sh --pg-version=10 --
depth=8 cluster2.pgcluster

sudo -iu walbackup crontab -e

# Каждые 19 минут проверяем заполненность диска, по достижении 86% сносим 100 самых
старых архивов WAL, и если самый новый оказывается моложе 34-х дней (4 * 8 + 2 на всякий
случай) -- шлём алерт
*/19 * * * * /srv/southbridge/bin/sb-pgsql-prunewal.sh --max-used=86 --rm-count=100 --
min-age=34
```

и для проверки запустим вручную скрипт бэкапа (от пользователя **basebackup**)

```
/srv/southbridge/bin/sb-pgsql-basebackupwrapper.sh --pg-version=10 --depth=8
cluster2.pgcluster --debug
```

(он создаст полную копию экземпляра в каталоге **cluster2.pgcluster/YYYY-MM-DDTHHMM**)

Настройка архивации WAL

Для начала следует обеспечить пользователю postgres беспарольный вход по SSH на сервер архивации как пользователю walbackup (при запросе пароля пользователя walbackup -- введите его)

```
[cluster]

sudo -iu postgres

ssh-keygen -t ed25519 -N '' -f $HOME/.ssh/id_ed25519

ssh-copy-id -i $HOME/.ssh/id_ed25519.pub walbackup@archive.pgcluster
```

Поместим на ноды кластера скрипт архивации

```
[cluster]

mkdir -p /srv/southbridge/libexec

curl 'https://raw.githubusercontent.com/nbw74/roles/trunk/postgresql/files/pgsql-archive-
command.sh' -o /srv/southbridge/libexec/pgsql-archive-command.sh

chmod a+x /srv/southbridge/libexec/pgsql-archive-command.sh
```

и внесём в конфигурационные файлы PostgreSQL требуемые параметры

/var/lib/pgsql/10/data/postgresql.conf

```
archive_mode    = on

archive_command = '/srv/southbridge/libexec/pgsql-archive-command.sh -V 10 -H
archive.pgcluster -D pgcluster -p %p -f %f'

archive_timeout = 10
```

(параметр archive_timeout указываем в учебных целях -- интенсивность записи в базу скриптом, запущенным нами в предыдущей лекции, довольно мала, и этот параметр форсирует ротацию WAL, чтобы мы могли наблюдать процесс архивации, не тратя на это десятки минут).

Затем произведём перезагрузку СУБД на мастере (не забыв установить кластеру режим обслуживания).

```
[master node]

pcs property set maintenance-mode=true

sleep 2

sudo -iu postgres /usr/pgsql-10/bin/pg_ctl -D /var/lib/pgsql/10/data restart

sleep 2

pcs property set maintenance-mode=false
```

Архивация запущена. Зайдя на архивную ноду, мы можем наблюдать увеличивающееся количество файлов в каталоге **/srv/walbackup/pgcluster**.

Упрощенное возвращение в строй выпавшего мастера

Архивация WAL позволяет проводить упрощенную процедуру ввода в строй отказавшего мастера PostgreSQL. Для этого достаточно лишь удалить лок-файл и сбросить ошибки ноды -- после этого в большинстве случаев (за исключением аварий, вызванных сложным нарушением сетевой связности и повреждениями ФС) бывший мастер подцепится слейвом к текущему.

Настройка restore_command

Для начала надо обновить параметр ресурса pgsql, отвечающий за установку параметра restore_command в **recovery.conf** (команды даны в предположении, что мастер у нас на **node1**: сначала отключаем СУБД на slave'ax, потом на master, поднимаем в обратном порядке); между командами делаем паузы в несколько две-три секунды.

```
pcs resource ban PGSQL node3.pgcluster
pcs resource ban PGSQL node2.pgcluster
pcs resource ban PGSQL node1.pgcluster
pcs resource update PGSQL restore_command='ssh walbackup@archive.pgcluster "cat
pgcluster/%f.bz2" | lbzip2 -d - > %p'
pcs resource clear PGSQL node1.pgcluster
pcs resource clear PGSQL node2.pgcluster
pcs resource clear PGSQL node3.pgcluster
```

Теперь у нас PostgreSQL при накатывании WAL берёт его из общего архива (который в **archive.pgcluster:/srv/walbackup/pgcluster**)

Пример ввода в строй экземпляра СУБД после нарушения работы

1. Имитируем сбой PostgreSQL на master

```
[master]

pkill postgres
```

   Через 20-30 секунд мы увидим, что мастер переехал на **node2** (или **node3**), слейв реплицируется с нового мастера, а старый лежит дохлым:

```
Node Attributes:

* Node node1.pgcluster:
    + PGSQL-data-status               : DISCONNECT
    + PGSQL-receiver-status           : ERROR
    + PGSQL-status                    : STOP
    + master-PGSQL                    : -INFINITY

* Node node2.pgcluster:
    + PGSQL-data-status               : LATEST
    + PGSQL-master-baseline           : 0000001472000298
    + PGSQL-receiver-status           : normal (master)
    + PGSQL-status                    : PRI
    + master-PGSQL                    : 1000

* Node node3.pgcluster:
    + PGSQL-data-status               : STREAMING|ASYNC
    + PGSQL-receiver-status           : normal
    + PGSQL-status                    : HS:async
    + master-PGSQL                    : -INFINITY
```

2. Удаляем лок-файл

```
[old dead master]

rm /var/lib/pgsql/10/tmp/PGSQL.lock
```

3. Выполняем сброс ошибок ресурса

```
pcs resource cleanup PGSQL
```

Всё. Выполнив **crm_mon -Afr** мы увидим, что через несколько секунд бывший мастер вернулся в строй как слейв. В логе СУБД при этом можно будет наблюдать что-то вроде

```
2018-10-15 19:01:57.668 +03 [6956] LOG:  restored log file "000000010000001900000033"
from archive
2018-10-15 19:01:57.905 +03 [6956] LOG:  restored log file "000000010000001900000034"
from archive
2018-10-15 19:01:58.146 +03 [6956] LOG:  restored log file "000000010000001900000035"
from archive
cat: pgcluster/000000010000001900000036.bz2: No such file or directory
lbzip2: stdin: not a valid bzip2 file
2018-10-15 19:01:58.396 +03 [7349] LOG:  started streaming WAL from primary at
19/36000000 on timeline 1
```

Восстановление из резервной копии

Мы можем восстановить данные из резервной копии двумя способами:

- Запустить СУБД непосредственно на сервере архивации, сдампить из неё нужный набор данных (базу или таблицу) и загрузить этот дамп в действующий экземпляр СУЬД;
- полностью удалить действующий экземпляр (например. в случае его полного разрушения -- DROP DATABASE или прорыва канализации над серверной) и заместить его резервной копией.

Для демонстрации восстановления из непрерывного архива рассмотрим первый случай (второй отличается от него только техническими шагами, вроде копирования файлов с одного сервера на другой).

Итак, нам нужно восстановить состояние экземпляра СУБД на некий момент времени -- например, 10 утра 18 июня 2022 г.

1. Подготовительный этап -- выставление нужных прав файлам архива WAL

   ```
   [archive]

   sudo -iu walbackup
   find /srv/walbackup/pgcluster -type f -name '*.bz2' -print0 | xargs -r0 chmod g+r
   exit
   ```

   Дальнейшие действия выполняем как пользователь basebackup. Проверить, доступен ли на чтение каталог с архивами WAL, и на запись -- каталог с сокетом; если есть проблемы - исправить.

   ```
   [archive]

   sudo -iu basebackup
   ls /srv/walbackup/pgcluster
   touch /var/run/postgresql/deleteme
   rm /var/run/postgresql/deleteme
   ```

2. заходим в каталог с резервной копией, ближайшей по времени "сзади" к моменту, на который восстанавливаемся

   ```
   cd cluster2.pgcluster/2022-06-17T0400/
   ```

3. закомментируем в файле настроек postgresql строки, отвечающие за архивацию WAL (10-я версия postgresql почему-то хочет сразу по окончании наката лога что-то заархивировать);

   ```
   sed -r -i 's|^(archive_.*)|# \1|g' postgresql.conf
   ```

4. создаём файл **recovery.conf**

   ```
   vim recovery.conf
   standby_mode = 'off'
   restore_command = 'lbzip2 -dc /srv/walbackup/pgcluster/%f.bz2 > %p'
   recovery_target_time = '2022-06-18 10:00:00 MSK'
   recovery_target_action = 'pause'
   recovery_target_timeline = 'latest'
   ```

5. запускаем СУБД

   ```
   /usr/pgsql-10/bin/pg_ctl -D $PWD start
   ```

6. смотрим в лог и ждём

```
tail -f log/$(ls -1tr log|tail -1)
```

7. увидев что-то вроде

```
2022-06-18 19:42:05.614 +03 [31697] LOG:  restored log file
"000000010000007E000000CD" from archive
2022-06-18 19:42:05.685 +03 [31697] LOG:  restored log file
"000000010000007E000000CE" from archive
2022-06-18 19:42:05.752 +03 [31697] LOG:  restored log file
"000000010000007E000000CF" from archive
2022-06-18 19:42:05.759 +03 [31697] LOG:  recovery stopping before commit of
transaction 323000, time 2022-06-18 10:00:00.307532+03
2022-06-18 19:42:05.759 +03 [31697] LOG:  recovery has paused
2022-06-18 19:42:05.759 +03 [31697] HINT:  Execute pg_wal_replay_resume() to
continue.
```

можем делать запросы, дапмы и прочие действия, например

```
pg_dump -h localhost4 -U postgres -Fp -f test.sql test
```