Обзор базового инструментария
Установка и управление, psql



Темы

Варианты установки PostgreSQL

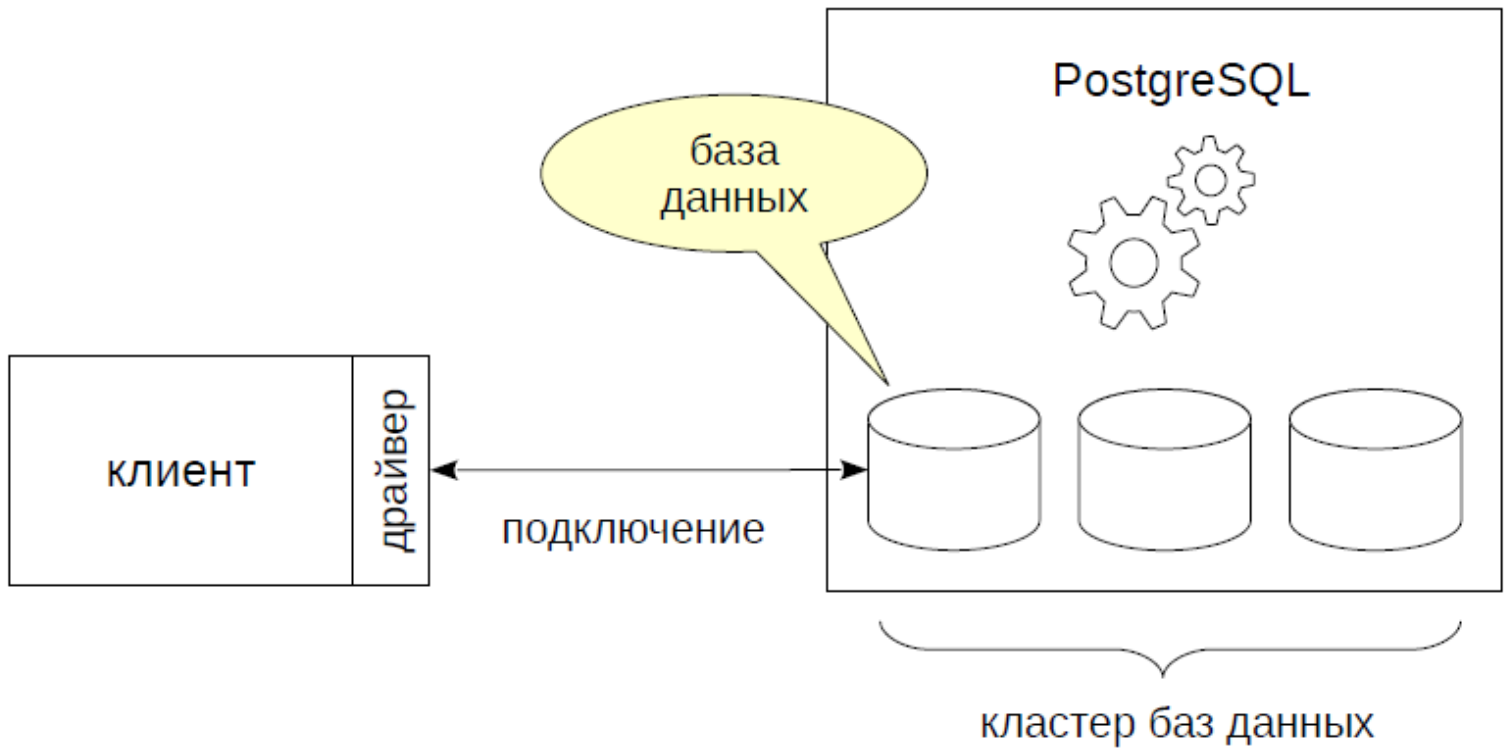
Управление сервером

Журнал сообщений сервера

Настройка параметров конфигурации

Использование psql

Кластер БД, сервер, клиент



Установка

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)

Управление сервером

Утилита для управления

pg_ctlcluster | pg_ctl

Основные задачи

- запуск сервера
- останов сервера
- обновление параметров конфигурации

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
```

Также можно получить информацию об установленных кластерах и их текущем состоянии при помощи команд:

```
$ pg_lsclusters
$ pg_ctlcluster status
```

Установка и управление

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

```
student$ ls -l /usr/lib/postgresql/16
```

```
total 8
drwxr-xr-x 2 root root 4096 июн 22 18:53 bin
drwxr-xr-x 4 root root 4096 июн 23 21:43 lib
```

Владелец ПО сервера — пользователь root.

Кластер баз данных автоматически инициализируется при установке из пакета и находится в каталоге /var/lib/postgresql/16/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 содержимое каталога:

```
student$ sudo ls -l /var/lib/postgresql/16/main
```

```
total 96
drwx------ 1 postgres postgres 4096 июн 24 04:29 base
drwx------ 1 postgres postgres 4096 июн 24 04:29 global
drwx------ 2 postgres postgres 4096 июн 23 21:43 pg_commit_ts
drwx------ 2 postgres postgres 4096 июн 23 21:43 pg_dynshmem
drwx------ 4 postgres postgres 4096 июн 23 21:46 pg_logical
drwx------ 4 postgres postgres 4096 июн 23 21:43 pg_multixact
drwx------ 2 postgres postgres 4096 июн 23 21:43 pg_notify
drwx------ 2 postgres postgres 4096 июн 23 21:43 pg_replslot
drwx------ 2 postgres postgres 4096 июн 23 21:43 pg_serial
drwx------ 2 postgres postgres 4096 июн 23 21:43 pg_snapshots
drwx------ 1 postgres postgres 4096 июн 24 04:29 pg_stat
drwx------ 2 postgres postgres 4096 июн 23 21:43 pg_stat_tmp
drwx------ 2 postgres postgres 4096 июн 23 21:43 pg_subtrans
drwx------ 2 postgres postgres 4096 июн 23 21:43 pg_tblspc
drwx------ 2 postgres postgres 4096 июн 23 21:43 pg_twophase
-rw------- 1 postgres postgres    3 июн 23 21:43 PG_VERSION
drwx------ 1 postgres postgres 4096 июн 23 21:43 pg_wal
drwx------ 2 postgres postgres 4096 июн 23 21:43 pg_xact
-rw------- 1 postgres postgres   88 июн 23 21:43 postgresql.auto.conf
-rw------- 1 postgres postgres  130 июн 24 04:29 postmaster.opts
-rw------- 1 postgres postgres  108 июн 24 04:29 postmaster.pid
```

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

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

Остановить сервер:

```
student$ sudo pg_ctlcluster 16 main stop
```

Запустить сервер:

```
student$ sudo pg_ctlcluster 16 main start
```

Перезапустить:

```
student$ sudo pg_ctlcluster 16 main restart
```

Обновить конфигурацию:

```
student$ sudo pg_ctlcluster 16 main reload
```

Получить информацию о сервере:

```
student$ sudo pg_ctlcluster 16 main status
```

```
pg_ctl: server is running (PID: 2882)
/usr/lib/postgresql/16/bin/postgres "-D" "/var/lib/postgresql/16/main" "-c"
"config_file=/etc/postgresql/16/main/postgresql.conf"
```

Список установленных экземпляров (можно без sudo):

```
student$ pg_lsclusters
```

```
Ver Cluster Port Status Owner    Data directory              Log file
16  main    5432 online postgres /var/lib/postgresql/16/main
/var/log/postgresql/postgresql-16-main.log
```

Журнал сообщений

В журнал записываются

- служебные сообщения сервера
- сообщения пользовательских сеансов
- сообщения приложений

Настройка журнала

- расположение
- формат записей
- какие события регистрировать

Журнал сообщений сервера

Журнал сообщений сервера находится здесь:

```
student$ ls -l /var/log/postgresql/postgresql-16-main.log
```

```
-rw-r----- 1 postgres adm 4871 июн 24 04:29 /var/log/postgresql/postgresql-16-main.log
```

Заглянем в конец журнала:

```
student$ tail -n 10 /var/log/postgresql/postgresql-16-main.log
```

```
2025-06-24 04:29:08.309 MSK [876] LOG:  background worker "logical replication launcher"
(PID 906) exited with exit code 1
2025-06-24 04:29:08.312 MSK [896] LOG:  shutting down
2025-06-24 04:29:08.336 MSK [896] LOG:  checkpoint starting: shutdown immediate
2025-06-24 04:29:08.562 MSK [896] LOG:  checkpoint complete: wrote 3 buffers (0.0%); 0
WAL file(s) added, 0 removed, 0 recycled; write=0.096 s, sync=0.029 s, total=0.250 s;
sync files=2, longest=0.015 s, average=0.015 s; distance=0 kB, estimate=0 kB;
lsn=0/1963B80, redo lsn=0/1963B80
2025-06-24 04:29:08.579 MSK [876] LOG:  database system is shut down
2025-06-24 04:29:09.257 MSK [2882] LOG:  starting PostgreSQL 16.9 (Ubuntu
16.9-1.pgdg24.04+1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu
13.3.0-6ubuntu2~24.04) 13.3.0, 64-bit
2025-06-24 04:29:09.257 MSK [2882] LOG:  listening on IPv4 address "127.0.0.1", port 5432
2025-06-24 04:29:09.284 MSK [2882] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
2025-06-24 04:29:09.359 MSK [2885] LOG:  database system was shut down at 2025-06-23
21:46:36 MSK
2025-06-24 04:29:09.473 MSK [2882] LOG:  database system is ready to accept connections
```

Параметры конфигурации

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`

`\dconfig`

Параметры конфигурации

Основной файл конфигурации postgresql.conf расположен в этом каталоге:

```
student$ ls -l /etc/postgresql/16/main
```

```
total 60
drwxr-xr-x 2 postgres postgres  4096 июн 23 21:43 conf.d
-rw-r--r-- 1 postgres postgres   315 июн 23 21:43 environment
-rw-r--r-- 1 postgres postgres   143 июн 23 21:43 pg_ctl.conf
-rw-r----- 1 postgres postgres  5743 июн 23 21:43 pg_hba.conf
-rw-r----- 1 postgres postgres  2640 июн 23 21:43 pg_ident.conf
-rw-r--r-- 1 postgres postgres 29960 июн 23 21:43 postgresql.conf
-rw-r--r-- 1 postgres postgres   317 июн 23 21:43 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 размещения 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 различные пути.

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

```
student$ grep '#work_mem' /etc/postgresql/16/main/postgresql.conf
```

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

Во-вторых, можно поместить определение параметра в файл с суффиксом .conf в каталоге /etc/postgresql/16/main/conf.d или в пользовательский файл конфигурации, местоположение которого следует задать в директиве include основного конфигурационного файла postgresql.conf.

В-третьих, можно изменить значение параметра 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 файл postgresql.auto.conf, который находится в каталоге PGDATA:

```
student$ sudo cat /var/lib/postgresql/16/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. Для этого можно воспользоваться pg_ctlcluster, либо использовать функцию SQL:

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

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



Убедимся, 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 прямо во время его выполнения. Например, если мы собираемся выполнить запрос, сортирующий большой объем данных, то для сеанса можно увеличить значение work_mem:

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

```
SET
```

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

Еще один способ проверить текущее значение — выполнить запрос к представлению:

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

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

Можно увидеть значение параметра и с помощью команды \dconfig:

```
=> \dconfig work_mem
```

```
List of configuration parameters
 Parameter | Value
-----------+-------
 work_mem  | 64MB
(1 row)
```

Клиент psql

Терминальный клиент для работы с PostgreSQL

Поставляется вместе с СУБД

Используется администраторами и разработчиками для интерактивной работы и выполнения скриптов

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 (host), 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 курса, 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 \c. Мы будем указывать необязательную часть имени команды в квадратных скобках: \c[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 PostgreSQL устанавливался с документацией, 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. Команды psql всегда однострочные — в отличие от команд SQL.

Внутри psql есть возможность получить список и краткое описание его собственных команд:

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

Форматирование вывода

Клиент psql умеет выводить результаты 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_in_place_tablespaces   | off        |
 allow_system_table_mods      | off        |
 application_name             | psql       |
 archive_cleanup_command      |            |
 archive_command              | (disabled) |
 archive_library              |            |
 archive_mode                 | off        |
(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_in_place_tablespaces|off|
allow_system_table_mods|off|
application_name|psql|
archive_cleanup_command||
archive_command|(disabled)|
archive_library||
archive_mode|off|
```

```
=> \a \t

Output format is aligned.
Tuples only is off.
```

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

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 \pset fieldsep ''

Output format is unaligned.
Tuples only is on.
Field separator is "".

=> \o dev1_tools.sql

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

На экран (в стандартный вывод) ничего не попало. Посмотрим в файле:

```
=> \! cat dev1_tools.sql

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

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_tools.sql
```

```
     tbl      | count
--------------+-------
 pg_statistic |   409
(1 row)

   tbl   | count
---------+-------
 pg_type |   613
(1 row)

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

Есть и другие способы выполнить команды, в том числе из файлов. После выполнения команд сеанс psql будет завершен:



- 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 подставить значение переменной, надо предварить ее имя 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 списка 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_proc        | 1245184
 pg_rewrite     |  745472
 pg_attribute   |  720896
 pg_description |  630784
 pg_statistic   |  294912
(5 rows)
```

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

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

```
\? variables
```

Выйти из psql можно с помощью команд \q[uit], quit, exit или нажав Ctrl+D:

```
=> \q
```

Итоги

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