vasco

By Guillaume Jacquet
CEO & co-founder of Vasco

CLAUDE COWORK IS IMPRESSIVE.

Here's the infrastructure it assumes you already have.

Last month, Claude told one of our customers their pipeline was on track. It had access to HubSpot, Gong, Stripe, and Slack via MCP. They were missing target by 21%. Nine deals hadn't been touched in 30 days. A Gong call flagged a budget blocker. A Slack thread flagged a competitor displacement. Claude had access to all of it — and the weekly summary said "stable, on track."

IN THIS PAPER

Claude Cowork connects to your tools, automates pipeline reports, and answers revenue questions in plain language. The productivity gains are real. But Claude doesn't verify. It reasons. And it reasons on whatever you give it — clean or dirty, complete or missing, reconciled or contradictory.

> Claude doesn't say "I'm not sure." It says "here's your weekly pipeline summary" — and it's wrong in a way that looks exactly like being right.

For RevOps teams building their GTM on top of AI agents, the question isn't whether Claude Cowork is useful. It's whether the data underneath it is trustworthy enough to act on.

1. Claude won't tell you it's wrong

Four MCPs and a false sense of security

MCP (Model Context Protocol) is the connector layer. It gives Claude access to your CRM records, call transcripts, billing events, Slack threads — whatever the connected systems hold. Most teams connect HubSpot, Gong, Stripe, and Slack and assume they've covered their bases.

They haven't. Four pipes are still pipes. Here's what even a full MCP setup doesn't give Claude:

No shared definitions

What's an SQL? How do you calculate NRR? A "qualified lead" in HubSpot, a "high-intent signal" in Gong, and a "converted trial" in Stripe might describe the same account — or three different ones. Claude can't reconcile what nobody has defined.

No identity resolution

The same contact exists as a HubSpot record, a Gong participant, and a Stripe customer. Without identity resolution, Claude treats them as separate entities. The Gong call where budget was flagged doesn't connect to the deal where it matters.

No plan or targets

Claude can pull closed-won totals from HubSpot and subscription data from Stripe, but doesn't know your number, which motion carries it, or how pace compares to target. $180K is a neutral number without a $240K benchmark next to it.

No outcome memory

Claude reasons on current state. It doesn't know the last 4 deals with this exact pattern — stalled at Stage 3, no executive contact, competitor mentioned in calls — all closed-lost. Every deal is assessed from scratch.

No cross-tool sequencing

Each MCP delivers its own data model. HubSpot thinks in deals and contacts. Gong thinks in calls and speakers. Stripe thinks in subscriptions. Slack thinks in threads. Nobody has told Claude how these relate to each other, how to sequence them into a deal journey, or which signal overrides which when they conflict.

The gap isn't data access — it's data architecture. What's missing is the layer between your tools and Claude that reconciles identities, enforces definitions, sequences events into journeys, and remembers what patterns led to which outcomes.

2. What "confidently wrong" actually looks like

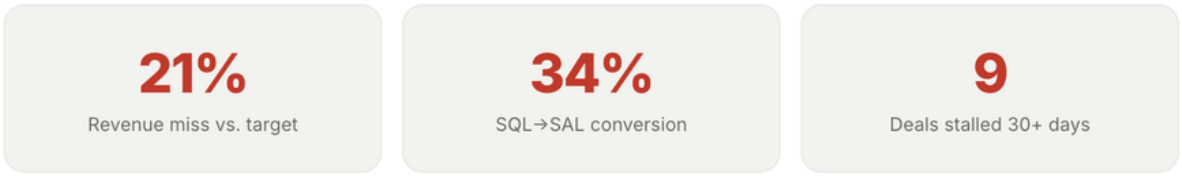
A real case study — anonymized data, real gap

This happened to one of our customers. B2B SaaS, mid-seven-figure ARR, 38 active accounts. Claude Cowork was connected to HubSpot, Gong, Stripe, and Slack via MCP — a full setup. We're anonymizing details, but the data is real.

To be fair: Claude with four MCPs doesn't produce zero value. It successfully pulled deal records, surfaced call summaries, and generated a readable weekly report. The problem wasn't that it failed — it's that it succeeded at the wrong question. It reported what the data said. It couldn't tell you what the data meant.

During the weekly review, Claude reported the pipeline as stable: 9 active deals, 75% MQL-to-SQL conversion, $240K target appearing on track. No alerts.

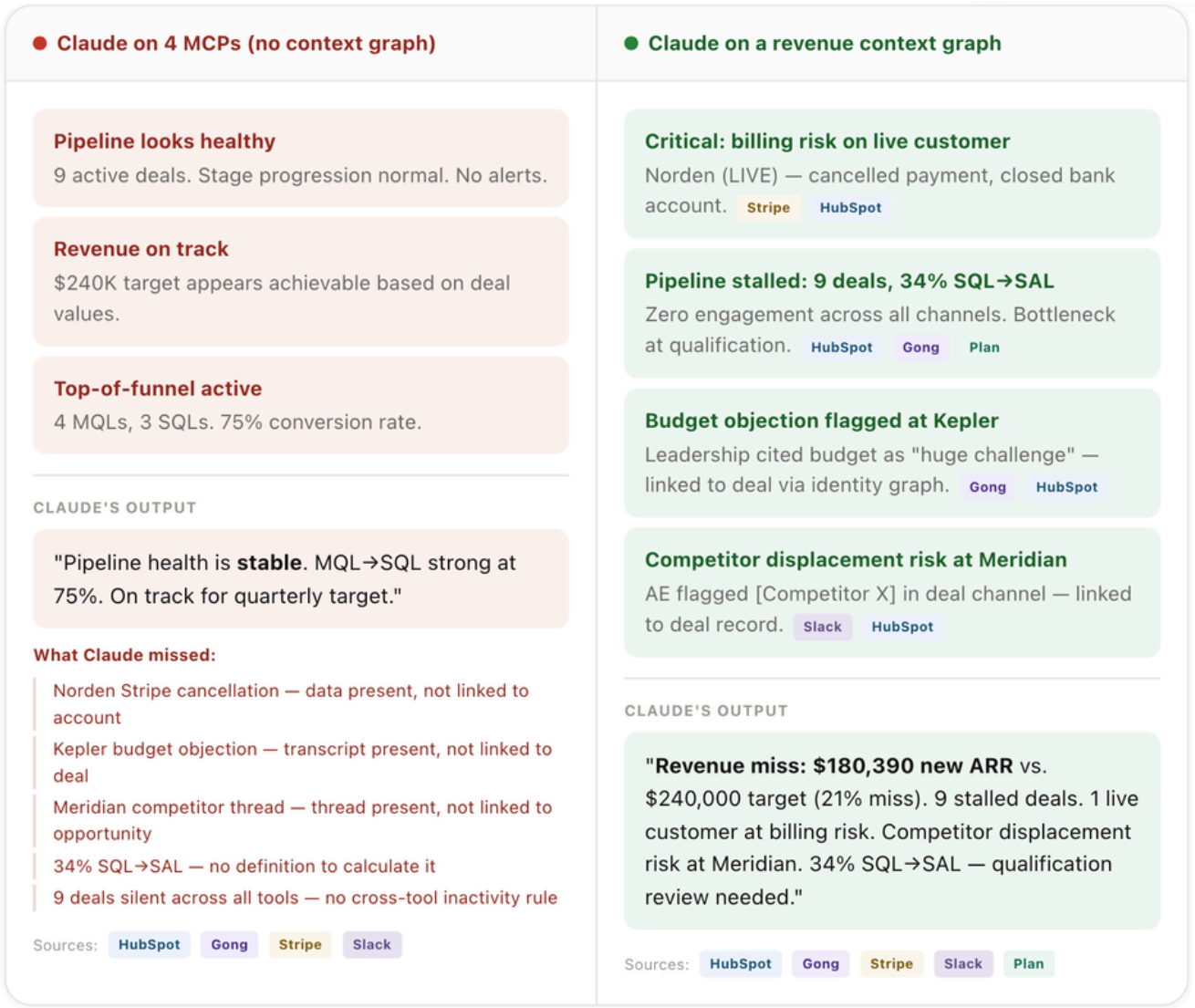
The same company's data, reconciled through a revenue context graph, told a different story:



- **$180,390 in new ARR against a $240,000 target.** A 21% miss — a structural gap no tool in the stack encoded because none of them held the target.
- **34% SQL-to-SAL conversion.** Claude reported MQL-to-SQL (75%) because that's what HubSpot tracks cleanly. The conversion that actually mattered — qualification to acceptance — required a definition the MCP doesn't carry.
- **9 deals with zero engagement for 30+ days.** No calls in Gong. No emails. No stage changes. HubSpot still showed them as "active." Claude could see the Gong silence and the HubSpot stasis separately — but had no rule that connects cross-tool silence to a stalled deal.
- **Norden (live customer) — cancelled payment, closed bank account.** The billing event was in Stripe. Claude had Stripe access. But the Stripe customer ID and the HubSpot company record aren't the same entity to Claude without identity resolution. It reported Norden as healthy.
- **Kepler's leadership flagged budget as a "huge challenge"** on a Gong-recorded call. Claude had access to Gong. But the call participants didn't map cleanly to HubSpot contacts — so the budget signal floated as an unconnected data point, never attached to Kepler's deal.
- **A Slack thread flagged a competitor displacement at Meridian.** The AE wrote "they're evaluating [Competitor X] — we need to move fast." Claude had Slack access. But it had no schema to associate a deal-channel thread with a HubSpot opportunity.

A RevOps lead checking a spreadsheet would have caught the revenue miss — that's a fair pushback. But the point isn't that this data was impossible to find manually. It's that t**he team was relying on Claude to find it**, and Claude reported "on track" with the same confidence it would have used if the data were clean. The automation didn't just miss signals — it replaced the manual check that would have caught them.

> Four MCPs. Every signal present in the data. The links between them missing. Claude reported "stable" — and the team moved on.



3. Won't Claude just get better at this?

What model improvements will and won't solve

Probably. MCP is evolving. Claude's reasoning capabilities improve with every model generation. It's reasonable to expect that cross-tool reasoning, entity matching, and multi-source synthesis will get meaningfully better over the next 12–24 months.

But there are categories of knowledge that no model improvement will conjure from your data, because the knowledge doesn't live in your data:

Your definitions

What counts as an SQL at your company? How do you define churn vs. downgrade? These are business decisions, not data patterns. No model will infer them from raw records because the same data supports multiple valid definitions.

Your plan

Quota targets, motion-level benchmarks, segment-specific thresholds. These exist in spreadsheets, board decks, and planning sessions — not in any system Claude connects to.

Your outcome history

Which deal patterns led to wins, losses, or churn? CRMs store deal status. They don't store the causal chain — and no reasoning model can reconstruct what was never recorded.

Your identity map

Even if Claude gets better at fuzzy matching, revenue decisions need deterministic, maintained, auditable identity resolution. "Probably the same account" isn't good enough for a billing risk alert.

Claude will get better at reasoning across data it can access. It won't get better at reasoning across data that doesn't exist. Definitions, plans, outcome tags, and identity maps are infrastructure — they need to be built, not inferred.

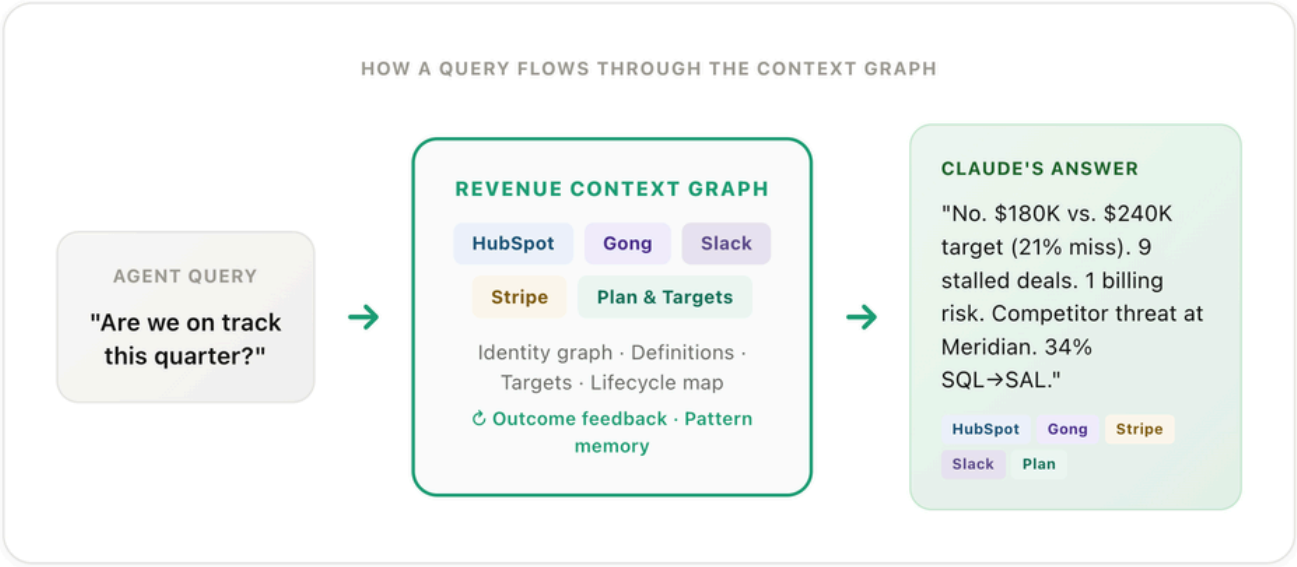
4. What Claude needs: a revenue context graph

The practical bridge between raw tools and AI reasoning

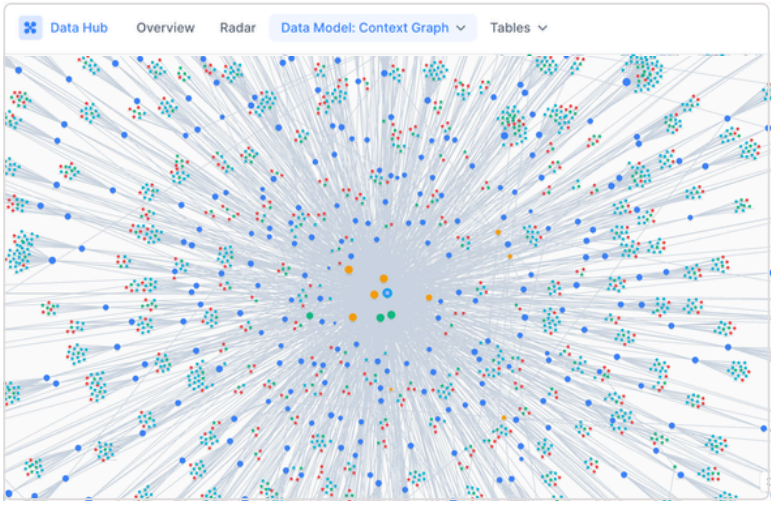
HubSpot's Dharmesh Shah recently wrote about context graphs — the idea that AI needs more than raw data; it needs the relationships and decision traces that connect data to meaning. He added a sharp caveat: most companies aren't ready.

He's right about the gap. But revenue teams don't need a fully instrumented decision-trace graph. They need a specific, bounded version: **a revenue context graph** — one that connects deal history, conversation data, billing signals, Slack decision threads, and plan targets into a single queryable layer an agent can reason across.

You don't need to instrument every agent decision. You need to connect the five or six systems that already hold 90% of your revenue context and give them a shared schema, shared identities, and shared definitions.



> Dashboards tell you what happened. Context graphs tell you why. Claude without a context graph just tells you what happened faster.



Vasco's context graph

5. The three layers Claude needs underneath it

Foundation, planning, and context

LAYER 1

🗺️ Foundational — your business map

Motions, channels, lifecycle stages, definitions, ownership rules — loaded before Claude speaks.

In the case study: This is what let the agent know the partner-referral channel contributed 2 of 3 SQLs. No MCP categorizes deals by motion — the foundational layer does.

LAYER 2

🎯 Planning — deterministic numbers

Targets by rep, quota by motion, benchmarks by segment.

In the case study: This turned "$180,390" from a neutral observation into a **21% miss**. Without the planning layer, Claude had no target to compare against — so it reported the number as "on track."

LAYER 3

🔗 Context — the connective tissue

Deal histories, transcripts, billing events, Slack threads — connected via identity resolution and sequenced into deal journeys. Every finding traces to its source.

In the case study: This linked Norden's Stripe cancellation to their HubSpot account, connected Kepler's Gong transcript to their deal record, and associated the Meridian Slack thread with the active opportunity. Same data the MCPs already had — now actually connected.

6. How the graph actually learns

Outcome tagging, pattern extraction, and continuous recalibration

A context graph that only connects systems is useful. A context graph that learns from outcomes is transformational. Here's how the feedback loop works mechanically.

1

Outcome tagging

Every deal, every customer relationship gets a tagged outcome — Customer Won, Churned, Expanded, Downgraded, Deal Stalled. This isn't AI-generated. It comes from the CRM close reason, validated against billing data and enriched with timing from stage transitions. The graph stores the outcome alongside the full deal journey: every stage change, every call, every Slack thread, every billing event, sequenced in order.

2

Pattern extraction

With enough tagged outcomes (typically 50+ closed-won and 50+ closed-lost for statistical relevance), the graph runs comparative analysis across deal journeys. This is closer to cohort analysis than machine learning — it groups deals by outcome and identifies which signals, timings, and sequences differ systematically between groups. The output is a set of correlations, each with a confidence score based on sample size and consistency.

3

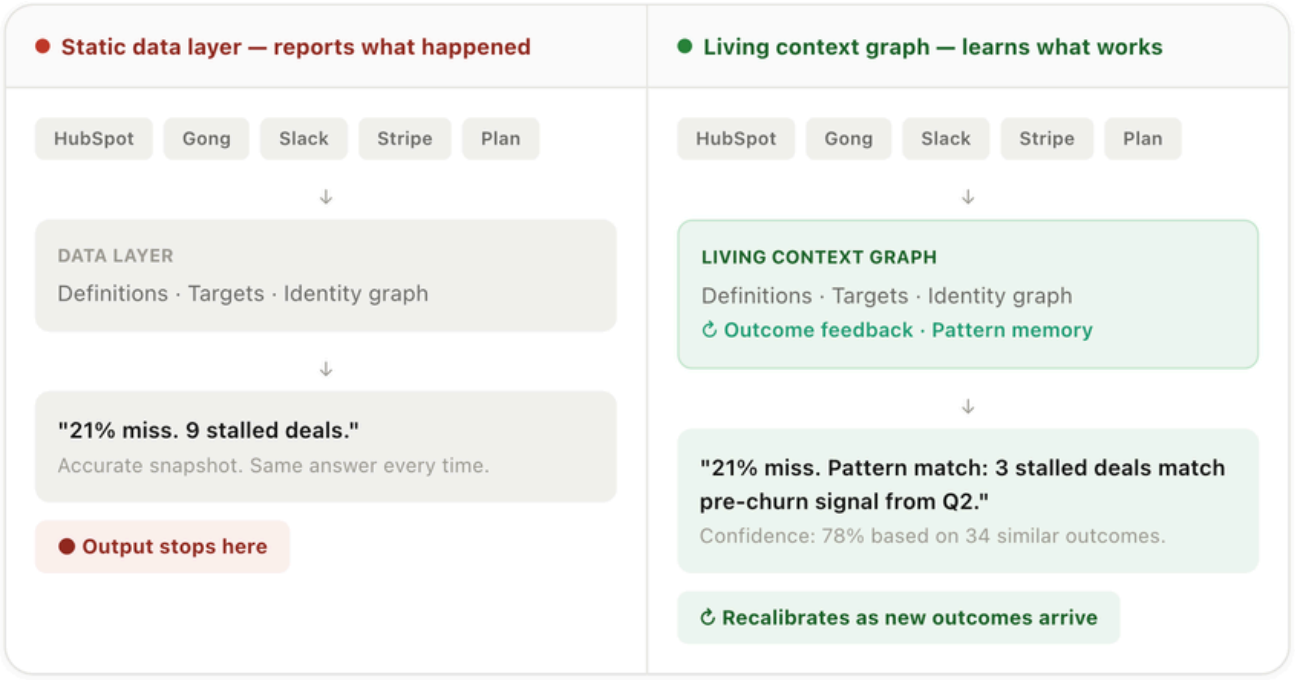
Correlation validation

Not every pattern is actionable. The graph surfaces correlations; RevOps validates them. 'Deals with CISO engagement before Stage 3 close 2.4× more' is a hypothesis until the team confirms it reflects a real causal mechanism versus a spurious correlation. The graph flags the pattern. The team decides whether it becomes a playbook rule.

4

Continuous recalibration

As new outcomes flow in, correlations strengthen or weaken. A pattern that held for 6 months may stop holding after a pricing change or market shift. The graph re-runs pattern extraction on a rolling basis and flags when a previously strong correlation degrades — so playbooks don't run on stale assumptions.



Example correlations the graph surfaces once it has sufficient outcome data:

↑ CORRELATES WITH CONVERSION

CISO engaged before Stage 3

Deals where a compliance stakeholder entered discovery before technical evaluation closed 2.4× more often. (n=47, confidence: high)

↓ CORRELATES WITH STALLING

No executive contact by day 30

72% of deals without a C-suite touchpoint by day 30 stalled before decision stage. (n=63, confidence: high)

↑ CORRELATES WITH EXPANSION

Multi-department usage by month 4

Customers who expanded to a second team within 4 months had 3.1× higher NRR at 12 months. (n=29, confidence: moderate)

↓ CORRELATES WITH CHURN

Support spike + no QBR

Accounts with 3× support ticket increase and no executive review in prior quarter churned at 68%. (n=22, confidence: moderate)

> The graph doesn't just report your pipeline. It writes, validates, and maintains your playbooks — based on what actually works. But it can only do this if it knows the outcomes.

7. What this changes in practice

ICP drift and rep coaching — from the same customer

ICP drift: 38% of pipeline was structurally unlikely to convert
Same customer. 142 won deals over six months. The context graph surfaced a clear winning profile: Series C+ Fintech and HealthTech, 200–500 employees, compliance pain points. These closed in 132 days at $62K ACV.

38% of active pipeline was outside this ICP. Sub-50-employee companies had an 8% win rate and churned within 12 months. Legacy Retail: 10-month cycles, 20% lower ACV.

✓ ICP-ALIGNED DEALS		× NON-ICP DEALS (38% OF PIPE)	
Avg. ACV	**$62K**	Avg. ACV	**$38K**
Cycle	**132 days**	Cycle	**10+ months**
Win rate	**High**	Win rate	**8%**
Retention	**12mo+**	Retention	**Churned <12mo**

The output: reallocate 60% of outbound SDR capacity to mid-market Fintech. Stop targeting sub-50 companies. Pivot messaging to "Compliance Automation for International Expansion" — the pain in 80% of won deals where the CISO was the buyer.

Rep coaching: speed, not effort — and the graph proved it
7 reps, 90 days. The graph revealed a pattern no CRM report could: **top performers closed 3× larger ACVs** ($25K vs. $7,560) despite lower activity volume. The differentiator wasn't effort — it was velocity.

$25K	$7.5K	<2d
Top performer ACV	Core group ACV	Top SQL→SAL velocity

Top performers moved SQL→SAL in under 2 days. Core group: 3.2+ days. Faster qualification correlated with larger deals and higher win rates — not because speed is inherently good, but because the behaviors that produce speed (engaging decision-makers early, qualifying budget in the first call, mapping the buying committee before Stage 3) are the same behaviors that produce wins.

The graph validated this from transcripts: top performers focused on 4 strategic personas (C-suite, Finance) early in discovery. Core reps spread across 10+ technical contacts. Won deals had 5.2 stakeholders. Lost deals had 2.1 — the wrong 2.1.

The coaching playbook — early qualification discipline, executive stakeholder mapping, proactive ROI defense — was written by the graph, from patterns, not from gut feel. And it updates as new deal data flows in.

vasco

8. What breaks when you build this yourself

Scope, timeline, and the maintenance trap

You can build a revenue context graph in-house. Some teams do. But the failure mode is almost always the same: you ship v1, it works for a quarter, then it drifts because no one maintains it.

Definitions

Encode every lifecycle stage, every conversion definition, every metric formula — across every system. Then get the entire GTM org to agree on them.

2–3 months for alignment alone

Cross-tool joins

Connect CRM records to call transcripts, billing events, Slack threads. Maintain identity resolution across systems that don't share IDs.

Breaks on every tool change

Historical reconstruction

Stage transitions, deal velocity, conversion trends — these require event-level history, not current-state snapshots. Most CRMs don't store this natively.

4–6 months to backfill

Ongoing maintenance

Definitions change. New reps join. Targets shift quarterly. A context graph that isn't maintained decays within weeks. The build is 30% of the work; the upkeep is 70%.

Permanent headcount

Realistic timeline from scratch: 12–18 months. Most teams restart at least once. If you've done all of this and maintain it continuously, you don't need a platform. If you haven't — that's exactly what Vasco was built for.

WHAT WE HANDLE SO YOU DON'T HAVE TO

Every one of these is a multi-week engineering project. Vasco does all of them — continuously, without code.

🔗

Identity Resolution

Same person, three emails, two CRMs.

Deterministic matching across HubSpot, Gong, Stripe, and Slack — so the budget objection on a call attaches to the deal it belongs to.

⚖️

Cross-Source Reconcile

Stripe says paid. HubSpot says churned.

When systems contradict, the graph reconciles the truth before Claude ever sees the data.

🏗️

Skipped Stages

Deals skip your process. We reconstruct what happened.

A deal jumps from MQL to Closed-Won. The graph reconstructs the missing transitions from activity data.

🏢

Contact → Account

47 contacts at Microsoft. One account, one story.

Roll up contact-level activity to a company-centric view — the level where revenue decisions actually happen.

✨

Enrichments

Missing ICP fit, intent, firmographics — filled before agents reason.

Incomplete records produce incomplete answers. The graph enriches before the agent queries.

💬

Conversation Placement

Every call mapped to the journey moment it mattered.

Map Gong transcripts to deal stages and lifecycle moments — not just dates.

🧮

Calculated Fields

Formula fields, roll-ups, overrides — absorbed silently.

Custom formulas and workflow-driven overrides that MCP may not surface. Absorbed into the graph automatically.

📊

Forecasts in Context

Actuals against targets. Both live in the graph.

Hold your plan and your actuals in the same queryable layer — so Claude can compare, not just report.

🔄

Reactivation Detection

Closed-lost came back. Full history preserved.

Detect re-engaged prospects, preserve prior deal history, and give Claude the full context of what happened last time.

9. The right mental model

	REVENUE DATA LAYER	CLAUDE COWORK	BOTH TOGETHER
What it does	Cleans, reconciles, structures. Resolves identities. Encodes definitions, plan, context. Learns from outcomes.	Reasons across data, answers in natural language.	Revenue intelligence that queries, trusts, acts — and improves itself.
Produces	Trustworthy revenue reality + pattern library from outcomes.	Fast outputs — only as accurate as the input.	Fast outputs on true numbers, with playbooks that evolve.
Time	12–18 months from scratch. Weeks with a platform.	Available today.	**As fast as your data foundation allows.**

10. FAQs

We have HubSpot, Gong, Stripe, and Slack connected via MCP. Isn't that enough?

You have access. You don't have architecture. Each MCP delivers its own schema, its own identities, its own vocabulary. Without a layer that reconciles those into a shared model — with deterministic identity resolution, shared definitions, and plan context — Claude reasons on disconnected fragments. The case study above had all four MCPs connected and still missed a 21% revenue gap.

Won't MCP evolve to handle cross-tool reasoning natively?

MCP will improve at data access and possibly basic entity matching. But definitions, plan targets, outcome history, and validated identity maps are business decisions encoded as infrastructure — not data patterns a model can infer. Even perfect cross-tool reasoning produces wrong answers if the definitions underneath are missing or inconsistent.

How is this different from Clari, Gong Analytics, or HubSpot's built-in reporting?

Those tools analyze the data they own. Clari focuses on pipeline and forecast. Gong Analytics surfaces call patterns. HubSpot reports on CRM data. None of them reconcile across all four systems, enforce shared definitions, hold your plan, or build an outcome-tagged pattern library. A context graph is an infrastructure layer underneath all of them — it doesn't replace any tool, it connects them.

Should I wait until my data is clean before using Claude?

No. Claude is useful today for drafting and summarizing. What you should solve first is trusting its output for pipeline reviews, forecasting, and anything where a wrong number leads to a wrong decision. Use Claude now. Just know where the trust boundary is.

What does the graph need to start learning from outcomes?

Tagged outcomes (Won, Lost, Churned, Expanded, Stalled) mapped to complete deal journeys — stage changes, calls, billing events, Slack threads, all sequenced. The statistical floor is roughly 50+ outcomes per category for reliable pattern extraction. Below that, the graph connects and reports; above it, the graph starts identifying what works and what doesn't.

Who should own the MCP configuration?

RevOps. MCP connections inherit whatever permissions the authorizing user holds. Treat Claude's data access like any integration: explicit authorization, documented scope, named owner.