

# Basic Toolkit Installation and Management, psql



## Copyright

© Postgres Professional, 2017, 2018, 2019.

Authors: Egor Rogov, Pavel Luzanov

## Use of course materials

Non-commercial use of course materials (presentations, demonstrations) is permitted without restrictions. Commercial use is possible only with the written permission of Postgres Professional. Changes to course materials are prohibited.

## Feedback

Send feedback, comments and suggestions to:

[edu@postgrespro.ru](mailto:edu@postgrespro.ru)

## Denial of responsibility

In no event shall Postgres Professional be liable to any party for direct, indirect, special, incidental, or consequential damages, including lost profit, arising out of the use of course materials. Postgres Professional disclaims any warranties on course materials. Course materials are provided on an “as is” basis and Postgres Professional has no obligations to provide maintenance, support, updates, enhancements, or modifications.

Server installation and management

Using psql

Installation options

Server management

Message log

Configuration parameters

Where to get PostgreSQL, how to start or stop the server, where to see the message log, how to check and set new values to the configuration parameters.

Minimal information on these topics is necessary for everyone working with PostgreSQL.

Details of the installation and management of the server are covered in courses for DBAs.

## Installation options

- prebuilt packages — preferred method
- build from source codes

## Extensions

- additional functionality
- installed separately
- 50 modules and programs are included in PostgreSQL distribution

The preferred installation option for PostgreSQL is to use prebuilt packages. In this case, a clear, supported and easily updated installation is obtained. Packages exist for most common operating systems.

Another installation option is to build the system from source codes. This option may be needed when building with non-standard parameters or when using a platform for which there is no ready-made package.

Ready packages and source codes are located here:

<http://www.postgresql.org/download/>

In the course we will use a virtual machine running Ubuntu and a PostgreSQL server installed from the package for this OS.

For PostgreSQL, there are a lot of extensions that add new functionality to the database on the fly, without changing the system kernel.


The distribution of PostgreSQL includes approximately 50 different extensions. Their description is included in the documentation:

<https://postgrespro.com/docs/postgresql/11/contrib>

<https://postgrespro.com/docs/postgresql/11/contrib-prog>

The list of available extensions and their installation status can be viewed in the `pg_available_extensions` view.

## Management utility

`pg_ctlcluster`  `pg_ctl`

## Main tasks

- server startup
- server shutdown
- reloading configuration parameters

The main server management operations include initialization (creation) of a database cluster, server start / stop, reloading of changed configuration parameters, and some others. The `pg_ctl` utility that comes with the server is designed to perform these actions.

In the package distribution for Ubuntu, you access `pg_ctl` utility not directly, but through a special `pg_ctlcluster` wrapper. Help on using `pg_ctlcluster` can be obtained with the command:

```
$ man pg_ctlcluster
```

More information about server management for database administrators:

<https://postgrespro.com/docs/postgresql/11/app-pg-ctl>

<https://postgrespro.com/docs/postgresql/11/runtime>

## What is logged

- server messages
- user session messages
- application messages

## Log setting

- destination for messages
- record format
- event types

Information about the DBMS operation is recorded in the server message log. This includes information about starting / stopping the server, various service messages, errors occurred etc.

Also, messages from backends can be included: what commands are executed and how much time they take, locks that appear, etc. This allows tracing user sessions.

Application developers can log their own messages.

PostgreSQL settings allow to determine exactly which messages and in what format should go into the server message log.

For example, output in csv format is convenient for automating log analysis.

<https://postgrespro.com/docs/postgresql/11/runtime-config-logging>

## Configuration parameters

main configuration file — `postgresql.conf`

ALTER SYSTEM — `postgresql.auto.conf`

## Making changes for the entire instance

change `postgresql.conf` or execute ALTER SYSTEM

reload configuration

check `pg_settings`

The PostgreSQL server has a large number of configuration parameters for various features: managing resource consumption, configuring processes and user sessions, managing server logs and more. During the course we will meet some of the parameters. As for now it is important to figure out how to check the current values and set new ones.

Configuration parameters are usually specified in configuration files. The main configuration file is `postgresql.conf`. In addition, there is the ALTER SYSTEM command, which allows to make changes to another configuration file `postgresql.auto.conf`. Parameters set via ALTER SYSTEM take precedence over parameters in `postgresql.conf`.

A typical scenario of changing the parameters for the entire system is as follows:

1. Make changes to the `postgresql.conf` file or execute ALTER SYSTEM SET *parameter* TO *new\_value*;
2. Reload the configuration (`pg_ctlcluster reload` or call the `pg_reload_conf` function). Some settings require a server restart.
3. Check that the changes have been applied. The `pg_settings` view contains the actual values of all parameters.

Options for setting and controlling parameters::

<https://postgrespro.com/docs/postgresql/11/config-setting>

`pg_settings` view description:

<https://postgrespro.com/docs/postgresql/11/view-pg-settings>

## Set in runtime

SET/RESET  
set\_config()

## View

SHOW  
current\_setting()

Most configuration parameters allow changing in user sessions right at run time.

You can change the parameters with the SET command or with the set\_config function.

To get the current values, you can use the SHOW command or the current\_setting function.



PostgreSQL command-line client

Comes with DBMS

Used by administrators and developers for interactive work and script execution

There are various third-party tools and IDEs designed for PostgreSQL, the consideration of which is out of scope of the course.

In the course we will use the command-line `psql` client:

1. `psql` is the only client that comes with the DBMS.
2. Skills of working with `psql` will be useful for developers and database administrators, regardless of which tool they will continue to work with.

For interactive work, `psql` has built-in support for `readline`, paging query results (`more`, `less`) programs. Features of `psql` allow you to interact with the OS, view the contents of the system catalog, create scripts to automate repetitive tasks.

`psql` full description:

<https://postgrespro.com/docs/postgresql/11/app-psql>

## Start psql

```
$ psql -d database -U role -h host -p port
```

## New connection in psql

```
=> \c[onnect] database role host port
```

## Information about the current connection

```
=> \conninfo
```

When running `psql`, you need to specify the connection parameters.

Required connection parameters include: database name, user (role) name, server name, port number. If these parameters are not specified, `psql` will try to connect using default values:

- *database* — same as database username,
- *role* — same as OS username,
- *host* — local connection,
- *port* — usually 5432.

Settings made for this course purposes allow you to connect to PostgreSQL without specifying any parameters at all.

If you want to make a new connection without leaving `psql`, run the `\connect` command.

The `\conninfo` command displays information about the current connection.

Additional information on connection settings:

<https://postgrespro.com/docs/postgresql/11/libpq-envars>

<https://postgrespro.com/docs/postgresql/11/libpq-pgservice>

<https://postgrespro.com/docs/postgresql/11/libpq-pgpass>

## Shell command line

```
$ psql --help  
$ man psql
```

## In psql

=> \?	psql commands list
=> \? variables	psql variables
=> \h[elp]	SQL commands list
=> \h <i>command</i>	SQL command syntax
=> \q	exit

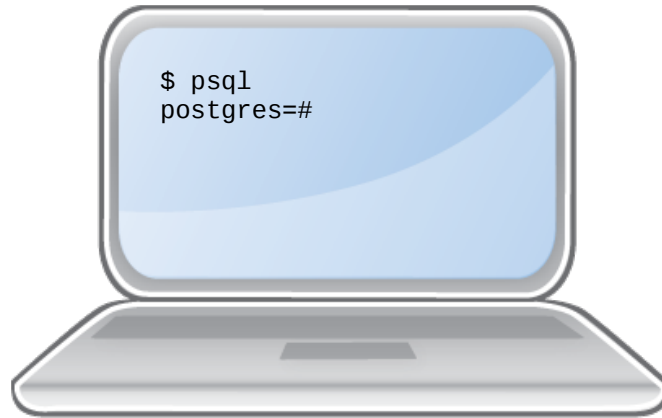
Information on `psql` can be obtained not only in the documentation, but also directly in the system.

`psql --help` gives general help. In case the documentation has been installed in the system, the manual can be obtained with the `man psql` command.

`psql` can execute SQL commands and its own commands.

Inside `psql` you can get a list and a brief description of the `psql` commands. All `psql` commands begin with a backslash.

The `\help` command lists the SQL commands that the server supports, as well as the syntax of the specific SQL command.



Preferred PostgreSQL installation method is to use prebuilt packages

Package distributions take into account OS features you need to know

- how to start and stop server
- configuration files locations
- message log location

psql is the standard command-line client for PostgreSQL

1. Set the `work_mem` parameter in `postgresql.conf` to 8 MB.
2. Reload the configuration and verify that changes took effect.
3. Write some SQL commands to a file.  
For example, create a table and insert some rows to it.
4. Start `psql`, run the SQL script you created, and check the results.
5. Find in the message server log lines related to today's work.

To perform practical tasks it is necessary to log into the operating system under *student* user (password *student*).

To run `psql` in a terminal window, just type `psql` without parameters. The default settings will be used for the connection:

```
student@student:~$ psql
psql (11.1 (Ubuntu 11.1-1.pgdg16.04+1))
Type "help" for help.
```

```
student=# \conninfo
```

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

```
student=#
```