

## Basic tools

# Server configuration



### 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 company specifically disclaims any warranties on course materials. Course materials are provided “as is,” and Postgres Professional company has no obligations to provide maintenance, support, updates, enhancements, or modifications.

Configuration parameters

Configuration files

Parameter management at the instance and session levels

## Purpose

managing DBMS operation and behavior

## Setting parameters

for an instance via configuration files

for a separate database or user

for the current session

There are multiple parameters in PostgreSQL that control the DBMS behavior. These parameters affect resource management, backend process operations, and much more.

For example, the *max\_connections* parameter limits the number of concurrent connections to the server.

The full list of configuration parameters and their descriptions is available in the documentation: <https://postgrespro.com/docs/postgresql/13/runtime-config.html>

In this topic, we will not cover any specific configuration parameters, but rather discuss how to set their values.

Configuration parameters are generally managed in configuration files. The parameter values defined in the configuration files affect the whole DBMS instance, unless explicitly configured otherwise.


Some parameters can be set for specific databases or for a specific user's sessions. Such parameters will overrule those declared in configuration files. These types of parameters will be discussed in further chapters of the course.

Lastly, many parameters can be changed at the session level, during server operation.


## Main configuration file


loaded when the server starts

 located in the data directory (PGDATA) by default

 /etc/postgresql/13/main

## After any changes to the parameters, the file has to be reloaded

 \$ pg\_ctl reload

 \$ pg\_ctlcluster 13 main reload

=> SELECT pg\_reload\_conf();

changes to some parameters require a server restart to apply

The main configuration file is postgresql.conf.

The file's location is defined during initial PostgreSQL compilation. By default, the file is located in the data catalog (PGDATA), but package distributions usually place it somewhere else, depending on the OS-specific conventions.

This is a well-documented plaintext file that stores parameters in a key-value format.

If the same parameter is defined in the file multiple times, only the most recently read value will be used.

For any changes to parameters to apply, the file must be reloaded. Some parameters require a server restart to apply.

## The postgresql.conf file and the pg\_file\_settings view

Take a look at a part of a config file.

```
=> SHOW config_file;
```

```
      config_file
-----
/etc/postgresql/13/main/postgresql.conf
(1 row)
```

```
=> SELECT pg_read_file('/etc/postgresql/13/main/postgresql.conf', 1516, 860)
\g (tuples_only=on format=unaligned)
```

```
#-----
# FILE LOCATIONS
#-----

# The default values of these variables are driven from the -D command-line
# option or PGDATA environment variable, represented here as ConfigDir.

data_directory = '/var/lib/postgresql/13/main'      # use data in another directory
                                                    # (change requires restart)
hba_file = '/etc/postgresql/13/main/pg_hba.conf'    # host-based authentication file
                                                    # (change requires restart)
ident_file = '/etc/postgresql/13/main/pg_ident.conf' # ident configuration file
                                                    # (change requires restart)

# If external_pid_file is not explicitly set, no extra PID file is written.
external_pid_file = '/var/run/postgresql/13-main.pid' # write an extra PID file
                                                    # (change requires restart)
```

To see all non-comment lines of a config file, use the pg\_file\_settings view:

```
=> SELECT sourceline, name, setting, applied
FROM pg_file_settings
WHERE sourcefile LIKE '/etc/postgresql/13/main/postgresql.conf';
```

sourceline	name	setting	applied
42	data_directory	/var/lib/postgresql/13/main	t
44	hba_file	/etc/postgresql/13/main/pg_hba.conf	t
46	ident_file	/etc/postgresql/13/main/pg_ident.conf	t
50	external_pid_file	/var/run/postgresql/13-main.pid	t
64	port	5432	t
65	max_connections	100	t
67	unix_socket_directories	/var/run/postgresql	t
101	ssl	on	t
103	ssl_cert_file	/etc/ssl/certs/ssl-cert-snakeoil.pem	t
105	ssl_key_file	/etc/ssl/private/ssl-cert-snakeoil.key	t
122	shared_buffers	128MB	t
143	dynamic_shared_memory_type	posix	t
229	max_wal_size	1GB	t
230	min_wal_size	80MB	t
530	log_line_prefix	%m [%p] %q%u@%d	t
564	log_timezone	Europe/Moscow	t
570	cluster_name	13/main	t
586	stats_temp_directory	/var/run/postgresql/13-main.pg_stat_tmp	t
679	datestyle	iso, mdy	t
681	timezone	Europe/Moscow	t
695	lc_messages	en_US.UTF-8	t
697	lc_monetary	en_US.UTF-8	t
698	lc_numeric	en_US.UTF-8	t
699	lc_time	en_US.UTF-8	t
702	default_text_search_config	pg_catalog.english	t

(25 rows)

The column name “applied” is misleading. In case any changes are made to the file, the table value tells you if the new configuration can be applied without restarting the server. The pg\_file\_settings view shows only the contents of the file, actual configuration parameter values may differ.

## The pg\_settings view

Current configuration parameter values are shown in the `pg_settings` view. For example, here's its output regarding the `work_mem` parameter:

```
=> SELECT name, setting, unit,
       boot_val, reset_val,
       source, sourcefile, sourceline,
       pending_restart, context
FROM pg_settings
WHERE name = 'work_mem'\gx

-[ RECORD 1 ]---+-----
name           | work_mem
setting        | 4096
unit           | kB
boot_val       | 4096
reset_val      | 4096
source         | default
sourcefile     |
sourceline     |
pending_restart | f
context        | user
```

The `work_mem` parameter defines how much memory is allocated for operations like sorting or hash join. The default value can be insufficient for queries working with massive amounts of data. To learn more about the `work_mem` parameter, check out our Query performance tuning course (QPT).

---

Below are the main columns of the `pg_settings` view:

- `name`, `setting`, `unit` — parameter name and value,
- `boot_val` — default value,
- `reset_val` — value to be set by the `RESET` command,
- `source` — the source of the current value,
- `pending_restart` — the value is changed in the configuration file, pending server restart.

---

The `context` column shows what action has to be taken for the parameter change to take effect. Possible values include:

- `internal` — cannot be changed, set during installation,
- `postmaster` — server restart required,
- `sighup` — reload of configuration files required,
- `superuser` — can be changed by superuser for their session,
- `user` — can be changed by any user for their session.

---

## postgresql.conf lines are applied in a certain order

If the same parameter is defined multiple times throughout the file, the latest value applies.

Add two lines to the end of `postgresql.conf` which both modify the `work_mem` parameter value:

```
student$ echo work_mem=12MB | sudo tee -a /etc/postgresql/13/main/postgresql.conf
```

```
work_mem=12MB
```

```
student$ echo work_mem=8MB | sudo tee -a /etc/postgresql/13/main/postgresql.conf
```

```
work_mem=8MB
```

```
=> SELECT sourceline, name, setting, applied
FROM pg_file_settings
WHERE name = 'work_mem';
```

sourceline	name	setting	applied
783	work_mem	12MB	f
784	work_mem	8MB	t

(2 rows)

The `applied = f` value for the 12MB line tells us that the configuration change cannot be applied.

---

The `context` field of the `work_mem` parameter says “user”. This means that the parameter can be changed by the user during their session. We will discuss how to do it in a bit.

To change the value for all sessions, you just need to reload the configuration file:

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

```
pg_reload_conf
```

```
-----
```

```
t
```

```
(1 row)
```

Verify that `work_mem` now has taken the value from the last line of the configuration file:

```
=> SELECT name, setting, unit,  
       boot_val, reset_val,  
       source, sourcefile, sourceline,  
       pending_restart, context
```

```
FROM pg_settings
```

```
WHERE name = 'work_mem'\gx
```

```
-[ RECORD 1 ]-----+-----
```

name		work_mem
setting		8192
unit		kB
boot_val		4096
reset_val		8192
source		configuration file
sourcefile		/etc/postgresql/13/main/postgresql.conf
sourceline		784
pending_restart		f
context		user

## Configuration file managed by SQL commands

ALTER SYSTEM	adds or changes a line
SET <i>parameter</i> TO <i>value</i> ;	
ALTER SYSTEM RESET <i>parameter</i> ;	removes a line
ALTER SYSTEM RESET ALL;	deletes all lines
read after postgresql.conf	

## Location

always in the data directory (PGDATA)

## Actions needed to apply

same as for postgresql.conf

The file postgresql.auto.conf is always loaded last. This file is always located in the data catalog (PGDATA).

It should never be modified manually, but only with the ALTER SYSTEM command. ALTER SYSTEM is an SQL interface for managing configuration parameters.

For any changes made with ALTER SYSTEM to take place, the server must reload the configuration files, as it does with postgresql.conf.

The contents of both files (postgresql.conf and postgresql.auto.conf) can be checked using the pg\_file\_settings view, and the current parameter values are shown in the pg\_settings view.

More about the ALTER SYSTEM command:

<https://postgrespro.com/docs/postgresql/13/runtime-config-logging.html#RUNTIME-CONFIG-LOGGING-WHERE>



## The ALTER SYSTEM command and the postgresql.auto.conf file

Let's set a new value for the work\_mem parameter:

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

ERROR: invalid value for parameter "work\_mem": "16mb"

HINT: Valid units for this parameter are "B", "kB", "MB", "GB", and "TB".

What happened?

ALTER SYSTEM checks if the provided value is valid.

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

ALTER SYSTEM

Now it works!

The command saves the new value 16MB to a file named postgresql.auto.conf:

```
=> SELECT pg_read_file('postgresql.auto.conf')
\g (tuples_only=on format=unaligned)
```

# Do not edit this file manually!

# It will be overwritten by the ALTER SYSTEM command.

work\_mem = '16MB'

However, the value has not been applied yet:

```
=> SHOW work_mem;
```

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

For the new work\_mem setting to apply, reload the configuration file first:

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

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

```
=> SELECT name, setting, unit,
       boot_val, reset_val,
       source, sourcefile, sourceline,
       pending_restart, context
FROM pg_settings
WHERE name = 'work_mem'\gx
```

```
-[ RECORD 1 ]---+-----
name          | work_mem
setting       | 16384
unit          | kB
boot_val      | 4096
reset_val     | 16384
source        | configuration file
sourcefile    | /var/lib/postgresql/13/main/postgresql.auto.conf
sourceline    | 3
pending_restart | f
context       | user
```

Lines are removed from postgresql.auto.conf with the ALTER SYSTEM RESET command:

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

ALTER SYSTEM

```
=> SELECT pg_read_file('postgresql.auto.conf')
\g (tuples_only=on format=unaligned)
```

```
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
```

-----

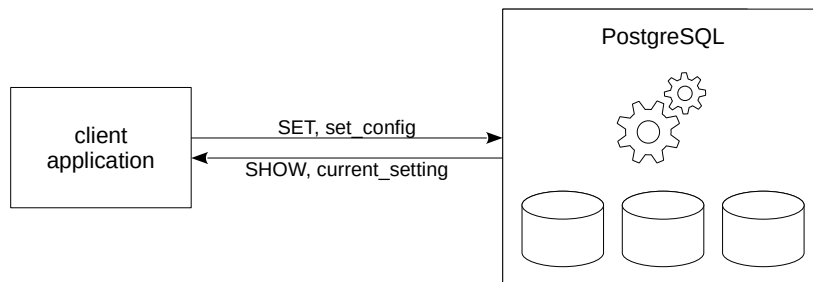
Reload the configuration file again. The old value from postgresql.conf is now applied:

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

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

```
=> SELECT name, setting, unit,
        boot_val, reset_val,
        source, sourcefile, sourceline,
        pending_restart, context
FROM pg_settings
WHERE name = 'work_mem'\gx
```

```
-[ RECORD 1 ]---+-----
name          | work_mem
setting       | 8192
unit          | kB
boot_val      | 4096
reset_val     | 8192
source        | configuration file
sourcefile    | /etc/postgresql/13/main/postgresql.conf
sourceline    | 784
pending_restart | f
context       | user
```



set until the end of the session or transaction  
the act of setting parameters is transactional  
custom parameters are allowed

Parameter values can be changed directly during the session with the SET command or the set\_config function. The SHOW command or the current\_setting function display the current parameter value.

By setting a new value, you can specify its effective range: until the end of the session (by default) or until the end of the transaction (SET LOCAL).

In any case, setting parameters is transactional: if the current transaction is rolled back, any parameters modified within it will return to the state they were in at the start of the transaction.

In addition to the PostgreSQL system parameters, the same commands and functions can be used to create and get the values of custom parameters.

## Configuring parameters for the duration of a session

The SET command is used to set parameter values for the duration of the current session:

```
=> SET work_mem TO '24MB';
```

SET

The set\_config function also works:

```
=> SELECT set_config('work_mem', '32MB', false);
```

```
set_config
-----
32MB
(1 row)
```

The third parameter of the function defines if the parameter should be used for the duration of the current transaction (true) or until the end of the current session (false). This may be important when using a connection pool, when transactions by multiple users may be performed within the same session.

---

## Checking parameter values during a session

There are multiple ways to check the current parameter value:

```
=> SHOW work_mem;
```

```
work_mem
-----
32MB
(1 row)
```

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

```
current_setting
-----
32MB
(1 row)
```

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

```
name | setting | unit
-----+-----+-----
work_mem | 32768 | kB
(1 row)
```

---

## Setting parameter values within a transaction

Open a transaction and set a new work\_mem value:

```
=> RESET work_mem;
```

RESET

```
=> BEGIN;
```

BEGIN

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

SET

```
=> SHOW work_mem;
```

```
work_mem
-----
64MB
(1 row)
```

If the transaction is rolled back, so is the new parameter value. If the transaction commits, however, the parameter remains changed.

```
=> ROLLBACK;
```

ROLLBACK

```
=> SHOW work_mem;
```

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

You can set a new parameter value just for the duration of the current transaction:

```
=> BEGIN;
```

BEGIN

```
=> SET LOCAL work_mem TO '64MB'; -- or set_config('work_mem','64MB',true);
```

SET

```
=> SHOW work_mem;
```

```
work_mem
-----
64MB
(1 row)
```

```
=> COMMIT;
```

COMMIT

When the transaction commits, the old value returns:

```
=> SHOW work_mem;
```

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

---

## Custom parameters

Custom parameters can be defined during session. You can also verify if such a parameter is already defined.

Custom parameter names must include the full stop sign (.) in order to distinguish them from standard parameters.

```
=> SELECT CASE
      WHEN current_setting('myapp.currency_code', true) IS NULL
      THEN set_config('myapp.currency_code', 'USD', false)
      ELSE
        current_setting('myapp.currency_code')
    END;
```

```
current_setting
-----
USD
(1 row)
```

Now myapp.currency\_code can be used as a global variable during the session:

```
=> SELECT current_setting('myapp.currency_code');
```

```
current_setting
-----
USD
(1 row)
```

Custom parameters can be defined in postgresql.conf. This will make them initialize in all sessions.

The main configuration file is postgresql.conf

ALTER SYSTEM is an SQL interface for managing configuration parameters stored in postgresql.auto.conf

When configuration files are modified, they have to be reloaded

Some parameters can be changed just for the current session

Changes to some parameters require a server restart to apply

## Practice



1. Get a list of parameters that require a server restart to apply.
2. In the `postgresql.conf` file, make an error when changing the *max\_connections* parameter value.

Restart the server. Make sure that the server does not start and check the message log.

Fix the error and start the server.

11

2. To get the location of the `postgresql.conf` file, check the value of the *config\_file* parameter.

Edit the `postgresql.conf` file either as the owner (the postgres user) or as a superuser.

To do the former, you can open a new terminal window and execute the command there:

```
sudo su postgres
```

To do the latter, open the file in a text editor from the command line using the `sudo` command, for example:

```
sudo vim postgresql.conf
```

## 1. Parameters that require a server restart to apply

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

name	setting	unit
archive_mode	off	
autovacuum_freeze_max_age	200000000	
autovacuum_max_workers	3	
autovacuum_multixact_freeze_max_age	400000000	
bonjour	off	
bonjour_name		
cluster_name	13/main	
config_file	/etc/postgresql/13/main/postgresql.conf	
data_directory	/var/lib/postgresql/13/main	
data_sync_retry	off	
dynamic_shared_memory_type	posix	
event_source	PostgreSQL	
external_pid_file	/var/run/postgresql/13-main.pid	
hba_file	/etc/postgresql/13/main/pg_hba.conf	
hot_standby	on	
huge_pages	try	
ident_file	/etc/postgresql/13/main/pg_ident.conf	
ignore_invalid_pages	off	
jit_provider	llvmjit	
listen_addresses	localhost	
logging_collector	off	
max_connections	100	
max_files_per_process	1000	
max_locks_per_transaction	64	
max_logical_replication_workers	4	
max_pred_locks_per_transaction	64	
max_prepared_transactions	0	
max_replication_slots	10	
max_wal_senders	10	
max_worker_processes	8	
old_snapshot_threshold	-1	min
port	5432	
recovery_target		
recovery_target_action	pause	
recovery_target_inclusive	on	
recovery_target_lsn		
recovery_target_name		
recovery_target_time		
recovery_target_timeline	latest	
recovery_target_xid		
restore_command		
shared_buffers	16384	8kB
shared_memory_type	mmap	
shared_preload_libraries		
superuser_reserved_connections	3	
track_activity_query_size	1024	B
track_commit_timestamp	off	
unix_socket_directories	/var/run/postgresql	
unix_socket_group		
unix_socket_permissions	0777	
wal_buffers	512	8kB
wal_level	replica	
wal_log_hints	off	

(53 rows)

## 2. Setting the max\_connections parameter

The current value of max\_connections:

=> `SHOW max_connections;`

```
max_connections
-----
100
(1 row)
```

Imagine we try to set it to 50, but mistype and enter the letter O instead of the zero:

```
student$ echo max_connections=50 | sudo tee -a /etc/postgresql/13/main/postgresql.conf
```

```
max_connections=50
```

The error can be discovered by looking at the pg\_file\_settings view:



```
=> SELECT * FROM pg_file_settings WHERE name = 'max_connections'\gx
```

```
-[ RECORD 1 ]-----  
sourcefile | /etc/postgresql/13/main/postgresql.conf  
sourceline | 782  
seqno      | 25  
name       | max_connections  
setting    | 50  
applied    | f  
error      | setting could not be applied
```

But here, we didn't look at pg\_file\_settings and restarted the server right away:

```
=> \q
```

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

Job for postgresql@13-main.service failed because the service did not take the steps required by its unit configuration. See "systemctl status postgresql@13-main.service" and "journalctl -xeu postgresql@13-main.service" for details.

The server does not start. The reason is recorded in the server log. Here is what it says:

```
student$ tail -n 5 /var/log/postgresql/postgresql-13-main.log
```

```
2024-03-07 13:52:57.360 MSK [147174] LOG:  database system is shut down  
2024-03-07 13:52:57.714 MSK [147569] LOG:  invalid value for parameter "max_connections": "50"  
2024-03-07 13:52:57.714 MSK [147569] FATAL:  configuration file "/etc/postgresql/13/main/postgresql.conf" contains errors  
pg_ctl: could not start server  
Examine the log output.
```

Let's fix the error:

```
student$ sudo sed -i 's/50/50/' /etc/postgresql/13/main/postgresql.conf
```

Start the server:

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

The server starts. Check the max\_connections value:

```
student$ psql
```

```
=> SHOW max_connections;
```

```
max_connections  
-----  
50  
(1 row)
```

1. Set the parameter `work_mem = 32MB` in the `psql` tool's command line options.
2. In the Ubuntu package distribution, the `postgresql.conf` file is not located in the `PGDATA` directory. How does the server find this configuration file at startup?

1. Use either the `options` key in the connection string or the `PGOPTIONS` environment variable.

More on how connection strings are formed: \_

<https://postgrespro.ru/docs/postgresql/13/runtime-config-logging.html#RUNTIME-CONFIG-LOGGING-WHERE>

2. To see the location of the `postgresql.conf` file, check the `config_file` parameter.

To find where the parameter is set, run the `ps` command for the postgres main process. The process ID (PID) is the first line of the `postmaster.pid` file, which is located in the data directory (`PGDATA`).

## 1. Setting parameters at application start

If the app uses the libpq library to connect to the server, there are two ways you can set parameters at application start.

The first way is to add the options key to the connection string:

```
student$ psql "options='-c work_mem=32MB'" -c 'SHOW work_mem'

 work_mem
-----
 32MB
(1 row)
```

The second way is to set the PGOPTIONS environment variable:

```
student$ export PGOPTIONS='-c work_mem=32MB'; psql -c 'SHOW work_mem'

 work_mem
-----
 32MB
(1 row)
```

## 2. The config\_file variable

The file postgresql.conf isn't located within the data catalog:

```
=> SHOW config_file;

 config_file
-----
 /etc/postgresql/13/main/postgresql.conf
(1 row)

=> SHOW data_directory;

 data_directory
-----
 /var/lib/postgresql/13/main
(1 row)
```

How does the server know where postgresql.conf is?

In the Ubuntu package distribution, the config\_file value is defined in the server startup command text. This allows the postgresql.conf file to be stored outside of PGDATA.

The server startup command text is stored in the pg\_ctlcluster utility's source code. It can also be found in the description of the postgres process. The process ID (PID) of the main server process (usually called postmaster) is stored in the first line of the postmaster.pid file. The file itself is stored in the PGDATA catalog.

```
student$ sudo cat /var/lib/postgresql/13/main/postmaster.pid
```

```
147963
/var/lib/postgresql/13/main
1709808788
5432
/var/run/postgresql
localhost
 655375    98351
ready
```

```
student$ ps 147963
```

PID	TTY	STAT	TIME	COMMAND
147963	?	Ss	0:00	/usr/lib/postgresql/13/bin/postgres -D /var/lib/postgresql/13/main -c config_file=/etc/postgresql/13/main/postgresql.conf