xata

Beyond monitoring

How we built an open-source, self-healing Postgres agent

Alex Francoeur

October 1, 2025

Beyond monitoring

How ~~we built~~ and why we’re building an open-source, self-healing Postgres agent

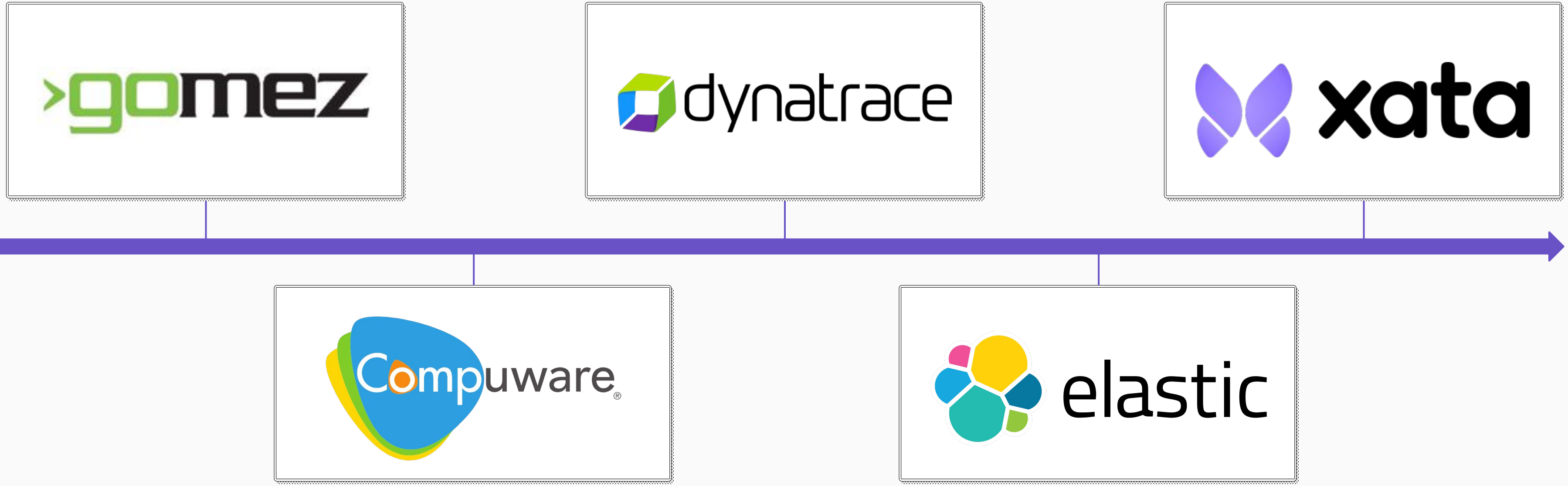
Alex Francoeur

October 1, 2025

Me, on a slide

- Product @ Xata
- Greater Boston
- Happily married with 2 kids
- Proud father of 6 chickens and 1 dog
- Soccer / football player, runner, snowboarder and year-round-coach
- Putzer-arounder, tinkerer, hacker (well, probably more of a vibe coder now)

Professional me, on a slide



xata

Postgres at scale

“Postgres at scale” for Xata means more than just handling large data or compute. It’s about scaling team productivity and operations.

xataio/pgroll

xataio/pgstream

xataio/agent

Foundations

1960s-1980s

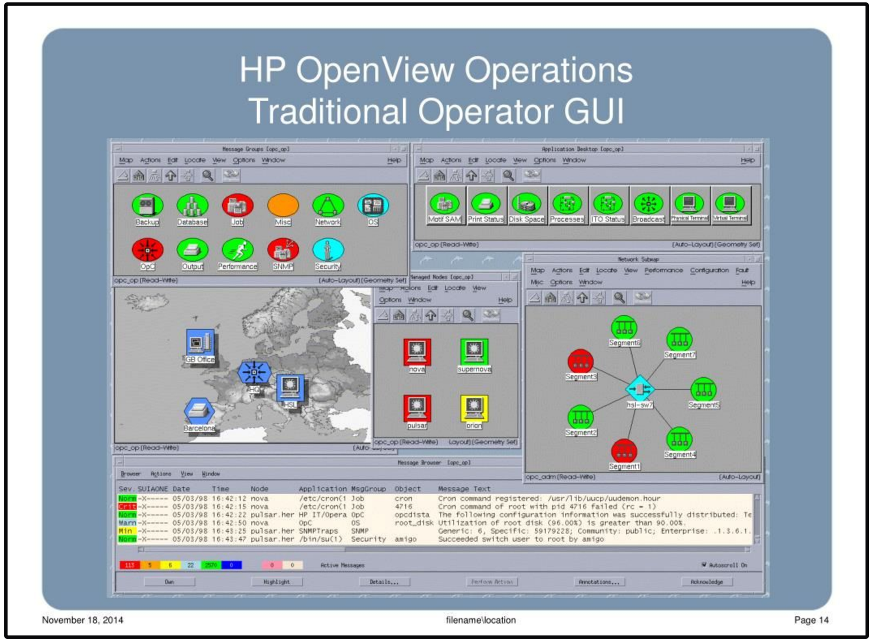
Monitoring relied on console lights, SMF records, and simple SNMP alerts.

DB administrators focused on mainframe RDBMS, backups, and console-based operations.

Centralization
1990s-Mid 2000s

System management platforms (HP OpenView, IBM Tivoli, BMC, CA) centralized infrastructure monitoring.

DBAs managed client-server databases with replication, clustering, and tuning.



HP OpenView Operations
Traditional Operator GUI

Message Browser [opc_op]

```
Sev. SUIAONE Date     Time     Node       Application MsgGroup Object    Message Text
Norm -X----- 05/03/98 16:42:12 nova       /etc/cron(1 Job      cron      Cron command registered: /usr/lib/uucp/uudemon.hour
Crit -X----- 05/03/98 16:42:15 nova       /etc/cron(1 Job      4716      Cron command of root with pid 4716 failed (rc = 1)
Norm -X----- 05/03/98 16:42:22 pulsar.her HP IT/Opera OpC      opcdista  The following configuration information was successfully distributed: Te
Warn -X----- 05/03/98 16:42:50 nova       OpC         OS       root_disk Utilization of root disk (96.00%) is greater than 90.00%.
Min  -X----- 05/03/98 16:43:25 pulsar.her SNMPTraps   SNMP               Generic: 6, Specific: 59179228; Community: public; Enterprise: .1.3.6.1.
Norm -X----- 05/03/98 16:43:47 pulsar.her /bin/su(1)  Security amigo     Succeeded switch user to root by amigo
```

November 18, 2014 filename\location Page 14

Logs & Apps
2000s

Splunk popularized log search; early APM tools tied performance to end-user experience

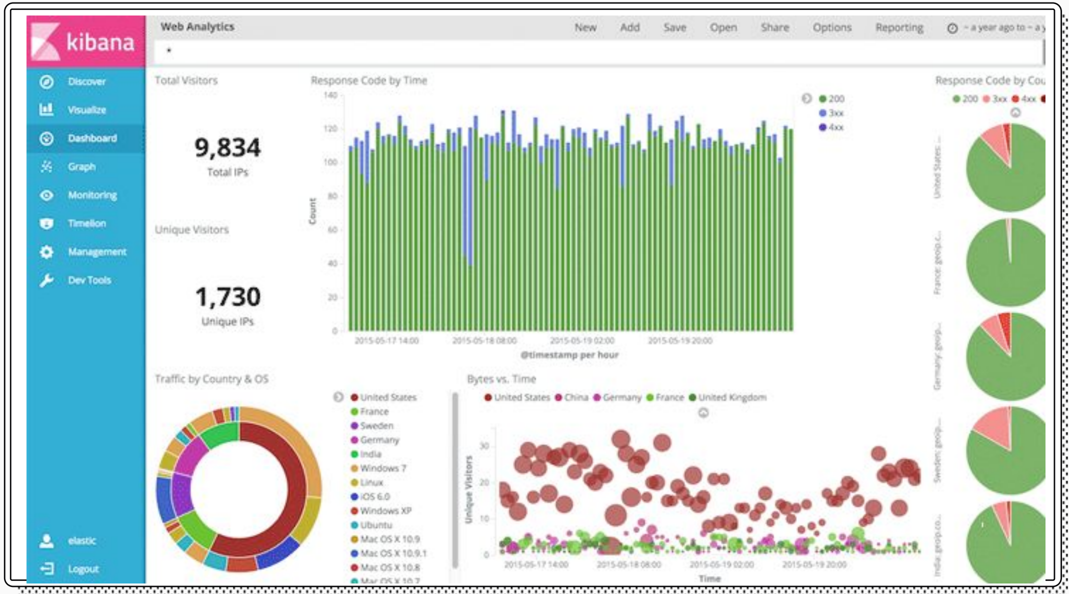
Ops DBAs supported web-scale apps, ensuring uptime and managing complex HA setups.

Cloud and DevOps

2010s

ELK, Prometheus, and SaaS platforms like Datadog enabled developer-centric, cloud-native monitoring.

Traditional DBA roles blurred into SRE and DevOps, with automation and CI/CD at the core.



Scale & Specialization 2018+

Loki and Tempo introduced efficient storage for logs and traces to manage telemetry growth.

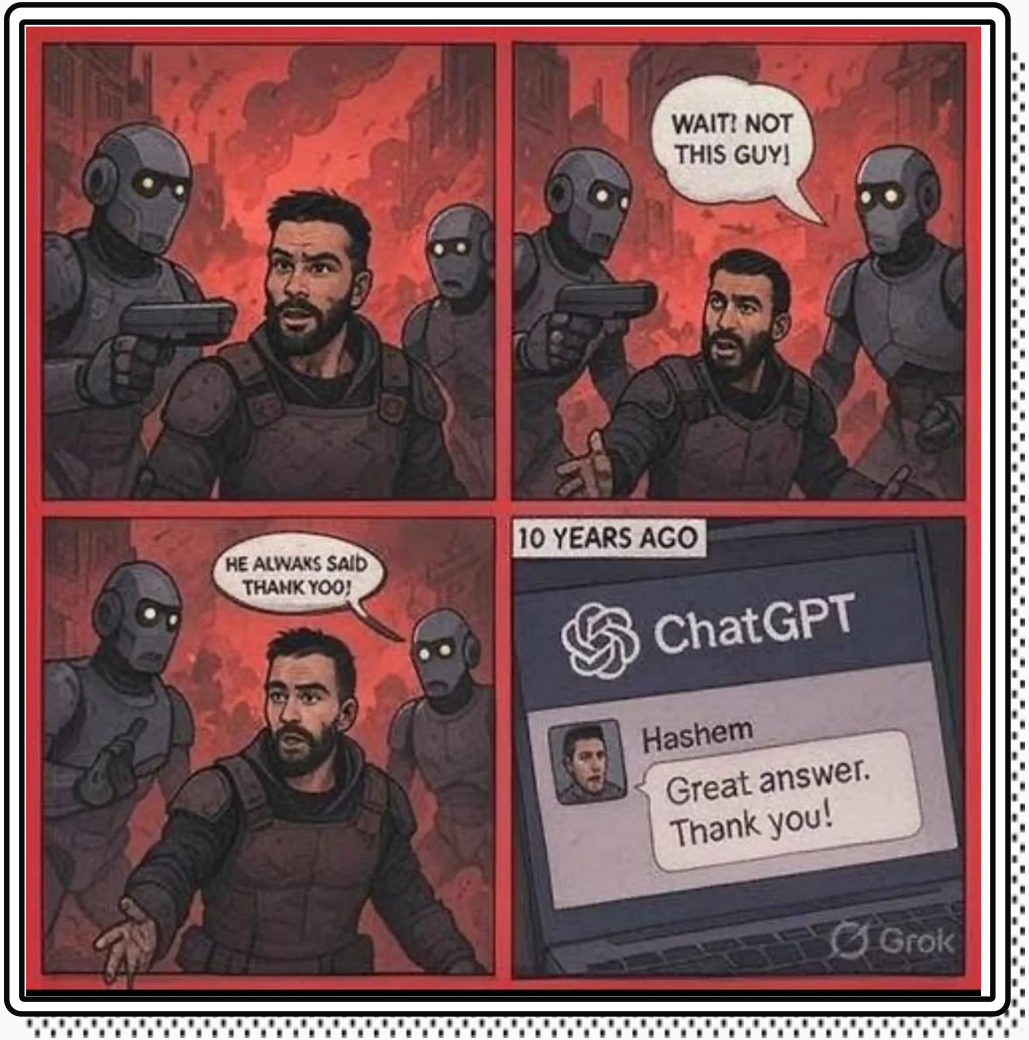
Platform and DataOps roles emerged, integrating database management with pipelines and analytics.

Intelligence Era

2020+

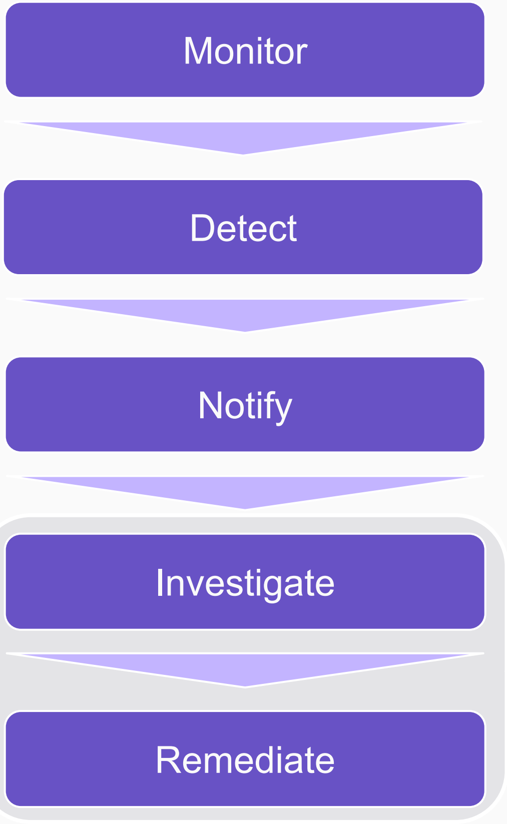
OpenTelemetry, eBPF, and ML brought predictive insights and auto-remediation into observability

DevSecOps practices embedded security and compliance into database workflows.



Hashem

Great answer. Thank you!



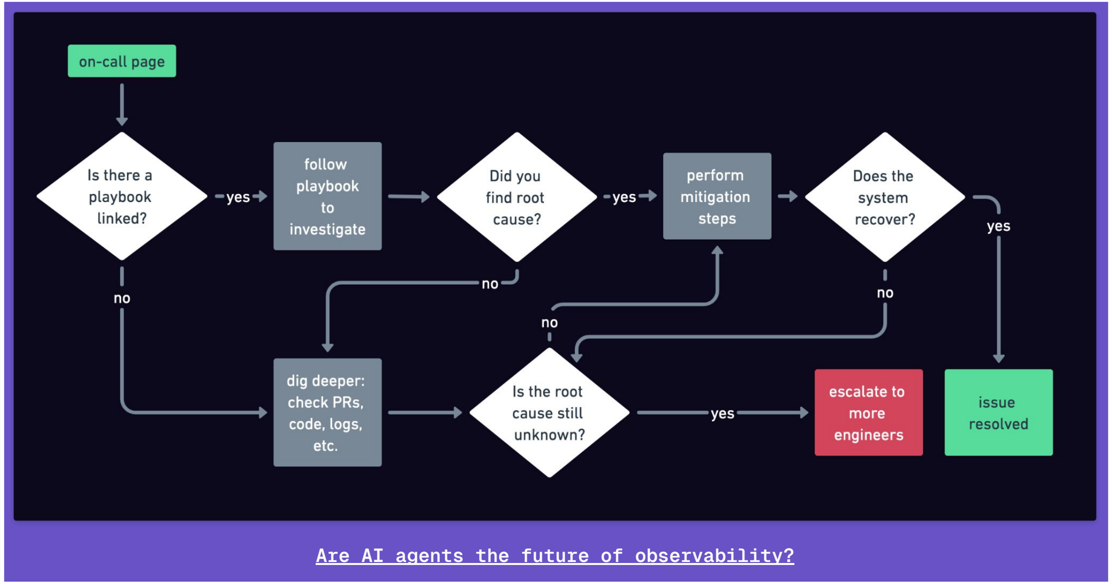
Monitor

Detect

Notify

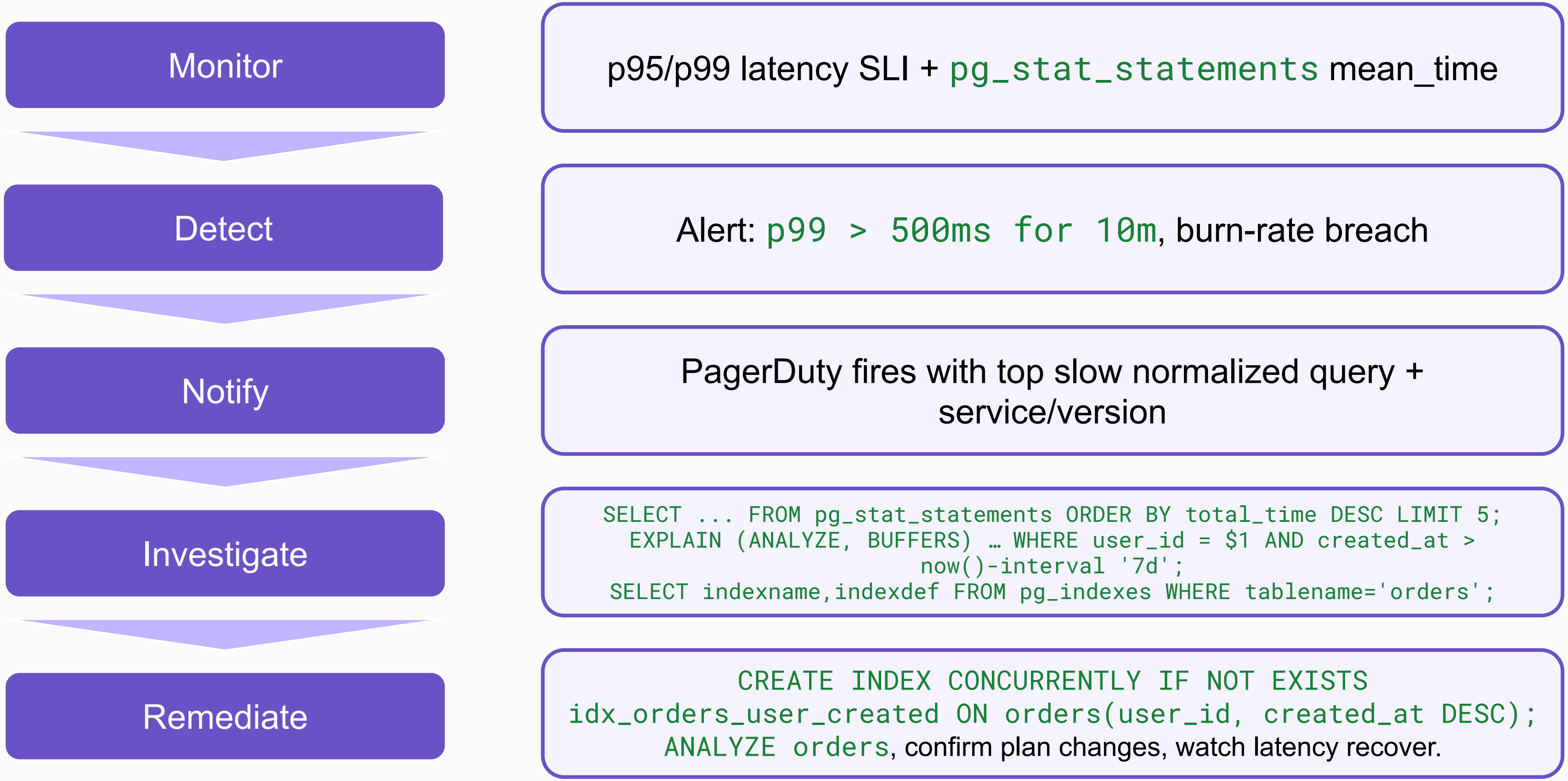
Investigate

Remediate



on-call page
Is there a playbook linked?
yes
no
follow playbook to investigate
Did you find root cause?
yes
no
perform mitigation steps
Does the system recover?
yes
no
dig deeper: check PRs, code, logs, etc.
no
Is the root cause still unknown?
yes
escalate to more engineers
issue resolved

Are AI agents the future of observability?



Monitor

p95/p99 latency SLI + `pg_stat_statements` mean_time

Detect

Alert: `p99 > 500ms for 10m`, burn-rate breach

Notify

PagerDuty fires with top slow normalized query + service/version

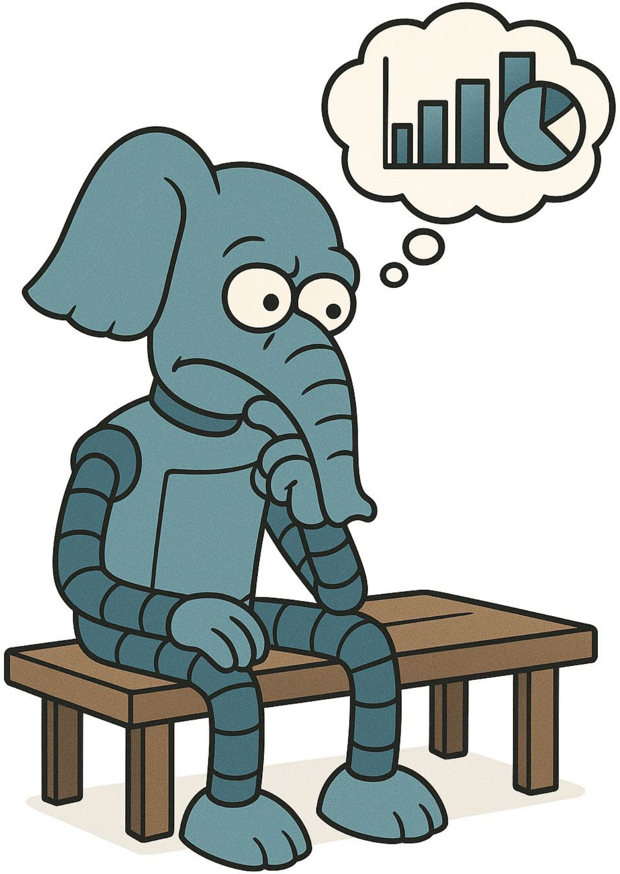
Investigate

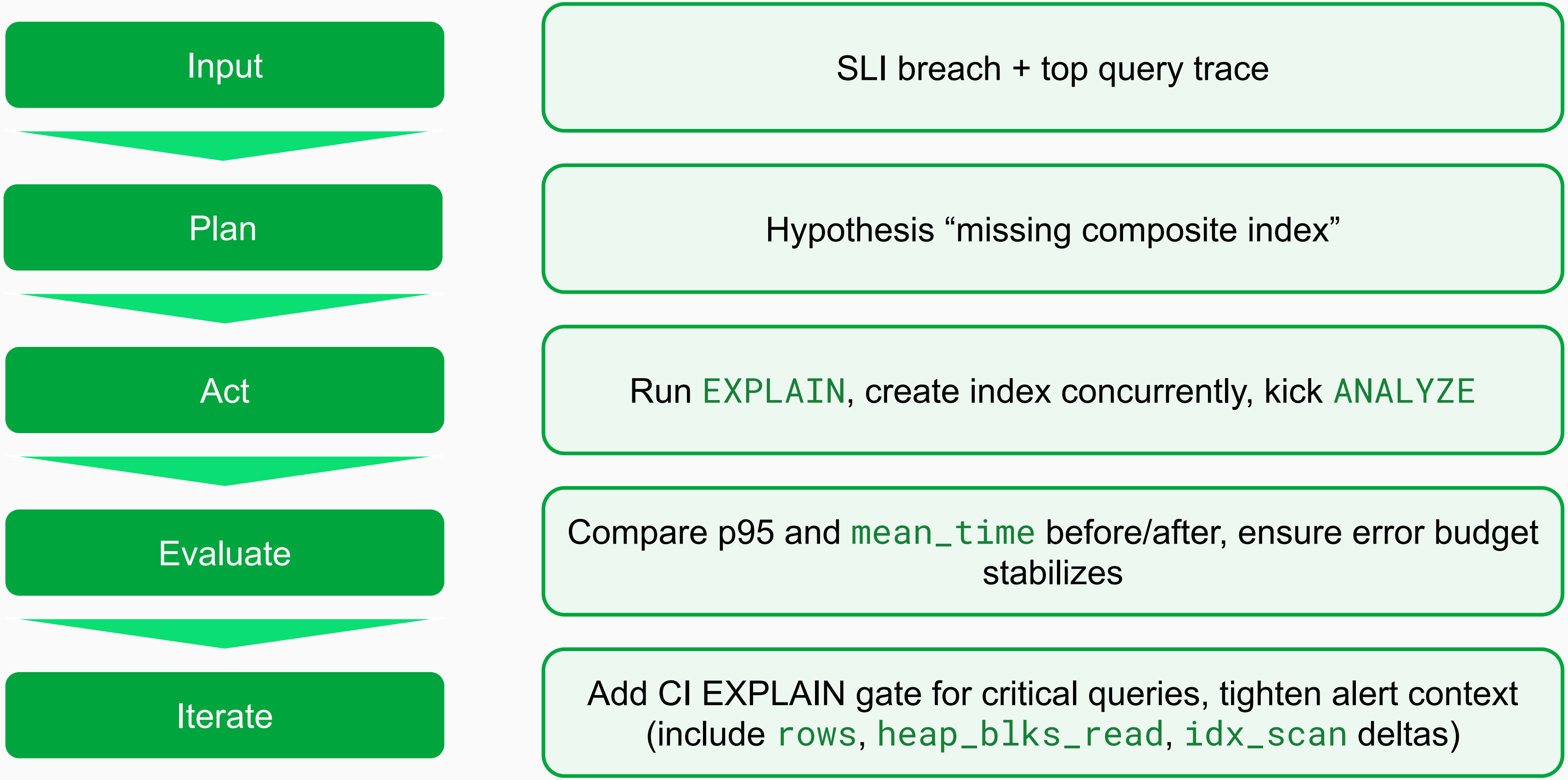
```
SELECT ... FROM pg_stat_statements ORDER BY total_time DESC LIMIT 5;
EXPLAIN (ANALYZE, BUFFERS) … WHERE user_id = $1 AND created_at >
now()-interval '7d';
SELECT indexname,indexdef FROM pg_indexes WHERE tablename='orders';
```

Remediate

```
CREATE INDEX CONCURRENTLY IF NOT EXISTS
idx_orders_user_created ON orders(user_id, created_at DESC);
```

`ANALYZE orders`, confirm plan changes, watch latency recover.





Input

SLI breach + top query trace

Plan

Hypothesis “missing composite index”

Act

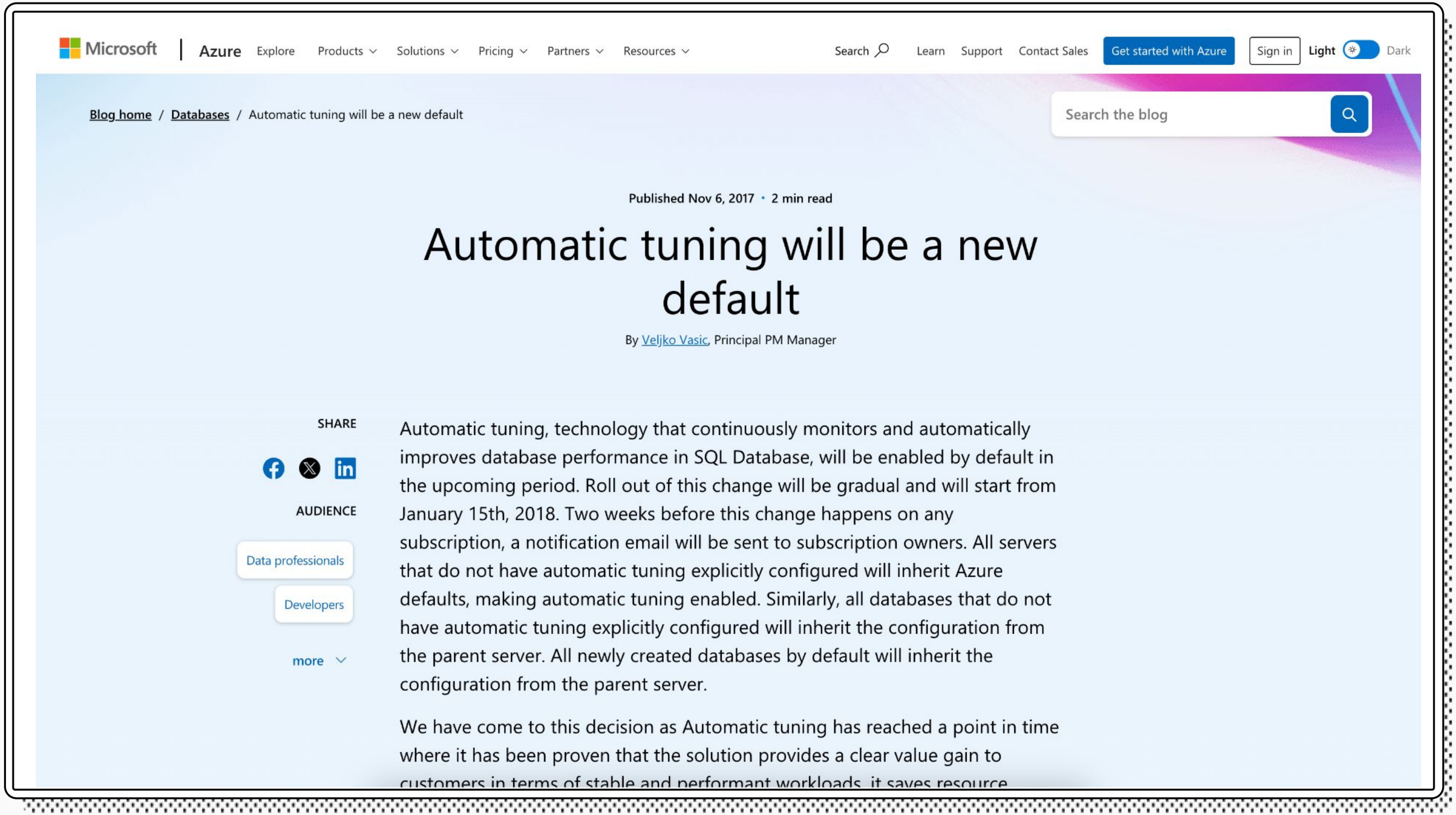
Run `EXPLAIN`, create index concurrently, kick `ANALYZE`

Evaluate

Compare p95 and `mean_time` before/after, ensure error budget stabilizes

Iterate

Add CI EXPLAIN gate for critical queries, tighten alert context (include `rows`, `heap_blks_read`, `idx_scan` deltas)



Blog home / Databases / Automatic tuning will be a new default

Published Nov 6, 2017 • 2 min read

Automatic tuning will be a new default

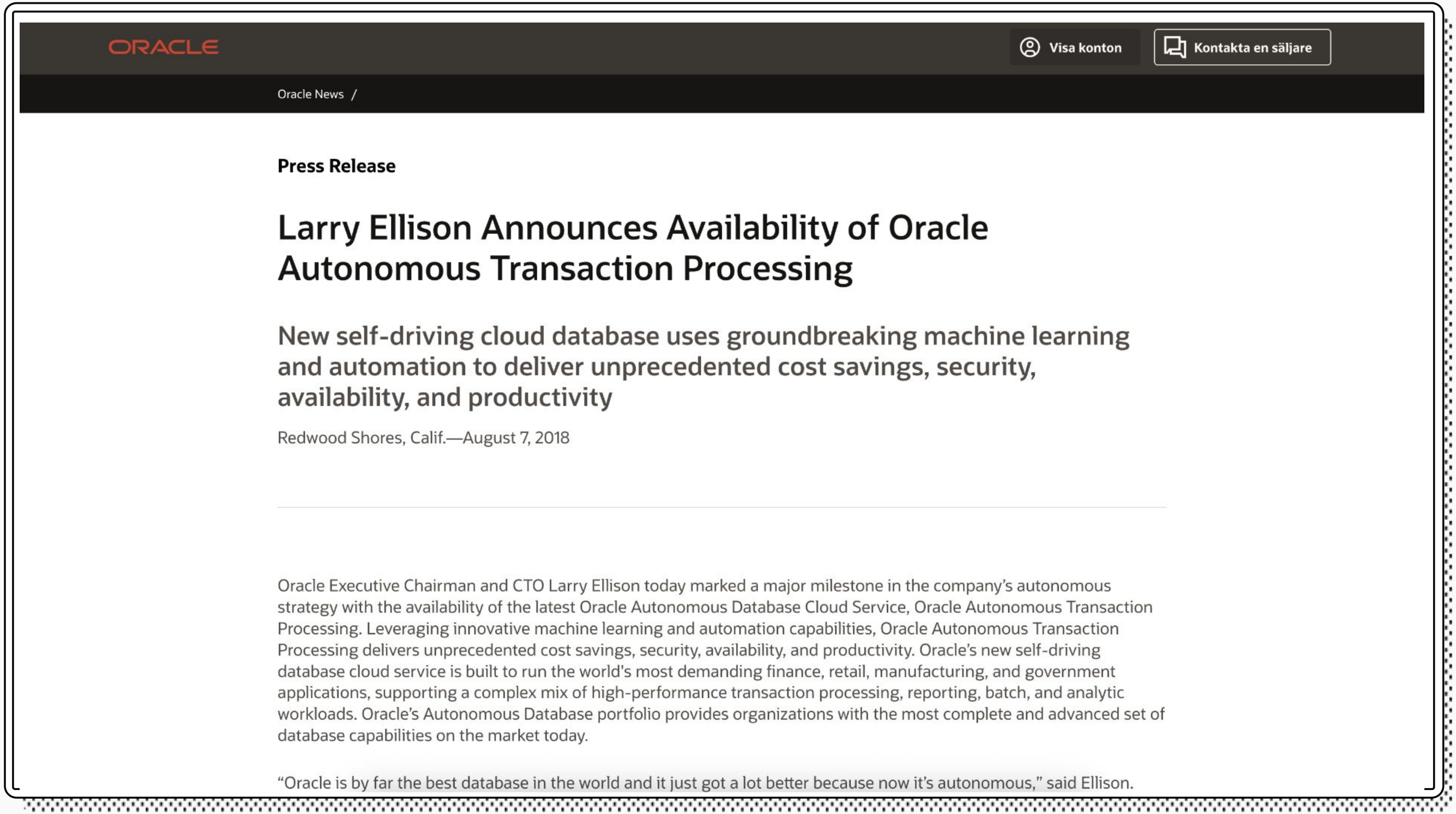
By Veljko Vasic, Principal PM Manager

Automatic tuning, technology that continuously monitors and automatically improves database performance in SQL Database, will be enabled by default in the upcoming period. Roll out of this change will be gradual and will start from January 15th, 2018. Two weeks before this change happens on any subscription, a notification email will be sent to subscription owners. All servers that do not have automatic tuning explicitly configured will inherit Azure defaults, making automatic tuning enabled. Similarly, all databases that do not have automatic tuning explicitly configured will inherit the configuration from the parent server. All newly created databases by default will inherit the configuration from the parent server.

We have come to this decision as Automatic tuning has reached a point in time where it has been proven that the solution provides a clear value gain to customers in terms of stable and performant workloads, it saves resource

https://azure.microsoft.com/en-us/blog/automatic-tuning-will-be-a-new-default

ORACLE



Press Release

Larry Ellison Announces Availability of Oracle Autonomous Transaction Processing

New self-driving cloud database uses groundbreaking machine learning and automation to deliver unprecedented cost savings, security, availability, and productivity

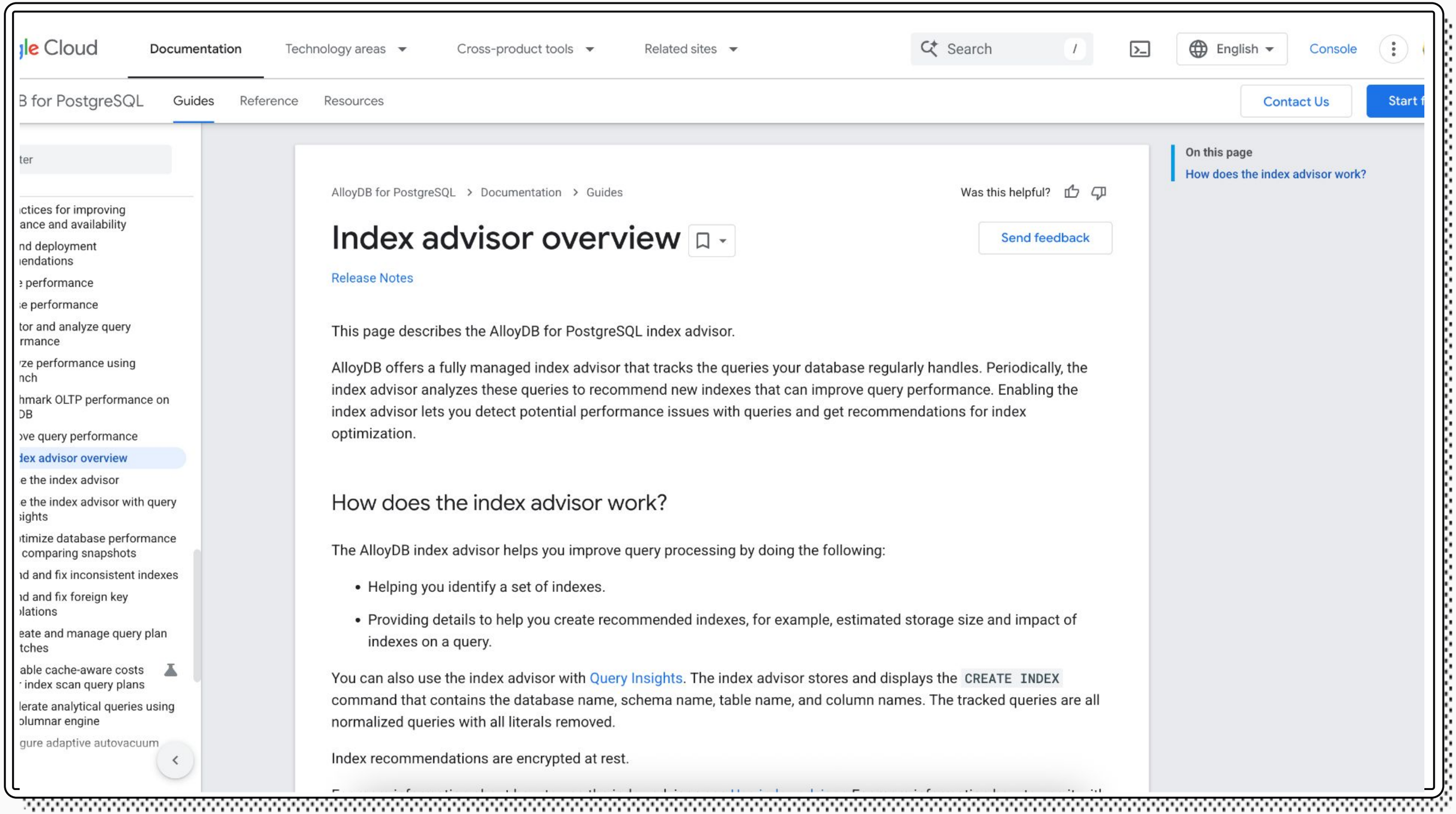
Redwood Shores, Calif.—August 7, 2018

Oracle Executive Chairman and CTO Larry Ellison today marked a major milestone in the company's autonomous strategy with the availability of the latest Oracle Autonomous Database Cloud Service, Oracle Autonomous Transaction Processing. Leveraging innovative machine learning and automation capabilities, Oracle Autonomous Transaction Processing delivers unprecedented cost savings, security, availability, and productivity. Oracle's new self-driving database cloud service is built to run the world's most demanding finance, retail, manufacturing, and government applications, supporting a complex mix of high-performance transaction processing, reporting, batch, and analytic workloads. Oracle's Autonomous Database portfolio provides organizations with the most complete and advanced set of database capabilities on the market today.

"Oracle is by far the best database in the world and it just got a lot better because now it's autonomous," said Ellison.

https://www.oracle.com/se/corporate/pressrelease/oracle-autonomous-transaction-processing-2018-08-07.html

Google Cloud



Index advisor overview

This page describes the AlloyDB for PostgreSQL index advisor.

AlloyDB offers a fully managed index advisor that tracks the queries your database regularly handles. Periodically, the index advisor analyzes these queries to recommend new indexes that can improve query performance. Enabling the index advisor lets you detect potential performance issues with queries and get recommendations for index optimization.

How does the index advisor work?

The AlloyDB index advisor helps you improve query processing by doing the following:

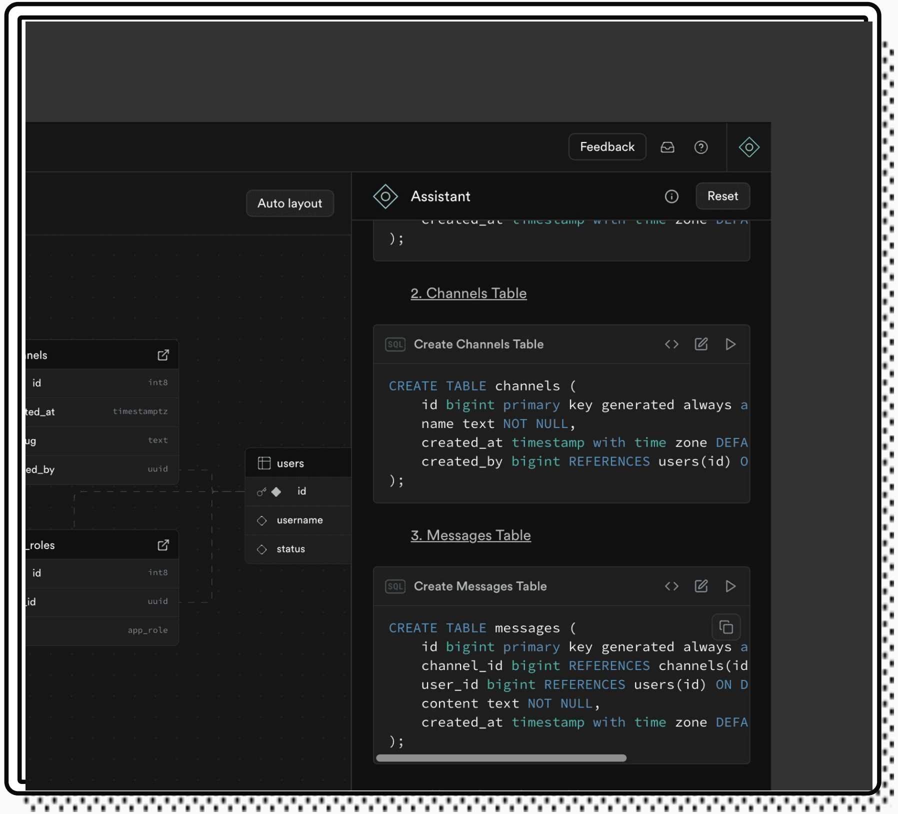
- Helping you identify a set of indexes.
- Providing details to help you create recommended indexes, for example, estimated storage size and impact of indexes on a query.

You can also use the index advisor with Query Insights. The index advisor stores and displays the `CREATE INDEX` command that contains the database name, schema name, table name, and column names. The tracked queries are all normalized queries with all literals removed.

Index recommendations are encrypted at rest.

https://cloud.google.com/alloydb/docs/index-advisor-overview

supabase



```
    created_at timestamp with time zone DEFA
);
```

2. Channels Table

Create Channels Table

```
CREATE TABLE channels (
    id bigint primary key generated always a
    name text NOT NULL,
    created_at timestamp with time zone DEFA
    created_by bigint REFERENCES users(id) O
);
```

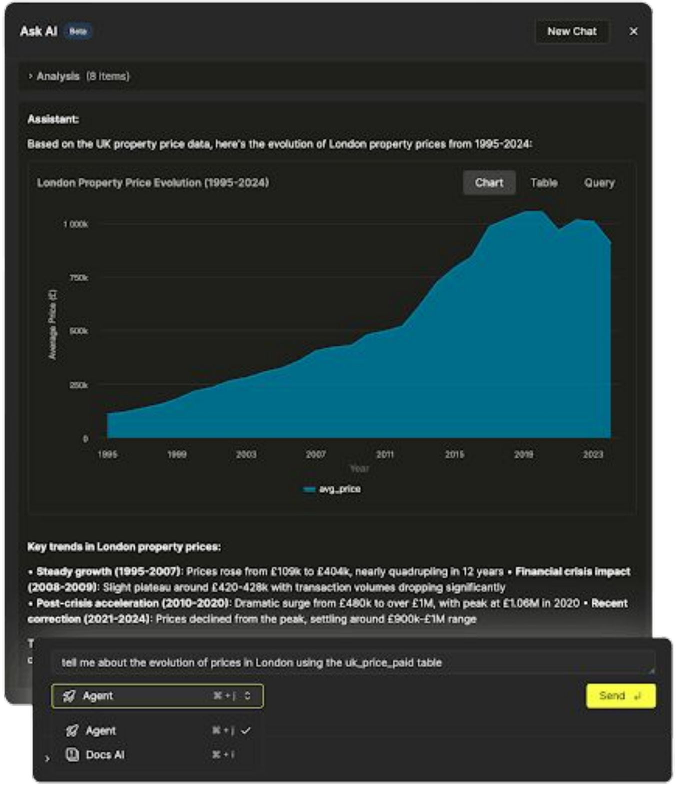
3. Messages Table

Create Messages Table

```
CREATE TABLE messages (
    id bigint primary key generated always a
    channel_id bigint REFERENCES channels(id
    user_id bigint REFERENCES users(id) ON D
    content text NOT NULL,
    created_at timestamp with time zone DEFA
);
```

https://supabase.com/blog/supabase-ai-assistant-v2

ClickHouse



Assistant:

Based on the UK property price data, here's the evolution of London property prices from 1995-2024:

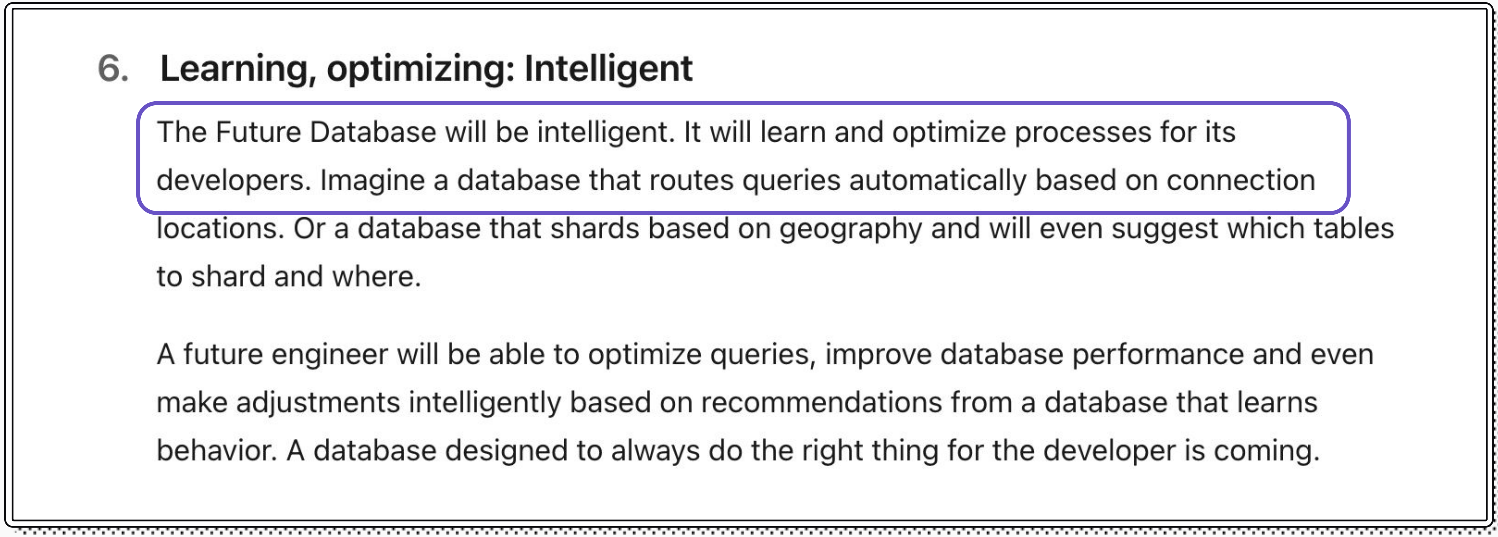
Key trends in London property prices:

• **Steady growth (1995-2007):** Prices rose from £109k to £404k, nearly quadrupling in 12 years • **Financial crisis impact (2008-2009):** Slight plateau around £420-428k with transaction volumes dropping significantly
• **Post-crisis acceleration (2010-2020):** Dramatic surge from £480k to over £1M, with peak at £1.06M in 2020 • **Recent correction (2021-2024):** Prices declined from the peak, settling around £900k-£1M range

tell me about the evolution of prices in London using the uk_price_paid table

https://clickhouse.com/blog/agentic-analytics-ask-ai-agent-and-remote-mcp-server-beta-launch

PlanetScale



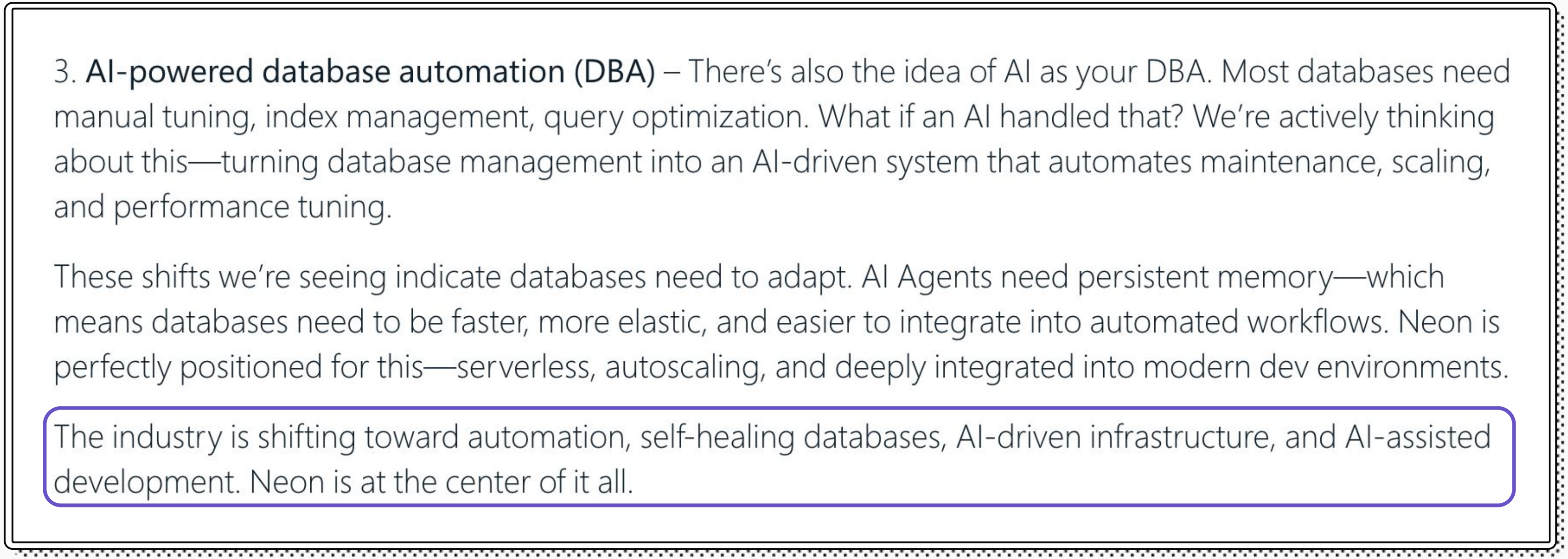
6. **Learning, optimizing: Intelligent**

The Future Database will be intelligent. It will learn and optimize processes for its developers. Imagine a database that routes queries automatically based on connection locations. Or a database that shards based on geography and will even suggest which tables to shard and where.

A future engineer will be able to optimize queries, improve database performance and even make adjustments intelligently based on recommendations from a database that learns behavior. A database designed to always do the right thing for the developer is coming.

https://planetscale.com/principles

NEON



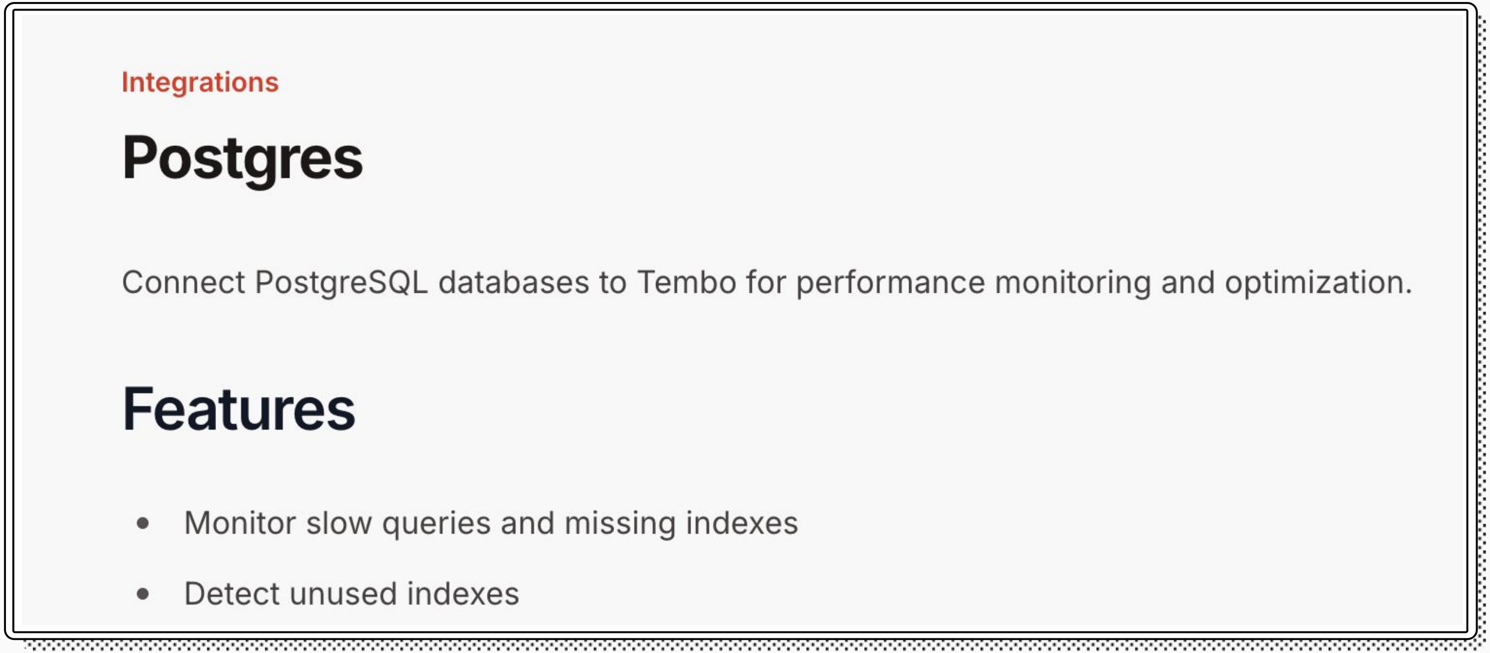
3. **AI-powered database automation (DBA)** – There's also the idea of AI as your DBA. Most databases need manual tuning, index management, query optimization. What if an AI handled that? We're actively thinking about this—turning database management into an AI-driven system that automates maintenance, scaling, and performance tuning.

These shifts we're seeing indicate databases need to adapt. AI Agents need persistent memory—which means databases need to be faster, more elastic, and easier to integrate into automated workflows. Neon is perfectly positioned for this—serverless, autoscaling, and deeply integrated into modern dev environments.

The industry is shifting toward automation, self-healing databases, AI-driven infrastructure, and AI-assisted development. Neon is at the center of it all.

https://m12.vc/news/founders-feature-neon/

tembo



Integrations

Postgres

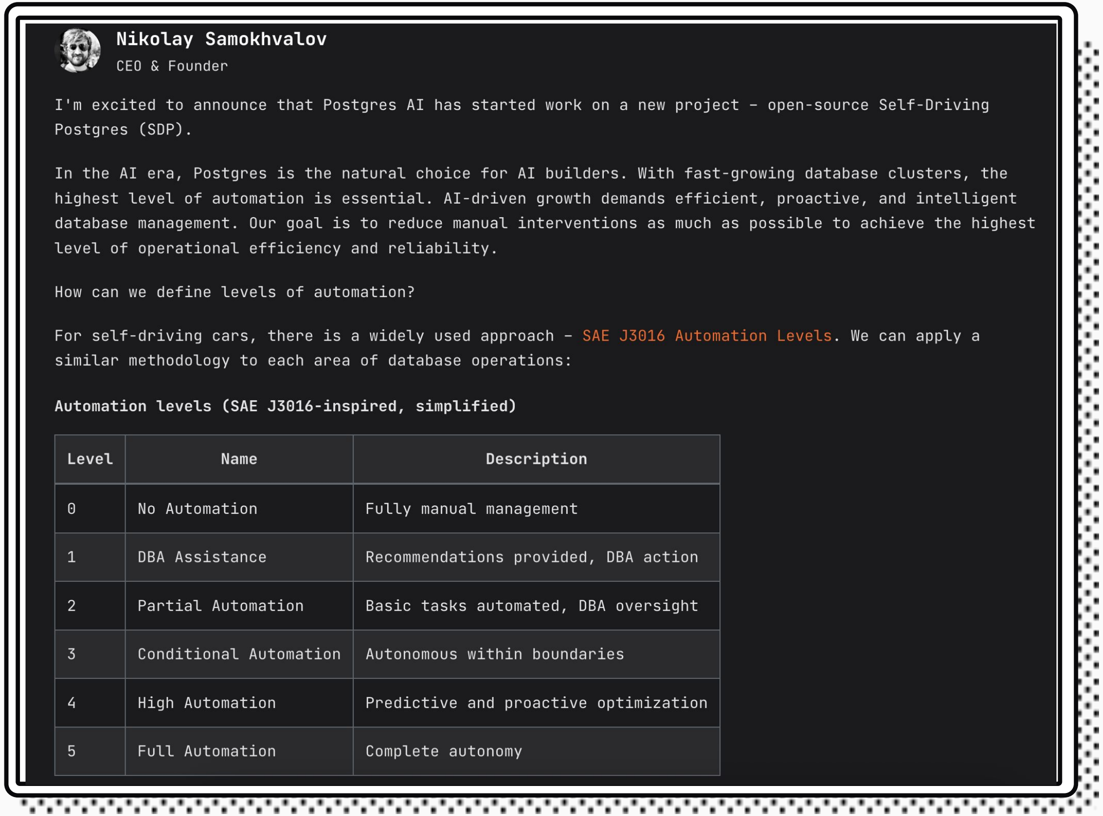
Connect PostgreSQL databases to Tembo for performance monitoring and optimization.

Features

- Monitor slow queries and missing indexes
- Detect unused indexes

https://postgres.ai/blog/20250725-self-driving-postgres

PostgresAI



Nikolay Samokhvalov
CEO & Founder

I'm excited to announce that Postgres AI has started work on a new project – open-source Self-Driving Postgres (SDP).

In the AI era, Postgres is the natural choice for AI builders. With fast-growing database clusters, the highest level of automation is essential. AI-driven growth demands efficient, proactive, and intelligent database management. Our goal is to reduce manual interventions as much as possible to achieve the highest level of operational efficiency and reliability.

How can we define levels of automation?

For self-driving cars, there is a widely used approach – SAE J3016 Automation Levels. We can apply a similar methodology to each area of database operations:

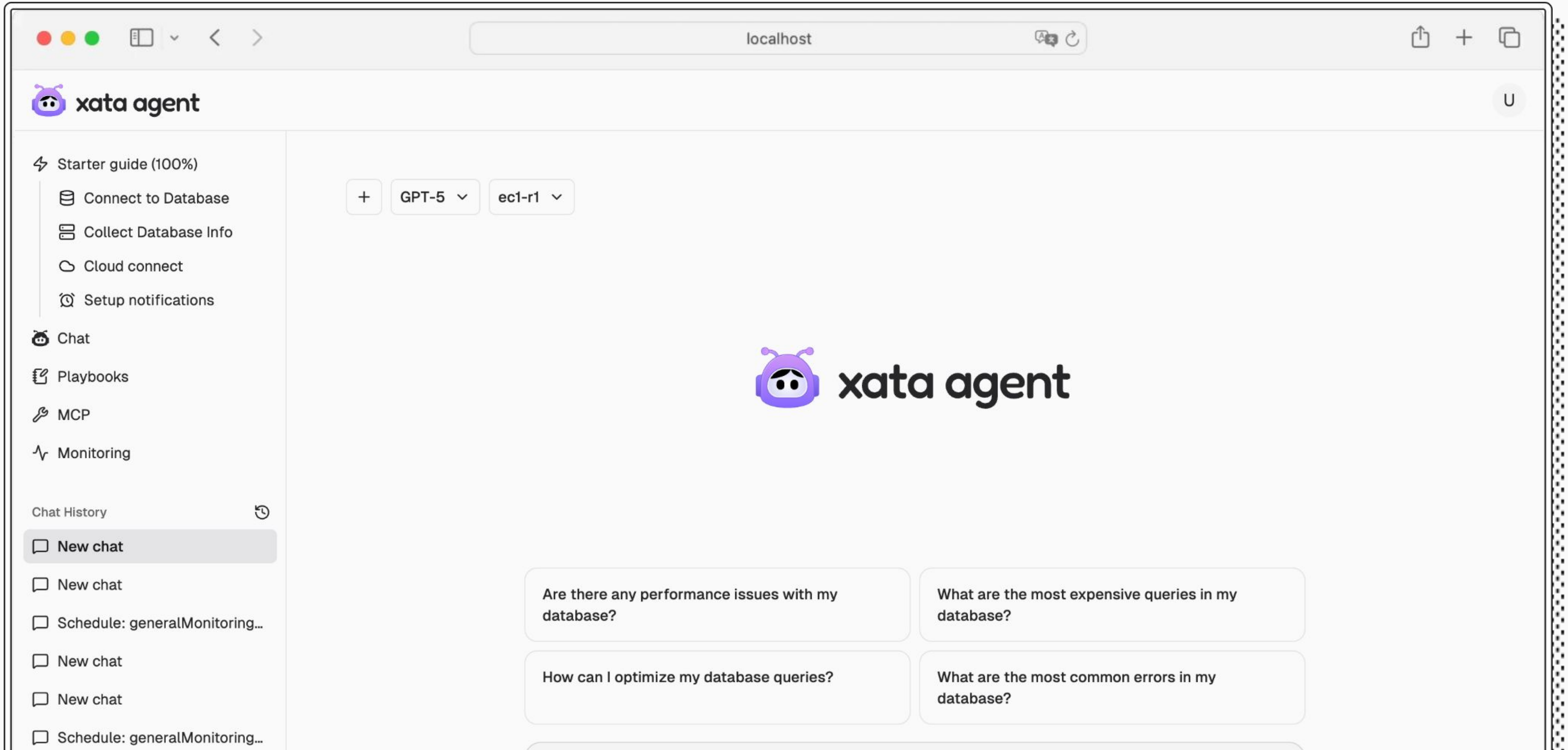
Automation levels (SAE J3016-inspired, simplified)

Level	Name	Description
0	No Automation	Fully manual management
1	DBA Assistance	Recommendations provided, DBA action
2	Partial Automation	Basic tasks automated, DBA oversight
3	Conditional Automation	Autonomous within boundaries
4	High Automation	Predictive and proactive optimization
5	Full Automation	Complete autonomy

https://postgres.ai/blog/20250725-self-driving-postgres

From DBA to DB Agent

https://xata.io/blog/dba-to-db-agent



Are there any performance issues with my database?

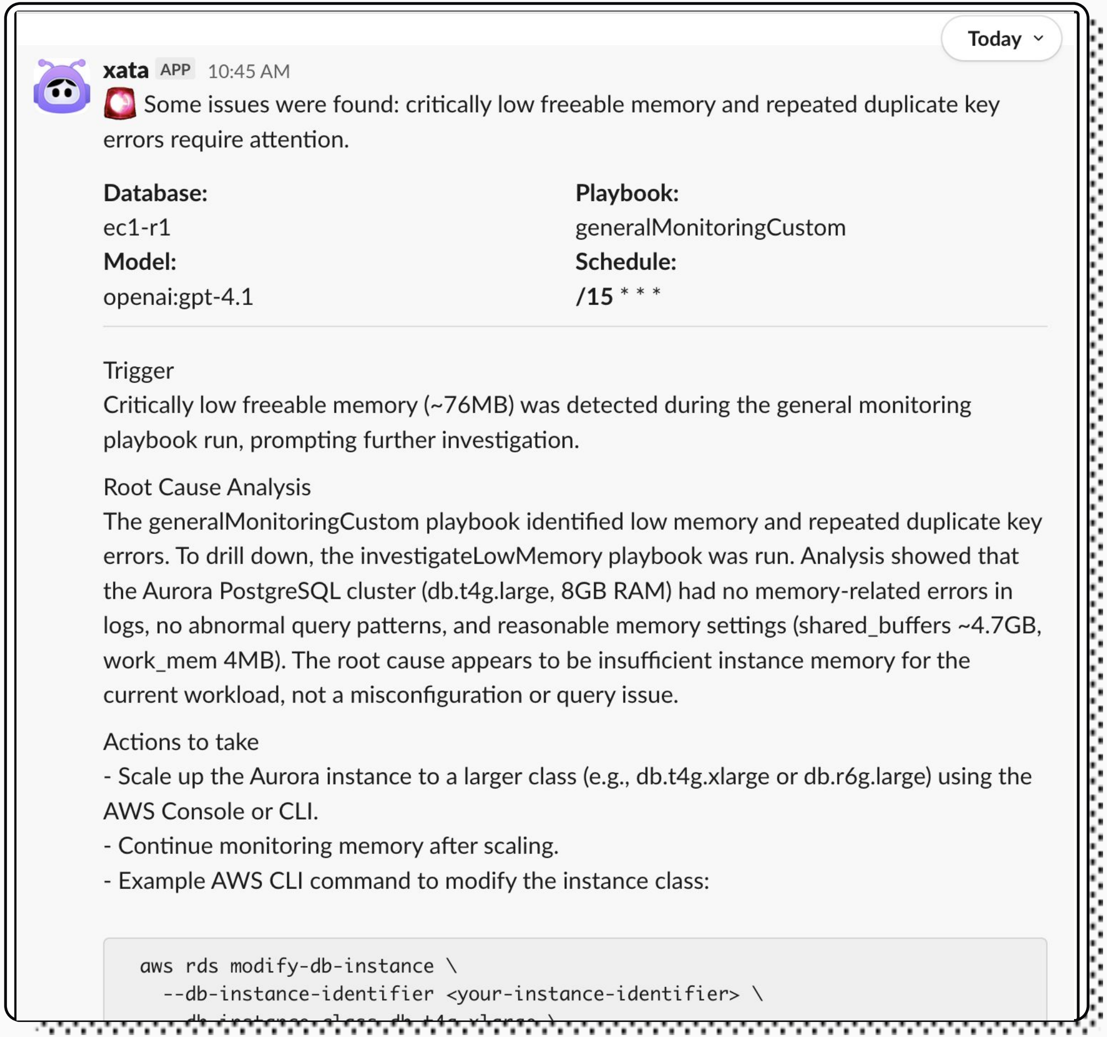
What are the most expensive queries in my database?

How can I optimize my database queries?

What are the most common errors in my database?

Tier 1 Triage

- CPU utilization
- Memory consumption
- Performance issues
- Query optimization
- Resource utilization



xata APP 10:45 AM

🔴 Some issues were found: critically low freeable memory and repeated duplicate key errors require attention.

Database:
ec1-r1

Model:
openai:gpt-4.1

Playbook:
generalMonitoringCustom

Schedule:
/15 * * *

Trigger

Critically low freeable memory (~76MB) was detected during the general monitoring playbook run, prompting further investigation.

Root Cause Analysis

The generalMonitoringCustom playbook identified low memory and repeated duplicate key errors. To drill down, the investigateLowMemory playbook was run. Analysis showed that the Aurora PostgreSQL cluster (db.t4g.large, 8GB RAM) had no memory-related errors in logs, no abnormal query patterns, and reasonable memory settings (shared_buffers ~4.7GB, work_mem 4MB). The root cause appears to be insufficient instance memory for the current workload, not a misconfiguration or query issue.

Actions to take

- Scale up the Aurora instance to a larger class (e.g., db.t4g.xlarge or db.r6g.large) using the AWS Console or CLI.
- Continue monitoring memory after scaling.
- Example AWS CLI command to modify the instance class:

```
aws rds modify-db-instance \
  --db-instance-identifier <your-instance-identifier> \
  --db-instance-class db.t4g.xlarge \
```

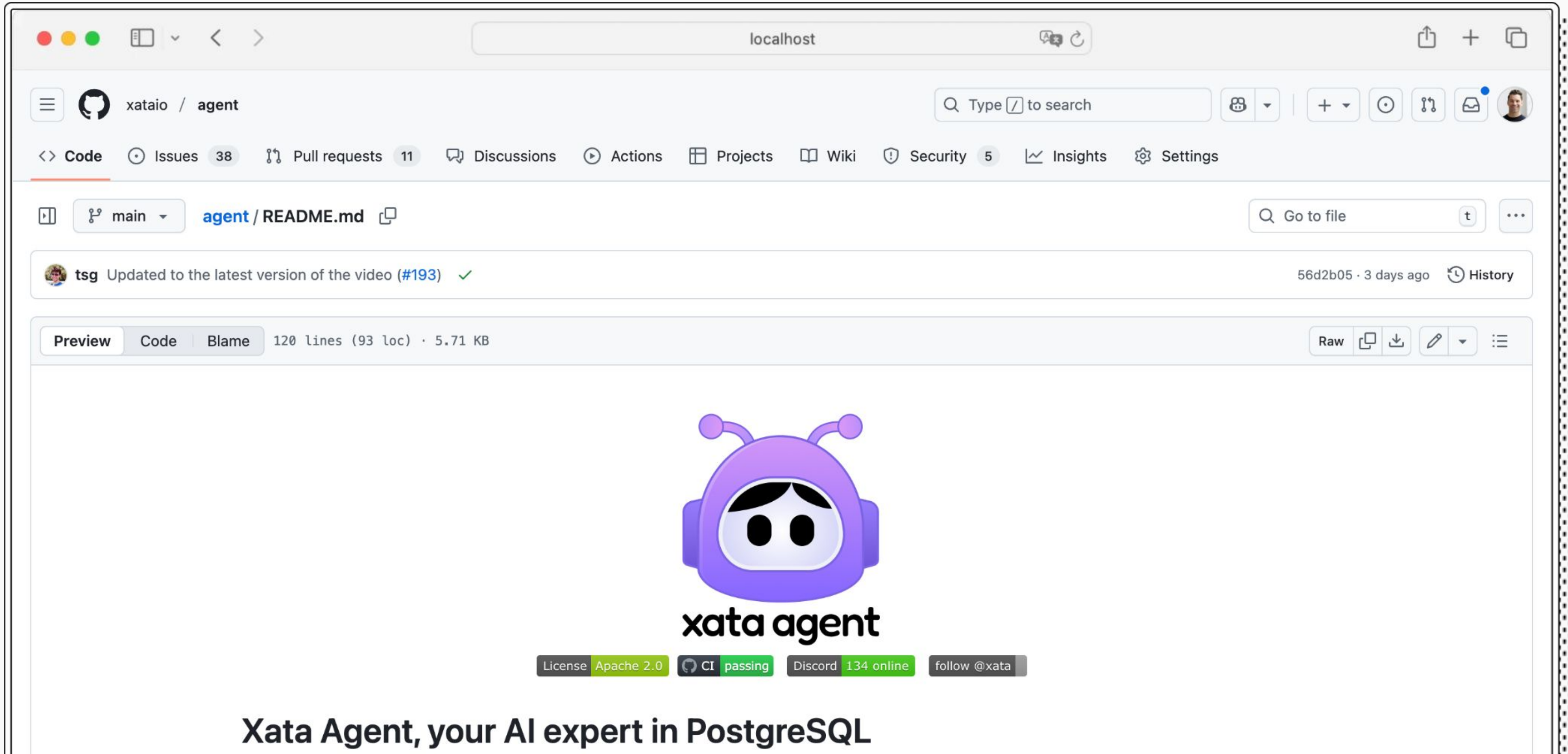
xata

Walk through

Building the Agent

Xata Agent

xataio/agent



agent / README.md

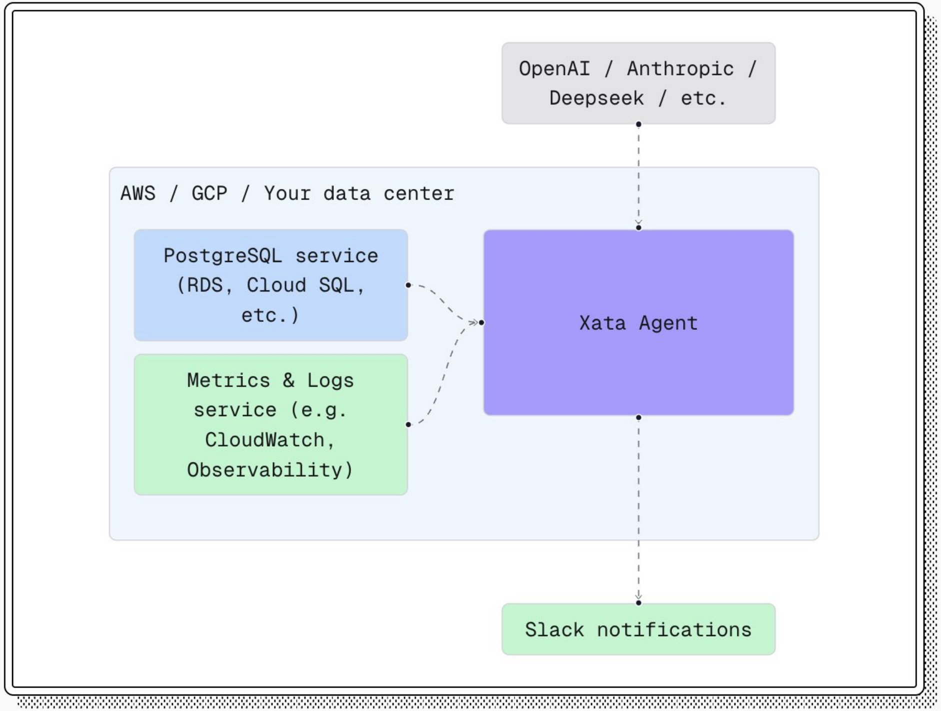
tsg Updated to the latest version of the video (#193) ✓ 56d2b05 · 3 days ago

120 lines (93 loc) · 5.71 KB

Xata Agent, your AI expert in PostgreSQL

Tech Stack

- Next.js
- Postgres
- TypeScript
- Vercel AI SDK
- MCP TypeScript SDK
- Self-host via docker-compose
- Mastra TypeScript AI Framework
- Ollama for local models
- LangFuse for observability
- LiteLLM as AI API gateway


OpenAI / Anthropic / Deepseek / etc.
AWS / GCP / Your data center
PostgreSQL service (RDS, Cloud SQL, etc.)
Metrics & Logs service (e.g. CloudWatch, Observability)
Xata Agent
Slack notifications

Schedule

Run playbooks at scheduled or agent defined intervals

Playbooks

Sequence of steps that the agent can follow to detect, diagnose, and resolve issues

Tools

Functions that can be called by the agent to gather more context and act

Approve

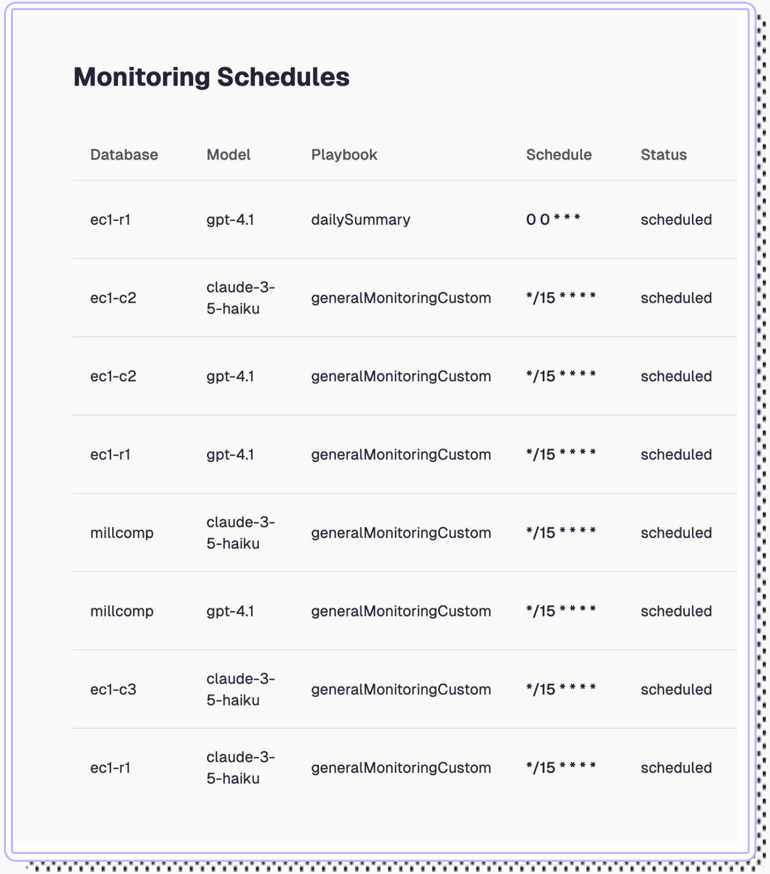
Workflows to approve recommendations from the agent

Schedule

Input

Schedules / Monitors

- Execute playbooks periodically
- Allows chaining of playbooks:
    - Based on the findings of a playbook, “drill down” using another playbook.
- Keep history of runs and outcomes
    - This will become the agent “memory”



Monitoring Schedules

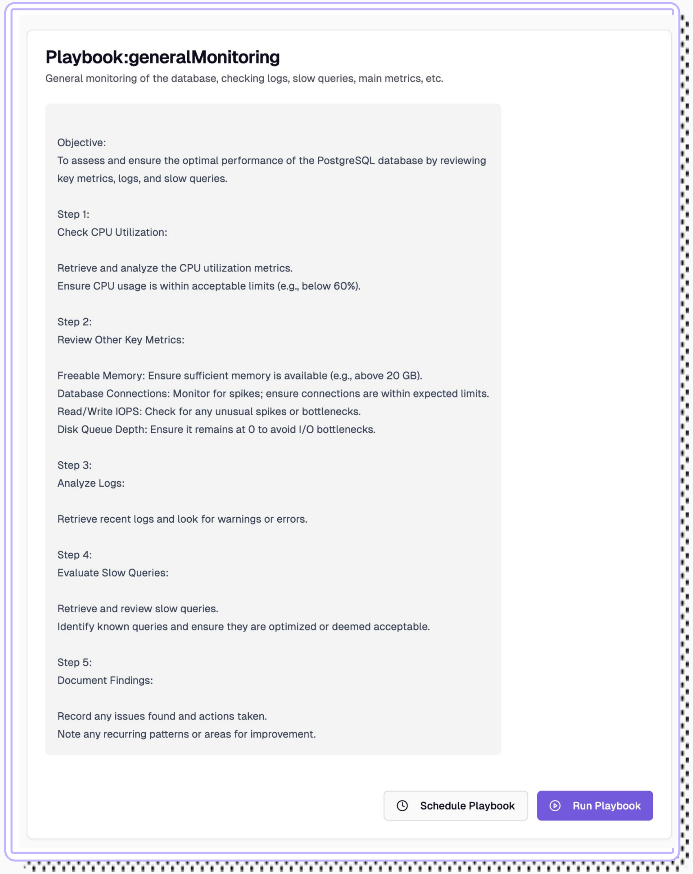
Database	Model	Playbook	Schedule	Status
ec1-r1	gpt-4.1	dailySummary	0 0 * * *	scheduled
ec1-c2	claude-3-5-haiku	generalMonitoringCustom	*/15 * * * *	scheduled
ec1-c2	gpt-4.1	generalMonitoringCustom	*/15 * * * *	scheduled
ec1-r1	gpt-4.1	generalMonitoringCustom	*/15 * * * *	scheduled
millcomp	claude-3-5-haiku	generalMonitoringCustom	*/15 * * * *	scheduled
millcomp	gpt-4.1	generalMonitoringCustom	*/15 * * * *	scheduled
ec1-c3	claude-3-5-haiku	generalMonitoringCustom	*/15 * * * *	scheduled
ec1-r1	claude-3-5-haiku	generalMonitoringCustom	*/15 * * * *	scheduled

Playbooks

Plan

Playbooks

- Playbook examples:
  - generalMonitoring
  - investigateHighCPU
  - investigateLowMemory
  - investigateHighConnectionCount
  - tuneSettings
  - dailySummary



Playbook:generalMonitoring

General monitoring of the database, checking logs, slow queries, main metrics, etc.

Objective:
To assess and ensure the optimal performance of the PostgreSQL database by reviewing key metrics, logs, and slow queries.

Step 1:
Check CPU Utilization:

Retrieve and analyze the CPU utilization metrics.
Ensure CPU usage is within acceptable limits (e.g., below 60%).

Step 2:
Review Other Key Metrics:

Freeable Memory: Ensure sufficient memory is available (e.g., above 20 GB).
Database Connections: Monitor for spikes; ensure connections are within expected limits.
Read/Write IOPS: Check for any unusual spikes or bottlenecks.
Disk Queue Depth: Ensure it remains at 0 to avoid I/O bottlenecks.

Step 3:
Analyze Logs:

Retrieve recent logs and look for warnings or errors.

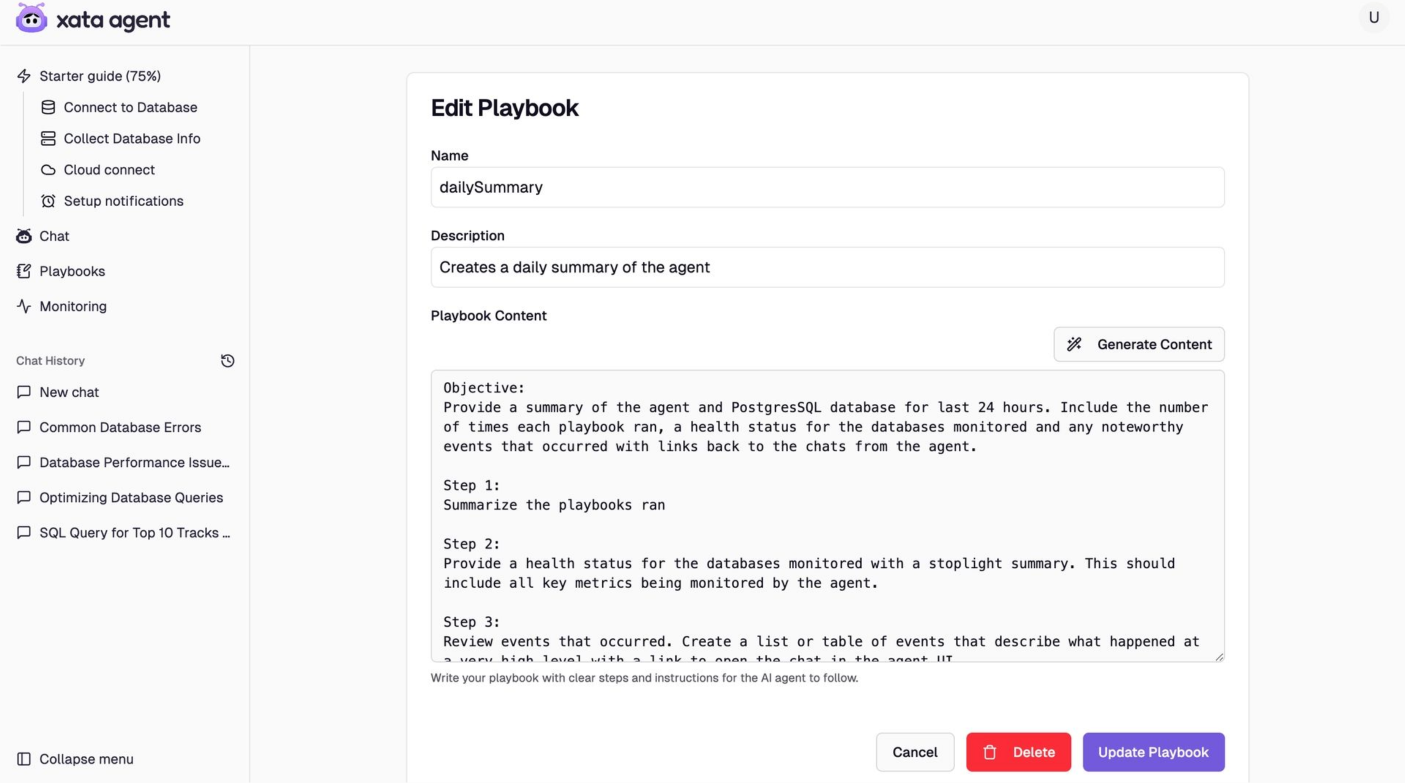
Step 4:
Evaluate Slow Queries:

Retrieve and review slow queries.
Identify known queries and ensure they are optimized or deemed acceptable.

Step 5:
Document Findings:

Record any issues found and actions taken.
Note any recurring patterns or areas for improvement.

Playbooks are editable



Edit Playbook

Name

dailySummary

Description

Creates a daily summary of the agent

Playbook Content

```
Objective:
Provide a summary of the agent and PostgresSQL database for last 24 hours. Include the number
of times each playbook ran, a health status for the databases monitored and any noteworthy
events that occurred with links back to the chats from the agent.

Step 1:
Summarize the playbooks ran

Step 2:
Provide a health status for the databases monitored with a stoplight summary. This should
include all key metrics being monitored by the agent.

Step 3:
Review events that occurred. Create a list or table of events that describe what happened at
a very high level with a link to open the chat in the agent UI
```

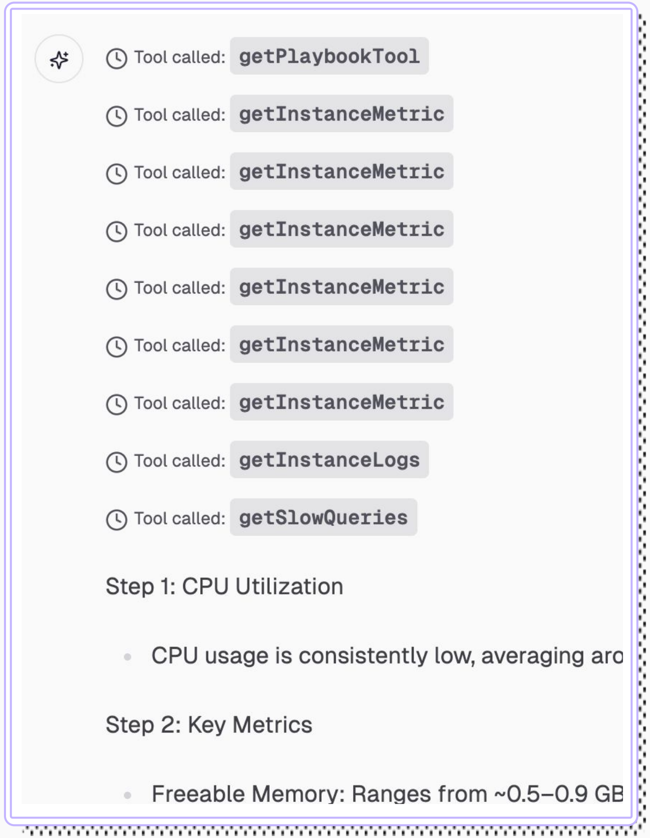
Write your playbook with clear steps and instructions for the AI agent to follow.

Tools

Act

Tools

- A mix of simple and complex tools for the agent to act
- Tool examples:
  - getInstanceMetric()
  - getInstanceLogs()
  - getSlowQueries()
  - explainQuery()
  - getSettings()
  - getTablesInfo()
  - getPostgresExtensions()



Tool called: `getPlaybookTool`

Tool called: `getInstanceMetric`

Tool called: `getInstanceMetric`

Tool called: `getInstanceMetric`

Tool called: `getInstanceMetric`

Tool called: `getInstanceMetric`

Tool called: `getInstanceMetric`

Tool called: `getInstanceLogs`

Tool called: `getSlowQueries`

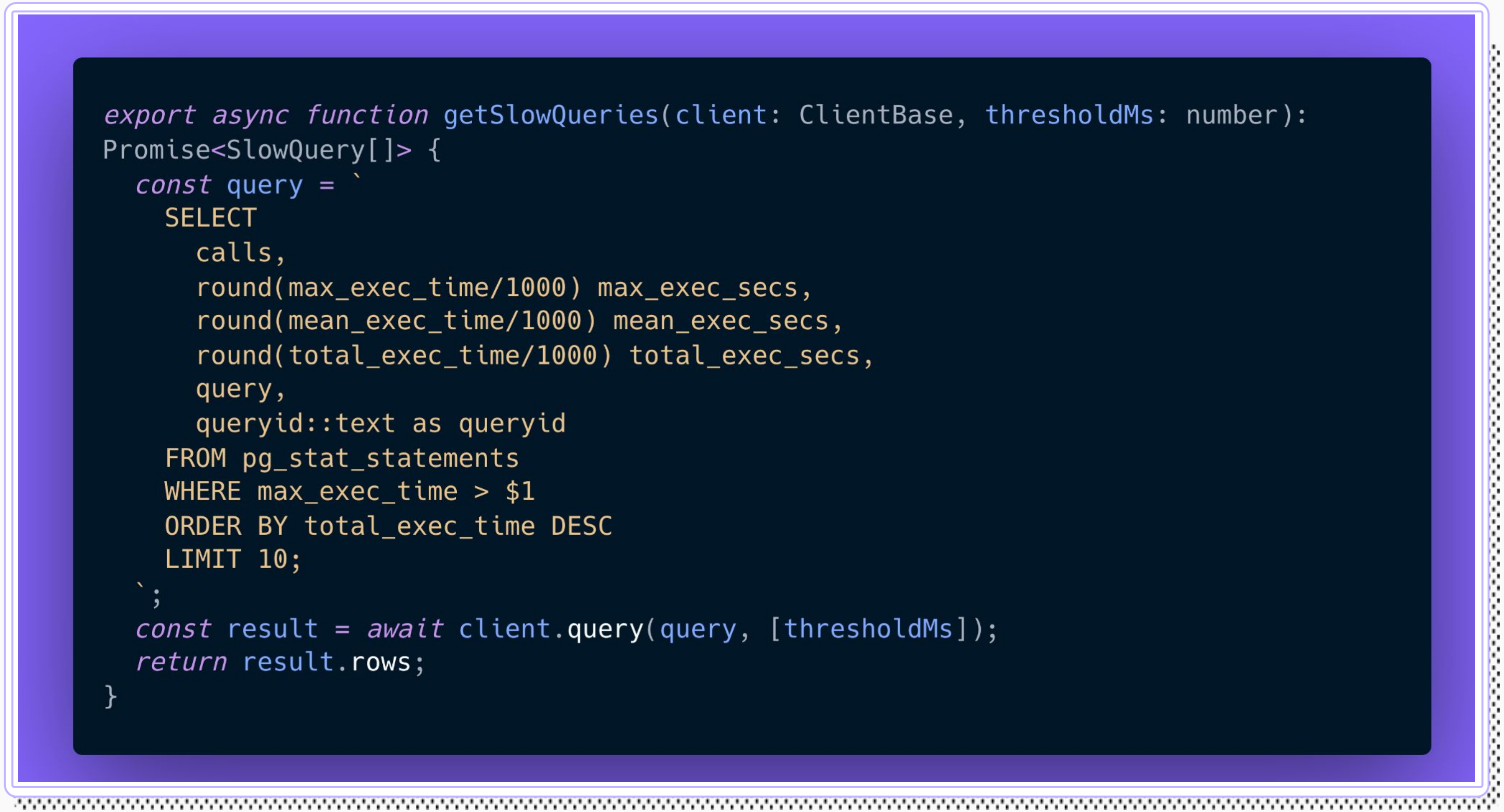
Step 1: CPU Utilization

- CPU usage is consistently low, averaging aro

Step 2: Key Metrics

- Freeable Memory: Ranges from ~0.5–0.9 GB

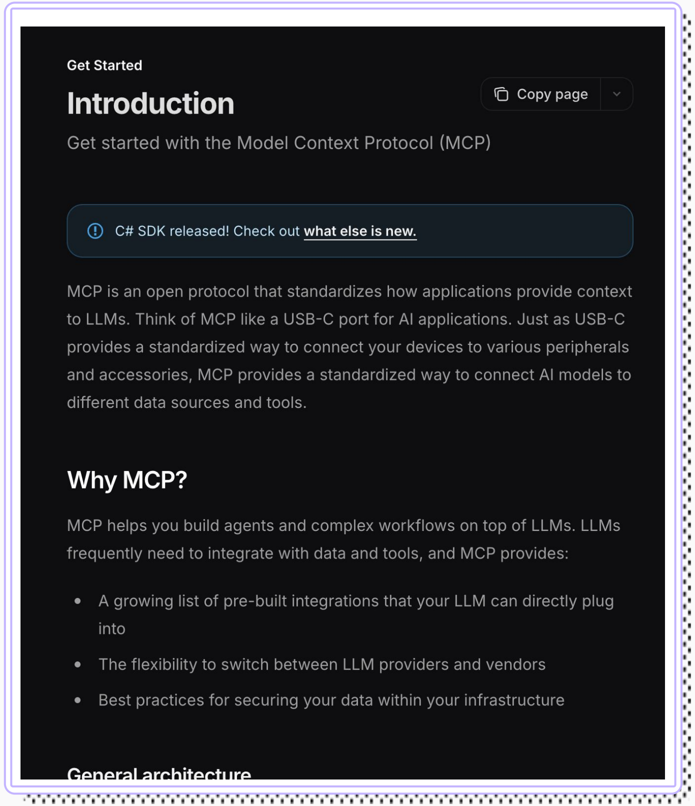
Example tool implementation: getSlowQueries



```
export async function getSlowQueries(client: ClientBase, thresholdMs: number):
Promise<SlowQuery[]> {
  const query = `
    SELECT
      calls,
      round(max_exec_time/1000) max_exec_secs,
      round(mean_exec_time/1000) mean_exec_secs,
      round(total_exec_time/1000) total_exec_secs,
      query,
      queryid::text as queryid
    FROM pg_stat_statements
    WHERE max_exec_time > $1
    ORDER BY total_exec_time DESC
    LIMIT 10;
  `;
  const result = await client.query(query, [thresholdMs]);
  return result.rows;
}
```

Model Context Protocol

- Proposed by Anthropic but now the industry standard
- An MCP server exposes
  - Resources
  - Tools
  - Prompts
- Many companies are adding MCP servers for their services
- Range from API wrapper to use case oriented



Introduction

Get started with the Model Context Protocol (MCP)

C# SDK released! Check out what else is new.

MCP is an open protocol that standardizes how applications provide context to LLMs. Think of MCP like a USB-C port for AI applications. Just as USB-C provides a standardized way to connect your devices to various peripherals and accessories, MCP provides a standardized way to connect AI models to different data sources and tools.

Why MCP?

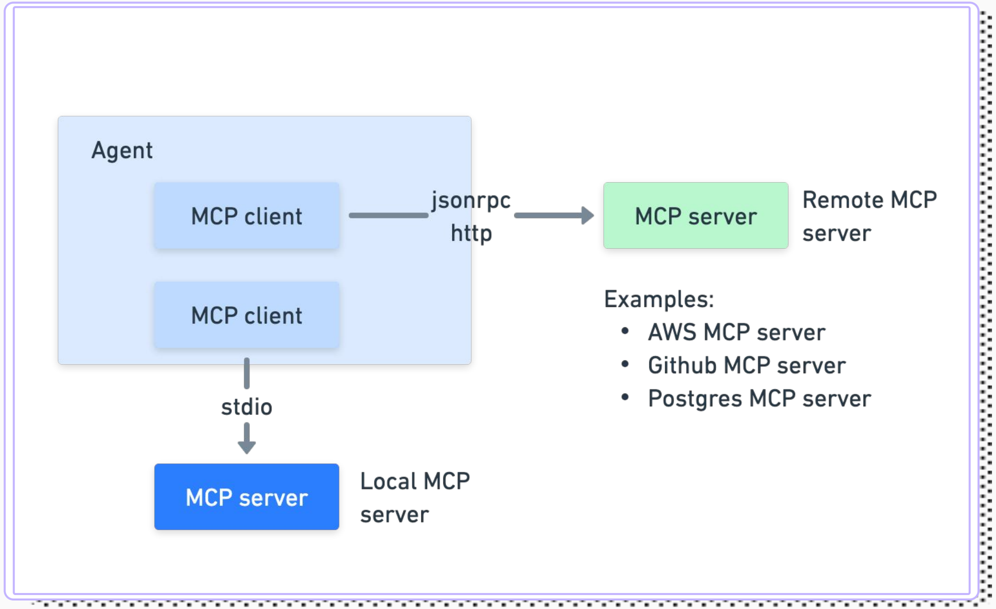
MCP helps you build agents and complex workflows on top of LLMs. LLMs frequently need to integrate with data and tools, and MCP provides:

- A growing list of pre-built integrations that your LLM can directly plug into
- The flexibility to switch between LLM providers and vendors
- Best practices for securing your data within your infrastructure

General architecture

Custom tools: MCP servers

- MCP
  - Local MCP server (stdio)
  - Remote MCP server (jsonrpc)
- Local MCP makes it easy to add your own tools:
  - Write them in TS or Python, drop them in a folder
- As cloud providers add MCP servers, the agent can just use them.



Examples:

- AWS MCP server
- Github MCP server
- Postgres MCP server

Approval

Evaluate

Recommend

Provide recommendations for end users to act on

Approval

Provide recommendations and the ability approve changes for the agent to acton

Autonomous

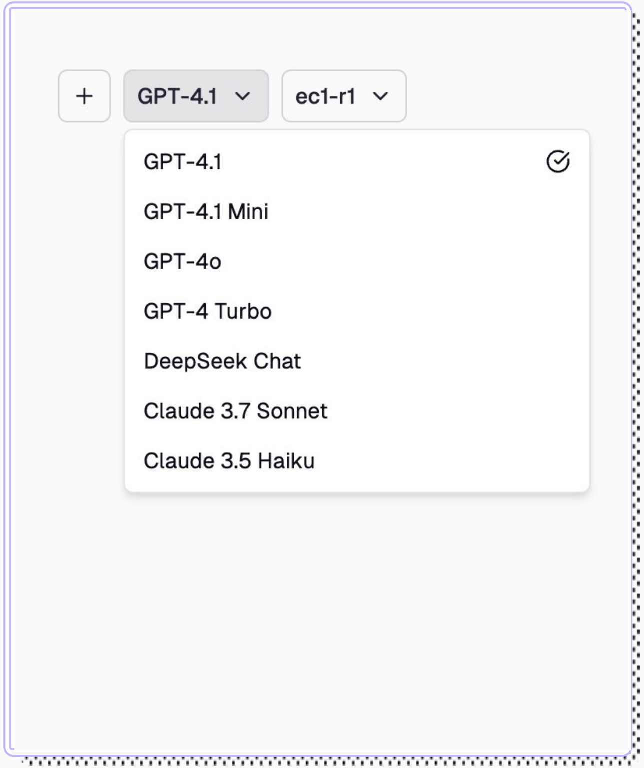
YOLO

Fine Tune

Iterate

Models

- Uses Vercel AI SDK as an abstraction layer for multiple models
  - OpenAI
  - Anthropic
  - Deepseek
  - Gemini
  - Ollama
  - LiteLLM
- Reasoning models tend to do better on agentic use cases
  - Better answers/reasoning > speed
  - More expensive



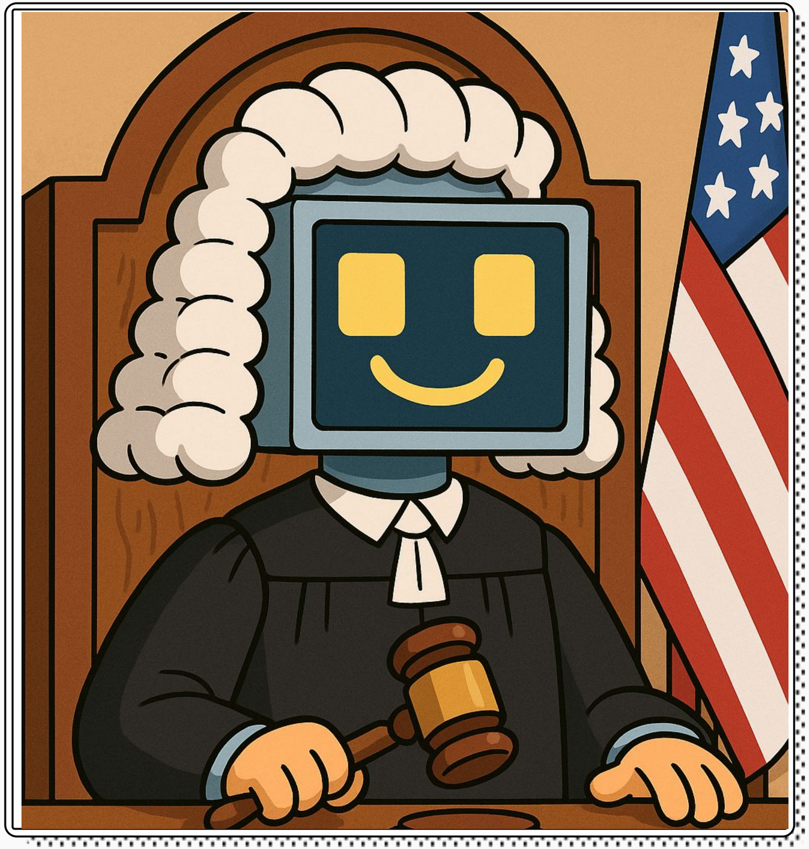
Evals

What

- LLM judges
- Repeatable test designed to measure some aspect of model behavior
- Crucial for tracking improvements, identifying regressions, and understanding limitations

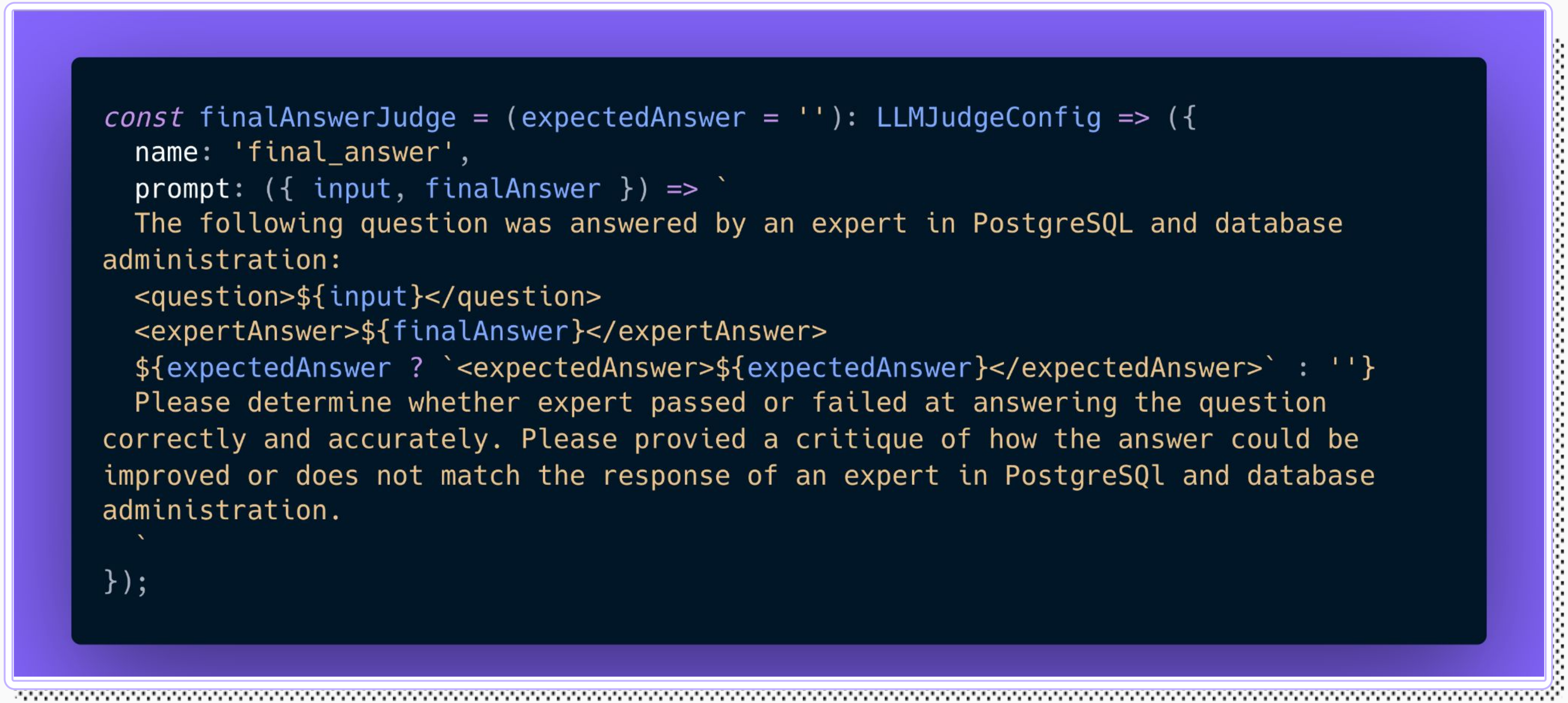
When

- A playbook / prompt is changed
- A new model is release
- A new tool is added



Evals

LLM Judges score 👩‍⚖️ Did it identify the root cause? Is the feedback actionable? Is it concise or too much to read?



```
const finalAnswerJudge = (expectedAnswer = ''): LLMJudgeConfig => ({
  name: 'final_answer',
  prompt: ({ input, finalAnswer }) => `
  The following question was answered by an expert in PostgreSQL and database
administration:
  <question>${input}</question>
  <expertAnswer>${finalAnswer}</expertAnswer>
  ${expectedAnswer ? `<expectedAnswer>${expectedAnswer}</expectedAnswer>` : ''}
  Please determine whether expert passed or failed at answering the question
correctly and accurately. Please provied a critique of how the answer could be
improved or does not match the response of an expert in PostgreSQL and database
administration.
  `
});
```

Challenges

Can I *really* trust an agent?

Is my data safe and private?

Is an agent cost effective for observability?

Should I YOLO?

What’s next

Looking ahead

Agent Roadmap

- ✓ Model tool / function calling support
- ✓ Safely run arbitrary SQL
- ✓ Better short & long term memory
- ✓ Code & observability integrations
- ✓ GitHub approval flows

Xata Demo

Not a product pitch, just setting context - I swear

Autonomous Postgres

- ✓ Store more context: metrics, logs, query stats, schema changes
- ✓ Recommendation engine: Slow queries first
- ✓ Apply to new CoW branch & test workflow
- ✓ Custom playbook & tooling support
- ✓ Non-functional requirements - evals, notifications, etc.

Join in on the fun

Deploy and provide feedback

- Works with any Postgres
- Native support for:
  - AWS RDS, Amazon Aurora, GCP Cloud SQL

Contribute to the project

- We are friendly to outside contributions
- Gain experience with any of these technologies
  - Next.js / Typescript
  - Evals
  - LLM Memory
  - GitHub integration

xata

Postgres at scale

Thank you!

alex-francoeur

alexf@xata.io

xata.io