



# PERCONA

Databases run better with Percona



# Most Common Mistakes and the Cost of Mistakes

Everyone Should Be Aware

JOBIN AUGUSTINE

**PostgreSQL Escalation Specialist**



# Copying “Oracle” or “XYZ” way

- Best Practices for XYZ or ABC software may not be relevant for PostgreSQL



# Infrastructure Mistakes

# Non-Standard binaries

- Generally triggered by the non-standard locations
- Source installation is easy. doesn't mean that you need to be doing that always.
- PostgreSQL is a database Kernel with thousands of extensions and tools around

## **Later Problems**

- Extensions and tools fails
- Difficult to troubleshoot, longer outages
- Too many edge cases.

# Recommendations

- Avoid source builds for Production use
- Use community / standard binaries for Production use
- Importance of testing every pieces together

# “postgres” user account

- Users ignores the Importance of OS user account
- It is possible to run Postgres in OS account. However, Special attention should be there starting from the **file permissions, peer/ident authentication, default super user, Service management**



# Misunderstanding of security

- Changing the Port number or Installation location won't add any security
- “Downloading” packages and installing is a frequent cause of security related incidents.
- Obstructions and deviations from standards and best practices often causes more vulnerabilities.

## Recommendation

- Involve security experts to assess if doubt exists
- Report CVE
- Avoid deviations from well tested, proven paths
- Use standard repository for installing packages
- Make the recommended path a “Happy Path”

# Selection of Hosting Infrastructure

- Database on Cloud
- Database on Kubernetes

No Silver Bullets.

- Virtual Machines and Bare metals


# Big tables - No data retention policies

- Big indexes
- Autovacuum has lot more work to do
- requirement for higher maintenance\_work\_mem.

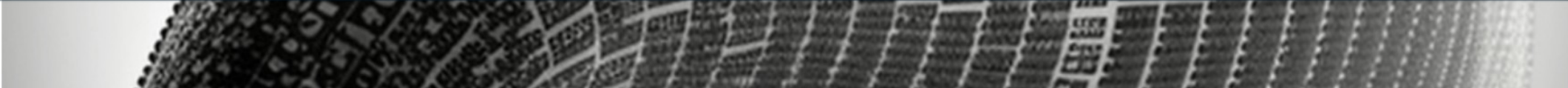


# Frequent Configuration Mistakes

# Memory and HugePages



Solutions ▾Resources ▾Community ▾About ▾




Insight for DBAsInsight for DevelopersPostgreSQL

Subscribe to RSS Feed

## Why Linux HugePages are Super Important for Database Servers: A Case with PostgreSQL

October 12, 2021



Jobin Augustin

# autovacuum\_max\_workers

autovacuum_freeze_max_age	200000000
autovacuum_max_workers	<b>6</b>
autovacuum_multixact_freeze_max_age	400000000
autovacuum_naptime	60
autovacuum_vacuum_cost_delay	20
autovacuum_vacuum_cost_limit	<b>-1</b>

- Workers need to share autovacuum\_vacuum\_cost\_limit
- Higher the number of workers, each worker runs slower
  - Slow running workers defeats its own purpose.

# max\_connections

max_connections	<b><u>10000</u></b>
-----------------	---------------------

<https://richyen.com/postgres/2021/09/03/less-is-more-max-connections.html>

<https://www.citusdata.com/blog/2020/10/08/analyzing-connection-scalability/>

<https://blog.anarazel.de/2020/10/07/measuring-the-memory-overhead-of-a-postgres-connection/>

<https://aws.amazon.com/blogs/database/resources-consumed-by-idle-postgresql-connections/>

<https://elephas.io/connection-scaling/>

- Avoid exceeding 10x of CPU count



Expectations



# vacuum tuning $\neq$ parameter tuning

- Expecting a vacuum tuning achieving by parameter  
Scheduled vacuum jobs are unavoidable.

# Concurrency by design - Blocking/Contention

Event	count
CPU	357119
transactionid	150775
WALWriteLock	48955
tuple	46401
ProcArrayLock	461
lock_manager	403
CLogControlLock	176
wal_insert	146
buffer_content	131
extend	96
XidGenLock	67

# Concurrency by design - Blocking/Contention

- SELECT ... FOR UPDATE
- Myths of throughput.
  - More sessions doesn't mean that faster response the the application or better throughput.

# Datatypes - Common mistakes

- Excessive use of NUMERIC
- Excessive use JSON
- Less use of ENUM

# Multi-Tenancy

DB Name	Avg.Commits	Avg.Rollbacks	Avg.DMLs	Cache hit ratio	Avg.Temp Files	Avg.Temp Bytes	DB size	Age
xxxxxxx	510528	1303	0	99	0	0	55571103	1651961
xxxxxxx	105096	0	0	99	0	0	20682797727	1395754
xxxxxxx	23194	0	0	99	0	0	8163999	321167
xxxxxxx	23194	0	0	99	0	0	8163999	321167
xxxxxxx	105003	0	0	99	0	0	8532639	1907745
xxxxxxx	105140	0	0	99	0	0	1678160543	1499295
xxxxxxxxxxx	105649	0	0	89	0	0	91183919775	1984670
xxxx	105237	0	0	99	0	0	121610891935	1991731
xxxx	105756	0	2	99	0	0	350966431	1907745
xxxxxxxxxxx	1134457	3	3732	99	0	16803529	15290626719	1907745
xxx_xxx_xx	160400	5	4752	99	0	0	56400245407	1907745
xxxxxx_xx	105010	0	0	99	0	0	22901407	1499295
xxxxxx_xxxx	105005	0	0	99	0	0	27808415	1499295
xxxx_xxxx	17639322	15	192823037	89	0	0	209958630047	1907745
xxxxxxxxxx_xxxxxxxxx_xxxx	634098	425	6222	99	0	2353834	6036501151	1991272
xxxxxxxxxx_xxxx	97978	0	381118	99	13	30591268	3490591391	1519132
xxxxxxx	105061	0	168	99	0	0	9622175	1321265
xxxx	120919	185829	42837	99	0	0	36835999	1656096
xxxxxxxxxxx	105868	0	0	99	0	0	21453722271	1878817
xxxxxxxx	105982	0	0	99	0	0	1393259167	1499538
xxxxxxxx_xxx	105212	0	0	99	0	0	24637665951	1848817
xxxxxxxx	116483	22071	121	99	0	297375309	2520076959	1647685
xxxxxxxx	106517	0	0	99	0	0	2283336351	1499295

- Hosting multiple databases / schema in single Instance has very less advantage, But has many disadvantages.

# Network latency - A common performance killer

	Statement since	State since	waits
	00:00:00.233427	00:00:00.233383	CPU: 9.45%, ClientRead: 9.15%, <u>Net/Delay*: 80.57%</u>
	00:00:00.001282	00:00:00.001243	CPU: 9.7%, ClientRead: 6.85%, <u>Net/Delay*: 78.08%</u>
	00:00:00.003834	00:00:00.003775	ClientRead: 13.7%, CPU: 6.35%, <u>Net/Delay*: 79.69%</u>
	00:00:14.832956	00:00:14.832862	ClientRead: 2.15%, CPU: 2.15%, <u>Net/Delay*: 19.87%</u>
	00:00:00.462914	00:00:00.462768	ClientRead: 5.05%, CPU: 2.95%, <u>Net/Delay*: 33.75%</u>
	00:00:00.00174	00:00:00.00169	ClientRead: 10.25%, CPU: 4.05%, <u>Net/Delay*: 33.86%</u>
	00:00:00.462711	00:00:00.46266	CPU: 6.85%, ClientRead: 5.9%, <u>Net/Delay*: 83.03%</u>
	00:00:00.005069	00:00:00.005017	CPU: 5.8%, ClientRead: 2.25%, <u>Net/Delay*: 86.35%</u>

<https://www.percona.com/blog/how-to-measure-the-network-impact-on-postgresql-performance/>





Good&Bad Practices

# Session / Connection termination

- Often done by DBAs
- Scripting for handling “idle” sessions
- Parameter settings like “idle\_session\_timeout”



# Overuse of Indexes

Table	Index	UK?	PK?	Scans	size	Fetch	C.Hit%	Last Use
XXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXX_XX_XXX	f	f	0	491634688	24	45	
XXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXX_XXXXXX	f	f	0	445046784	24	45	
XXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXX_XXXXX_XXXX	f	f	0	417546240	29	44	
XXXXXXXXXXXXXX	XX_XXXXXXXXXXXXXX_XXXXXXXX	t	f	0	343605248	1	0	
XXXXXXXXXXXXXX	XX_XXXXXXXXXXXXXX_XXXXXX	t	f	0	343605248	1	0	
XXXXXXXXXXXXXX	XX_XXXXXXXXXXXXXX_XXXXXX	t	f	0	340058112	1	0	
XXXXXXXXXXXXXX	XX_XXXXXXXXXXXXXX_XXXXXX	t	f	0	340058112	1	0	
XXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXX_XXXX	t	t	0	205537280	24	45	
XXXXXXXXXXXXXXXXXX	XXX_XXXXXXXXXXXXXX_XXXXXXXXXX	t	f	0	123011072	2425	99	
XXXXXXXXXXXXXXXXXX	XXX_XXXXXXXXXXXXXX_XXXXXXXXXX	t	f	0	122200064	1	0	
XXXXXXXXXXXXXXXXXX_XXX	XXXXXXXXXXXXXXXXXX_XXXXXXXXXX_XXX	f	f	0	112123904	1	0	
XXXXXXXXXXXXXXXXXXXXXX	XX_XXXXXXXXXXXXXXXXXXXXXX	t	t	0	93642752	2425	99	
XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX_XXX_XXXX	f	f	0	93642752	2425	99	
XXXXXXXXXXXXXXXXXXXXXX_XXX	XXXXXXXXXXXXXXXXXXXXXX_XXXX	t	t	0	86327296	1	0	
XXXXXXXXXXXXXXXXXXXXXX_XXX	XXXXXXXXXXXXXXXXXXXXXX_XXXX_XXX	f	f	0	76161024	1	0	
XXXXXXXXXXXXXXXXXXXXXX	XXX_XXXXXXXXXXXXXXXXXXXXXX_XXX	f	f	0	65191936	30	50	
XXXXXXXXXXXXXXXXXXXXXX_XXX	XXXXXXXXXXXXXXXXXXXXXX_XXXXXXXXXX_XXX	f	f	0	61562880	1	0	
XXXXXXXXXXXXXXXXXXXXXX_XXX	XXXXXXXXXXXXXXXXXXXXXX_XXXXXXXXXXXXXX	f	f	0	61562880	1	0	
XXXXXXXXXXXXXXXXXXXXXX_XXX	XXXXXXXXXXXXXXXXXXXXXX_XXXXXXXXXXXXXX	f	f	0	61562880	1	0	
XXXXXXXXXXXXXXXXXXXXXX_XXX	XXXXXXXXXXXXXXXXXXXXXX_XX_XXX_XX_XXX	f	f	0	45842432	1	0	
XXXX_XXXXXXXXXX_XXXX_XXXXXXXXXX	XXXX_XXXXXXXXXX_XXXX_XXXXXXXXXX_XXXX	t	t	0	41418			

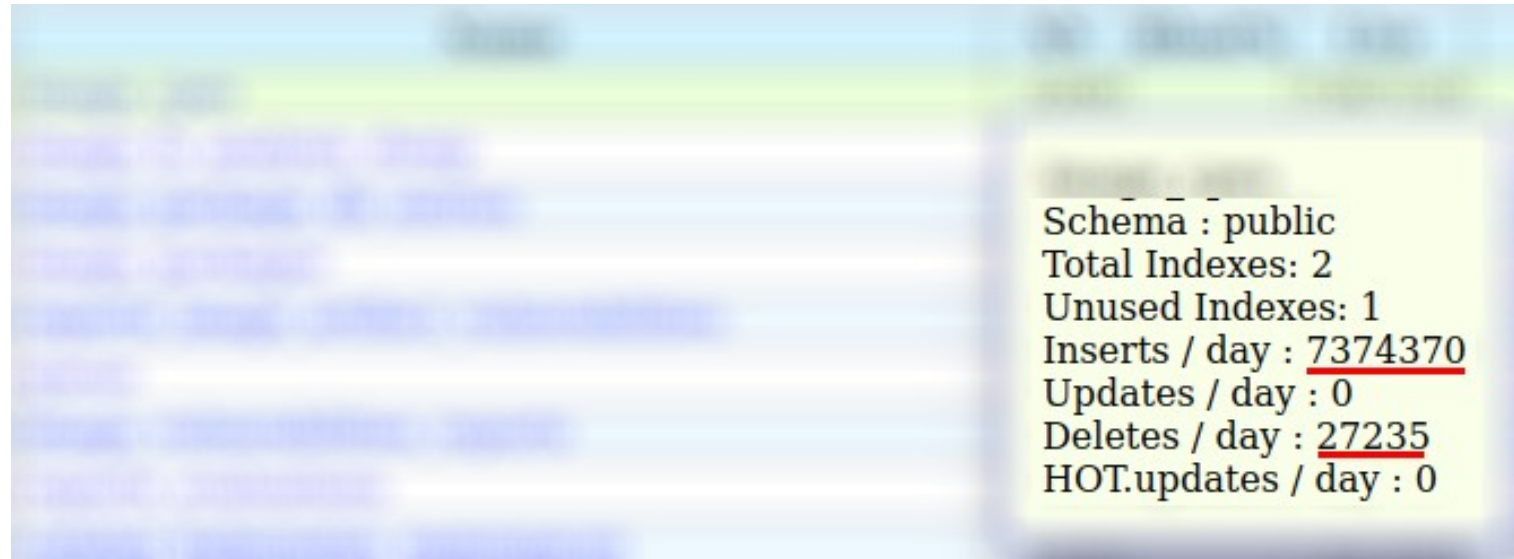


● Indexes are **very very very** costly

<https://www.percona.com/blog/postgresql-indexes-can-hurt-you-negative-effects-and-the-costs-involved/>

# Data retention policy implementation

- Keep an eye on rapidly growing tables.



The screenshot shows a table's statistics in a database management tool. The table is located in the 'public' schema. The statistics are as follows:

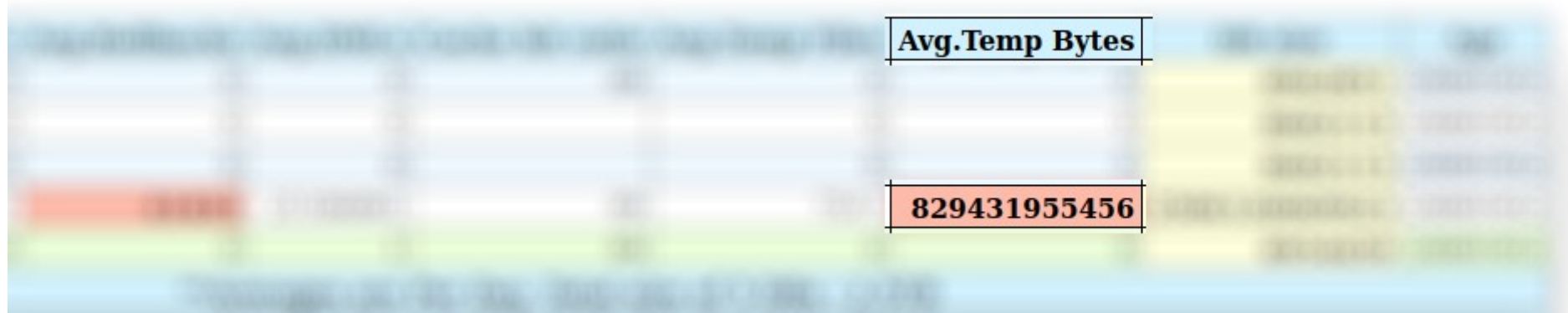
Schema	Total Indexes	Unused Indexes	Inserts / day	Updates / day	Deletes / day	HOT.updates / day
public	2	1	<u>7374370</u>	0	<u>27235</u>	0

# Aborts / Rollbacks

2014-2015	2015-2016	2016-2017	2017-2018	2018-2019	2019-2020	2020-2021
						</

- Inspect PostgreSQL logs
- DBAs/Ops responsibility to report it back and get it fixed

# Temp file generation



The image shows a blurred screenshot of a table with multiple columns. Two specific rows are highlighted with red boxes and labels. The first row is labeled 'Avg.Temp Bytes' and the second row shows the value '829431955456'.

Avg.Temp Bytes	
829431955456	

- Causes Disk I/O
- Cloud vendors may charge hefty

# Over Engineering = Overkill + Shooting on leg

- More the complexity, more the problems, unavailability and security vulnerabilities.
1. Security incidents because server dont have access to PostgreSQL repository
  2. Certificate authentication need to be used with care, Expiry can lead to outage, its a ticking bomb.
  3. Custom developed scripts for HA causing problems.

# External Pools everywhere

- External connection poolers like pgBouncers are over used
- External connection poolers are required if there is no connection pooler on the application side.

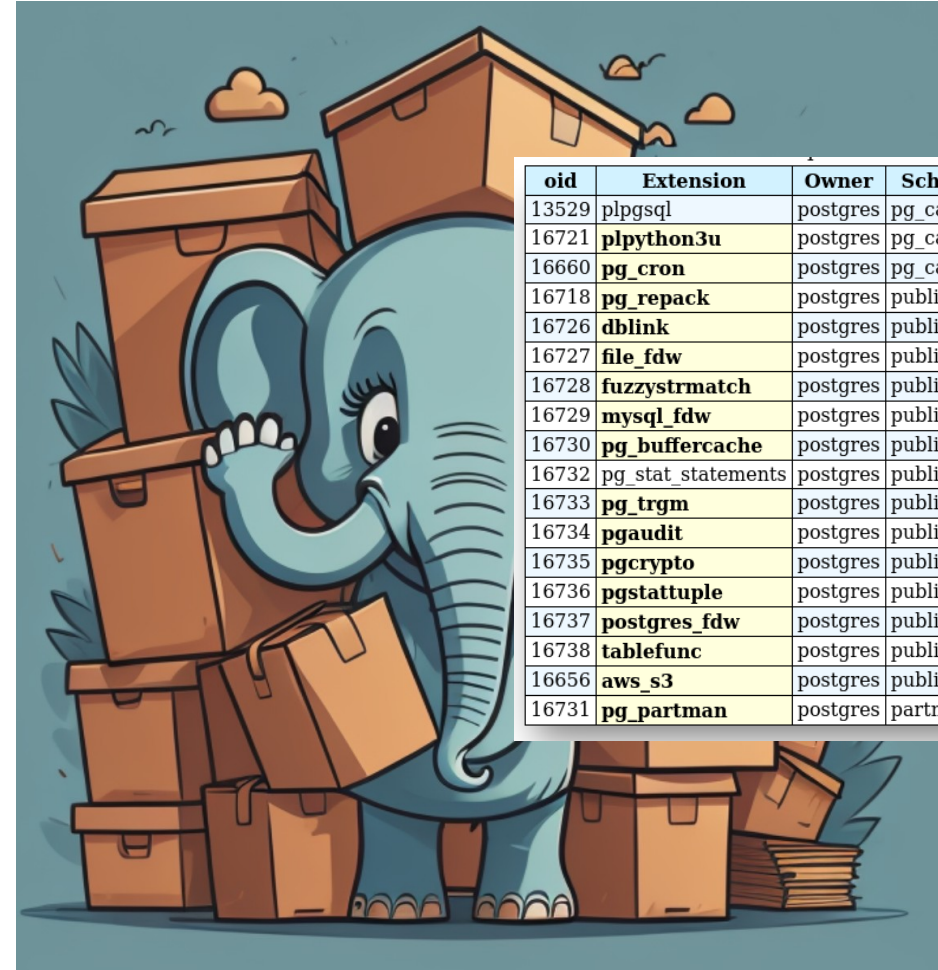
## Problems

- Scalability issues
- Extra network hop and Latencies
- Additional point of failure
- Pooler on the top of pooler
- Timeouts, if there is another pooler which is not releasing the connection.



# Extensions are not cheap

- Extensions uses hooks
- Performance overhead.
- The most frequent cause of crash.



oid	Extension	Owner	Schema	Relocatable?	Version
13529	plpgsql	postgres	pg_catalog	f	1.0
16721	plpython3u	postgres	pg_catalog	f	1.0
16660	pg_cron	postgres	pg_catalog	f	1.6
16718	pg_repack	postgres	public	f	1.4.8
16726	dblink	postgres	public	t	1.2
16727	file_fdw	postgres	public	t	1.0
16728	fuzzystrmatch	postgres	public	t	1.1
16729	mysql_fdw	postgres	public	t	1.2
16730	pg_buffercache	postgres	public	t	1.3
16732	pg_stat_statements	postgres	public	t	1.10
16733	pg_trgm	postgres	public	t	1.6
16734	pgaudit	postgres	public	t	1.7
16735	pgcrypto	postgres	public	t	1.3
16736	pgstattuple	postgres	public	t	1.5
16737	postgres_fdw	postgres	public	t	1.1
16738	tablefunc	postgres	public	t	1.0
16656	aws_s3	postgres	public	t	0.0.1
16731	pg_partman	postgres	partman	f	4.7.3

# Myths of High Availability

- High Availability must be measured
- MTBF - Mean Time Between Failures
- MTTR (mean time to recovery, repair, or resolve)
- Availability 9s. - Big claims
- Multi-Master myths.





# DR (Disaster Recovery) $\neq$ HA (High Availability)

- **High Availability** : A substitute with same configuration on same location without affecting the the application
- **Disaster Recovery** : Moving the Operations to a far away location.
- Vice-versa Another node on same region is not DR as well

# Security

User	Super?	Repl?	Limit	Enc	Active	IdleInTrans	Idle	Total	SSL	NonSSL
XXXXXXXX	t	t								
XXX_XXXXXXXX	t	f								
XXXXXXXXXXXX	t	f								
XXXXXXXXXXXX	t	f								
XXXXXXX	t	f			0	0	3	3	0	3
XXXXXX	t	f								
XXXX_XXXX_XXXX	t	f			5	0	25	30	0	30
XXXXXXX	t	t			0	0	3	3	0	3
XXXXXX	t	f								
XXXXX	t	f								
XXXXXXXXXX	t	f								
XXXXXXXXXX	t	f								
XXXXXXX	t	f			0	0	3	3	0	3
XXXXX	t	f								
-----	f	f			0	0	15	10	0	10

4. High temp on processors : 100.2°C on postgres, 107.4°C on cronos

5. Number of unencrypted connections : **388**

6. There are **14 Super user accounts**, consider this from the security standpoint

7. High memory pressure. Consider increasing RAM and shared\_buffers

# Security

Line	Type	DB	USER	Address	CIDR Mask	DDN/Binary Mask	IP	Method	err
117	local	{all}	{all}					trust	
119	host	{all}	{all}	127.0.0.1	32	255.255.255.255	IPv4	trust	
120	host	{all}	{all}	10.	0	0.0.0.0	IPv4	md5	
121	host	{all}	{all}	10.	24	255.255.255.0	IPv4	md5	
122	host	{all}	{all}	10.	24	255.255.255.0	IPv4	md5	
123	host	{all}	{all}	127	32	255.255.255.255	IPv4	md5	
124	host	{all}	{all}	10.	32	255.255.255.255	IPv4	md5	
125	host	{all}	{all}	10.	32	255.255.255.255	IPv4	md5	
126	host	{all}	{all}	10.	32	255.255.255.255	IPv4	md5	
127	host	{all}	{all}	10.	32	255.255.255.255	IPv4	md5	
128	host	{all}	{all}	10.	32	255.255.255.255	IPv4	md5	
129	host	{all}	{all}	10.	32	255.255.255.255	IPv4	md5	
130	host	{all}	{all}	10.	32	255.255.255.255	IPv4	md5	
131	host	{all}	{all}	10.	32	255.255.255.255	IPv4	md5	
132	host	{all}	{all}	10.	32	255.255.255.255	IPv4	md5	
133	host	{all}	{all}	10.	32	255.255.255.255	IPv4	md5	

# Checkpoint tuning

- Default checkpointing in every 5 minutes
- Causes too many full page writes
- In an HA cluster, Standby will be promoted if primary is lost. So instance recovery time is not affecting anything.

Forced Checkpoint %	avg mins between CP	Avg CP write time (s)	Avg CP sync time (s)	Tot MB Written	MB per CP	Checkpoint MBps	Bgwriter MBps	Backend MBps	Total MBps	New buffers ratio	Clean by checkpoints (%)
6.0	4.57	165.8197	0.0419	59718.26	30.3765	0.1107	0.0071	0.0177	0.1355	2.581	81.7

# Beware of overkill

- Always ask, What are the proven methods and recommendations for production use.
- Develop “Ops” mindset.
- Assess the value for business. Avoid experimenting with production systems

# Tool for help

The screenshot shows the GitHub repository page for `jobinau/pg_gather`. The repository is public and has 102 stars, 16 forks, and 8 unwatchers. It includes a search bar, navigation tabs for Code, Issues (20), Pull requests, Discussions, Actions, Projects (1), Wiki, and Security. The repository description states: "Scan PostgreSQL Instance for potential problems. pg\_gather is a SQL-only scanner leveraging the built-in features of psql". The file list shows a `dev` folder (4 days ago), a `docs` folder (3 months ago), and a `LICENSE.md` file (2 years ago). The `About` section contains tags for `postgres`, `database`, `postgresql`, `performance-analysis`, and `scanner`, along with a `Readme` link.

jobinau / **pg\_gather** Type  to search

**Code** Issues 20 Pull requests Discussions Actions Projects 1 Wiki Security

**pg\_gather** Public Unpin Unwatch 8 Fork 16 Starred 102

main Go to file **<> Code**

**jobinau** Additional analysis query to get pids with Ne... fc914b2 · 4 days ago 396 Commits

dev	Additional analysis query to get pids...	4 days ago
docs	Doc about catalog bloat and fixes	3 months ago
LICENSE.md	Update LICENSE.md	2 years ago

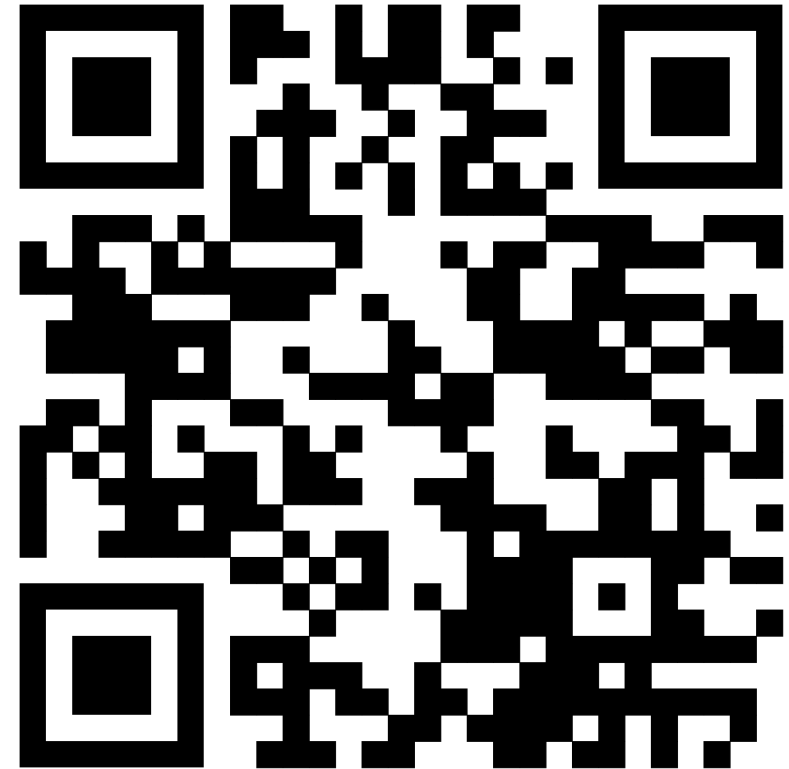
**About**

Scan PostgreSQL Instance for potential problems. pg\_gather is a SQL-only scanner leveraging the built-in features of psql

postgres database postgresql performance-analysis scanner

Readme

# Encryption



[https://github.com/Percona-Lab/pg\\_tde](https://github.com/Percona-Lab/pg_tde)