The “default” postgresql.conf, step by step

data egret

Your remote PostgreSQL DBA team

Ilya Kosmodemiansky

ik@dataegret.com

Securing your PostgreSQL database availability and high performance



Contributing sponsor to PostgreSQL

MIGRATION
POSTGRESQL SUPPORT
CLOUD COST MANAGEMENT

data egret

Your PostgreSQL DBA team

EXPERTISE

- DBA with 10+ years of experience in PostgreSQL administration.
- Significant Contributors

EDUCATION

Co-founders of Open Alliance for PostgreSQL Education

Open Alliance For PostgreSQL Education

OPEN SOURCE ADVOCATES

Accompanying you on your journey to Open Source PostgreSQL

OSBA Mitglied

COMMUNITY

Recognised as Significant Contributing Sponsor to PostgreSQL.

COURSES
FOR DBA
AND DEVELOPERS

Scan to explore curriculum →

Before we start…

- 271 settings in version 10 → 348 in version 14 → 399 in version 18
- Defaults on the following slides are PostgreSQL 18 defaults
- Settings in **postgresql.conf** are changed manually
- **postgresql.auto.conf** is written by **ALTER SYSTEM** and wins over **postgresql.conf**
  - since 17, `allow_alter_system = off` switches ALTER SYSTEM off
- **pg_settings** combines everything together

pg_settings

```
postgres=# \x
Expanded display is on.
postgres=# select * from pg_settings where name ~ 'checkpoint_timeout';
-[ RECORD 1 ]---+------------------------------------------------------------
name            | checkpoint_timeout
setting         | 3600
unit            | s
category        | Write-Ahead Log / Checkpoints
short_desc      | Sets the maximum time between automatic WAL checkpoints.
extra_desc      |
context         | sighup
vartype         | integer
source          | configuration file
min_val         | 30
max_val         | 86400
enumvals        |
boot_val        | 300
reset_val       | 3600
sourcefile      | /etc/postgresql/18/main/postgresql.conf
sourceline      | 267
pending_restart | f
```

pg_settings category

```
postgres=# select distinct(category) from pg_settings order by 1;
                             category
--------------------------------------------------------------------
 Client Connection Defaults / Locale and Formatting
 Client Connection Defaults / Other Defaults
 Client Connection Defaults / Shared Library Preloading
 Client Connection Defaults / Statement Behavior
 Connections and Authentication / Authentication
 Connections and Authentication / Connection Settings
 Connections and Authentication / SSL
 Connections and Authentication / TCP Settings
 Developer Options
 Error Handling
 File Locations
 Lock Management
 Preset Options
 Query Tuning / Genetic Query Optimizer
 Query Tuning / Other Planner Options
 Query Tuning / Planner Cost Constants
 Query Tuning / Planner Method Configuration
 Replication / Primary Server
 Replication / Sending Servers
 Replication / Standby Servers
 Replication / Subscribers
 Reporting and Logging / Process Title
 Reporting and Logging / What to Log
 Reporting and Logging / When to Log
 Reporting and Logging / Where to Log
 Resource Usage / Background Writer
 Resource Usage / Disk
 Resource Usage / I/O
 Resource Usage / Kernel Resources
 Resource Usage / Memory
 Resource Usage / Worker Processes
 Statistics / Cumulative Query and Index Statistics
 Statistics / Monitoring
 Vacuuming / Automatic Vacuuming
 Vacuuming / Cost-Based Vacuum Delay
 Vacuuming / Default Behavior
 Vacuuming / Freezing
 Version and Platform Compatibility / Other Platforms and Clients
 Version and Platform Compatibility / Previous PostgreSQL Versions
 Write-Ahead Log / Archive Recovery
 Write-Ahead Log / Archiving
 Write-Ahead Log / Checkpoints
 Write-Ahead Log / Recovery
 Write-Ahead Log / Recovery Target
 Write-Ahead Log / Settings
 Write-Ahead Log / Summarization
(46 rows)
```

pg_settings context

```
postgres=# select distinct(context) from pg_settings;
      context
-------------------
 postmaster
 superuser-backend
 user
 internal
 backend
 sighup
 superuser
(7 rows)
```

postgresql.conf - main config file

- We usually advise not to change the order of the settings when you edit them manually
- If a setting appears twice, the last one wins
- **postgresql.conf** supports includes: `include`, `include_if_exists`, `include_dir`
- Always check **pg_settings** if in doubt
- And off we go

listen_addresses

- We usually use `*` or `127.0.0.1` (default: `localhost`). One can use `127.0.0.1` if Postgres works together with PgBouncer.
- If Postgres listens on an external IP address, this IP address must be protected by a firewall (and by **pg_hba.conf**).
- There are *arguments* that using a Unix socket could bring more performance, but generally using TCP is more convenient because of keepalives.

max_connections

- Each client connection makes Postgres spawn a “heavy” Unix process
- That’s why things like `max_connections = 1000` would never work
- A much better idea: `max_connections = 100` (default) or `200`, and really small pool sizes in PgBouncer or another connection pooler

Reserved connections

- When all `max_connections` are in use, a DBA still needs to get in to troubleshoot
- `superuser_reserved_connections = 5` (default 3): last resort, superusers only
- Since 16: `reserved_connections = 10` for roles with `pg_use_reserved_connections`, e.g. monitoring and DBA accounts that are not superusers
- Both are taken out of `max_connections`

Don’t forget

Postgres already has some processes:

```
54  0:00 /usr/lib/postgresql/18/bin/postgres -D /var/lib/postgresql/18/main -c config_file=/etc/postgresql/18/main/postgresql.conf
55  0:00 postgres: 18/main: io worker 0
56  0:00 postgres: 18/main: io worker 1
57  0:00 postgres: 18/main: io worker 2
58  0:00 postgres: 18/main: checkpointer
59  0:00 postgres: 18/main: background writer
61  0:00 postgres: 18/main: walwriter
62  0:00 postgres: 18/main: autovacuum launcher
63  0:00 postgres: 18/main: logical replication launcher
```

- New in 18: I/O workers (`io_method = worker` , `io_workers = 3`); the stats collector is gone since 15
- None of them uses a `max_connections` slot

Keepalives

- `tcp_keepalives_idle = 5` If the network is unstable, 5 seconds can really help
- `tcp_keepalives_interval = 1`
- `tcp_keepalives_count = 5`
- Even if you have very good network quality between your app and database, it can become *suddenly* unstable

shared_buffers

- Rule of thumb: 25% of RAM
- But to use 16/32/64GB of `shared_buffers` efficiently, fast disks are required
- Our exception (the docs say more than 40% of RAM rarely helps): if the database is definitely smaller than RAM, 75% of RAM can also work

huge_pages

- Rule of thumb: with at least 8–16GB of `shared_buffers`, use huge pages
- `huge_pages = on` (and not `try`, the default)
- Huge pages must be enabled in the kernel first: `vm.nr_hugepages`
- How many: `postgres -D $PGDATA -C shared_memory_size_in_huge_pages` (15+, with the server stopped)
- If you use `try`, check `huge_pages_status` (17+)

temp_file_limit

- Maximum disk space for temporary files, per process (default -1: no limit)
- Each parallel worker gets its own allowance
- Good starting point: **number of processes × temp_file_limit < 10% of your disk**
- But it depends a lot on the particular server and application

work_mem

- Memory per sort or hash operation, not per process: one query can use it several times
- 128MB is a good starting point (default 4MB)
- Hash operations can use `hash_mem_multiplier` × `work_mem`: 2.0 by default since 15, so 256MB
- Too high a setting can cause OOM
- Can be set per session, role or database

maintenance_work_mem

- Memory for maintenance operations in any session: `VACUUM`, `CREATE INDEX`, `ALTER TABLE ADD FOREIGN KEY` (default 64MB)
- 256–512MB, if there is enough RAM
- Quite helpful for `CREATE INDEX CONCURRENTLY`
- `autovacuum_work_mem` overrides it for autovacuum, per worker (default -1: use `maintenance_work_mem`)

Write Ahead Log

- `wal_level = replica` (default) unless you use logical replication
- `checkpoint_timeout = 60min` can improve performance, if longer crash recovery is acceptable
- `max_wal_size = 32GB`, if `pg_wal` has the space
- `checkpoint_completion_target = 0.9` (default since 14)

bgwriter

- The background writer writes dirty buffers in advance, so that backends find clean ones
- Sadly, it is not the best part of the PostgreSQL codebase
  - Make it as aggressive as possible:
    - `bgwriter_delay = 10ms` (the minimum)
    - `bgwriter_lru_maxpages = 1000`
    - `bgwriter_lru_multiplier = 10.0` (the maximum)
  - It might help if your disks are not the best

Must have optimizer settings

- `effective_cache_size = 2 * shared_buffers` or less: a planner hint, it allocates nothing
- `default_statistics_target = 100` (default)
  - raise it per column: `ALTER TABLE … ALTER COLUMN … SET STATISTICS`

Autovacuum

- There is no practical use case where autovacuum should be disabled
- Deserves a separate talk (or maybe a separate tutorial)
- We try to provide some good starting points

Autovacuum

- Autovacuum checks tables one after another to figure out if they need to be vacuumed
- A table is vacuumed when its dead rows exceed `autovacuum_vacuum_threshold` + `autovacuum_vacuum_scale_factor` × rows
- `autovacuum_vacuum_threshold = 50` (default): 50 rows
- `autovacuum_vacuum_scale_factor = 0.05` (default 0.2): plus 5% of the rows
- Since 18, `autovacuum_vacuum_max_threshold` (default 100M) caps it for huge tables

Autovacuum

- If there is a lot of write activity in your database,
- the default `autovacuum_max_workers = 3` might not be enough
- Consider `autovacuum_max_workers = 10`, but beware of I/O
- All workers share one cost limit: raise `autovacuum_vacuum_cost_limit` together with them
- Since 18 it changes with a reload, up to `autovacuum_worker_slots` (default 16, restart)

Freezing and long transactions

Setting	Default	Our advice
`vacuum_freeze_min_age`	50000000	keep the default
`vacuum_freeze_table_age`	150000000	keep the default
`autovacuum_freeze_max_age`	200000000	keep the default
`idle_in_transaction_session_timeout`	0 (off)	**set to `'6h'`**
`transaction_timeout` (17+)	0 (off)	optional

- The freezing defaults are enough today: failsafe (14), page-level freezing (16), eager freezing (18)
- Idle and long transactions hold vacuum back: end them

Autovacuum also collects statistics

- `autovacuum_analyze_threshold = 50` (default)
- `autovacuum_analyze_scale_factor = 0.05` (default 0.1)
- Same formula (without the 18 cap), counting rows changed since the last analyze

Autovacuum

- `autovacuum_naptime = 1s` (default 1min)
- `autovacuum_vacuum_cost_delay = 2ms`: the default since 12 (it was 20ms)
- If autovacuum still can't keep up, raise `autovacuum_vacuum_cost_limit` (default -1: `vacuum_cost_limit` = 200, shared by all workers)

Those settings were designed to reduce the impact of vacuuming on the system, but that doesn't work, so minimize them: a small delay and a bigger budget.

Don’t forget manual vacuum

- There are cases when you need it
- Manual `VACUUM` is not throttled: `vacuum_cost_delay = 0` (default)
- The page costs are shared with autovacuum: `vacuum_cost_page_hit` / `_miss` / `_dirty` = 1 / 2 / 20 (miss was 10 before 14)
- `vacuum_cost_limit = 200` is also autovacuum’s budget, unless `autovacuum_vacuum_cost_limit` is set

Logging

```
logging_collector = on            # without it, the next four lines do nothing
log_directory = '/var/log/postgresql'
log_filename = 'postgresql-%Y-%m-%d.log'
log_rotation_age = 1d             # default
log_rotation_size = 0
log_min_error_statement = error   # default
log_min_duration_statement = 1000 # -1 disabled, 0 all, in ms
log_checkpoints = on              # default since 15
log_line_prefix = '%m %p %u@%d from %h [vxid:%v txid:%x] [%i] '
log_lock_waits = on
log_statement = 'none'            # default
log_replication_commands = on
log_temp_files = 0
log_timezone = 'Europe/Berlin'
```

Logging

```
track_activities = on                # default
track_counts = on                    # default
track_io_timing = on
track_functions = pl
track_activity_query_size = 8192     # needs a restart
log_autovacuum_min_duration = 1000 # in ms; default 10min since 15
```

pg_stat_statements

```
shared_preload_libraries = 'pg_stat_statements'
pg_stat_statements.max = 10000
pg_stat_statements.track = top          # default
pg_stat_statements.track_utility = false
pg_stat_statements.save = false
```

Do not forget to create the extension in the postgres database:

```
psql -d postgres -c "create extension pg_stat_statements"
```

Replication

- `hot_standby = on` (default since 10)
- `max_standby_streaming_delay = 120s` (default 30s): max delay before canceling conflicting queries
- `hot_standby_feedback = on`
  - against `ERROR: canceling statement due to conflict with recovery`
  - you need it only if you have intensive OLTP on primary
  - do not switch it on until your replica has caught up!

New since PostgreSQL 14

Setting	Since	Why
`io_method = worker`, `io_workers = 3`	18	asynchronous I/O for reads
`effective_io_concurrency`, `maintenance_io_concurrency`	18	default 16, were 1 and 10
`wal_compression = lz4` or `zstd`	15	less WAL from full-page images
`idle_replication_slot_timeout`	18	invalidates abandoned slots
`log_connections = 'all'`	18	now a list of what to log
`track_cost_delay_timing = on`	18	shows how long vacuum sleeps

Already on earlier slides: `allow_alter_system`, `reserved_connections`, `huge_pages_status`, `autovacuum_worker_slots`, `autovacuum_vacuum_max_threshold`, `transaction_timeout`

Questions?



ik@dataegret.com

Securing your PostgreSQL database availability and high performance



Contributing sponsor to PostgreSQL

MIGRATION
POSTGRESQL SUPPORT
CLOUD COST MANAGEMENT

data egret

Your PostgreSQL DBA team

EXPERTISE

- DBA with 10+ years of experience in PostgreSQL administration.
- Significant Contributors

EDUCATION

Co-founders of Open Alliance for PostgreSQL Education

Open Alliance For PostgreSQL Education

OPEN SOURCE ADVOCATES

Accompanying you on your journey to Open Source PostgreSQL

OSBA Mitglied

COMMUNITY

Recognised as Significant Contributing Sponsor to PostgreSQL.

COURSES
FOR DBA
AND DEVELOPERS

Scan
to explore
curriculum
→