

PostgreSQL Logging in Practice

Choose settings by the questions
you need answered at 09:00.

Alicja Kucharczyk · Guardioz.com

22 October 2026 · 13:15-13:45 · Audit 3



Alicja Kucharczyk

15 YEARS OF POSTGRESQL

Performance tuning, troubleshooting
Microsoft, Azure Database for PostgreSQL
PostgreSQL consulting, teams worldwide

COMMUNITY

Founder, Warsaw PostgreSQL Users Group
Organizer, PGConf.PL



PRODUCT OWNER

**Performance guidance for teams that run
PostgreSQL in production without a dedicated DBA.**

Ranked findings, each with the reasoning and a safe next step



TECHNICAL BOARD LEAD

Tomorrow: the first DBA-1 exam
23 October · Community Events Day

 **Guardioz: my 15 years of PostgreSQL experience, in one product.**

A powerful mechanism, easy to get lost in.

The scope: Error Reporting and Logging, section 19.8 of the PostgreSQL documentation.

THE MECHANISM

42

settings in Where, When
and What to Log (PG 18)

Some overlap. Some names mislead:

`log_min_error_statement`

`log_duration`

`log_hostname`

THIS TALK

Clear up the misunderstandings
I see most often with customers.

A ready-made recipe.

NOT IN THIS TALK

pgAudit, compliance

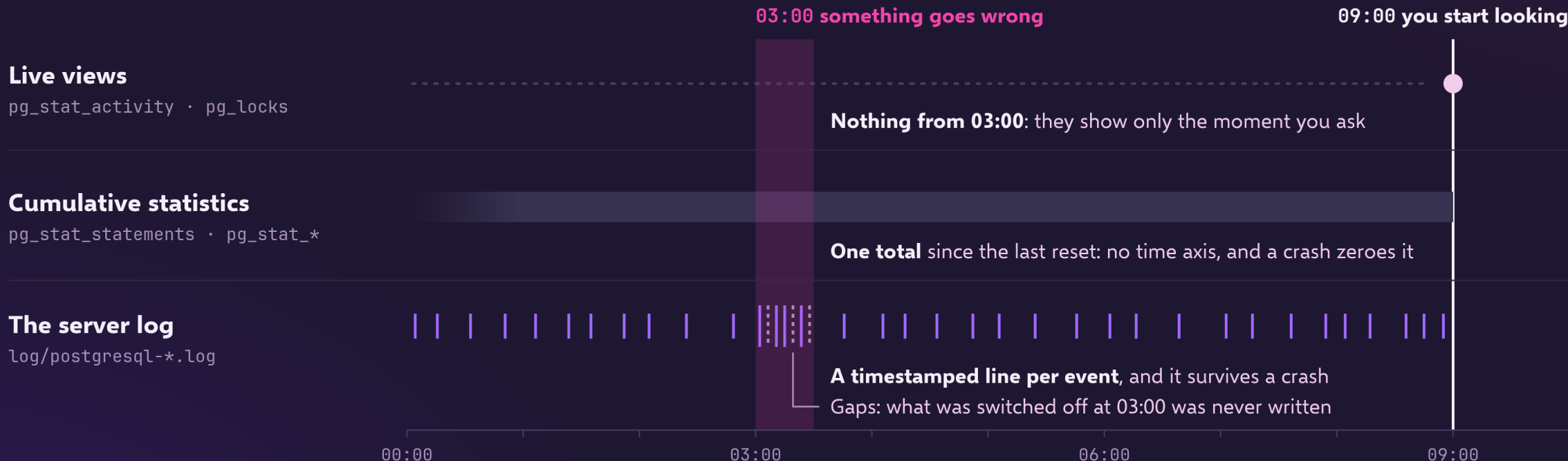


The main point: when an incident starts, the information is already in the log.

the server log

At 09:00, only the log still remembers 03:00

The views show you now, or a total since the last reset. Only the log keeps a timeline.



The log holds only what you asked for before 03:00.

Switching a setting on at 09:00 helps with the next incident, not with this one.

after the incident · THIS TALK IN SEVEN QUESTIONS

By default, the log answers two of these seven

Start with **logging_collector = on**. It is off by default: then PostgreSQL writes no log file of its own.

THE QUESTION	THE STOCK SETTING	IN THE LOG
What failed, or crashed?	<code>log_min_messages = warning</code>	☒ logged with the failing SQL
Which application ran it?	<code>log_line_prefix = '%m [%p] '</code>	☐ <i>time and PID only</i>
Why was it slow at 03:00?	<code>log_min_duration_statement = -1</code>	☐ <i>no log line</i>
Who blocked the one-row UPDATE?	<code>log_lock_waits = off</code>	☐ <i>no log line</i> (on by default from PG 19)
Who filled the disk with temp files?	<code>log_temp_files = -1</code>	☐ <i>no log line</i>
Was that stall a checkpoint?	<code>log_checkpoints = on</code>	☒ logged (default since PG 15)
Is autovacuum keeping up?	<code>log_autovacuum_min_duration = 10min</code>	☐ partly only runs over 10 min

Each gap is one line in `postgresql.conf`, and a reload.

Set them on day one, in the same Terraform or Ansible code that creates the server.

Three sections of postgresql.conf.

WHERE TO LOG

- Where to Log -

**Where does the log go,
and will it last?**

ANSWERS AFTER AN INCIDENT

Where is the 03:00 log?
Is it still there in a week?

WHEN TO LOG

- When to Log -

**Serious enough?
Slow enough?**

ANSWERS AFTER AN INCIDENT

What failed, or crashed?
Why was it slow at 03:00?

WHAT TO LOG

- What to Log -

**What does a line carry?
Which events get one?**

ANSWERS AFTER AN INCIDENT

Which application ran it?
Who blocked the UPDATE?
Who filled the disk?
Was it a checkpoint?
Is autovacuum keeping up?

Where to log

WHERE TO LOG

YOU ARE HERE

- Where to Log -

**Where does the log go,
and will it last?**

ANSWERS AFTER AN INCIDENT

Where is the 03:00 log?
Is it still there in a week?

WHEN TO LOG

- When to Log -

**Serious enough?
Slow enough?**

ANSWERS AFTER AN INCIDENT

What failed, or crashed?
Why was it slow at 03:00?

WHAT TO LOG

- What to Log -

**What does a line carry?
Which events get one?**

ANSWERS AFTER AN INCIDENT

Which application ran it?
Who blocked the UPDATE?
Who filled the disk?
Was it a checkpoint?
Is autovacuum keeping up?

The collector is off by default.

```
# - Where to Log -

#log_destination = 'stderr'           # Valid values are combinations of
                                       # stderr, csvlog, jsonlog, syslog, and
                                       # eventlog, depending on platform.
                                       # csvlog and jsonlog require
                                       # logging_collector to be on.

# This is used when logging to stderr:
#logging_collector = off               # Enable capturing of stderr, jsonlog,
                                       # and csvlog into log files. Required
                                       # to be on for csvlogs and jsonlogs.
                                       # (change requires restart)

# These are only used if logging_collector is on:
#log_directory = 'log'                # directory where log files are written,
                                       # can be absolute or relative to PGDATA
```

 **On a VM or bare metal, turn it on before the server goes live.**

A week of logs, no cron job.

DEFAULT NAME

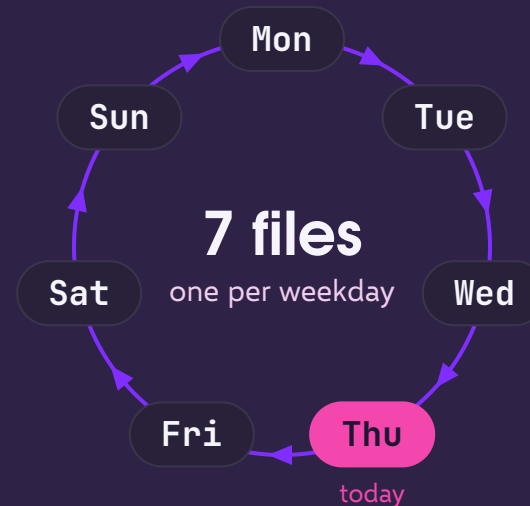
`postgresql-%Y-%m-%d_%H%M%S.log`

```
...
postgresql-2026-09-27_000000.log
postgresql-2026-09-28_000000.log
postgresql-2026-09-29_000000.log
postgresql-2026-09-30_000000.log
postgresql-2026-10-01_000000.log
postgresql-2026-10-02_000000.log
```

A new name every day, and nothing deletes the old files: cron or logrotate has to.

WEEKDAY NAME + TRUNCATE

`postgresql-%a.log`



At midnight the server moves to the next name: last week's file.

Truncate on rotation empties it first.

**Nothing to delete,
nothing to schedule.**



No size limit: one busy day can fill the disk.

At most 7 days of history.

WHERE: five lines, a week of logs.

Where is the 03:00 log, and will it last a week?

```
WHERE  logging_collector = on           # RESTART: set it before go-live
        log_filename = 'postgresql-%a.log' # Mon ... Sun: 7 files
        log_rotation_age = 1d             # new file every midnight
        log_rotation_size = 0             # %a can't rotate on size
        log_truncate_on_rotation = on     # overwrite last week's file
```

[COPY](#)

WHEN

WHAT

NEXT We have a file. What goes into it?

When to log

WHERE TO LOG

- Where to Log -

**Where does the log go,
and will it last?**

ANSWERS AFTER AN INCIDENT

Where is the 03:00 log?
Is it still there in a week?

WHEN TO LOG

YOU ARE HERE

- When to Log -

**Serious enough?
Slow enough?**

ANSWERS AFTER AN INCIDENT

What failed, or crashed?
Why was it slow at 03:00?

WHAT TO LOG

- What to Log -

**What does a line carry?
Which events get one?**

ANSWERS AFTER AN INCIDENT

Which application ran it?
Who blocked the UPDATE?
Who filled the disk?
Was it a checkpoint?
Is autovacuum keeping up?

Same values, different jobs.

POSTGRESQL.CONF

```
#log_min_messages = warning          # debug5 ... debug1, info, notice, warning, error, log, fatal, panic  
#log_min_error_statement = error     # debug5 ... debug1, info, notice, warning, error, log, fatal, panic
```

WHAT EACH ONE PRINTS · select 1/0;

ERROR: division by zero	← log_min_messages = warning
STATEMENT: select 1/0;	← log_min_error_statement = error

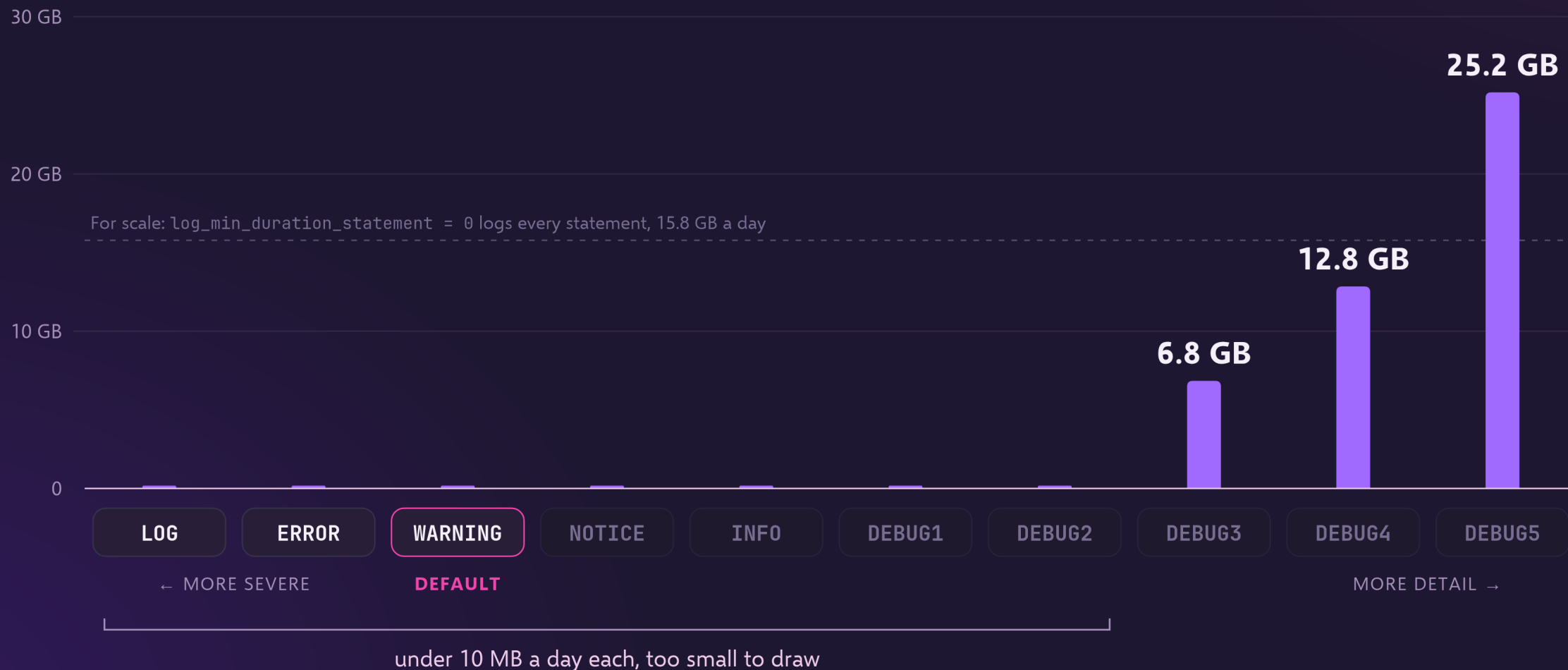
 **log_min_messages: is the message written? log_min_error_statement: does its SQL come along?**

postgresql.conf · log_min_messages

debug5 writes 25.2 GB of log a day. The default: 115 kB.

Same server, same 200 transactions per second. Only log_min_messages changes.

pgbench on PostgreSQL 18.6, 4 vCPU · 10 minutes per level, scaled to a day



The slow-query log: one setting.

log_min_duration_statement

-1

off - the default

0

every statement -
watch the overhead

N (e.g. '1s')

every statement that
takes N or longer

```
2026-10-02 12:03:05.813 CEST [1143805] LOG:  duration: 6972.697 ms ← how long it took  
statement: select count(*) from generate_series(1, 40000000) as g(i) ← the query  
where i % 7 = 0;
```

 **Start with '1s', then watch the overhead on your database.**

Keep every slow query. Sample the middle.

```
log_min_duration_statement = '1s'  
log_min_duration_sample = '200ms'  
log_statement_sample_rate = 0.01
```

< 200 ms

Outside this policy

200 ms to 1 s

Sample about 1%

≥ 1 s

Log every completion



Estimate bytes/event × events/second × 86,400 before enabling everything.

WHEN: one line. Errors are logged already.

What failed, and what was slow? Keep `log_min_messages` and `log_min_error_statement` at their defaults.

```
WHERE    logging_collector = on                # RESTART: set it before go-live
          log_filename = 'postgresql-%a.log'    # Mon ... Sun: 7 files
          log_rotation_age = 1d                 # new file every midnight
          log_rotation_size = 0                 # %a can't rotate on size
          log_truncate_on_rotation = on         # overwrite last week's file

WHEN      log_min_duration_statement = '1s'     # why was it slow at 03:00?
```

COPY

WHAT

```
_____
_____
_____
```

NEXT The line says only: time and PID. Who was it?

What to log

WHERE TO LOG

- Where to Log -

**Where does the log go,
and will it last?**

ANSWERS AFTER AN INCIDENT

Where is the 03:00 log?
Is it still there in a week?

WHEN TO LOG

- When to Log -

**Serious enough?
Slow enough?**

ANSWERS AFTER AN INCIDENT

What failed, or crashed?
Why was it slow at 03:00?

WHAT TO LOG

YOU ARE HERE

- What to Log -

**What does a line carry?
Which events get one?**

ANSWERS AFTER AN INCIDENT

Which application ran it?
Who blocked the UPDATE?
Who filled the disk?
Was it a checkpoint?
Is autovacuum keeping up?

Give every line an identity.

DEFAULT

```
log_line_prefix = '%m [%p] '
```

```
┌────────── %m ─────────┐ ┌ [%p] ─┐
```

```
2026-10-06 15:15:08.778 CEST [2179046]
```

```
LOG: duration: 302.343 ms statement: SELECT pg_sleep(0.3)
```

RECOMMENDED

```
log_line_prefix = '%m [%p] %qdb=%d,user=%u,app=%a,client=%h '
```

```
┌────────── %m ─────────┐ ┌ [%p] ─┐
```

```
2026-10-06 15:16:06.594 CEST [2179367] db=orders,user=app_rw,app=billing,client=127.0.0.1
```

```
LOG: duration: 302.122 ms statement: SELECT pg_sleep(0.3)
```

Real lines · PostgreSQL 18.6 · the same query under both prefixes

Let the application send its name.

IN THE CONNECTION STRING

`postgresql://db/shop?application_name=checkout-api`

Django, Node, Go

`jdbc:postgresql://db/shop?ApplicationName=checkout-api`

Hibernate, Spring

%a in the log · application_name in pg_stat_activity

A shared database role does not have to mean anonymous services.



application_name is set by the application: ask the developers for it.

A one-row UPDATE can wait for seconds.

```
LOG:  process 315963 still waiting for ShareLock on transaction 768 after 1000.273 ms
```

```
DETAIL:  Process holding the lock: 315964. Wait queue: 315963. ← match with [%p]
```

```
CONTEXT:  while updating tuple (0,1) in relation "t_orders"
```

```
STATEMENT:  update t_orders set amount = 1 where id = 1;
```

```
...
```

```
LOG:  process 315963 acquired ShareLock on transaction 768 after 2701.787 ms
```

PG18.6 · prefixes omitted

THE THRESHOLD

deadlock_timeout = 1s: the first line appears after it.

 **log_lock_waits = on** shows if lock waits are causing poor performance.

The threshold is per file: set it to 0.

```
log_temp_files = 0
```

every temp file · the threshold is per file

```
LOG:  temporary file: path "base/pgsql_tmp/pgsql_tmp317684.0", size 34095104  
STATEMENT:  select id, customer_id, amount from orders order by created_at;
```

 **Spilling right now? Ask `pg_ls_tmpdir()`, not the log.**

Already on since PostgreSQL 15.

CHECKPOINTS

`log_checkpoints = on`

starting: time → scheduled

starting: wal → check `max_wal_size`

AUTOVACUUM

`log_autovacuum_min_duration = 10min`

Only runs of 10 minutes or more.

Shorter runs leave no line.

 **On by default since PG15: keep the defaults. On PG13 and 14, set them yourself.**

WHAT: three lines, three answers.

Who ran it, who blocked it, who filled the disk?

```
WHERE    logging_collector = on           # RESTART: set it before go-live
          log_filename = 'postgresql-%a.log' # Mon ... Sun: 7 files
          log_rotation_age = 1d            # new file every midnight
          log_rotation_size = 0            # %a can't rotate on size
          log_truncate_on_rotation = on     # overwrite last week's file

WHEN      log_min_duration_statement = '1s' # why was it slow at 03:00?

WHAT      log_line_prefix = '%m [%p] %qdb=%d,user=%u,app=%a,client=%h ' # who ran it, from which IP?
          log_lock_waits = on              # who blocked the UPDATE?
          log_temp_files = 0               # who filled the disk?
```

[COPY](#)

NEXT Some settings from different sections act on the same event.

Three settings. Three different answers.

Setting	Section	What you get
log_duration = on	What	duration: 24.968 ms
log_statement = 'all'	What	statement: select 'DUPLICATE-TEST-MARKER' ...
log_min_duration_statement = 0	When	duration: 23.625 ms statement: select count(*) ...

Captured PG18.3 message bodies · SQL shortened for the slide



Time and SQL in one line: only log_min_duration_statement. Keep log_duration = off.

More logging can split your evidence.

```
log_statement = 'all'  
log_min_duration_statement = 0
```

```
LOG:  statement: select 'DUPLICATE-TEST-MARKER' as probe; ← log_statement  
LOG:  duration: 0.173 ms ← log_min_duration_statement
```

**One statement, two lines:
first the SQL, then the time without the SQL.**

 **So: log_duration = off and log_statement = 'none' (or 'ddl'). Never 'all'.**

%h prints an IP, or a name.

```
log_line_prefix = '%m [%p] %qdb=%d,user=%u,app=%a,client=%h '
```

IN THE LOG

log_hostname = off

from the socket

client=10.42.1.5

log_hostname = on

reverse DNS lookup

client=vm-logtalk-hostname-cli.internal.cloudapp.net

What one DNS lookup per connection costs.

Azure · PostgreSQL 18.6 · TLS + SCRAM · a new connection for every SELECT 1 · no local DNS cache

Scenario	Per connection	%h prints
log_hostname = off	14 ms	10.42.1.5
on, DNS healthy	20 ms (+42%)	vm-logtalk-hostname-cli...
on, DNS answers 100 ms late	123 ms	the same name
on, DNS not answering	20 s	??? (host=??? port=???)
off, DNS not answering	14 ms	10.42.1.5

```
... bench@postgres/pgbench ??? LOG:  connection ready: setup total=20033.681 ms,  
    fork=0.540 ms, authentication=5.641 ms
```

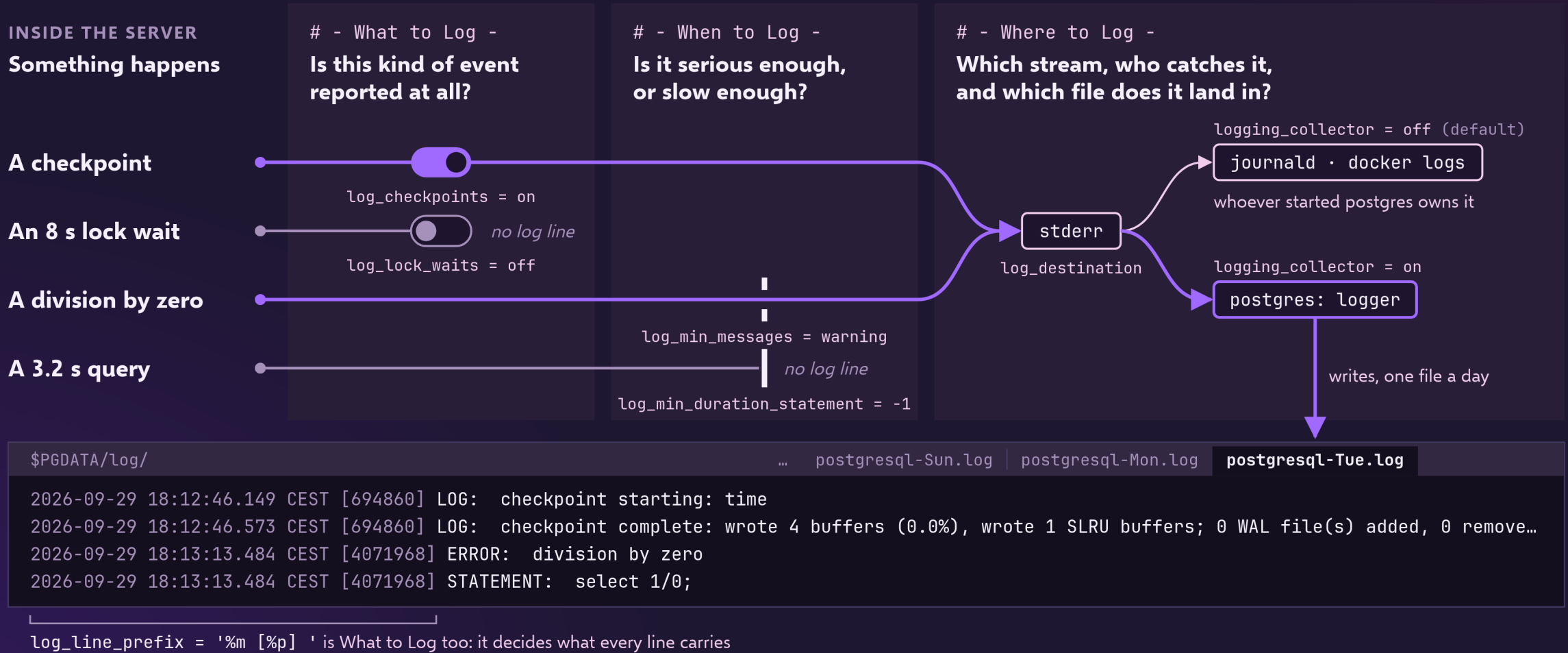
Real line, DNS not answering: 20 s of setup, 5.6 ms of it authentication, and ??? where the address should be



You pay one DNS lookup per connection. When DNS fails, you wait 20 s and lose the IP you already had.

Three questions stand between an event and your log

postgresql.conf lists them Where → When → What. An event meets them in reverse order.



The complete block: 12 lines.

WHERE	<code>logging_collector = on</code> <code>log_filename = 'postgresql-%a.log'</code> <code>log_rotation_age = 1d</code> <code>log_rotation_size = 0</code> <code>log_truncate_on_rotation = on</code>	<code># RESTART: set it before go-live</code> <code># Mon ... Sun: 7 files</code> <code># new file every midnight</code> <code># %a can't rotate on size</code> <code># overwrite last week's file</code>	RESTART COPY
WHEN	<code>log_min_duration_statement = '1s'</code>	<code># why was it slow at 03:00?</code>	
WHAT	<code>log_line_prefix = '%m [%p] %qdb=%d,user=%u,app=%a,client=%h '</code> <code>log_lock_waits = on</code> <code>log_temp_files = 0</code>	<code># who ran it, from which IP?</code> <code># who blocked the UPDATE?</code> <code># who filled the disk?</code>	
KEEP OFF	<code>log_duration = off</code> <code>log_statement = 'none'</code> <code>log_hostname = off</code>	<code># durations without SQL</code> <code># or 'ddl'; never 'all'</code> <code># %h already logs the IP, no DNS</code>	

`SELECT pg_reload_conf();` 11 lines apply now. logging_collector waits for the next maintenance window.

 **New server: put the whole block in the Terraform or Ansible code that creates it.**

PGCONF.PL 2026

See you in Gdańsk on 24 November.



ARCHE Dwór Uphagena, Gdańsk

One day, one track, seven talks

Tickets €60 + VAT



2026.pgconf.pl

Thank you!

POSTGRESQL 19 · BETA

log_lock_waits = on
becomes the default.
log_autoanalyze_min_duration
gets its own setting.

**Where, when and what:
decide them before
the incident, in code.**



FEEDBACK

Rate this talk

PGConf.EU feedback form



LINKEDIN

Alicja Kucharczyk

questions after the talk



BLOG POST

log_hostname

explained and measured