10 способов сделать резервную копию в PostgreSQL

**5 мин**

**227K**

[Системное администрирование\*](https://habr.com/ru/hub/sys_admin/)[Администрирование баз данных\*](https://habr.com/ru/hub/db_admins/)

Многие разговоры про бэкапы начинаются с присказки что люди делятся на две категории… так вот я отношусь к тем людям которые делают бэкапы. Правильно настроенное резервное копирование и проверка резервных копий укрепляет сон. А наличие заранее написаных и проигранных инструкций по восстановлению вообще укрепляет пищеварение и иммунитет. Так вот, за время работы с PostgreSQL мне довелось часто настраивать резервное копирование, при этом условия и требования были самые разные. Однако при этом набор инструментов за редким исключением оставался неизменным. В этой статье поделюсь своим опытом в деле, как можно брать резервные копии PostgreSQL.



Если рассматривать резервное копирование как вполне конкретный процесс, то возникает два простых вопроса:  
1. откуда запускать резервное копирование?  
2. какие инструменты следует использовать для резервного копирования?  
  
На первый вопрос есть два варианта ответа: можно запускать задачу резервного копирования с выделенного backup сервера, на мой взгляд это наиболее подходящий вариант. Либо запускать задачу непосредственно с сервера БД, это в случае если нет выделенного сервера бэкапов.  
  
С инструментами все гораздо интереснее. Здесь я выделяю две группы, основные инструменты и вспомогательные. Основные это те, которые собственно и выполняют резервное копирование. Вспомогательные это те которые добавляют что-то особенное к процессу резервного копирования, например архивирование, шифрование, управление нагрузкой и т.д.  
  
В комплекте PostgreSQL есть 2 утилиты которые позволяют делать резервные копии, это [pg\_dump](http://www.postgresql.org/docs/9.3/static/app-pgdump.html)/[pg\_dumpall](http://www.postgresql.org/docs/9.3/static/app-pg-dumpall.html) и [pg\_basebackup](http://www.postgresql.org/docs/9.3/static/app-pgbasebackup.html). Кроме того есть возможность использовать утилиты файлового копирования, такие как rsync, tar, cp и т.п.  
Итак, каким инструментом запускать бэкап?  
pg\_dump — подходит для случаев когда нужно сделать резервную копию таблицы, базы, схемы или данных.  
pg\_basebackup — подходит для случаев когда нужно сделать резервную копию целиком всего кластера БД или настроить [hot standby реплику](http://www.postgresql.org/docs/current/static/warm-standby.html#STREAMING-REPLICATION).  
rsync/tar/cp — также используются для случаев копирования всего кластера.  
  
Когда только случился релиз PostgreSQL 9.0 резервное копирование выполнялось с помощью rsync, однако уже в 9.1 появился pg\_basebackup, который имеет некоторыми преимуществами перед rsync:

* pg\_basebackup не требует ssh доступа, но требует доступа к базе указанного в [pg\_hba.conf](http://www.postgresql.org/docs/9.3/static/auth-pg-hba-conf.html);
* pg\_basebackup богаче по функциональности (копирование WAL, создание recovery.conf, встроенное сжатие gzip и пр.);
* pg\_basebackup не требует отдельного вызова функций [pg\_start\_backup/pg\_stop\_backup](http://www.postgresql.org/docs/9.3/static/functions-admin.html#FUNCTIONS-ADMIN-BACKUP-TABLE) как это требуется при использовании rsync/tar/cp;
* pg\_basebackup выполняет копирование быстрее чем rsync за счет использования протокола потоковой репликации.

но есть и некоторые недостатки:

* pg\_basebackup идет out-of-the-box, и соответственно требует установленного postgres;
* pg\_basebackup не имеет встроенных функций для ограничения скорости копирования (обещают только в 9.4);
* pg\_basebackup требует включенных опций wal\_level = hot\_standby, max\_wal\_senders в postgresql.conf.

Здесь я буду рассматривать pg\_basebackup, хотя и pg\_dump тоже может использоваться в нижеперечисленных способах.  
  
1. Простое и без изысков резервное копирование с backup сервера в каталог /backup (каталог должен быть предварительно создан):

backup@backup ~ $ pg\_basebackup -x -h db01.example.com -U backup -D /backup

2. Копирование с пониженным приоритетом IO операций с помощью ionice, для случаев когда нужно уменьшить нагрузку на дисковый ввод-вывод от резервного копирования:

postgres@db01 ~ $ ionice -c 3 pg\_basebackup -x -h db01.example.com -U backup -D /backup

3. Копирование с сжатием в bzip2, для случаев когда нужно использовать нестандартный для pg\_basebackup алгоритм сжатия (gzip). Здесь мы передаем данные через стандартный вывод (stdout) на стандартный ввод (stdin) программе bzip2.

backup@backup ~ $ pg\_basebackup -x --format=tar -h db01.example.com -U backup -D - |bzip2 -9 > /backup/db01/backup-$(date +%Y-%m-%d).tar.bz2

4. Копирование с сжатием в несколько потоков (используем lbzip2 и задействуем 6 ядер). При таком раскладе можно задействовать простаивающие ядра и ускорить процесс сжатия.

backup@backup ~ $ pg\_basebackup -x --format=tar -h db01.example.com -U backup -D - |lbzip2 -n 6 -9 > /backup/db01/backup-$(date +%Y-%m-%d).tar.bz2

5. Здесь копирование запускается на сервере БД. Формируемая резервная копия отправляется на удаленный сервер по ssh.

postgres@db01 ~ $ pg\_basebackup -x --format=tar -h 127.0.0.1 -U backup -D - |ssh backup@backup.example.com "tar xf - -C /backup/"

6. Здесь копирование также запускается на сервере БД и выполняется отправка на удаленный сервер, но уже с архивированием в 6 потоков с помощью lbzip2.

backup@backup ~ $ pg\_basebackup -x --format=tar -h 127.0.0.1 -U backup -D - |ssh backup@backup.example.com "lbzip2 -n 6 -9 > /backup/db01/backup-$(date +%Y-%m-%d).tar.bz2"

7. Копирование на удаленный сервер с ограничением пропускной полосы до 10Мб с помощью pv и последующее архивирование на удаленной стороне. Этот вариант для случаев когда нужно передать не нагружая сеть.

backup@backup ~ $ pg\_basebackup -x --format=tar -h 127.0.0.1 -U backup -D - |pv -r -b -L 10M |ssh backup@backup.example.com "bzip2 -9 > /backup/db01/backup-$(date +%Y-%m-%d).tar.bz2"

Тут стоит отметить что c 9.4 в pg\_basebackup уже есть возможность ограничения скорости передачи (-r, --max-rate).  
8. Копирование запускается на backup сервере, а далее происходит раздваивание потока на две части. Один поток сжимается с bzip2 (сам бэкап) и второй поток через tar копируется во временный каталог для последующей валидации. Способ редкоиспользуемый, но тут интересна сама реализация.

backup@backup ~ $ pg\_basebackup -x --format=tar -h db01.example.com -U backup -D - |tee >(bzip2 -9 -c > /backup/db01/backup-$(date +%d-%b-%Y).tar.bz2) |tar xf - -C /backup/validation/

9. Копирование с задействование lbzip2 на обоих узлах, для случаев когда у сети маленькая пропускная способность, сначала поток сжимается, затем передается по сети и затем расжимается на удаленной стороне. Здесь используется tar и требуется выполнение pg\_start\_backup('label\_name') на стороне postgres.

postgres@master # cd /var/lib/pgsql/9.3/data

postgres@master # tar cfO - ./ |lbzip2 -n 2 -5 |ssh postgres@standby "lbunzip2 -c -n 2 |tar xf - -C /var/lib/pgsql/9.3/data"

10. бэкапирование с шифрованием через GPG, для случаев когда нужно зашифровать резервную копию. Предварительно следует создать ключи через gpg --gen-key (в моем случае ключи созданы с именем backup)

backup@backup ~ $ pg\_basebackup -x --format=tar -h db01.example.com -U backup -D - |gpg -r backup -e |bzip2 -9 > /backup/db01/backup-$(date +%d-%b-%Y).tar.bz2

Для расшифровки резервной копии следует выполнить такую команду

backup@backup ~ $ bzcat /backup/backup-09-May-2014.tar.bz2 |gpg -r backup -d |tar xf - -C /example/dir/

На этом все, подведем итоги по инструментам:

* pg\_basebackup — утилита для создания резервных копий postgres;
* [lbzip2](http://lbzip2.org/) — bzip2 сжатие с использованием несокльких ядер — если нужно запаковать быстрее (аналоги: [pbzip2](http://compression.ca/pbzip2/), [pigz](http://zlib.net/pigz/));
* ionice — регулировка класса и приоритета для планировщика ввода-вывода (также можно использовать nice для регулировки приоритета процессов для CPU планировщика);
* [pv](http://linux.die.net/man/1/pv) — контролируем объем передаваемых данных через pipe и т.о. используем для ограничения объема передаваемых данных в единицу времени (аналог — [throttle](http://linux.die.net/man/1/throttle));
* [tar](http://www.gnu.org/software/tar/) — утилита архивирования, нужна для вспомогательных целей когда неиспользуется сжатие bzip2/gzip;
* tee — чтение с stdin c записью в stdout и другие файлы (является частью coreutils);
* [gpg](https://www.gnupg.org/) — решает задачи по шифрованию.

Всем спасибо за внимание!