

Basic tools

# Server installation and management



## Copyright

© Postgres Professional, 2015–2022

Authors: Egor Rogov, Pavel Luzanov, Ilya Bashtanov

Translated by Alexander Meleshko

## Use of course materials

Non-commercial use of course materials (presentations, demonstrations) is allowed without restrictions. Commercial use is possible only with the written permission of Postgres Professional. It is prohibited to make changes to the course materials.

## Feedback

Please send your feedback, comments and suggestions to:

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

## Disclaimer

Postgres Professional assumes no responsibility for any damages and losses, including loss of income, caused by direct or indirect, intentional or accidental use of course materials. Postgres Professional does not provide any guarantees for the course materials. Course materials are provided "as is" and Postgres Professional is not obligated to provide maintenance, support, updates, extensions and changes.

Basic concepts

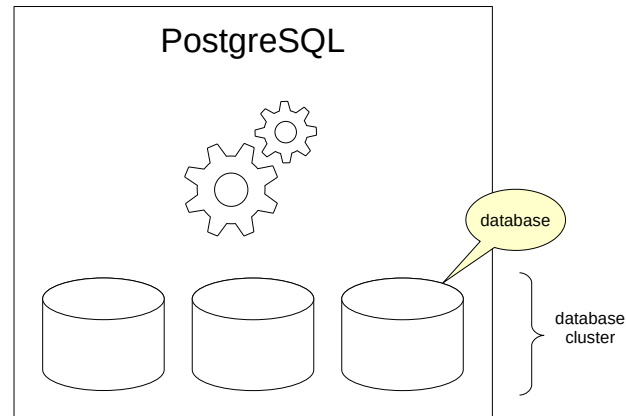
Installation from source code

Server management

Package installation

Server management in Ubuntu

Cloud solutions



Before talking about installation, let's review some basic concepts.

PostgreSQL is a program that belongs to the class of *database management systems*.

When this program is running, we call it a PostgreSQL *server* or a *server instance*. So far, the server seems to be a “black box” for us, but gradually we will get acquainted with how it works.

The data managed by PostgreSQL is stored in *databases*. One instance of PostgreSQL simultaneously manages several databases. This set of databases is called a *database cluster*. We will talk more about databases in the “Data organization. Databases and schemas” module of the course.

So, a database cluster is data in files. A server or a server instance is a program that manages a database cluster.

## Installation from source code

- stable server version

- you can use non-standard parameters or build the server for a non-standard architecture

## Installation from git

- current server version

- used primarily by kernel developers; requires a broader set of tools

You should at least have an idea of how to install the program from source code, so that you could build PostgreSQL with non-standard parameters or on a non-standard architecture, if necessary.

You can find the version 13.6 source code in gzip and bzip2 at <https://www.postgresql.org/ftp/source/v13.6/> as well as on the VM in the postgres user home catalog.

You can also get the source code straight from the project's git repository: [git://git.postgresql.org/git/postgresql.git](https://git.postgresql.org/git/postgresql.git) (there are other mirrors, including github).

This way, you can build not just the stable version, but also a version at any specific commit, including the most recent one.

When installing from the git repository, a wider set of tools is required. For example, the lexer and parser are made using Flex and Bison, and git stores the source codes for these tools. They generate C files when built, which are then compiled. The source code archives include the pre-built C files already.

# Required software



## Hard requirements

tar, gzip/bzip2, GNU make, C compiler (C89)

## Used, but can be disabled

GNU Readline library, zlib library

## Extras

Perl, Python, and Tcl programming languages  
for using PL/Perl, PL/Python, PL/Tcl

Kerberos, OpenSSL, OpenLDAP, PAM — authentication and encryption  
tools

ICU library for cross-platform UNICODE support

## Separate tools when building from the git repository

5

A number of programs and utilities are required when building from source.

The readline library allows you to edit the command line, use the command history and auto-completion. In a server installation, it may not be necessary if you are never going to run `psql` via the console.

The zlib library is used to compress `pg_dump` archives.

## Unpacking the archive

The source code package is located in the student directory. Let's unpack and open it:

```
student$ tar xzf /home/student/postgresql-13.6.tar.gz
```

```
student$ ls -ld /home/student/postgresql-13.6
```

```
ls: cannot access '/home/student/postgresql-13.6': No such file or directory
```

```
student$ cd /home/student/postgresql-13.6
```

---

## Configuration

First, remove all previous configuration settings:

```
student$ make distclean
```

The configure command accepts a number of parameters, such as:

- `--prefix` — installation catalog, default is `/usr/local/pgsql`,
- `--enable-debug` — enables debug messages.

See the documentation for the full list of parameters.

The command also makes use of environment variables. For example, `CC` and `CFLAGS` configure the C compiler.

---

In production, you should install PostgreSQL into a catalog that is write-protected for everyone but the system admin. Here, we will install the system into the student catalog instead. We're also using port 5555 instead of the default 5432:

```
student$ ./configure --prefix=/home/student/pgsql13 --with-pgport=5555
```

The output of this command is extremely verbose, so we are skipping it entirely.

---

## PostgreSQL compilation and installation

Options:

- `make` — compile only the server,
- `make world` — compile the server, all extensions and the documentation.

Let's compile the server. Depending on your computer performance, this may take minutes or even tens of minutes.

```
student$ make
```

Now, install the server.

```
student$ make install
```

To compile the extensions, repeat the last two commands from within the contrib catalog.

```
student$ cd /home/student/postgresql-13.6/contrib
```

```
student$ make
```

```
student$ make install
```

If you are installing the server together with the extensions and docs (`make world`), you can install everything with just the `make install-world` command.



## Tool

initdb

## Notes

a new database cluster cannot belong to the OS superuser

PGDATA is a convenient variable that represents the cluster directory

configuration files are created in the PGDATA directory

⚠ enabling checksum calculation for data pages is a good idea

After installing the server, you need to create a database cluster. The `initdb` tool does that for you.

For security reasons, the directory in which the cluster is initialized cannot belong to the OS superuser. Usually, the cluster ownership is assigned to the `postgres` user.

The cluster owner can define the `PGDATA` environment variable pointing to the cluster directory. This variable is used by some server tools when they need to find out the location of the cluster. Such tools include `initdb`, as well as `pg_ctl`, the main server management tool, which we will discuss in a bit.

During cluster initialization, `initdb` creates configuration files in the `PGDATA` directory. We will talk more about configuration files in another topic.

`initdb` has many keys that affect its operation. One important key is `-k` or `--data-checksums`. It enables or disables calculating checksums for data pages. The checksum check is performed when accessing any data page in the cluster. This slightly reduces performance, but allows you to quickly detect data corruption.

For more details, see <https://postgrespro.com/docs/postgresql/13/app-initdb>

## Creating a cluster

Now, let's create a data catalog. In production, it's a good practice to have a separate OS user as the owner of the data catalog. Here, we will just use the student user and the /home/student/pgsql13 catalog.

```
student$ mkdir /home/student/pgsql13/data
```

This catalog is often referred to as PGDATA. PGDATA is the environment variable pointing to it, and it's a convenient shortcut.

```
student$ export PGDATA=/home/student/pgsql13/data
```

All server tools and utilities are located in the bin catalog. You should always add it into the PATH variable:

```
student$ export PATH=/home/student/pgsql13/bin:$PATH
```

initdb is a tool used to initialize a database cluster.

- The -U key lets you define the PostgreSQL superuser. If omitted, the OS username is used ("student", in this case). Usually, the PostgreSQL superuser is called postgres, so let's stick with that.
- The -k key enables page checksum calculation, which can help spot data corruption.
- If PGDATA isn't set, the -D key should be used to define the data catalog. We don't need to do it here.

```
student$ initdb -U postgres -k -D /home/student/pgsql13/data
```





## Tool

pg\_ctl

## Main operations

starting, stopping and checking the status of the server

updating configuration parameters

switching to a replica

The main server management operations include starting and stopping the server, getting the current status of the server, updating the configuration, and some others.

pg\_ctl, a stock PostgreSQL tool, is designed to perform these actions.

pg\_ctl should be run as the owner of the database cluster.

For more details about server management for database administrators, see:

<https://postgrespro.ru/docs/postgresql/13/app-pg-ctl>

<https://postgrespro.ru/docs/postgresql/13/runtime>

## Server management

We can start the server now.

- The `-l` key defines the server log file location.
- The `-D` key is unnecessary, as we have defined the `PGDATA` variable.

```
student$ pg_ctl start -l /home/student/logfile
```

Let's verify that the server is running: use the `psql` tool to connect to the server and return current time:

```
student$ psql -U postgres -p 5555 -c 'SELECT now();'

      now
-----
2024-03-07 13:48:12.698426+03
(1 row)
```

---

The following command stops the server:

```
student$ pg_ctl stop -m fast
```

The `-m` key selects the shutdown mode:

- `fast` — terminates all sessions and writes changes to disk before shutdown,
- `smart` — waits for all sessions to finish, then writes changes to disk before shutdown,
- `immediate` — terminates all sessions and shuts down, will start in recovery mode.

The `fast` mode is used by default.

# Installation from a package



Package installation is usually preferable

Linux (**ubuntu**®, Debian, Red Hat, CentOS and others)

comes pre-installed with the OS

repository (yum, apt) or RPM/DEB packages

FreeBSD, OpenBSD

packages from the Ports and Packages Collection

macOS

Windows

11

The preferred option is to use ready-made packages, since in this case a clear, supported and easily updated installation is obtained.

Packages exist for most popular systems. Each of them may have its own features that you should get to know before installing.

Package repositories of some systems already include PostgreSQL, but usually not the latest version. The latest version is always available in the PostgreSQL Global Development Group (PGDG) repository:

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

(Postgres Pro versions: <https://postgrespro.com/products/download>)


We will be working with the Ubuntu PostgreSQL package.



# Creating a cluster



## Tool

pg\_createcluster  initdb

## Notes

initialization is performed when installing the package, creates a cluster named “main”



checksum calculation is not enabled by default

defines the location of server configuration and log files

sets up automatic server startup at OS startup

12

pg\_createcluster is a wrapper for the initdb tool designed to initialize the cluster in Ubuntu.

The help for the pg\_createcluster command can be obtained using man:

```
$ man pg_createcluster
```

pg\_createcluster executes automatically when the package is installed and creates a database cluster named “main”.

Note that cluster initialization is done with checksum calculation for data pages disabled.

Executables, configuration files and the server log are stored in accordance with the Ubuntu practices.

In addition, the installation sets up the PostgreSQL server to start and stop when Ubuntu starts and stops.

To delete a cluster, the pg\_dropcluster tool is used.

Both pg\_createcluster and pg\_dropcluster are specific to Ubuntu.

In other systems, you need to explicitly initialize the cluster using initdb and use the appropriate operating system tools.

## Creating a cluster in Ubuntu

On the provided VM, PostgreSQL is installed from the distribution package:

```
student$ sudo apt install -y postgresql-13
```

PostgreSQL catalog:

```
student$ sudo ls -l /usr/lib/postgresql/13
```

```
total 8
drwxr-xr-x 2 root root 4096 Jun  7  2023 bin
drwxr-xr-x 4 root root 4096 Jun  7  2023 lib
```

The owner of the server is the root user.

pg\_config shows the parameters the server was installed with:

```
student$ sudo /usr/lib/postgresql/13/bin/pg_config --configure
```

```
'--build=x86_64-linux-gnu' '--prefix=/usr' '--includedir=${prefix}/include' '--mandir=${prefix}/share/man' '--infodir=${prefix}/share/info' '--sysconfdir=/etc' '--localstatedir=/var'
```

A database cluster called main is already initialized and is located in /var/lib/postgresql/13/main.

The owner of the catalog is postgres. Here's what the catalog contains:

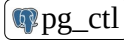
```
student$ sudo ls -l /var/lib/postgresql/13/main
```

```
total 84
drwx----- 6 postgres postgres 4096 Mar  7 13:48 base
drwx----- 2 postgres postgres 4096 Mar  7 13:48 global
drwx----- 2 postgres postgres 4096 Mar  7 13:48 pg_commit_ts
drwx----- 2 postgres postgres 4096 Mar  7 13:48 pg_dynshmem
drwx----- 4 postgres postgres 4096 Mar  7 13:48 pg_logical
drwx----- 4 postgres postgres 4096 Mar  7 13:48 pg_multixact
drwx----- 2 postgres postgres 4096 Mar  7 13:48 pg_notify
drwx----- 2 postgres postgres 4096 Mar  7 13:48 pg_replslot
drwx----- 2 postgres postgres 4096 Mar  7 13:48 pg_serial
drwx----- 2 postgres postgres 4096 Mar  7 13:48 pg_snapshots
drwx----- 2 postgres postgres 4096 Mar  7 13:48 pg_stat
drwx----- 2 postgres postgres 4096 Mar  7 13:48 pg_stat_tmp
drwx----- 2 postgres postgres 4096 Mar  7 13:48 pg_subtrans
drwx----- 2 postgres postgres 4096 Mar  7 13:48 pg_tblspc
drwx----- 2 postgres postgres 4096 Mar  7 13:48 pg_twophase
-rw----- 1 postgres postgres   3 Mar  7 13:48 PG_VERSION
drwx----- 3 postgres postgres 4096 Mar  7 13:48 pg_wal
drwx----- 2 postgres postgres 4096 Mar  7 13:48 pg_xact
-rw----- 1 postgres postgres  88 Mar  7 13:48 postgresql.auto.conf
-rw----- 1 postgres postgres 130 Mar  7 13:48 postmaster.opts
-rw----- 1 postgres postgres 110 Mar  7 13:48 postmaster.pid
```



## Tool

pg\_ctlcluster



## Main operations

- starting, stopping and checking the status of the server
- reloading configuration parameters
- switching to a replica

In the Ubuntu PostgreSQL package, pg\_ctl is not run directly, but through a special wrapper pg\_ctlcluster. The help for pg\_ctlcluster can be obtained using man:

```
$ man pg_ctlcluster
```

In other systems, pg\_ctl may be used directly.

## Server management in Ubuntu

When installing from package, PostgreSQL is added to the list of applications to launch on OS startup, so there's no need to launch PostgreSQL manually.

Use the following commands to control the server, either as the OS user postgres or with sudo:

Stop:

```
student$ sudo pg_ctlcluster 13 main stop
```

Start:

```
student$ sudo pg_ctlcluster 13 main start
```

Restart:

```
student$ sudo pg_ctlcluster 13 main restart
```

Get server status:

```
student$ sudo pg_ctlcluster 13 main status
```

```
pg_ctl: server is running (PID: 122560)
/usr/lib/postgresql/13/bin/postgres "-D" "/var/lib/postgresql/13/main" "-c" "config_file=/etc/postgresql/13/main/postgresql.conf"
```

Reload configuration:

```
student$ sudo pg_ctlcluster 13 main reload
```

The pg\_ctlcluster tool can manage multiple servers of different versions. It has the following parameters:

- 13 — server version number,
- main — server name.

---

For a package installation, the server log file is located at:

```
student$ ls -l /var/log/postgresql/postgresql-13-main.log
```

```
-rw-r----- 1 postgres adm 2819 Mar  7 13:48 /var/log/postgresql/postgresql-13-main.log
```

Let's look at the last messages:

```
student$ tail -n 10 /var/log/postgresql/postgresql-13-main.log
```

```
2024-03-07 13:48:15.854 MSK [122514] LOG:  aborting any active transactions
2024-03-07 13:48:15.863 MSK [122514] LOG:  background worker "logical replication launcher" (PID 122521) exited with exit code 1
2024-03-07 13:48:15.864 MSK [122516] LOG:  shutting down
2024-03-07 13:48:15.893 MSK [122514] LOG:  database system is shut down
2024-03-07 13:48:16.032 MSK [122560] LOG:  starting PostgreSQL 13.11 (Ubuntu 13.11-1.pgdg22.04+1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 11.3.0-1ubuntu1-22.04) 11.3.0, 64-bit
2024-03-07 13:48:16.032 MSK [122560] LOG:  listening on IPv4 address "127.0.0.1", port 5432
2024-03-07 13:48:16.035 MSK [122560] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2024-03-07 13:48:16.042 MSK [122561] LOG:  database system was shut down at 2024-03-07 13:48:15 MSK
2024-03-07 13:48:16.049 MSK [122560] LOG:  database system is ready to accept connections
2024-03-07 13:48:18.427 MSK [122560] LOG:  received SIGHUP, reloading configuration files
```

## PostgreSQL in a virtual environment

- virtualization overhead
- the administrator has full access to the instance

## PostgreSQL as a Service

- offered by many cloud providers
- the provider takes over the administration routine
- limited management, backup and monitoring tools

Virtualization solutions can be used to run databases, including PostgreSQL. The trade off for the convenience of this approach is considerable overhead costs. For high-load systems, any additional intermediate layers between the DBMS and the hardware are undesirable.

In addition, many leading cloud providers offer PostgreSQL as a service (Database as a Service, managed database).

In this case, the provider takes over most of the administration tasks. This limits your control over the instance. Moreover, management capabilities for tasks such as performance monitoring or backup and recovery are limited by the tools offered by the service provider.

We will not consider the features of individual cloud solutions within the course. You can study the documentation offered by different service providers to learn more.



# Takeaways



Two installation options — package or source code

Cloud solutions available

The server is run by a dedicated OS user

Initialize a database cluster before using the server

OS-specific commands for server management

Enabling checksum calculation for a cluster.

1. Stop the server.
2. Check whether checksums are being calculated for the cluster.
3. Enable checksum calculation.
4. Start the server.

2, 3. Use the `pg_checksums` tool. To run it, enter the full path:

`/usr/lib/postgresql/13/bin/pg_checksums`

<https://postgrespro.com/docs/postgresql/13/app-pgchecksums>

## 1. Stopping the server

```
student$ sudo pg_ctlcluster 13 main stop
```

## 2. Verification

To see if page checksum calculation is enabled, run the `pg_checksums` utility with the `--check` key:

```
student$ sudo /usr/lib/postgresql/13/bin/pg_checksums --check -D /var/lib/postgresql/13/main
pg_checksums: error: data checksums are not enabled in cluster
```

## 3. Enabling checksum calculation

Run `pg_checksums` with the `--enable` key:

```
student$ sudo /usr/lib/postgresql/13/bin/pg_checksums --enable -D /var/lib/postgresql/13/main
```

```
Checksum operation completed
Files scanned: 1494
Blocks scanned: 4945
pg_checksums: syncing data directory
pg_checksums: updating control file
Checksums enabled in cluster
```

Checksum calculation is now enabled.

## 4. Start the server

```
student$ sudo pg_ctlcluster 13 main start
```

1. Install PostgreSQL from source code as shown in the demo.
2. Create a database cluster, start the server.
3. Make sure the server is running.
4. Stop the server.

## **1. Compiling PostgreSQL from source code**

All necessary commands were shown during the demo, but were not executed. Repeat the process by yourself within the provided virtual environment.

Note that compiling from source takes a while (several to tens of minutes, depending on your machine specifications).