AI-Powered Root Cause Analysis at Scale

From Theory to Production — Lessons from Nubank's 120M+ Customers

ObservabilityCon 2026 · Minneapolis

Yevgeny Gladun · Letícia Mota · Nubank

What we'll cover today

Why we built it

The gaps no vendor could close at our scale

How it works

Three production problems → three concrete solutions

What we learned

Real numbers, hard tradeoffs, what's next

Nubank at a glance — why AI-powered RCA matters at our scale

Customers	**120+ million customers** across **Brazil**, **Mexico**, and **Colombia** (and counting). So, every minute of downtime can affect millions of real people's financial lives.
Scale	**Thousands of microservices** running on AWS/EKS with **millions of daily transactions** across the core products and services (account, credit card, loans, investments, crypto, insurance, etc.)
Time sensitive	When engineers get paged at 3 AM, they're swimming in data and need answers fast. Manual investigation and correlation is too slow.
Goal	**Build an AI agent that investigates like a senior SRE — fast enough to matter, deep enough to find root cause.**

Three gaps driving MTTR variability and incident volume

Gaps in Training

Inconsistent application of best practices. Engineers rely on runbooks that don't scale, assume permissions they don't have, can't adapt to novel failures.

Overload

23,000+ metrics on Prometheus, billions of log rows on Trino, dozens of dashboards on Grafana, HoneyComb and GitHub — engineers drown in data sources.

Tooling Limits

Hours of lost productivity: High stress events requiring effort from multiple engineers to get to Root Cause Analysis (RCA).

What an AI SRE agent must do

Lead engineers to the right questions, explore the right data, and deliver root cause in seconds.

The SRE Agent: From Idea to Production

Core Platform Pillars

01. Autonomous RCA

Orchestrates multi-source analysis: Prometheus, Honeycomb, Trino, GitHub, K8s, Grafana and internal incident data.

02. LLM-Driven

Python orchestrator (GPT-5.x/Claude 4.x) with MCP access to observability tools.

03. Engineer UX

Chat interface delivering structured RCA in seconds or minutes for the users.

Wins

ROI-Positive: Run in production at under $0.20 per investigation vs. engineer time.

Better Focus: Reduced context-switching allows engineers to stay in the flow.

Speed to Market

2 Months

Launched by 2 engineers

Engineering Challenges at Scale

Yevgeny Gladun
Staff Engineer — Runtime Platforms

Why we decided to build

We evaluated 4 vendors. None cleared all 5 boxes for our environment. So we built.

Vendor A	Investigation demo not impressive; unclear value delivered
Vendor B	Not SOC compliant; high per-investigation cost
Vendor C	Required sending all telemetry externally; blocked by network policy
Vendor D	Two years in production, <40% alert coverage, poor CSAT

5 reasons we chose to build

1. **Unpredictable pricing** — Costs scale with investigation length and LLM tokens at our size. Unclear ROI.
2. **No RCA confidence guarantee** — Investigation quality varied wildly across incident types.
3. **Custom observability stack** — Heavy customizations made vendor integration difficult out of the box.
4. **Data privacy & compliance** — Petabytes of observability data; SOC 2 alone wasn't enough.
5. **2-engineer PoC** — Validate the approach internally before any vendor commitment.

SRE Agent Architecture

Behind the scenes, it's a **Python service** that uses the **LLM's reasoning** to answer questions, while supplying the context it needs to do that well.

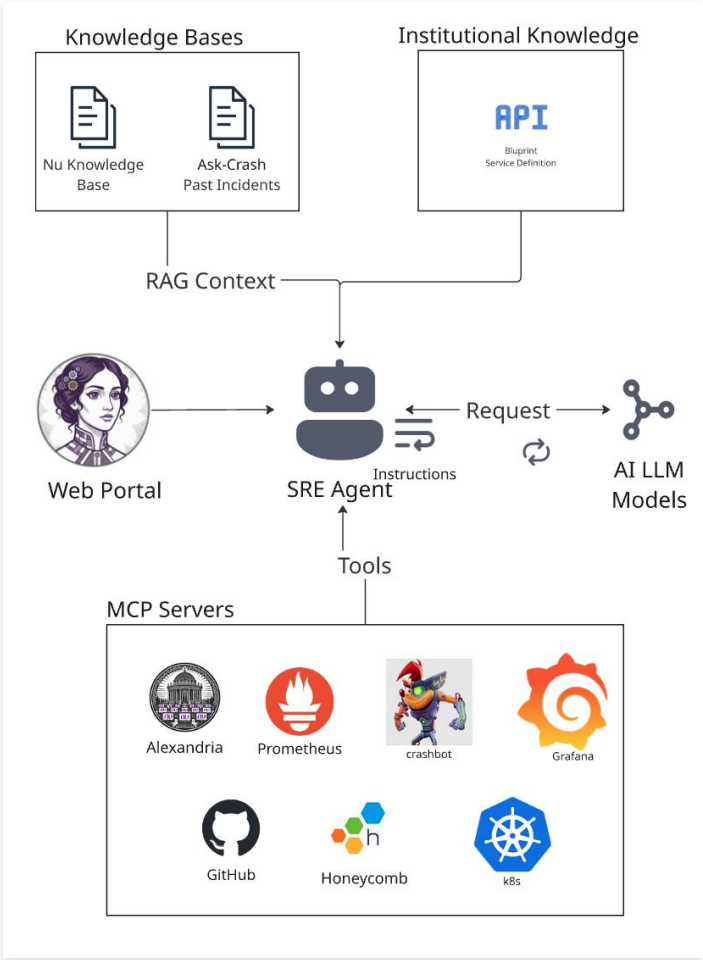
Core Functions:

Orchestration: Coordinates LLM reasoning with Nu's internal data.

Observability: Provides real-time access via **MCP servers**.

Institutional Knowledge: Injects tribal knowledge and incident history.

RAG Context: Supplies relevant historical data for grounded answers.



The Anatomy of an Effective AI SRE Agent

The effectiveness of any AI SRE agent comes down to three core pillars — intelligence, data, and knowledge — orchestrated by a central loop.

Model Intelligence

The LLM is the single most important component. A more capable model produces sharper reasoning, better tool selection, and more accurate conclusions.

Data Access

The LLM can only reason about data it can see. Integrations with observability, source control, and infra APIs translate to richer context and better answers.

Institutional Knowledge

Beyond foundational knowledge of tech like Kubernetes and Prometheus, agents must understand internal architecture, naming conventions, and past incident logs.

The Agent Loop

The orchestrator makes these pillars work together in a cycle of reasoning, tool invocation, and learning from new information.

Key Insight: The chain only works if every pillar holds — weakness in one collapses the whole loop.

Three fundamental problems — and how we solved them

The Dory Problem

Challenge

LLM forgets the entire investigation between turns.

Solution

Stateful orchestrator loop that reconstructs the relevant context on each turn.

Context Overflow

Challenge

23K+ metrics, billions of log rows, MBs per tool execution instantly fills context.

Solution

Statistical summarization and data analysis (zero LLM cost) + concurrent LLMs.

Generic Tool Use

Challenge

The LLM knows your tools exist but doesn't know how to use *your* tools — your label conventions, your scale, your gotchas.

Solution

Expert guides for each MCP (Model Context Protocol).

The Stateful Orchestration Loop

Solving the LLM memory problem through iterative refinement

01. Discovery

Classify question and extract service context.

Load matching flow and prompts directly from DB.

02. MCP Loaded

LLM selects specific MCPs to load.

Tools and expert guides are injected together into session context.

03. Investigating

Core Recursive Loop

- Up to 30 iterations (configurable per flow).
- Execute tools.
- Compress results.
- Condense prompt.

04. Finalizing

Final summary with key findings.

- Reference evidence URLs.
- Cost breakdown.
- Full state persistence.

The stateful loop enables the agent to navigate complex reasoning paths while keeping the context window focused and cost-efficient. 40-60% prompt-token reduction via section variants.

Expert Guides: Responsible Tool Use

The Insight

The LLM doesn't inherently know your tool's nuances. **Expert Guides** inject institution-specific instructions during tool loading.

Example

"When using Prometheus..."

- DON'T list all metrics (23K!).
- DO search by name pattern & apply label filters.
- DO limit time ranges.

Efficiency Gain: 23,000 raw metrics reduced to **~14 targeted queries** per investigation.

Progressive MCP Discovery

Let the LLM decide what to load

Discovery Tool Approach: Instead of dumping 200 tools into context at the start, give the LLM a discovery tool.

`get_tools_from_mcp("mcp_name")`

allows the LLM to discover tools available from specific MCPs.

Dynamic Selection: The LLM chooses based on context.

"This is a latency question — I need Prometheus and Honeycomb data, probably not Crashbot MCP."

Efficient Scaling: On MCP load, tools and expert guides are injected together. The prompt only grows with tools that are **actually relevant**.

Pluggable by design — adding a new MCP is a configuration entry, not a code change.

The LLM Matters

Same Expert Guides, same data — the model changes everything.

GPT-4 Behavior

01 **Shallow**
Finds one data point and rushes to conclude. "Investigation complete!"

02 **No correlation**
Like a developer who finds the first Stack Overflow answer and calls it a day

03 **Couldn't compensate**
Expert Guides helped, but shallow reasoning limited outcomes

GPT-5 Behavior

01 **In-depth**
Keeps going, cross-references sources, follows the evidence chain until it finds reasonable RCA signals

02 **Senior SRE behavior**
Says "let me check one more thing" and actually finds the root cause

03 **Biggest quality leap**
Same Expert Guides, dramatically better outcomes. The model upgrade mattered most.

The model is one of the main pillars.

Data Compression

Two-tier compression for high-density observability data

Observability data doesn't fit in LLM context. This two-tier approach solves it by prioritizing efficiency before intelligence.

Tier 1: Statistical (Zero LLM cost)

01. Percentiles: Mean, median, P25/P75/P95/P99 from raw timeseries.

02. Change Points: Detect exactly when the trend shifted.

03. Anomaly Detection: Z-score flags >2σ deviations automatically.

Real example: Honeycomb response → 26,991 → 2,343 chars (≈93% reduction). Triggered when tool output > 4,000 chars.

Tier 2: LLM Summarization

01. Non-statistical data: Independent LLM summarization call for data that can't be statistically summarized.

02. MCP-Specific prompts: Each MCP has its own summary prompt. Trino log summarization ≠ GitHub diff summarization.

Intelligence is used only where statistical reduction fails.

Six investigation flows headlining the platform

Different questions need different strategies. The LLM auto-classifies and loads the right playbook.

RCA Investigation

Adaptive general troubleshooting

"Why is service X behaving oddly?"

Latency Deep Dive

P50-P99 analysis, bottlenecks

"High P99 on endpoint Y"

Kafka Deep Dive

Consumer lag, partitions

"Kafka lag increasing"

Dependency Upgrade Audit

GitHub + CI/CD — correlate library upgrades with health

"Was the latest upgrade successful?"

Service Topology

Upstream, downstream, infra dependencies

"Show me topology for service X"

Alert Investigation

RCA from alerts

"Investigate this Alert Manager alert"

…and more (Service Mesh, HTTP Service Deep Dive, Prometheus Canonical Dashboard, custom flows defined by teams)

What an investigation actually looks like

One real production example — anonymized service name, real numbers.

User question

"Investigate if there are any issues with payments-service-x"

SRE Agent output

Service Status: Operational — elevated rate limiting

- 16 pods running · 1,698 req/s
- 200s: 1,396/s · 429s: 282/s (16.7%) · 500s: <1/s
- P95 latency <0.4s · No pod restarts

Primary finding: Excessive rate limiting (429s) — likely client retry storm or limits too restrictive.

Recommendation: Identify throttled clients via per-client aggregation on services_http_requests_total{status="429"}.

14 targeted Prometheus queries · 4 iterations · ~6,800 tokens

The architecture generalizes beyond SRE into an agentic platform

1 Fraud Investigation Flows

Patterns across transaction logs and customer signals — detect coordinated attacks across thousands of accounts.

2 RCA Flows

The original use case — reliability investigations across the full o11y stack.

3 Customer Support Flows

Ticket triage with tribal knowledge from past tickets and service docs.

4 Marinator

AI bot powered by the SRE Agent that posts Sev1/Sev2 leadership updates every 30 min.

Six production lessons from building an AI SRE agent

1 Model quality is highest-leverage

GPT-4.x -> 5.x had more impact than any architecture change. Better reasoning = fewer wasted iterations.

2 One-size-fits-all fails at scale

Custom flows per investigation type produce targeted, deep analysis.

3 Expert Guides are your moat

Any agent can call Prometheus. Only yours knows not to list 23K metrics first.

4 Your agent needs same o11y as your services

Real-time streaming of tool calls, cost tracking, full persistence - engineers need to see what it's doing.

5 Separate prompts from code

DB-backed versioned prompts let experts improve quality without deployments. Roll back instantly if things worsen.

6 Live updates turn engineers into participants

Streaming reasoning keeps engineers engaged. They follow along, catch wrong turns early, and have "aha" moments.

Three things to remember when building AI-powered RCA

Evaluate the Pillars

Data access, tribal knowledge, and the LLM. If your data and tribal knowledge can leave your network, consider buying. If they can't (compliance, scale, or sensitivity), you build.

Invest in Guardrails

Expert prompts, custom flows, stateful pipelines - the difference between a toy demo and production-grade RCA.

Ship Fast, Iterate Faster

Each release solved real production problems. Prototype in weeks, not months. Let real incidents teach your agent.

Questions?

Yevgeny Gladun
Staff Engineer · Runtime Platforms
linkedin.com/in/yevgeny-gladun

Letícia Mota
Product Manager · Nubank
linkedin.com/in/leticiamirandaf

About Nu - Careers
https://international.nubank.com.br/careers/