

Understanding MCP and using AI Agents for PostgreSQL database management

Divya Sharma

PostgreSQL Database Engineer

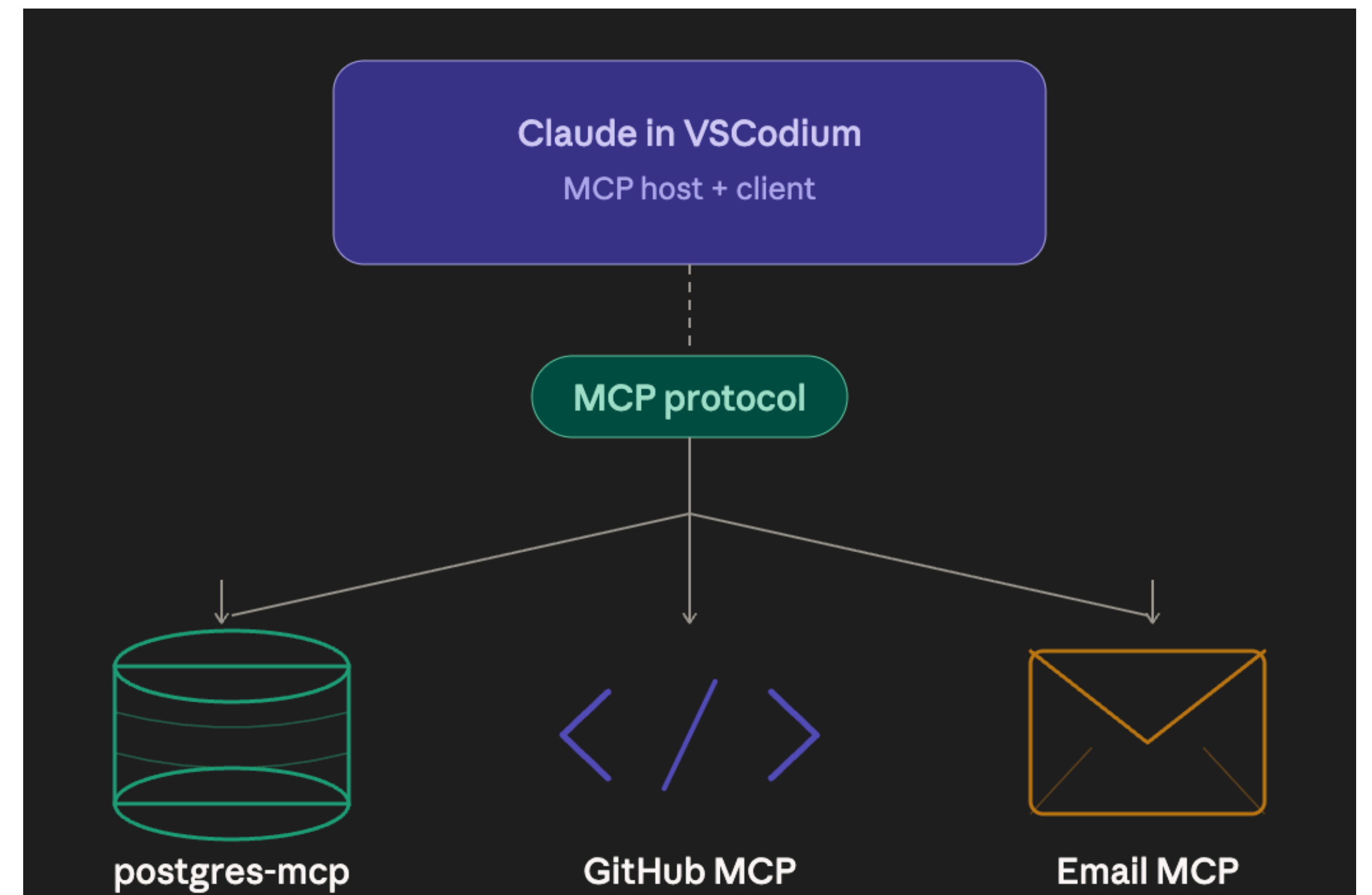
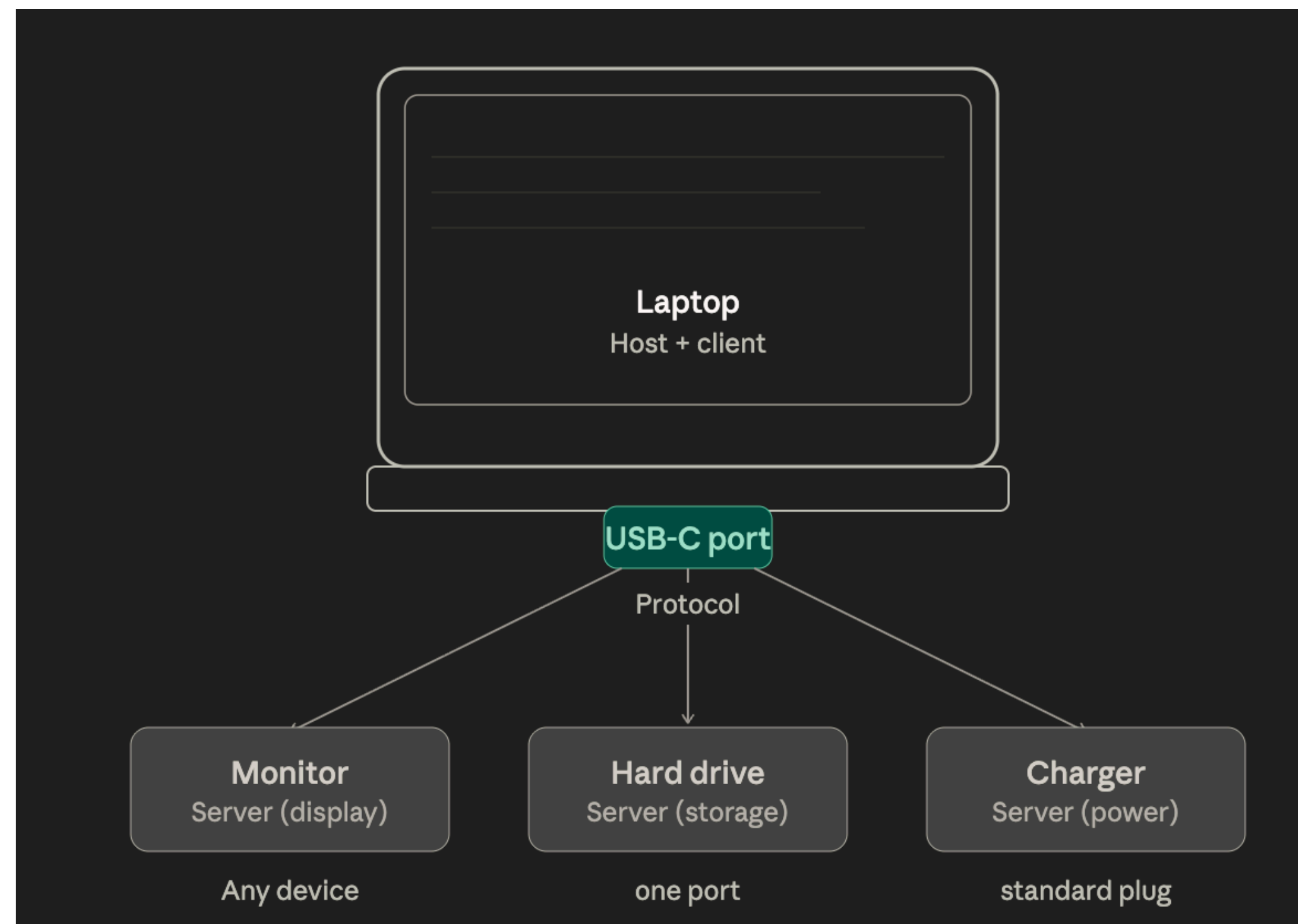
PostgreSQL Conference Germany 2026

What to expect

- We will cover :
 - Basics of MCP and AI agents
 - Using existing MCP servers to help you troubleshoot - demo
 - Frameworks for building an AI agent for postgresql troubleshooting
- We will NOT cover :
 - Claude expert usage
 - Agentic AI advanced concepts
 - How to do everything with AI

AI Agents and MCP

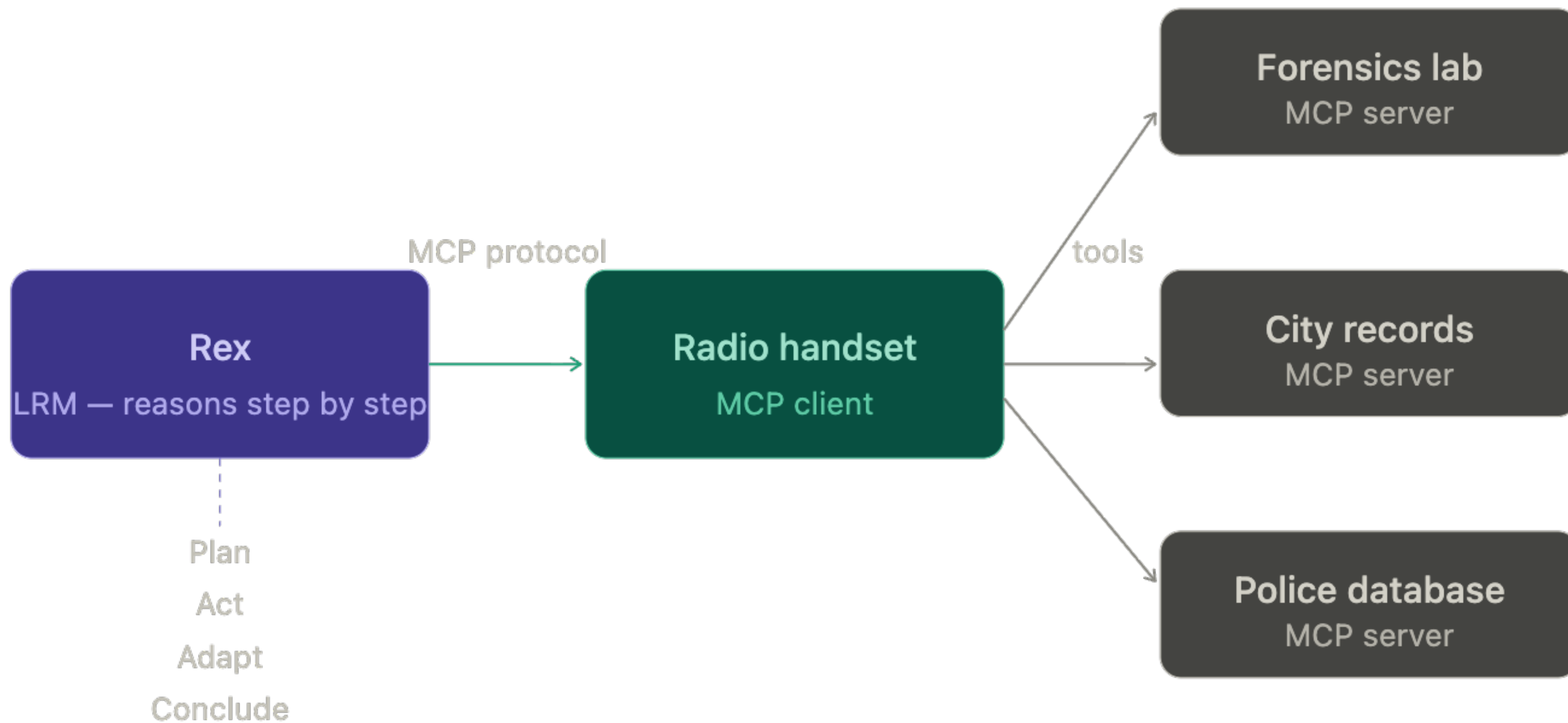
Model Context Protocol



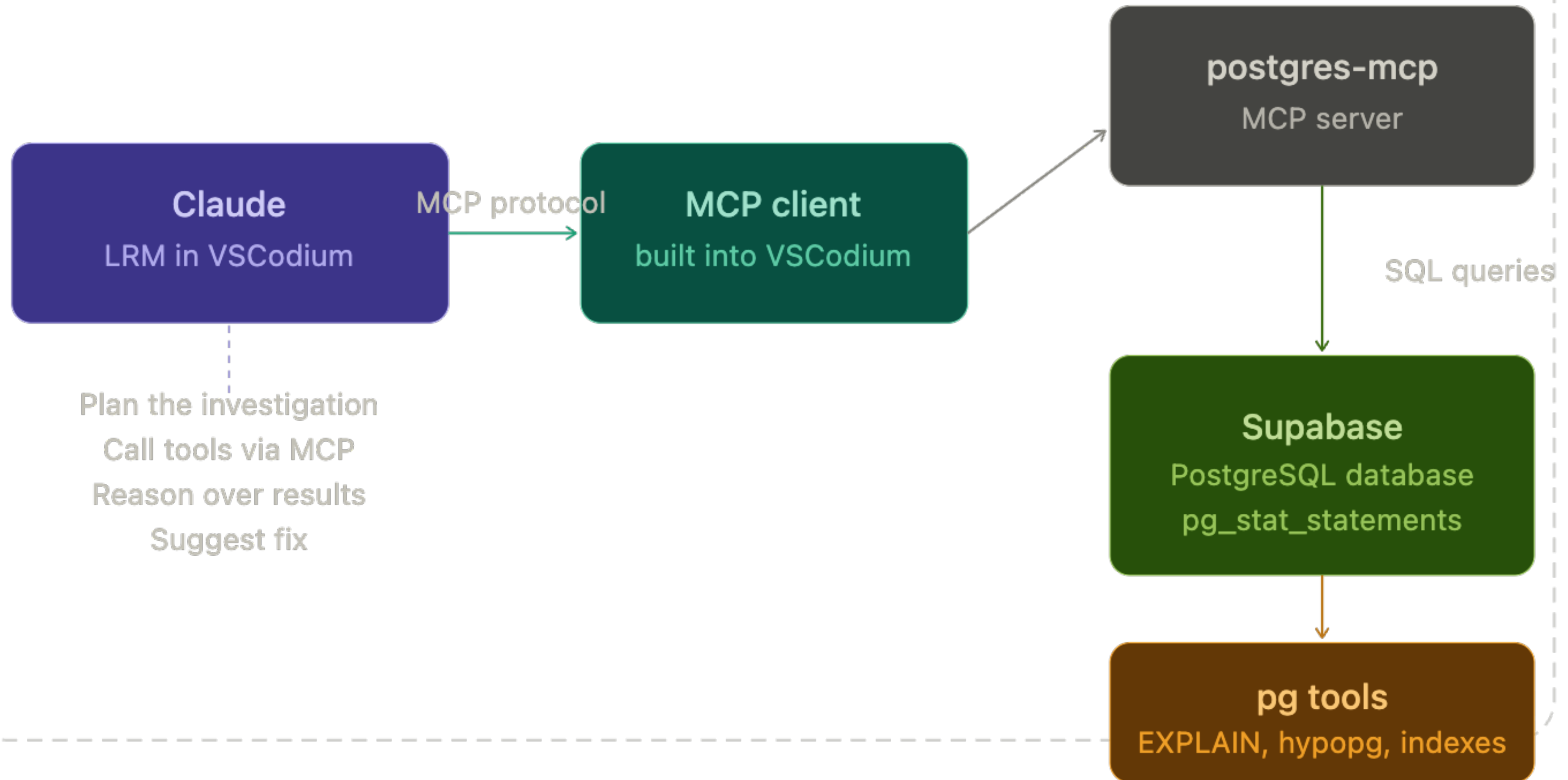
AI agents are models using tools in a loop

```
env = Environment()  
tools = Tools(env)  
system_prompt = "Goals, constraints, and how to act"  
  
while True:  
    action = llm.run(system_prompt + env.state)  
    env.state = tools.run(action)
```

AI agent — Rex runs the whole investigation autonomously



AI agent — autonomously troubleshoots performance issues end-to-end



MCP client/server setup



Postgres MCP Pro

License MIT pypi v0.3.0 Discord 0 online follow @auto_dba contributors 10

A Postgres MCP server with index tuning, explain plans, health checks, and safe sql execution.

[Overview](#) · [Demo](#) · [Quick Start](#) · [Technical Notes](#) · [MCP API](#) · [Related Projects](#) · [FAQ](#)

Overview

Postgres MCP Pro is an open source Model Context Protocol (MCP) server built to support you and your AI agents throughout the entire development process—from initial coding, through testing and deployment, and to production tuning and maintenance.

Postgres MCP Pro does much more than wrap a database connection.

Features include:

- **Database Health** - analyze index health, connection utilization, buffer cache, vacuum health, sequence limits, replication lag, and more.
- **Index Tuning** - explore thousands of possible indexes to find the best solution for your workload, using industrial-strength algorithms.
- **Query Plans** - validate and optimize performance by reviewing EXPLAIN plans and simulating the impact of hypothetical indexes.
- **Schema Intelligence** - context-aware SQL generation based on detailed understanding of the database schema.
- **Safe SQL Execution** - configurable access control, including support for read-only mode

Slow query analysis and troubleshooting



SQL Editor

Search queries... +

> SHARED

> FAVORITES

> PRIVATE (1)

SQL Sample eCommerce Schema ...

> COMMUNITY

Templates

Quickstarts

SQL Sample eCommerce Schema With Seed Data

```

40 SELECT
45   NOW() - (random() * interval '180 days')
46 FROM generate_series(1, 1000000);
47
48 EXPLAIN ANALYZE SELECT o.id, o.total, o.created_at, p.name
49 FROM orders o
50 JOIN customers c ON o.customer_id = c.id
51 JOIN products p ON o.product_id = p.id
52 WHERE c.name = 'Customer 111'
53        AND o.status = 'completed'
54        AND o.created_at >= NOW() - INTERVAL '30 days'
55 ORDER BY o.created_at DESC;
56
57 select count(*) from orders;
58
59 ALTER TABLE orders SET (autovacuum_enabled = true);
60
61 UPDATE orders
62 SET total = total + 1

```

Results Explain Chart Export Source Primary Database Role postgres Run selected

Query Execution Plan / Total time 161.28ms

Explain with AI Raw

> LIMIT (cost 24363.4, estimated 1 row)	6.78ms / 3 rows
> GATHER MERGE (cost 24363.4, estimated 1 row)	6.78ms / 3 rows
> SORT (cost 23363.2, estimated 1 row)	0.002ms / 2 rows
> NESTED LOOP (cost 23363.2, estimated 1 row)	85.23ms / 2 rows
> HASH JOIN (cost 23363.1, estimated 1 row)	85.22ms / 2 rows
> PARALLEL SEQ SCAN (cost 22259.8, estimated 60,675 rows)	142.75ms / 400,000 → -348,298 → 51,702 rows

View running queries

PI-OPENCLAW
OUTLINE
TIMELINE

Users > amandhingra > .claude.json > { } projects > { } /Users/amandhingra > { } mcpServers > { } postgres > [] args

36 "cachedDvnmaticConfias": {
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Cline - amandhingra

● Database Summary

Metric	Count	Total Size
Tables	38	148 MB
Indexes	119	45 MB

Top Tables by Size

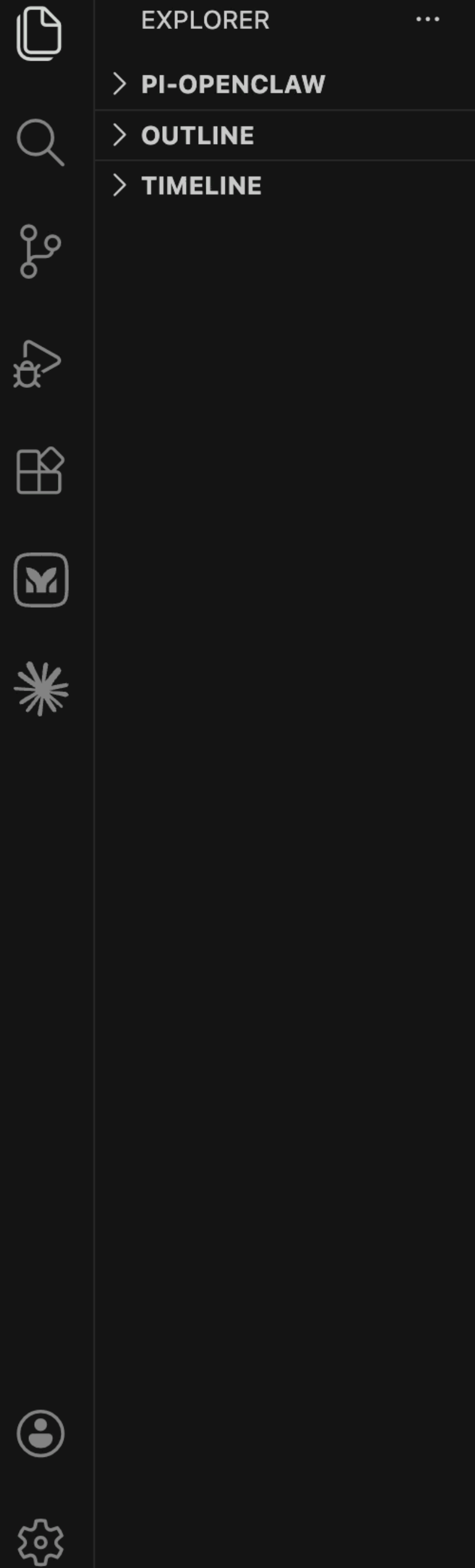
Schema	Table	Total Size	Table Size
public	orders	143 MB	100 MB
public	customers	3704 kB	2552 kB
auth	users	128 kB	0 bytes
auth	one_time_tokens	88 kB	0 bytes
auth	refresh_tokens	64 kB	0 bytes
public	products	32 kB	8192 bytes

Top Indexes by Size

Schema	Index	Table	Size
public	orders_pkey	orders	43 MB
public	customers_pkey	customers	1112 kB
auth	one_time_tokens_relates_to_hash_idx	one_time_tokens	32 kB
auth	one_time_tokens_token_hash_hash_idx	one_time_tokens	32 kB

The public.orders table dominates the database at 143 MB total (including indexes), with its primary key index alone taking 43 MB. The public.customers table is the second largest at 3.7 MB.

> This query is taking close to 3 seconds, suggest optimizatin



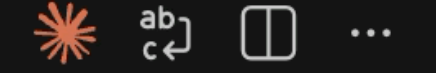
EXPLORER ...
> PI-OPENCLAW
> OUTLINE
> TIMELINE

{} cline_mcp_settings.json

Welcome

Settings

{}.claude.json X

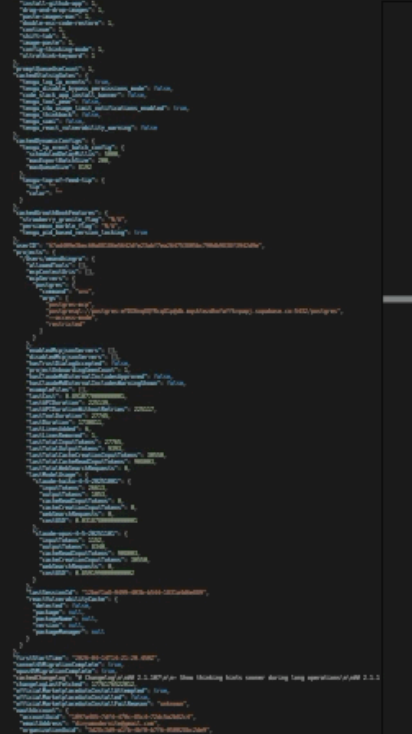


Users > amandhingra > {}.claude.json > {} projects > {} /Users/amandhingra > {} mcpServers > {} postgres > [] args > abc 3

```

54     "projects": {
55         "/Users/amandhingra": {
56             "allowedTools": [],
57             "mcpContextUris": [],
58             "mcpServers": {
59                 "postgres": {
60                     "command": "uvx",
61                     "args": [
62                         "postgres-mcp",
63                         "postgresql://postgres:eYDX6nq8QYRcqU1p@db.mqsktezdhofaffkvpapj.supabase.co:5432/postgres",
64                         "--access-mode".

```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Cline - amandhingra [warning icon] [plus icon] [minus icon] [refresh icon] [close icon]

```

"result": [
  {
    ... +9 lines (ctrl+o to expand)

```

• The hypothetical indexes show a massive improvement:

Performance Comparison

Metric	Without Indexes	With Indexes
Estimated Cost	24,363	6.01
Improvement	-	~4,000x better

Execution Plan With Indexes

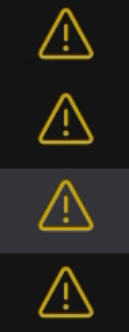
- Sort (Cost: 6.01)
 - Nested Loop (Cost: 0.24..6.00)
 - Nested Loop (Cost: 0.10..5.68)
 - Index Scan on customers ← Uses idx_customers_name
 - Index Scan on orders ← Uses idx_orders_customer_status_created
 - Index Scan on products

All sequential scans are eliminated and replaced with efficient index scans.

Create These Indexes

Run this in your database to make it permanent:

```
CREATE INDEX idx_customers_name ON customers(name);
```





SQL Editor

Search queries... +

> SHARED

> FAVORITES

> PRIVATE (1)

SQL Sample eCommerce Schema ...

> COMMUNITY

Templates

Quickstarts

SQL Sample eCommerce Schema With Seed Data +

```

64
65 DELETE FROM orders
66 WHERE id % 5 = 0;
67
68 UPDATE orders
69 SET total = total + 2
70 WHERE id % 3 = 0;
71
72 CREATE EXTENSION hypopg;
73
74 CREATE INDEX idx_customers_name ON customers(name);
75
76 CREATE INDEX idx_orders_customer_status_created
77 ON orders(customer_id, status, created_at DESC);

```

Results Explain Chart Export



Source

Primary Database

Role postgres

Run selected

Running...

Bloat detection and vacuum tuning

EXPLORER ...

- > PI-OPENCLAW
- > OUTLINE
- > TIMELINE

Users > amandhingra > { } .claude.json > { } projects > { } /Users/amandhingra > { } mcpServers > { } postgres > [] args > abc 3

```
54 "projects": {
55   "/Users/amandhingra": {
56     "allowedTools": [],
```

```
-- Safe to drop these redundant indexes:
DROP INDEX auth.custom_oauth_providers_identifier_idx;
DROP INDEX auth.mfa_factors_user_id_idx;
DROP INDEX auth.refresh_tokens_instance_id_idx;
DROP INDEX auth.sessions_user_id_idx;
```

⚠ Unused Indexes (50+)

Most are in auth schema with 0 scans. The largest unused index is idx_orders_customer_status_created at 62 MB (newly created, expected).

Summary

Category	Status
Connections	✓ Healthy (14 active, 0 idle)
Buffer Cache	✓ Excellent (99.9% hit rate)
Vacuum	✓ No wraparound risk
Sequences	✓ Healthy
Constraints	✓ Valid
Replication	✓ Primary, no issues
Index Bloat	✓ None
Duplicate Indexes	⚠ 4 redundant
Unused Indexes	⚠ 50+ rarely used

Overall: Database is healthy. The only action items are cleaning up duplicate/unused indexes to save space and reduce write overhead.

> check for bloat in the tables in database postgres. check if there ar



EXPLORER

- > PI-OPENCLAW
- > OUTLINE
- > TIMELINE

Users > amandhingra > .claude.json > {} projects > {} /Users/amandhingra > {} mcpServers > {} postgres > [] args > abc 3

```
54 "projects": {
55   "/Users/amandhingra": {
56     "allowedTools": [],
```

```

schemaname NOT IN ('pg_catalog', 'information_schema')\nORDER BY n_dead_tup DESC\nLIMIT 20;")
L {
  "result": [
    {
      ... +60 lines (ctrl+o to expand)
  ]
}

• postgres - execute_sql (MCP)(sql: "SELECT \n  name, \n  setting, \n  unit,\n  short_desc\nFROM pg_settings \nWHERE name
LIKE '%vacuum%' OR name LIKE '%autovacuum%'\nORDER BY name;")
L {
  "result": [
    {
      ... +41 lines (ctrl+o to expand)
  ]
}

• postgres - execute_sql (MCP)(sql: "SELECT \n  c.relname as table_name,\n  c.relnamespace::regnamespace as schema,\n
pg_size_pretty(pg_total_relation_size(c.oid)) as total_size,\n
pg_size_pretty(pg_relation_size(c.oid)) as table_size,\n  c.reltuples::bigint as estimated_rows,\n
CASE \n    WHEN c.reltuples > 0 THEN \n        round((pg_relation_size(c.oid) /
NULLIF(c.reltuples, 0))::numeric, 2)\n    ELSE 0 \n  END as bytes_per_row\nFROM pg_class
c\nJOIN pg_namespace n ON n.oid = c.relnamespace\nWHERE c.relkind = 'r' \n  AND n.nspname NOT IN
('pg_catalog', 'information_schema')\nORDER BY pg_total_relation_size(c.oid) DESC\nLIMIT 15;")
L {
  "result": [
    {
      ... +26 lines (ctrl+o to expand)
  ]
}

• postgres - execute_sql (MCP)(sql: "SELECT \n  relname as table_name,\n  reloptions\nFROM pg_class\nWHERE reloptions IS NOT
NULL \n  AND relkind = 'r';")
```



Tool use

```
postgres - execute_sql(sql: "SELECT \n  relname as table_name,\n  reloptions\nFROM pg_class\nWHERE reloptions IS NOT
NULL \n  AND relkind = 'r';") (MCP)
Execute a read-only SQL query
```

Do you want to proceed?

- > 1. Yes
- 2. Yes, and don't ask again for postgres - execute_sql commands in /Users/amandhingra
- 3. No, and tell Claude what to do differently (esc)

EXPLORER ...

- > PI-OPENCLAW
- > OUTLINE
- > TIMELINE

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Cline - amandhingra

{ } cline_mcp_settings.json

Welcome

Settings

{ } .claude.json X

Users > amandhingra > { } .claude.json > { } projects > { } /Users/amandhingra > { } mcpServers > { } postgres > [] args > abc 3

```

54     "projects": {
55         "/Users/amandhingra": {
56             "allowedTools": [],

```

```

ALTER TABLE public.orders SET (autovacuum_enabled = true);

2. Run manual VACUUM to clean up immediately

-- Standard vacuum (reclaims space for reuse, doesn't shrink file)
VACUUM VERBOSE public.orders;

-- Or VACUUM FULL to reclaim disk space (locks table!)
VACUUM FULL VERBOSE public.orders;

3. For high-write tables, consider aggressive settings

ALTER TABLE public.orders SET (
    autovacuum_enabled = true,
    autovacuum_vacuum_scale_factor = 0.05, -- vacuum at 5% dead tuples
    autovacuum_vacuum_threshold = 1000,
    autovacuum_analyze_scale_factor = 0.02
);

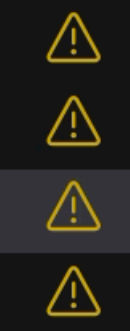
```

Summary

Issue	Severity	Fix
Autovacuum disabled on orders	● Critical	Re-enable autovacuum
71% dead tuples (2M rows)	● Critical	Run VACUUM or VACUUM FULL
~106 MB wasted space	⚠ High	VACUUM FULL to reclaim

Would you like me to provide more details on any of these fixes?

> I have run vacuum on orders table. re-check bl



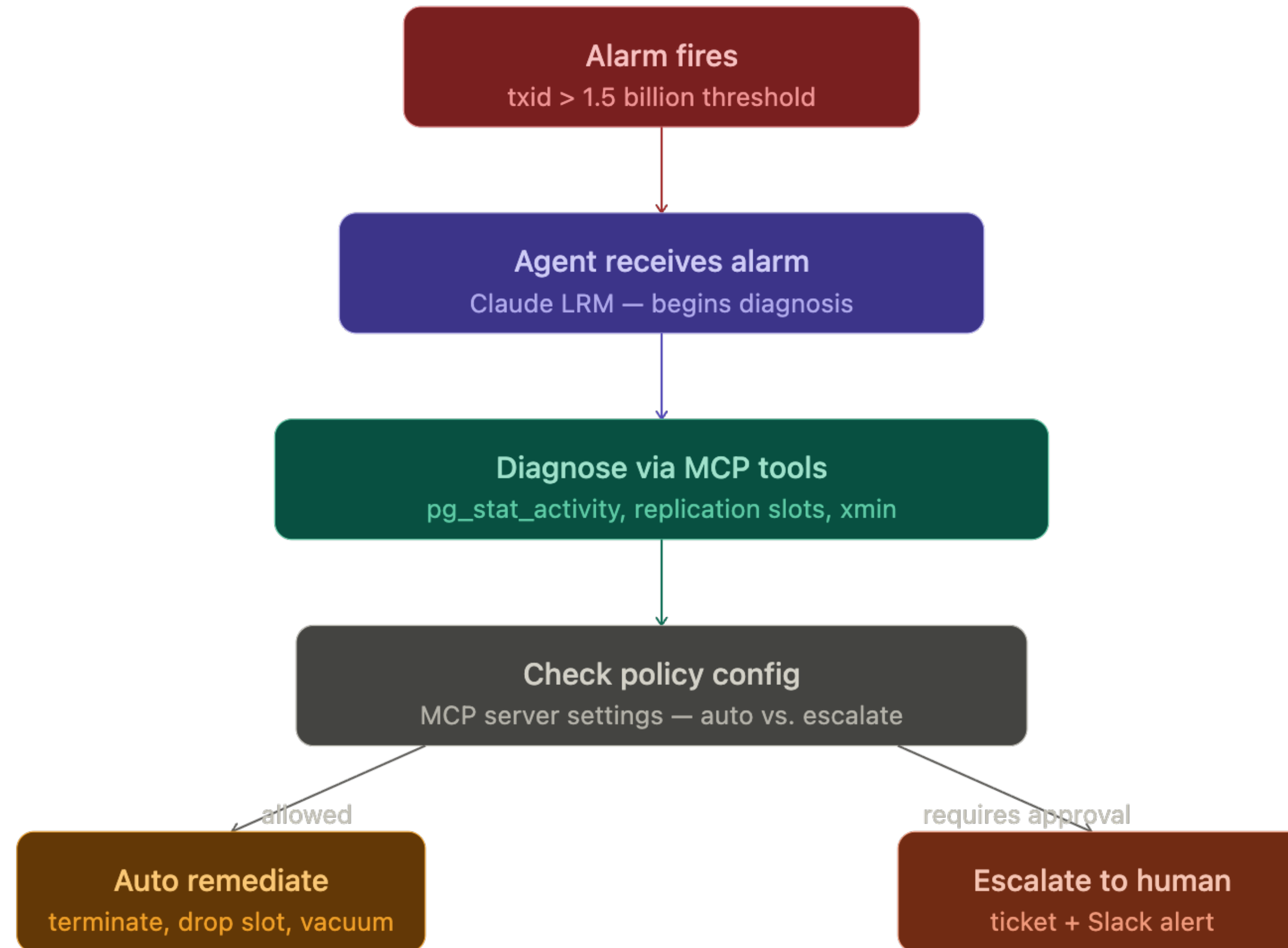
Component alternatives

Component	What we used	Alternatives
AI reasoning model (LRM)	Claude Sonnet 4.5 / Opus 4	OpenAI o3, o4-mini · Google Gemini 2.5 Pro · DeepSeek R1 · Grok 3
Agent IDE / coding environment	VSCodium + Claude Code extension	Cursor · Windsurf · GitHub Copilot (VS Code) · Zed · JetBrains AI
Agent runtime / framework	Claude Code (CLI)	LangChain · LangGraph · CrewAI · AutoGen · Pydantic AI · Semantic Kernel
MCP server — PostgreSQL	crystaldba/postgres-mcp	@modelcontextprotocol/server-postgres · Supabase MCP · pgmcp (Rust)
MCP server — observability	postgres-mcp + pg_stat_statements	Datadog MCP · Grafana MCP · Prometheus MCP · Honeycomb MCP
Database	Supabase (PostgreSQL)	Any PostgreSQL installation · AWS RDS PostgreSQL · AlloyDB · Azure Database for PostgreSQL
Secret / config management	MCP server settings.json	AWS Secrets Manager · HashiCorp Vault · Doppler · 1Password Secrets · Infisical

AI agent use case for production

AI agent use case for production

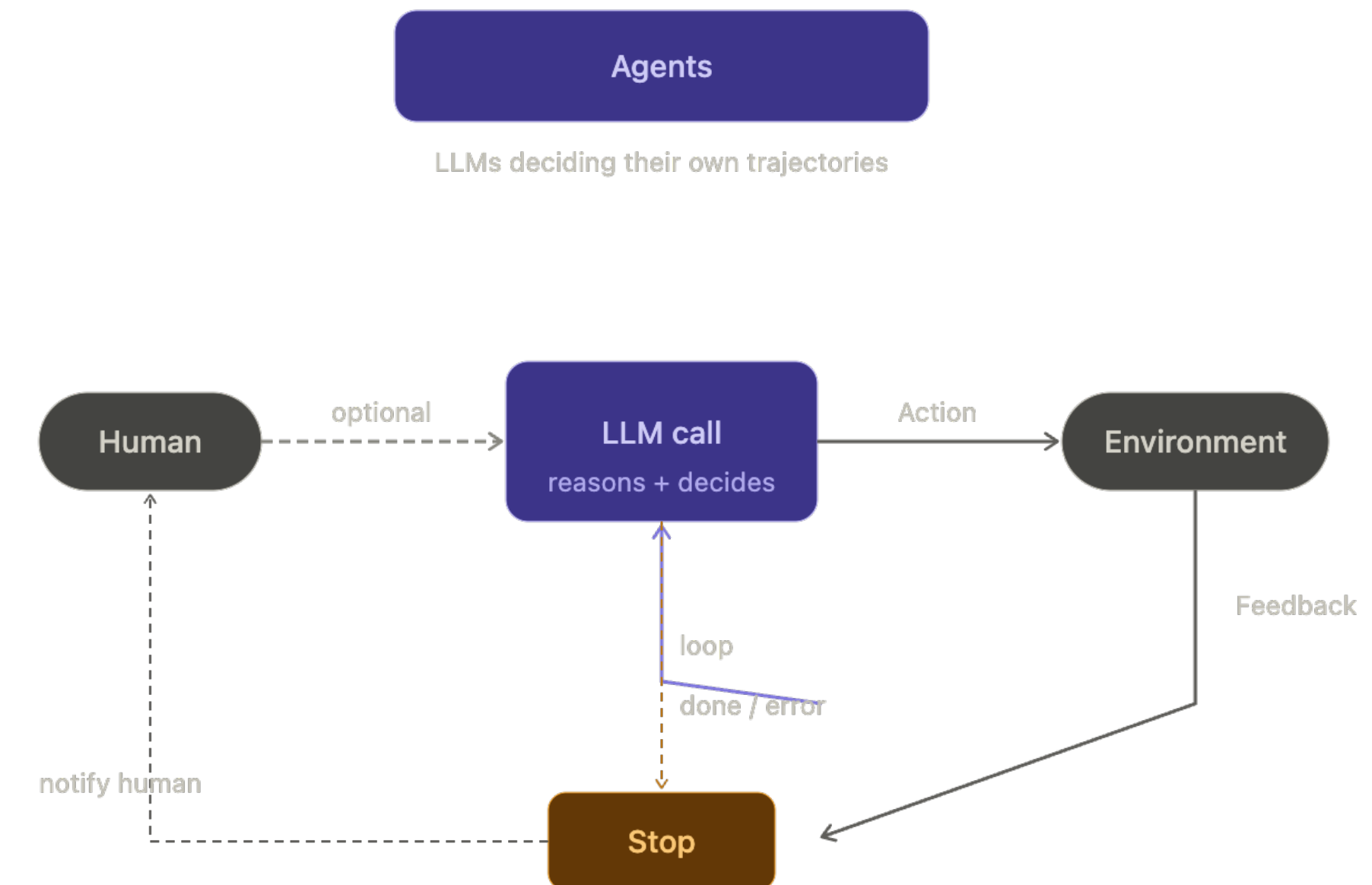
Avoiding transactionID wraparounds using a custom AI agent



Avoiding transactionID wraparounds using a custom AI agent

1. Agent trigger - receiving the alarm - datadog, cloudwatch etc.

```
python
# Entry point - called by your alerting system
def on_alarm(event: dict):
    agent.run(
        goal="Diagnose and remediate wraparound risk",
        context=event,          # txid, db name, severity
        policy=load_config()   # user-set thresholds and allowed actions
    )
```



Avoiding transactionID wraparounds using a custom AI agent

2. MCP server config - the policy layer

```
json
{
  "thresholds": {
    "idle_session_mins": 30,
    "xmin_age_warning": 500000000,
    "slot_lag_bytes": 1073741824
  },
  "auto_allowed": [
    "terminate_idle_session",
    "run_vacuum_analyze"
  ],
  "escalate_only": [
    "cancel_active_query",
    "cancel_replica_query"
  ],
  "human_only": [
    "drop_replication_slot",
    "alter_system"
  ],
  "notify": {
    "slack_channel": "#db-alerts",
    "ticket_system": "jira",
    "oncall_pagerduty": true
  }
}
```

Avoiding transactionID wraparounds using a custom AI agent

3. Diagnostic tools the agent uses via MCP servers

```
sql

-- Tool: find_wraparound_causes
SELECT pid, username, state, wait_event_type,
       age(backend_xmin) as xmin_age,
       now() - xact_start as txn_duration,
       query
FROM pg_stat_activity
WHERE backend_xmin IS NOT NULL
ORDER BY age(backend_xmin) DESC;

-- Tool: check_replication_slots
SELECT slot_name, active, pg_size_pretty(
       pg_wal_lsn_diff(pg_current_wal_lsn(), restart_lsn)
) AS lag
FROM pg_replication_slots;

-- Tool: get_database_age
SELECT datname, age(datfrozenxid) as txid_age,
       2147483647 - age(datfrozenxid) as txids_remaining
FROM pg_database ORDER BY txid_age DESC;
```

Avoiding transactionID wraparounds using a custom AI agent

4. Remediation action

Action	SQL	Safety check
terminate_idle_session	<code>SELECT pg_terminate_backend(\$pid)</code>	Only if state = 'idle in transaction' AND duration > config threshold
run_vacuum_analyze	<code>VACUUM ANALYZE \$table</code>	Always safe, can run concurrently
cancel_active_query	<code>SELECT pg_cancel_backend(\$pid)</code>	Requires escalate approval
drop_replication_slot	<code>SELECT pg_drop_replication_slot (\$name)</code>	Human-only, never automated

Avoiding transactionID wraparounds using a custom AI agent

5. Production safety guardrails :
 - a. Dry-run mode
 - b. Action log
 - c. Blast radius limit
 - d. Post-action verification
 - e. Circuit breaker

Avoiding transactionID wraparounds using a custom AI agent

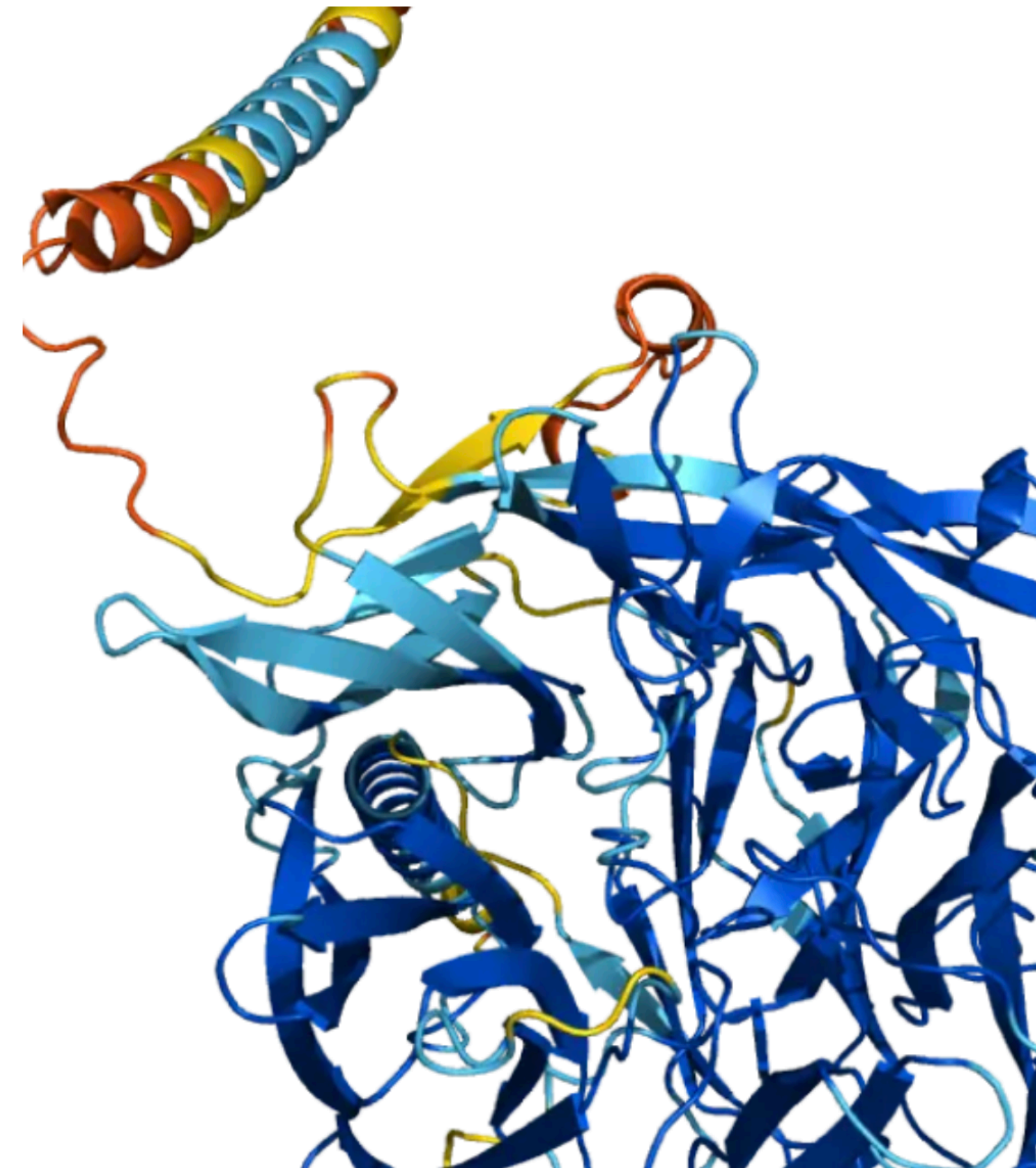
6. Notification and ticketing

```
[WRAPAROUND AGENT] Run #42 - 2026-04-15 14:32 UTC  
Database: prod-main | txid age: 1,623,441,201  
Cause identified: 2 idle-in-transaction sessions (PIDs 18234, 18291, 47min, 52min)  
Action taken: terminated both sessions (auto-allowed per config)  
Result: age reducing - current: 1,622,980,003 (-461,198)  
No escalation required.
```

**Lastly, something to REALLY
think about**

AlphaFold is an AI system developed by **Google DeepMind** that predicts a protein's 3D structure from its amino acid sequence. It regularly achieves accuracy competitive with experiment.

Google DeepMind and EMBL's European Bioinformatics Institute ([EMBL-EBI](#)) have partnered to create AlphaFold DB to make these predictions freely available to the scientific community. The latest database release contains over 200 million entries, providing broad coverage of [UniProt](#) (the standard repository of protein sequences and annotations). We provide individual [downloads](#) for the human proteome and for the proteomes of 47 other key organisms important in research and global health. We also provide a download for the manually curated subset of UniProt ([Swiss-Prot](#)).



Q8I3H7: May protect the malaria parasite against attack by the immune system. Mean pLDDT 85.57.

[View protein](#)

Source : <https://alphafold.com/>



CLIMATE CHANGE AND ENERGY

We did the math on AI's energy footprint. Here's the story you haven't heard.

The emissions from individual AI text, image, and video queries seem small—until you add up what the industry isn't tracking and consider where it's heading next.

*“ by **2028** more than half of the electricity going to data centers will be used for AI. At that point, AI alone could consume **as much electricity annually as 22% of all US households.**”*

Source : <https://www.technologyreview.com/2025/05/20/1116327/ai-energy-usage-climate-footprint-big-tech/>

Drive carefully

and form your prompts even more carefully!

Thank you!

Divya Sharma

<https://www.linkedin.com/in/divyasharma95/>



Your feedback is very valuable to me