

Базовый инструментарий Установка и управление, psql



Авторские права

© Postgres Professional, 2017–2021

Авторы: Егор Рогов, Павел Лузанов

Использование материалов курса

Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

Обратная связь

Отзывы, замечания и предложения направляйте по адресу:

edu@postgrespro.ru

Отказ от ответственности

Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или косвенным, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

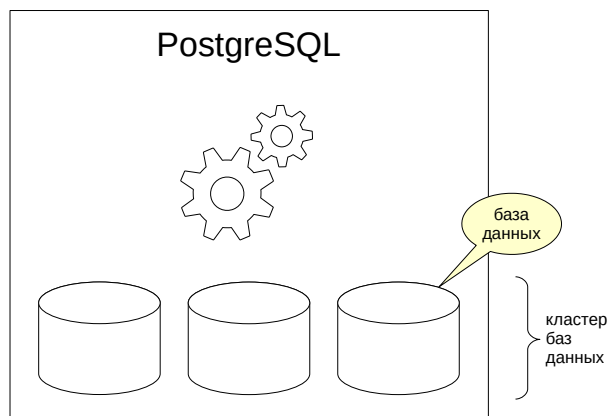
Варианты установки PostgreSQL

Управление сервером

Журнал сообщений сервера

Настройка параметров конфигурации

Использование psql



Начнем с основных понятий.

PostgreSQL — программа, которая относится к классу *систем управления базами данных*.

Когда эта программа выполняется, мы называем ее *сервером PostgreSQL* или *экземпляром сервера*. Пока сервер представляется для нас «черным ящиком», но постепенно мы познакомимся с тем, как он устроен.

Данные, которыми управляет PostgreSQL, хранятся в *базах данных*. Один экземпляр PostgreSQL одновременно работает с несколькими базами данных. Этот набор баз данных называется *кластером баз данных*. Подробнее мы будем говорить о базах данных в теме «Организация данных. Логическая структура».

Итак: кластер баз данных — это данные в файлах; сервер или экземпляр сервера — программа, управляющая кластером баз данных.

Варианты

- готовые пакеты (предпочтительный способ)
- установка из исходных кодов
- без установки — облачные сервисы

Расширения

- дополнительный функционал
- устанавливаются отдельно
- в поставке с сервером — модули и программы (~50 штук)

Предпочтительный вариант установки PostgreSQL — использование пакетных менеджеров (таких, как `rpm` или `dpkg`) и готовых пакетов. В этом случае получается понятная, поддерживаемая и легко обновляемая установка. Пакеты существуют для большинства операционных систем.

Другой вариант — самостоятельная сборка PostgreSQL из исходных кодов. Это может понадобиться для установки нестандартных значений параметров или при использовании не очень популярной платформы.

Готовые пакеты и исходные коды: <http://www.postgresql.org/download/>

Кроме того, можно использовать готовые облачные решения, что позволяет обойтись вообще без установки. Такую возможность дают все ведущие зарубежные (Amazon RDS, Google Cloud SQL, Microsoft Azure) и отечественные (Яндекс.Облако, облако Mail.ru) платформы.

В курсе мы будем использовать виртуальную машину с ОС Xubuntu и сервер PostgreSQL, установленный из пакета для этой ОС.

Для PostgreSQL существует большое количество расширений, которые подключают новый функционал к СУБД «на лету», без изменения ядра системы. В состав дистрибутива входит примерно 50 расширений.


<https://postgrespro.ru/docs/postgresql/12/contrib>

<https://postgrespro.ru/docs/postgresql/12/contrib-prog>

Список доступных расширений и статус их установки можно посмотреть в представлении `pg_available_extensions`.

Утилита для управления

pg_ctlcluster

 pg_ctl

Основные задачи

запуск сервера

останов сервера

обновление параметров конфигурации

К основным операциям управления сервером относятся начальная инициализация кластера баз данных, запуск и останов сервера, обновление конфигурации и некоторые другие. Для выполнения этих действий предназначена утилита `pg_ctl`, идущая в составе PostgreSQL.

В пакетном дистрибутиве для Ubuntu доступ к утилите `pg_ctl` осуществляется не напрямую, а через специальную обертку `pg_ctlcluster`. Справку по использованию `pg_ctlcluster` можно получить командой:

```
$ man pg_ctlcluster
```

Более подробная информация об управлении сервером для администраторов баз данных:

<https://postgrespro.ru/docs/postgresql/12/app-pg-ctl>

<https://postgrespro.ru/docs/postgresql/12/runtime>

Установка и управление

В виртуальной машине курса установка выполнена из пакета. Каталог установки PostgreSQL:

```
student$ ls -l /usr/lib/postgresql/12
```

```
total 8
drwxr-xr-x 2 root root 4096 дек  9  2020 bin
drwxr-xr-x 4 root root 4096 дек  9  2020 lib
```

Владелец ПО сервера — пользователь root.

Кластер баз данных автоматически инициализируется при установке из пакета и находится в каталоге /var/lib/postgresql/12/main.

В последующих темах мы будем ссылаться на этот каталог как PGDATA, по имени переменной ОС, которую можно установить для использования в некоторых утилитах сервера.

Владельцем каталога является пользователь postgres. Вот его содержимое:

```
student$ sudo ls -l /var/lib/postgresql/12/main
```

```
total 84
drwx----- 8 postgres postgres 4096 июл 26 11:03 base
drwx----- 2 postgres postgres 4096 июл 26 11:03 global
drwx----- 2 postgres postgres 4096 июл 26 11:02 pg_commit_ts
drwx----- 2 postgres postgres 4096 июл 26 11:02 pg_dynshmem
drwx----- 4 postgres postgres 4096 июл 26 11:03 pg_logical
drwx----- 4 postgres postgres 4096 июл 26 11:02 pg_multixact
drwx----- 2 postgres postgres 4096 июл 26 11:03 pg_notify
drwx----- 2 postgres postgres 4096 июл 26 11:02 pg_replslot
drwx----- 2 postgres postgres 4096 июл 26 11:02 pg_serial
drwx----- 2 postgres postgres 4096 июл 26 11:02 pg_snapshots
drwx----- 2 postgres postgres 4096 июл 26 11:02 pg_stat
drwx----- 2 postgres postgres 4096 июл 26 11:02 pg_stat_tmp
drwx----- 2 postgres postgres 4096 июл 26 11:02 pg_subtrans
drwx----- 2 postgres postgres 4096 июл 26 11:02 pg_tblspc
drwx----- 2 postgres postgres 4096 июл 26 11:02 pg_twophase
-rw----- 1 postgres postgres   3 июл 26 11:02 PG_VERSION
drwx----- 3 postgres postgres 4096 июл 26 11:02 pg_wal
drwx----- 2 postgres postgres 4096 июл 26 11:02 pg_xact
-rw----- 1 postgres postgres  88 июл 26 11:02 postgresql.auto.conf
-rw----- 1 postgres postgres 130 июл 26 11:03 postmaster.opts
-rw----- 1 postgres postgres 108 июл 26 11:03 postmaster.pid
```

При установке из пакета в настройки запуска ОС добавляется запуск PostgreSQL. Поэтому после загрузки операционной системы отдельно стартовать PostgreSQL не нужно.

Можно явным образом управлять сервером с помощью следующих команд, которые выдаются либо от имени пользователя ОС postgres, либо через sudo:

Остановить сервер:

```
student$ sudo pg_ctlcluster 12 main stop
```

Запустить сервер:

```
student$ sudo pg_ctlcluster 12 main start
```

Перезапустить:

```
student$ sudo pg_ctlcluster 12 main restart
```

Обновить конфигурацию:

```
student$ sudo pg_ctlcluster 12 main reload
```

В журнал записываются

- служебные сообщения сервера
- сообщения пользовательских сеансов
- сообщения приложений

Настройка журнала

- расположение
- формат записей
- какие события регистрировать

Информация о ходе работы СУБД записывается в журнал сообщений сервера. Сюда попадают сведения о запуске и останове сервера, различные служебные сообщения о возникающих проблемах.

Также сюда могут попадать сообщения о выполняющихся командах и времени их работы, о возникающих блокировках и тому подобное. Это позволяет выполнять трассировку пользовательских сеансов.

Разработчики приложений могут записывать в журнал сервера свои собственные сообщения.

Настройки PostgreSQL позволяют гибко определять, какие именно сообщения и в каком формате должны попадать в журнал сервера.

Например, вывод в формате csv удобен для автоматизации анализа журнала.

<https://postgrespro.ru/docs/postgresql/12/runtime-config-logging>

Журнал сообщений сервера

Журнал сообщений сервера находится здесь:

```
student$ ls -l /var/log/postgresql/postgresql-12-main.log
```

```
-rw-r----- 1 postgres adm 838 июл 26 11:03 /var/log/postgresql/postgresql-12-main.log
```

Заглянем в конец журнала:

```
student$ tail -n 10 /var/log/postgresql/postgresql-12-main.log
```

```
2023-07-26 11:03:06.213 MSK [2322] LOG:  starting PostgreSQL 12.5 (Ubuntu 12.5-1.pgdg20.04+1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0, 64-bit
2023-07-26 11:03:06.214 MSK [2322] LOG:  listening on IPv4 address "127.0.0.1", port 5432
2023-07-26 11:03:06.224 MSK [2322] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2023-07-26 11:03:06.267 MSK [2323] LOG:  database system was shut down at 2023-07-26 11:02:58 MSK
2023-07-26 11:03:06.326 MSK [2322] LOG:  database system is ready to accept connections
2023-07-26 11:03:12.711 MSK [2322] LOG:  received SIGHUP, reloading configuration files
2023-07-26 11:03:12.753 MSK [2443] student@student ERROR:  database "tools_overview" does not exist
2023-07-26 11:03:12.753 MSK [2443] student@student STATEMENT:  DROP DATABASE tools_overview;
```


Для всего экземпляра

основной файл параметров — postgresql.conf

ALTER SYSTEM — postgresql.auto.conf

Для текущего сеанса

SET/RESET

set_config()

Просмотр текущего значения

SHOW

current_setting()

pg_settings

Сервер PostgreSQL настраивается с помощью разнообразных параметров конфигурации, которые служат для управления потреблением ресурсов, настройки служебных процессов и пользовательских сеансов, управления журналом сервера и для многих других задач. В ходе курса мы будем встречаться с некоторыми из этих параметров. А сейчас важно разобраться с тем, как проверить текущие значения и установить новые.

Настройки сервера обычно задаются в конфигурационных файлах. Основной конфигурационный файл — postgresql.conf, он редактируется вручную. Второй конфигурационный файл — postgresql.auto.conf — предназначен для изменения специальной командой ALTER SYSTEM. Параметры, установленные через ALTER SYSTEM, имеют приоритет над параметрами в postgresql.conf.

Большинство параметров конфигурации допускает изменение значений в пользовательских сеансах прямо во время выполнения.

Варианты установки и управления параметрами:

<https://postgrespro.ru/docs/postgresql/12/config-setting>

Текущие значения доступны в представлении pg_settings:

<https://postgrespro.ru/docs/postgresql/12/view-pg-settings>

Параметры конфигурации

Основной файл конфигурации postgresql.conf расположен в этом каталоге:

```
student$ ls -l /etc/postgresql/12/main
```

```
total 56
drwxr-xr-x 2 postgres postgres 4096 июл 26 11:03 conf.d
-rw-r--r-- 1 postgres postgres 315 июл 26 11:03 environment
-rw-r--r-- 1 postgres postgres 143 июл 26 11:03 pg_ctl.conf
-rw-r----- 1 postgres postgres 4933 июл 26 11:03 pg_hba.conf
-rw-r----- 1 postgres postgres 1636 июл 26 11:03 pg_ident.conf
-rw-r--r-- 1 postgres postgres 26922 июл 26 11:03 postgresql.conf
-rw-r--r-- 1 postgres postgres 317 июл 26 11:03 start.conf
```

Здесь же находятся и другие конфигурационные файлы.

Проверим значение параметра work_mem:

```
=> SHOW work_mem;
```

```
work_mem
-----
4MB
(1 row)
```

Параметр work_mem задает объем памяти, который будет использоваться для внутренних операций сортировки и хеш-таблиц, прежде чем будут задействованы временные файлы на диске.

4MB — это значение по умолчанию, и оно слишком мало. Допустим, мы хотим увеличить его до 16MB для всего экземпляра. Для этого есть два пути.

Во-первых, можно внести изменение в postgresql.conf, раскомментировав и изменив строку, где определяется параметр:

```
student$ grep '#work_mem' /etc/postgresql/12/main/postgresql.conf
```

```
#work_mem = 4MB                                # min 64kB
```

Во-вторых, можно изменить параметр с помощью команды SQL — что мы и сделаем:

```
=> ALTER SYSTEM SET work_mem TO '16MB';
```

```
ALTER SYSTEM
```

Такое изменение попадает не в postgresql.conf, а в другой файл (он находится в каталоге PGDATA):

```
student$ sudo cat /var/lib/postgresql/12/main/postgresql.auto.conf
```

```
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
work_mem = '16MB'
```

Чтобы изменение вступило в силу, нужно перечитать конфигурационные файлы. Можно либо воспользоваться pg_ctlcluster, либо использовать функцию SQL:

```
=> SELECT pg_reload_conf();
```

```
pg_reload_conf
-----
t
(1 row)
```

Проверим, что новое значение параметра применилось. Кроме команды SHOW можно сделать это таким образом:

```
=> SELECT current_setting('work_mem');
```

```
current_setting
-----
16MB
(1 row)
```

Чтобы восстановить значение параметра по умолчанию, достаточно вместо SET использовать команду RESET (и, конечно, перечитать конфигурационные файлы):

```
=> ALTER SYSTEM RESET work_mem;
```

```
ALTER SYSTEM
```

```
=> SELECT pg_reload_conf();
```

```
pg_reload_conf
-----
t
(1 row)
```

Большинству параметров можно установить новое значение для текущего сеанса во время выполнения. Например, если мы собираемся выполнить запрос, сортирующий большой объем данных, то для сеанса можно увеличить значение `work_mem`:

```
=> SET work_mem = '64MB';
```

SET

Проверим, что значение применилось, еще одним способом — выполним запрос к представлению:

```
=> SELECT name, setting, unit FROM pg_settings WHERE name = 'work_mem';
```

```
name | setting | unit
-----+-----+-----
work_mem | 65536 | kB
(1 row)
```

Новое значение действует только в текущем сеансе или в текущей транзакции (при указании `SET LOCAL`).

Терминальный клиент для работы с PostgreSQL

Поставляется вместе с СУБД

Используется администраторами и разработчиками для интерактивной работы и выполнения скриптов

Для работы с СУБД PostgreSQL существуют различные сторонние инструменты, рассмотрение которых не входит в рамки курса.

В курсе мы будем использовать терминальный клиент psql:

1. psql — это единственный клиент, поставляемый вместе с СУБД.
2. Навыки работы с psql пригодятся разработчикам и администраторам вне зависимости от того, с каким инструментом они будут работать.

Для интерактивной работы в psql встроена поддержка readline, программ постраничного просмотра результатов запросов (таких, как less). Возможности psql позволяют взаимодействовать с ОС, просматривать содержимое системного каталога, создавать скрипты для автоматизации повторяющихся задач.

<https://postgrespro.ru/docs/postgresql/12/app-psql>

Подключение

При запуске `psql` нужно указать параметры подключения. К обязательным параметрам относятся:

- имя базы данных, по умолчанию совпадает с именем пользователя;
- имя пользователя (роль), по умолчанию совпадает с именем пользователя ОС;
- узел, по умолчанию — локальное соединение;
- порт, по умолчанию — обычно 5432.

Параметры указываются так:

```
student$ psql -d база -U роль -h узел -p порт
```

Настройки, сделанные в виртуальной машине, позволяют подключаться к PostgreSQL без указания параметров:

```
student$ psql
```

Проверим текущее подключение:

```
=> \conninfo
```

You are connected to database "student" as user "student" via socket in "/var/run/postgresql" at port "5432".

Команда `\connect` выполняет новое подключение, не покидая `psql`. Ее можно сократить до `\с`. Мы будем указывать необязательную часть имени команды в квадратных скобках: `\с[onnect]`.

Справочная информация

Справку по `psql` можно получить не только в документации, но и прямо в системе. Команда

```
student$ psql --help
```

выдает справку по запуску. А если в системе установлена документация, то справочное руководство можно получить командой

```
student$ man psql
```

Утилита `psql` умеет выполнять команды SQL и свои собственные команды, которые начинаются с обратной косой черты, как `\conninfo`.

Внутри `psql` есть возможность получить список и краткое описание команд:

- `\h[elp]` выдает список команд SQL, которые поддерживает сервер, а также синтаксис конкретной команды SQL.
- `\?` выдает список команд `psql`.

Форматирование вывода

Клиент `psql` умеет выводить результат запросов в разных форматах:

- формат с выравниванием значений;
- формат без выравнивания;
- расширенный формат.

Формат с выравниванием используется по умолчанию:

```
=> SELECT name, setting, unit FROM pg_settings LIMIT 7;
```

name	setting	unit
allow_system_table_mods	off	
application_name	psql	
archive_cleanup_command		
archive_command	(disabled)	
archive_mode	off	
archive_timeout	0	s
array_nulls	on	

(7 rows)

Ширина столбцов выровнена по значениям. Также выводится строка заголовков и итоговая строка.

Команды `psql` для переключения режима выравнивания:

- `\a` — переключатель режима: с выравниванием/без выравнивания.
- `\t` — переключатель отображения строки заголовка и итоговой строки.

Отключим выравнивание, заголовок и итоговую строку:

```
=> \a \t
```

Output format is unaligned.
Tuples only is on.

```
=> SELECT name, setting, unit FROM pg_settings LIMIT 7;
```

```
allow_system_table_mods|off|
application_name|psql|
archive_cleanup_command||
archive_command|(disabled)|
archive_mode|off|
archive_timeout|0|s
array_nulls|on|
```

```
=> \a \t
```

Output format is aligned.
Tuples only is off.

Такой формат неудобен для просмотра, но может оказаться полезным для автоматической обработки.

Расширенный формат удобен, когда нужно вывести много столбцов для одной или нескольких записей. Для этого вместо точки с запятой указываем в конце команды \gx:

```
=> SELECT name, setting, unit, category, context, vartype,
        min_val, max_val, boot_val, reset_val
FROM pg_settings
WHERE name = 'work_mem' \gx
```

```
-[ RECORD 1 ]-----
name       | work_mem
setting    | 4096
unit       | kB
category   | Resource Usage / Memory
context    | user
vartype     | integer
min_val    | 64
max_val    | 2147483647
boot_val   | 4096
reset_val  | 4096
```

Если расширенный формат нужен не для одной команды, а постоянно, можно включить его переключателем \x.

Все возможности форматирования результатов запросов доступны через команду \pset.

Взаимодействие с ОС и выполнение скриптов

Из psql можно выполнять команды shell:

```
=> \! pwd
```

```
/home/student
```

С помощью запроса SQL можно сформировать несколько других запросов SQL и записать их в файл, используя команду \o[ut]:

```
=> \a \t
```

Output format is unaligned.
Tuples only is on.

```
=> \pset fieldsep ''
```

Field separator is "".

```
=> \o dev1_psql.log
```

```
=> SELECT format('SELECT %L AS tbl, count(*) FROM %I;', tablename, tablename)
FROM pg_tables LIMIT 3;
```

На экран ничего не попало. Посмотрим в файле:

```
=> \! cat dev1_psql.log
```

```
SELECT 'pg_statistic' AS tbl, count(*) FROM pg_statistic;
SELECT 'pg_type' AS tbl, count(*) FROM pg_type;
SELECT 'pg_foreign_server' AS tbl, count(*) FROM pg_foreign_server;
```

Вернем вывод на экран и восстановим форматирование по умолчанию.

```
=> \o \t \a
```

Tuples only is off.
Output format is aligned.

Выполним теперь эти команды из файла с помощью \i[nclude]:

```
=> \i dev1_psql.log
```

```
      tbl      | count
-----+-----
pg_statistic |    422
(1 row)

      tbl      | count
-----+-----
pg_type      |    406
(1 row)

      tbl      | count
-----+-----
pg_foreign_server |      0
(1 row)
```

Впрочем, то же самое можно получить за один шаг, используя команду \gexec:

```
=> SELECT format('SELECT %L AS tbl, count(*) FROM %I;', tablename, tablename)
FROM pg_tables LIMIT 3 \gexec
```

```

      tbl      | count
-----+-----
pg_statistic |    422
(1 row)

      tbl      | count
-----+-----
pg_type      |    406
(1 row)

      tbl      | count
-----+-----
pg_foreign_server |      0
(1 row)

```

Другие способы выполнить команды из файла:

- `psql < имя_файла`
- `psql -f имя_файла`
- `psql -c 'команда'` (работает только для одной команды)

Переменные psql

По аналогии с shell, psql имеет собственные переменные.

Установим переменную:

```
=> \set TEST Hi!
```

Чтобы получить значение, надо предварить имя переменной двоеточием:

```
=> \echo :TEST
```

```
Hi!
```

Значение переменной можно сбросить:

```
=> \unset TEST
```

```
=> \echo :TEST
```

```
:TEST
```

Переменные можно использовать, например, для хранения текста часто используемых запросов. Вот запрос на получение пяти самых больших по размеру таблиц:

```
=> \set top5 'SELECT tablename, pg_total_relation_size(schemaname||'.'||tablename) AS bytes FROM pg_tables ORDER BY bytes DESC LIMIT 5;'
```

Для выполнения запроса достаточно набрать:

```
=> :top5
```

```

  tablename      | bytes
-----+-----
pg_depend       | 1130496
pg_proc         | 1015808
pg_attribute    |  688128
pg_rewrite      |  679936
pg_description  |  573440
(5 rows)

```

Присвоение значения переменной top5 лучше записать в стартовый файл .psqlrc в домашнем каталоге пользователя. Команды из .psqlrc будут автоматически выполняться каждый раз при старте psql.

Результат запроса можно записать в переменную с помощью \gset:

```
=> SELECT current_setting('work_mem') AS current_work_mem \gset
```

```
=> \echo Значение work_mem: :current_work_mem
```

```
Значение work_mem: 4MB
```

Без параметров \set выдает значения всех переменных, включая встроенные. Справку по встроенным переменным можно получить так:

```
\? variables
```

Установка PostgreSQL из готовых пакетов —
предпочтительный способ установки

Пакетные дистрибутивы учитывают особенности ОС,
которые нужно знать

- как запускать и останавливать сервер
- расположение файлов конфигурации
- расположение журнала сервера

psql — клиент для работы с PostgreSQL

1. Установите в `postgresql.conf` для параметра `work_mem` значение 8 Мбайт.
Обновите конфигурацию и проверьте, что изменения вступили в силу.
2. Запишите в файл `ddl.sql` команду `CREATE TABLE` на создание любой таблицы.
Запишите в файл `populate.sql` команды на вставку строк в эту таблицу.
Войдите в `psql`, выполните оба скрипта и проверьте, что таблица создалась и в ней появились записи.
3. Найдите в журнале сервера строки за сегодняшний день.

Для выполнения практических заданий нужно войти в операционную систему под пользователем `student` (пароль `student`).

Для запуска `psql` в окне терминала наберите `psql` без параметров. Для подключения будут использованы настройки по умолчанию.

```
student:~$ psql
```

Для выполнения заданий каждой темы удобно создавать отдельную базу данных:

```
student/student=# CREATE DATABASE tools_overview;
```

```
CREATE DATABASE
```

```
student/student=# \c tools_overview
```

```
You are now connected to database "tools_overview" as user "student".
```

```
student/tools_overview=#
```

1. Воспользуйтесь любым текстовым редактором. В виртуальной машине установлены `mousepad`, `gedit`, `vim`, `nano`.

Обратите внимание: если вы запускаете редактор не из терминала, а из графической среды, он будет запущен под пользователем ОС `student`. В этом случае у вас не будет прав на редактирование файла `postgresql.conf`, который принадлежит пользователю `postgres`.

1. Параметры конфигурации

Добавим в конец конфигурационного файла строку:

```
student$ echo 'work_mem = 8MB' | sudo tee -a /etc/postgresql/12/main/postgresql.conf
```

```
work_mem = 8MB
```

Это можно сделать и любым текстовым редактором.

Обновляем конфигурацию:

```
student$ sudo pg_ctlcluster 12 main reload
```

Проверяем:

```
student$ psql
```

```
=> SELECT current_setting('work_mem') AS work_mem;
```

```
work_mem
-----
8MB
(1 row)
```

2. Выполнение скриптов в psql

Запишем в файл ddl.sql команду на создание таблицы с ключевыми словами PostgreSQL (для этого можно использовать и любой текстовый редактор):

```
student$ cat >ddl.sql <<EOF
CREATE TABLE keywords (
    word text,
    category text,
    description text
);
EOF
```

Запишем команды для заполнения таблицы keywords в файл populate.sql:

```
student$ cat >populate.sql <<EOF
INSERT INTO keywords
SELECT * FROM pg_get_keywords();
EOF
```

Создаем базу данных и подключаемся к ней:

```
=> CREATE DATABASE tools_overview;
```

```
CREATE DATABASE
```

```
=> \c tools_overview
```

You are now connected to database "tools_overview" as user "student".

Выполняем скрипты и проверяем записи в таблице:

```
=> \i ddl.sql
```

```
CREATE TABLE
```

```
=> \i populate.sql
```

```
INSERT 0 442
```

```
=> SELECT * FROM keywords LIMIT 10;
```

word	category	description
abort	U	unreserved
absolute	U	unreserved
access	U	unreserved
action	U	unreserved
add	U	unreserved
admin	U	unreserved
after	U	unreserved
aggregate	U	unreserved
all	R	reserved
also	U	unreserved

(10 rows)

3. Просмотр журнала

Журнал можно открыть любым текстовым редактором. Каждая запись в журнале начинается с даты (настройки журнала после установки из пакета) и может состоять из нескольких строк. Записи за сегодняшний день будут в конце файла.

```
student$ tail /var/log/postgresql/postgresql-12-main.log
```

```
2023-07-26 11:09:31.168 MSK [30660] bob@access_overview CONTEXT:  SQL function "foo" statement 1
2023-07-26 11:09:31.168 MSK [30660] bob@access_overview STATEMENT:  SELECT foo();
2023-07-26 11:09:31.469 MSK [30660] bob@access_overview ERROR:  permission denied for function foo
2023-07-26 11:09:31.469 MSK [30660] bob@access_overview STATEMENT:  SELECT foo();
2023-07-26 11:09:31.646 MSK [30660] bob@access_overview ERROR:  permission denied for function baz
2023-07-26 11:09:31.646 MSK [30660] bob@access_overview STATEMENT:  SELECT baz();
2023-07-26 11:10:08.465 MSK [33595] student@student ERROR:  database "tools_overview" does not exist
2023-07-26 11:10:08.465 MSK [33595] student@student STATEMENT:  DROP DATABASE tools_overview;
2023-07-26 11:10:08.597 MSK [6602] LOG:  received SIGHUP, reloading configuration files
2023-07-26 11:10:08.598 MSK [6602] LOG:  parameter "work_mem" changed to "8MB"
```