CASE STUDY

N8N · 5 AGENTS · 31 NODES

PRODUCTION MULTI-AGENT SYSTEM

Groundwork

Multi-agent AI for B2B cold email — grounded in real research, verified before it sends.

Reply rates on cold email fell to **3.43%** in 2026, down from **8.5%** in 2019. This is the system built to reverse that curve for one sender at a time.

RESEARCH · SYNTHESIS · GENERATOR · CRITIC · SMYKM

Tesfa Michael

Applied AI Engineer — Agentic Systems & n8n Automation

Built & validated end-to-end

Multi-agent orchestration, agentic workflow design, MCP integration · 2026

The Summary

Reply rates on cold email fell to 3.43% in 2026, down from 8.5% in 2019 —

less than half, in seven years, across an industry that spent that entire period adding more automation, not less.

That is the *Instantly 2026 Cold Email Benchmark Report*'s headline number, and it is worth sitting with, because the story most vendors tell is the opposite one: more sending, more personalization tokens, more AI in the loop, better results. The data says the tools got faster and the replies got rarer. Somewhere in that gap is the actual problem this project set out to solve.

Groundwork is a production multi-agent AI system built to never fabricate a personal detail. It runs a cold-email lead through five purpose-built AI agents (Research, Synthesis, Generator, Critic, and an orchestrating SMYKM Agent) before a single word reaches a prospect's inbox, implementing Samantha McKenna's "Show Me You Know Me®" (SMYKM) cold-email methodology: every email must open with a personal fact specific enough that it would mean nothing to anyone else, and every one of those facts has to be traceable, source URL and all, back to something the recipient actually posted, said, or wrote.

The system sources leads through an Apollo-style Apify actor, researches each one individually, drafts a first pass, has an independent model critique it against a six-dimension rubric, allows exactly one revision round, and writes the result to Airtable, flagged either `ready_to_send` or `needs_manual_review` and never silently either one. It does, in short, the research-before-writing work a human SDR would do by hand, then verifies it before anything sends.

6.6%	4%	100+
REPLY RATE	SPAM-FLAG RATE	EMAIL TEST BATCH

↳ Meaningfully ahead of the 4.1% reply / 7.8% spam baseline documented for AI-alone cold outreach.

Across a real test batch of just over 100 emails, it produced a 6.6% reply rate and a 4% spam-flag rate. That's meaningfully ahead of the 4.1%/7.8% baseline documented for AI-alone cold outreach in the research that shaped this build, on a sample too small to call definitive, which this case study says plainly rather than rounds up.

02 CAPABILITIES

Skills Demonstrated

What this build proves, mapped to where it's proven.

Every capability below is tied to a specific decision documented later in this case study — nothing here is a claimed skill without a build behind it. The six that follow are the differentiators; the full set rounds out the toolkit.

2.1 Signature Capabilities

Multi-Agent Orchestration

Five purpose-built agents — Research, Synthesis, Generator, Critic, SMYKM orchestrator — with explicit handoffs and shared state, not one model doing every job.

→ §6 ARCHITECTURE

Context Engineering

System prompts as structured contracts: source-priority rules, banned-phrase lists, and a tool-call budget that keeps "thorough" from becoming unbounded.

→ §6.1 ARCHITECTURE

Evaluator-Optimizer Design (LLM-as-Judge)

A Critic drawn from a different reasoning family than the Writer scores every draft against a 6-dimension rubric — the documented fix for same-family self-preference bias.

→ §7.3-7.4 DECISIONS

Tool-Use / Function-Calling Design

A hard 7-call research budget with a strict source-priority order, so agentic tool selection stays bounded instead of open-ended.

→ §6.1 ARCHITECTURE

Structured Outputs (JSON-Schema Enforcement)

Every handoff — Brief, draft, critique, final record — is a validated schema, never free text passed downstream on hope.

→ §6.3 ARCHITECTURE

MCP Integration

Apify MCP and Airtable MCP wired directly into the Research Agent's toolbelt, not simulated with plain REST calls.

→ §6.1 ARCHITECTURE

2.2 Full Capability Set

Agentic AI & LLM Engineering

Multi-Agent Orchestration · Agentic Workflow Design · Context Engineering · System-Prompt Design · Function Calling / Tool Use · Structured Outputs (JSON-Schema) · Evaluator-Optimizer / LLM-as-Judge · Grounding & Anti-Hallucination Design · Multi-Provider Model Routing · MCP Integration

Reliability & Systems Engineering

Agent Reliability Engineering · Idempotency & Checkpointing · Deterministic Gating / Harness Design · Production-Grade System Design · Audit Logging & Observability · Error Handling & Fallbacks

Data & Integration

REST API & Webhook Integration · Structured Data Transformation (JSON) · Data Validation & Normalization · API Authentication (OAuth2 / API Keys) · n8n Workflow Automation · Relational Data Modeling

Domain & Delivery

Email Deliverability & Compliance (SPF/DKIM/DMARC, GDPR/CAN-SPAM)

Technical Documentation & Case-Study Writing

The Context

PROJECT	Groundwork — SMYKM-based B2B cold-email automation system
ROLE	Applied AI / Automation Engineer — agentic workflow design, prompt & context engineering, build and validation
CORE PLATFORM	n8n (self-hosted workflow orchestration)
DATA LAYER	Airtable (three-table relational model), Postgres (per-agent conversational memory)
MODELS IN PRODUCTION	Qwen 3.7 Max (Research), DeepSeek V4 Pro (Synthesis), Claude Sonnet 4.6 via AWS Bedrock (Generator), Gemini 3.5 Flash via Google Vertex (Critic), Mistral Medium 3.5 (SMYKM orchestrator)
LEAD SOURCING	A third-party, Apollo-style lead-sourcing API, keyed on LinkedIn URL
WORKFLOW SIZE	31 nodes, 5 agents, 3 Airtable tables
STATUS AT WRITING	Fully built and validated end-to-end through Airtable persistence. Email sending intentionally not wired — a deliberate, documented scope boundary, not an oversight (see Stage 5, §6.1).

This is not a demo built to look impressive in a screen recording. The engineering that makes it work is mostly invisible in a system map: a Research Agent constrained to a hard tool-call budget and a strict source-priority order so “thorough” never quietly becomes “unbounded,” a Synthesis Agent whose only job is discarding anything without a source URL, and a Generator and Critic built against the exact same rubric so the model writing the email and the model judging it are checking for the same thing.

That discipline, not any single clever prompt, is why five separate model calls produce one coherent, source-traceable email instead of five disconnected opinions. It also ran against real credentials, real API rate limits, and a real batch of prospects. Section 9 documents what broke along the way, individually and without flattering edits, because a case study that only shows the finished pipeline teaches the reader nothing about the part of the job that is actually hard.

The Problem

4.1 The pain point, plainly stated

A founder or small sales team wants to reach 50, 100, or 500 specific people: the exact ICP for their product, not a purchased list, with an email good enough to get a reply. They have two realistic paths. Pay for an SDR (or a team of them), which prices out most companies below a certain size and takes months to hire and train well. Or use one of the AI cold-email tools that promise "personalization at scale," which in practice means inserting `{{first_name}}` and a company-name mail-merge token into a template and calling it personal.

The second path is the one that actually got measured, and the measurement is unflattering.

AI-ALONE OUTREACH		HUMAN-WRITTEN CONTROL	
4.1%	7.8%	10.4%	2.9%
REPLY	SPAM-FLAG	REPLY	SPAM-FLAG

↳ A 12,000-email controlled test, AI-generated vs. human-written outreach.

The researchers' own summary of why is the important part: every AI tool sounded monotonously the same. Technically correct, grammatically clean, strategically sound in structure. Recipients were not fooled by any of it.

4.2 Why the current tools fall short

The failure is not a prompting problem. It's an architectural one. Most AI cold-email tools run a single model call: stuff a lead's name, title, and company into a template prompt, generate a paragraph, send it. There is no separation between finding something true and specific about this person and writing convincingly. A single model asked to do both will, reliably, do the second one at the expense of the first, because it has no external signal telling it that "I saw you're the VP of Sales at Acme" is not personalization. It's mail merge with better manners.

> "AI-assisted personalization works, but only when AI handles the research, not the writing."

The 12,000-email study's own conclusion names the fix directly. Give AI the research job, keep the writing discipline tight. That single sentence is the architectural thesis behind every decision documented in Section 7, and, not coincidentally, exactly what separates a merge-tag email from an SMYKM email.

4.3 Who actually faces this

The reader this pipeline was built for is a small-to-mid-size company, anywhere from a solo founder to an established sales team, who has decided outreach matters but doesn't have the budget or headcount to staff a dedicated research function for every account, and doesn't want to be one more sender in the inbox teaching prospects to ignore cold email faster. That's the affordable-enterprise-AI angle running underneath this whole project: the research discipline a well-staffed SDR org applies to its top accounts, actually reading someone's LinkedIn activity before writing to them, reproduced at a cost structure a small-to-mid-size company can run without hiring a research function to do it.

05 METHODOLOGY

Show Me You Know Me

What it is, and why it works.

5.1 What is Show Me You Know Me?

Show Me You Know Me® (SMYKM) is Samantha McKenna's cold-email methodology, documented in her own Apollo Academy materials and interview transcripts under the #samsales banner. It's built around a single, almost uncomfortably literal design goal: write a subject line that makes absolutely no sense except to the person reading it. Not a mail-merge token. Not an inferred pain point dressed up as a compliment. An actual, verifiable fact about that one person, specific enough that it would mean nothing to anyone else, traceable back to something they actually posted, said, or wrote.

Her own worked examples make the point better than a description of the method could. "Leopard Shorts + SoulCycle + #samsales," sent to a social-media lead at Nike, replied to within a day. "Toby is Australian + 'deep dives' + samsales," sent to the CRO of a recently-acquired company, got a 19-minute reply and a booked meeting. A fraternity reference sent to the Chief Logistics Officer at Tory Burch drew a reply ten hours later that said, verbatim:

> "Your email stood out in a sea of emails."

That last line is the whole thesis in five words. The goal was never cleverness. It was standing out in a sea of emails that all read like they were written by the same tool, because increasingly, they were.

5.2 Why it actually works

Underneath the worked examples, McKenna's structure resolves into seven repeatable elements, and this pipeline's Generator prompt encodes them in this exact order.

1. **Subject line**
   Two or three hyper-specific personal facts, or one plus the sender's company name. Banned from the subject: the recipient's role, company, or industry — the three things every other cold email leads with.

2. **Personalization opener**
   Open directly with the hook. No "I hope this finds you well." No throat-clearing.

3. **Transition bridge**
   One casual sentence connecting the personal hook to the actual reason for writing.

4. **Persona-specific challenge**
   Name a real business problem this role plausibly faces. Describe the problem, not the product.

5. **Value proposition**
   What's at stake if the challenge goes unaddressed, framed as loss rather than gain — loss-framing measures out at roughly twice the persuasive power of the equivalent gain-framed sentence.

   "Six percent of a buyer's inbound leads come from people who change jobs. But what about the other 94 percent you don't hear from?"

6. **Objection pre-handling**
   Name the obvious objection before the reader has to, collaboratively rather than defensively: "While you may already have a project management tool…"

7. **Soft CTA**
   One low-pressure, interest-based question. Never a specific day or time. Never a calendar link on the first touch.

The data backing element seven is, on its own, close to sufficient to explain why SMYKM outperforms a generic template.

4.2% vs 1.4%
Soft, open-ended CTAs vs. hard "book a meeting" asks — nearly 3×. Puzzle Inbox, 200,000+ emails.

21.4% 75–100 words
Response rate for the 75–100-word bracket; past 200 words it collapses toward 3.8%. Grammarly × Boomerang, 4.7M emails.

SMYKM's own structural constraints (one hook, one bridge, one challenge, one soft ask) land an email in almost exactly that band without the writer having to think about word count at all.

None of this is really a copywriting insight, when you look at it from underneath. It's a psychology insight wearing a copywriting costume. The subject-line rule works because specificity is the one thing a template literally cannot fake: either the sender read your LinkedIn post about your half-marathon or they didn't, and there is no clever phrasing that substitutes for having actually looked. The objection pre-handling works because it signals the sender has sent this kind of email before and knows what a skeptical reader is thinking, which is a different signal than "sounds professional." And the loss-framing works because it always does — prospect theory's oldest finding, applied to a cold email instead of an insurance pitch.

SMYKM isn't a clever format. It's what a genuinely well-briefed human SDR already does, systematized into seven checkable steps. The engineering problem this project solved was making an AI pipeline actually earn the right to use that format, rather than performing it over a fabricated hook.

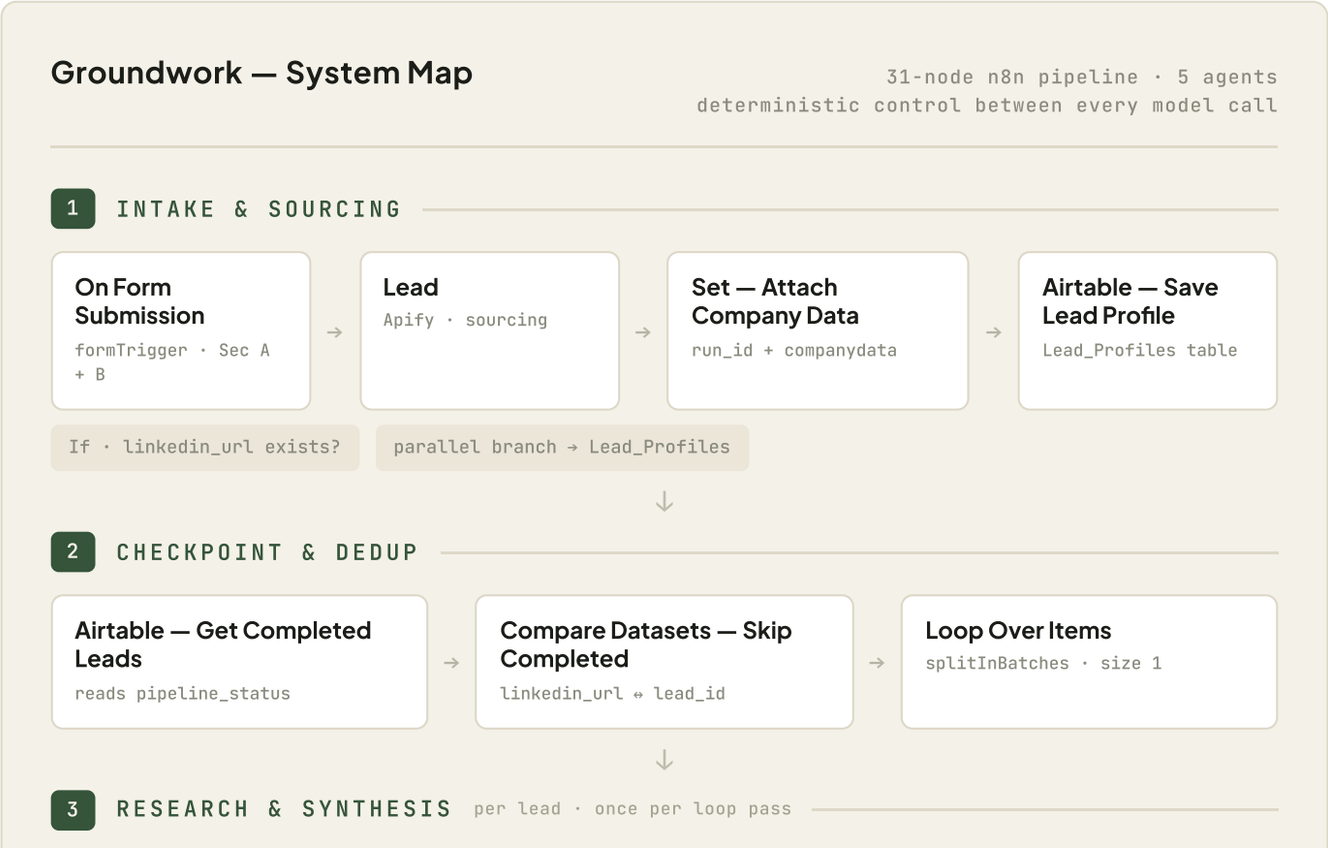
06 ARCHITECTURE

The Architecture

How it actually works.

6.1 The full pipeline

The workflow is 31 nodes, organized into five stages that map directly onto the system map that follows: intake and sourcing, checkpoint and dedup, research and synthesis, the bounded SMYKM draft/critique loop, and persistence and handoff.



Stage 1 — Intake and Sourcing. A form trigger collects two categories of input in one submission. Section A is the sender's own company data (company name, ICP description, target persona, one-line offering, value proposition, common objections) — everything the Generator and Critic agents need to write and evaluate on the sender's behalf. Section B is the targeting criteria (industry, location, company size, job titles, lead count) that gets handed to a third-party, Apollo-style lead-sourcing tool. A `Set — Attach Company Data` node immediately packages the sender's data into a single `companydata` object alongside a `run_id`. Critically, this happens once, before the per-lead loop begins, specifically so five downstream agent calls per lead don't each need their own copy of the same static company data. From there, an `If` node filters out any sourced record missing a `linkedin_url`, since the entire research phase depends on having a stable identifier and a place to look for personal signal. In parallel, every sourced lead is also written to a `Lead_Profiles` table, independent of whether that particular lead ever finishes the pipeline (more in §6.3).

Stage 2 — Checkpoint and Dedup. Before any lead enters the expensive part of the pipeline, an `Airtable — Get Completed Leads` node reads back every lead already marked `ready_to_send`, `sent`, or `needs_manual_review` from a prior run, and a

`Compare Datasets` node filters the current batch down to only the leads that haven't already been through this. This is what makes the workflow safely re-runnable. A campaign that dies partway through, or a nightly re-run against the same target list, never re-researches, re-writes, or re-emails a lead that's already been finalized.

Stage 3 — Research and Synthesis. A `Loop Over Items` node (n8n's Split In Batches, size 1) takes leads through the rest of the pipeline one at a time. Each lead first passes through the Research Agent — the one place in this system that genuinely uses dynamic, unpredictable tool selection, because which platform actually has a usable personal signal for a given person (a LinkedIn post? a podcast appearance? nothing at all?) cannot be known in advance. It has a hard budget of 7 tool calls, a source-priority order (personal LinkedIn activity first, personal blog/newsletter second, podcast or conference appearance third, company-level news dead last), and a strict grading rubric: a fact only counts if it's personal-first, specific, verifiable with a direct source URL, and recent. Its output feeds the Synthesis Agent, whose entire job is condensing raw research into one Personalization Brief, discarding anything without a source URL, ranking what's left, and selecting a single `primary_angle` the rest of the pipeline is not permitted to deviate from.

Stage 4 — The SMYKM Draft/Critique Loop. This is the core of the system and the subject of its own diagram in §7.3. The SMYKM Agent is a coordinator, not a creator; it does not write or judge emails itself. It delegates to two agent-as-tool nodes: the Generator Agent ("The Writer"), running Claude Sonnet 4.6, which drafts a subject and body using only the Personalization Brief and the sender's company data, and the Critic Agent ("The Judge"), running Gemini 3.5 Flash on Google Vertex, which scores the draft against a six-dimension rubric and checks every claimed fact against the Brief. The loop runs a maximum of two rounds. If the draft passes, it's finalized. If it fails twice, it's escalated to a human reviewer. Never silently dropped, never sent anyway.

Stage 5 — Persistence and Handoff. The finalized (or escalated) record is read back from Airtable and passed through a `Format for Sending` node that produces a clean, minimal output: subject, body, `ready_to_send`, `pipeline_status`, `personalization_tier`, `research_gaps`, and `review_notes`. A sticky note on the canvas, deliberately left in place and not hidden, documents that sending itself is not wired up yet, and names exactly where an SMTP, SendGrid, or Hunter.io node would attach.

6.2 Model routing, and why each seat is filled the way it is

AGENT	MODEL IN PRODUCTION	WHY THIS SEAT
Research Agent	`Qwen 3.7 Max`	Agentic tool-use: deciding which tool to call and when to stop, not prose quality

Synthesis Agent	DeepSeek V4 Pro OpenAI-compatible endpoint	Structured extraction, not creative writing; condensing, not composing
Generator Agent "The Writer"	Claude Sonnet 4.6 AWS Bedrock	The one place prose quality is decisive — reserved exclusively for final email composition
Critic Agent "The Judge"	Gemini 3.5 Flash Google Vertex	A genuinely different reasoning cluster from the Writer — see §7.4
SMYKM Orchestrator	Mistral Medium 3.5	Coordination and state-tracking, not generation or judgment

Worth being precise about something here: early research into this pipeline originally recommended Gemini Flash for both the Research and Synthesis seats, on the strength of its 83.6% score on the MCP Atlas tool-use benchmark and its 4× throughput advantage. The live workflow runs Qwen 3.7 Max and DeepSeek V4 Pro in those two seats instead — a pragmatic substitution made during the build itself, not a design change made lightly or silently. The two model decisions that research was most insistent on — Claude reserved solely for prose, and a Critic drawn from a different reasoning cluster than the Writer — were kept exactly as designed. That's not an accident. Those are the two seats where the research showed the model choice actually changes the outcome; the other three are where a decent model with the right harness around it is what matters, per §7.1.

6.3 The data model

Three Airtable tables carry the workflow's state, and each exists for a distinct, single reason rather than as an arbitrary split of one wide table.

Groundwork — Airtable Data Model — three tables · one shared key

all three tables key on the same identifier — the lead's LinkedIn URL

Lead_Profiles — written once, right after sourcing · durable identity record

lead_id	text · PK	first_name	text
last_name	text	email	email
linkedin_url	url	headline	long text
title	text	sanitized_address	text
organization_name	text	organization_primary_domain	text
organization_linkedin_url	url	organization_founded_year	number
run_id	text	sourced_at	datetime

Leads — pipeline state / checkpoint · read on every run start

lead_id	text · PK	pipeline_status	single select

run_id	text	final_subject	text
final_body	long text	critic_score_json	long text
error_log	long text	retry_count	number
personalization_tier	single select	research_gaps	long text
last_updated_at	datetime	created_at	datetime

Lead_Research_Events — polymorphic audit trail · one row per agent event

event_id	text · PK	lead_id	text · FK
agent_name	single select	event_type	single select
payload	long text	recorded_at	datetime

WHY THREE TABLES, NOT ONE WIDE ONE

Lead_Profiles	The durable identity directory, kept in its own table so it survives even when `pipeline_status` resets.
Leads	The checkpoint the workflow reads on every run — `pipeline_status` is the resume mechanism, so a re-run never re-processes or re-emails a finished lead.
Lead_Research_Events	A polymorphic log, not a fixed schema — every agent writes its own event type (`draft`, `critique`, `research_finding`) with a JSON payload, so the schema doesn't change when the process does.
tier + gaps	`personalization_tier` and `research_gaps` exist so a human scanning `ready_to_send` can catch a fabricated or thin personalization before it reaches an inbox — the audit trail is worthless if nobody reads it.

Airtable data layer · Groundwork

`Lead_Profiles` is written once, immediately after a lead is sourced, independent of whether it ever finishes the pipeline — a durable identity and firmographic record meant to outlive any single run's `pipeline_status`, kept in its own table specifically so identity data doesn't share a schema with fast-changing pipeline state. `Leads` is the pipeline's actual checkpoint table; the one field that matters most operationally is `pipeline_status`, since Stage 2's dedup logic reads it on every run to decide what's already done. `Lead_Research_Events` is a polymorphic audit log: every agent in the chain writes its own event type with its own JSON payload into the same table, so the schema doesn't need to change every time the process does. An audit trail nobody reads is decoration, not engineering.

6.4 Why n8n?

A workflow like this could have been built as custom Python or LangGraph orchestration, on a platform like Make or Zapier, or hand-rolled directly against model APIs. n8n won for reasons specific to what this pipeline actually needed, not because it's the only tool capable of the job.

Visual, inspectable workflow definition mattered more than it sounds like it should for a project meant to be shown, not just run. A reviewer can open the canvas and see the five-stage shape of the pipeline directly — the same reason Airtable won the data-layer decision in §7.6: a system that's legible to someone other than the person who built it is a real engineering property, not a nice-to-have. A LangGraph implementation of the same architecture would be at least as capable, but it would live entirely in code, legible only to someone willing to read it.

Against Make or Zapier specifically, the deciding factor was narrower: n8n's AI Agent node has first-class, framework-backed support for tool-calling, per-agent memory, and structured output — the exact primitives a five-model, tool-using pipeline needs. Building the same system on a platform without native agent and tool-calling nodes would mean fighting the platform to bolt on equivalent behavior, rather than using what it's actually built for.

Self-hosting mattered for a more practical reason, too: leads' personal information — name, LinkedIn URL, research findings written about them — stays on infrastructure this project controls, rather than passing through an additional SaaS vendor with its own data-retention policy to audit, which matters given the data-handling discussion in §7.7.

None of that makes n8n free of real cost. Section 9 documents two genuine platform quirks this project ran into directly: paired-item lineage breaking across a loop boundary, and an undocumented ceiling in the tool used to test-execute the workflow during development. n8n was the right tool for this pipeline's shape. It wasn't the right tool by default — and §7.1 is where the more important argument actually lives: the harness matters more than which platform assembles it.

07 DECISIONS

Decisions and Trade-offs

Everything in this section was a real fork in the road, not a rhetorical one. Each decision below was argued through, sometimes across several long research passes, before it was locked in. The two that mattered most (§7.2 and §7.5) came from a real disagreement between what looked impressive and what was actually correct for this pipeline's shape.

7.1 The harness is the point

The term "harness" has a specific, useful meaning in agent engineering. The harness is everything wrapping the model: the tools it can call, the memory it keeps, the checks that run on its output, the cost and iteration limits, the logging. There's a benchmark data point worth repeating exactly because it's counterintuitive. Simply changing the harness around a coding agent, while keeping the same underlying model, moved that agent from rank #30 to rank #5 on a public leaderboard. The model didn't get smarter. The scaffolding around it did.

Agent = Model + Harness

The harness is the tools, memory, output checks, cost and iteration limits, and logging around the model.

#30 → #5

Same model, better harness — the leaderboard jump for a coding agent from harness changes alone.

Applied to this workflow, the harness is everything that isn't an LLM call: the Apify sourcing pipe, the per-lead research-brief schema that constrains what the Synthesis Agent is allowed to hand forward, the Loop-Over-Items checkpoint logic, the Critic's rubric validator. None of that is glamorous. All of it is why five agents produce a coherent email instead of five disconnected opinions. It's also why n8n, in this build, is a tool the agent lives inside, not the identity of the project. The interesting engineering here is the harness design; n8n is simply where that harness happens to be assembled.

7.2 Rejecting the super-orchestrator

The first architecture draft for this pipeline included a top-level supervisor agent: one model watching the entire research–write–critique–revise–send sequence, deciding at each step whether to continue, retry, or escalate. It would have looked good in a demo. It was also the wrong pattern for this specific pipeline, and the reasoning for why is worth walking through in full, because it's the one architectural argument in this project that required pushing back on the more impressive-looking option.

Anthropic's own guidance on the orchestrator-workers pattern is specific about when it applies: workflows where the subtasks genuinely can't be predicted in advance, and the orchestrator has to determine them dynamically based on the specific input. This pipeline's actual sequence — research, then synthesize, then write, then critique, then maybe revise once, then finalize or escalate — is fixed and fully predictable at design time. It's the same five steps for lead #1 and lead #500.

> **Fixed sequence → Prompt Chaining.** Unpredictable subtasks → Orchestrator-Workers. This pipeline is the same five steps every time, so it gets a deterministic harness with validation gates between calls — not an autonomous supervisor.

There's a second problem with a monitoring super-orchestrator, more serious than a mismatched pattern label. It puts the LLM in charge of counting its own iteration cap. The research converged hard on this point from multiple independent directions: agents need well-defined execution boundaries, and those boundaries have to be enforced by the surrounding system, not by the agent's own self-assessment. A deterministic controller given identical inputs always produces identical outputs. An LLM re-deriving “have I already tried twice?” from natural-language context on every single lead, across a fifty-lead run, is fifty independent chances to miscount, lose track after a long context window, or reason itself into “just one more revision.” That’s the exact failure mode loop-engineering research warns about, and it’s compounded by cost. Anthropic’s own production data on their orchestrator-workers research system puts the token cost of that pattern at roughly fifteen times a single chat conversation, with the orchestrator itself as an additional, compounding bottleneck since its context keeps growing on every hand-off.

So the top-level orchestrator was removed entirely, not scaled down. The one place this pipeline still uses genuine dynamic, orchestrator-style tool selection is the Research Agent’s own toolbelt, because which platform holds a usable personal signal for a specific human being truly cannot be predicted before the agent looks. That’s the one part of this pipeline where the shape of the problem actually calls for it. Everywhere else, the sequence is fixed, and fixed sequences get a deterministic harness, not an autonomous supervisor. When this was on the table, the instinct to add a more impressive-looking layer for its own sake was real, and worth naming, because “will this look good in the case study” is a genuinely bad reason to add architecture. It’s a temptation every portfolio project should be honest about resisting.

7.3 Generator–Critic: a second model, not a bigger one

The decision to separate writing from judging into two distinct model calls, rather than asking one model to draft and then re-read its own work, rests on a specific, replicated finding: Huang et al. (ICLR 2024) demonstrated that large language models cannot reliably self-correct without external feedback. A model asked to critique its own output tends to confirm its own priors, not surface its own blind spots. The CRITIC line of research reinforces the same point from the other direction: tool-interactive, externally-grounded critiquing outperforms self-critique specifically because it has something outside the model's own reasoning to check against.

The revision cap itself — exactly two rounds and not an open-ended refine loop — comes from Self-Refine's own reported numbers: roughly 20% absolute quality improvement across two to four iterations, with the bulk of that gain landing in the first pass or two. Past that point, additional rounds buy shrinking quality improvement for real, compounding token cost. Two rounds is close to the empirically-supported point of diminishing returns, not an arbitrary round number.

The knockout rule closes a gap the round cap alone doesn't: if a revision scores lower than the draft it was meant to improve, the pipeline keeps the earlier, stronger draft rather than finalizing or escalating with whichever one happened to come last. A revision loop with no memory of "was this actually an improvement" can make a lead's outcome worse than doing nothing.

7.4 Why Gemini judges Claude, specifically

The choice of which second model does the judging isn't arbitrary either, and it's more specific than "use a different vendor." A controlled similarity test found that language models cluster into two distinct reasoning families: Gemini and GPT align closely, as do Claude and Mistral, while cross-cluster similarity drops sharply. The practical implication is that pairing Claude with a Mistral critic would still risk correlated blind spots, because the two models reason in similar ways even though they come from different labs. Gemini, from the other cluster, is a genuinely independent check rather than a same-family rubber stamp.

Critic (Gemini) and Writer (Claude) sit in different clusters — the judge is a real second opinion, not a reflection.

That independence matters more than it might sound like it should, because self-preference bias in LLM judging is well documented and large. UXBench found that frontier judge models assign their own model family a 75–84% win rate in pairwise evaluation — a number large enough that same-family critique is closer to confirmation than evaluation. Gemini judging Claude's draft is deliberately the harder, more honest version of this check.

7.5 The gap we chose to ship with, and named on purpose

Here's the part most write-ups quietly smooth over: early research into this pipeline explicitly recommended enforcing the two-round cap with a deterministic n8n IF/loop-counter node — a hard, code-level exit condition immune to an LLM's own miscounting, for exactly the reasons argued in §7.2. The live system does not do that. The SMYKM Agent's system prompt instructs it to "maintain an explicit, visible iteration counter in your reasoning at every step," and the cap is currently enforced by the model following that instruction, not by a gate outside its control.

KNOWN LIMITATION · SHIPPED ON PURPOSE

The two-round cap is enforced by instruction, not by code.

A deterministic loop-counter gate would be the harder, structural guarantee. The simpler version shipped first so real behavior could be observed before paying for the robust one — a trade made in the open, not discovered by a reader.

This was a real, considered decision, not an oversight that slipped through. Testing since the build, including a live execution that actually reached round two, shows the self-counting behaving correctly so far. But "behaving correctly so far" and "structurally guaranteed to behave correctly" are different claims, and only one of them is true today. This is named here, plainly, as a known limitation of the current build. The honest version of this decision: we know the more robust pattern, we chose to ship the simpler one first and observe real behavior before adding the harder guarantee.

7.6 Data storage: Airtable over Postgres, three tables over one

Postgres and Google Sheets were both live options for this pipeline's state layer. Airtable won for a reason that's easy to undervalue: portfolio demoability. A reviewer, a hiring manager, a prospective client, another engineer — any of them can open an Airtable base and see the pipeline's state in a way a Postgres table never demonstrates without a separate admin tool. For a project explicitly meant to be shown, not just run, that visibility is a legitimate engineering criterion, not a compromise.

The three-table split (§6.3) over one wide table follows the same instinct that drove the orchestrator decision: match the structure to what the data actually is, not to what's easiest to bolt on. A durable lead directory, a pipeline checkpoint, and a polymorphic event log are three different things with three different write patterns and three different read patterns. Collapsing them into one table would mean every future addition fighting the shape of a schema built for something else.

7.7 Lead sourcing: cost, and the legal ground it stands on

The lead-sourcing tool was chosen specifically because it's a cost-effective, production-grade Apollo-style option that still returns structured, enrichable records keyed on LinkedIn URL. That URL becomes the identifier every table in §6.3 is keyed on. Email verification beyond that — a verified-only filter applied at the source, rather than a separate downstream verification step — was a deliberate choice to keep deliverability checks lightweight rather than adding another paid tool to the pipeline.

This is also a place where publishing the workflow publicly means the legal posture has to be stated plainly rather than assumed.

CRIMINAL · CFAA

hiQ v. LinkedIn

Scraping public data doesn't by itself violate the CFAA — a public page needs no authorization that can be denied.

CONTRACTUAL · TOS

The real exposure

LinkedIn's ToS prohibit automated extraction; the risk is a banned account or cease-and-desist. No-login sourcing is mitigation, not a shield.

PRIVACY · GDPR

Underneath all of it

Public availability isn't a lawful basis to store personal data. EU leads need a retention policy and a real deletion path.

The governing precedent, hiQ Labs v. LinkedIn, held that scraping publicly available data does not by itself violate the Computer Fraud and Abuse Act, because accessing a public page requires no authorization that can be denied. That's narrower protection than it sounds. LinkedIn's own Terms of Service separately prohibit automated extraction, and the exposure that actually matters in practice is contractual — a banned account, a cease-and-desist — rather than criminal. Preferring no-cookie, no-login sourcing tools, as this pipeline does, is a real mitigation, not a legal shield. And GDPR sits underneath all of it regardless of scraping method: public availability is not, by itself, a lawful basis for storing someone's personal data. Anyone adapting this template for EU leads needs their own retention policy and a real answer for how a data-subject deletion request gets honored. This case study states that requirement rather than leaving it implied.

08 OPERATIONS

Best Practices to Maximize Output

Running this pipeline well is a different skill from building it, and a handful of operational habits make a measurable difference in output quality and cost.

01 **Feed the intake form real inputs, not placeholders.**

The Generator and Critic both work from the sender's `companydata` object captured once at Stage 1. A vague ICP description produces a vague persona-specific challenge (element four in §5.2), no matter how good the personal hook is. The time spent writing a precise ICP and objection list before the first run pays for itself across every lead in the batch.

02 **Batch size matters more than it looks like it should.**

A modest batch — large enough to make the per-run fixed costs worth it, small enough that one bad list doesn't burn a lot of Research Agent budget on leads with genuinely thin public signal — is close to the practical sweet spot. Extremes in either direction cost more than they save.

03 **Watch `personalization_tier` before you watch reply rate.**

A batch where a large share of leads land on `company_fallback` rather than a real personal angle is telling you something about the input list, not the pipeline: either the ICP is too broad, or the lead source isn't returning people with an active public footprint. Fixing the input list is cheaper than making the Generator write around thin material.

04 **Read `research_gaps` before sending anything, every time.**

This field exists specifically so a human catches a thin or fabricated-adjacent personalization before it reaches an inbox. Skipping this step to save a few minutes defeats a design decision this entire project is built around.

05 **Cost scales with tool calls, not with model size.**

The Research Agent's tool-call budget is the single largest cost lever in the pipeline. Loosening it produces more thorough research at a real, linear token cost; tightening it too far starts pushing more leads into `company_fallback`. Tune this number against lead quality, not against a fixed rule.

06 **Don't skip the schema read.**

Every agent that writes to Airtable confirms field names via a schema read before its first write in a run. It's an easy step to cut under time pressure — and it's the exact thing that prevents a silent field-name mismatch from corrupting a batch partway through.

09 POSTMORTEM

What Failed

Every one of these actually happened during this build. None are hypothetical "things that could go wrong." They're things that did, roughly in the order they did, along with what each one cost and what it taught.

FAILURE 01 · SILENT CREDENTIAL

A credential broke silently, and it looked like a code problem.

The Postgres Chat Memory node giving the Research Agent continuity across tool calls started failing with an `ENOTFOUND` error against its host — a stale or misconfigured credential, not a bug in any node's logic. From inside the workflow, a broken database credential and broken expression logic can look identical at first glance. The fix was outside the workflow (the credential needed correcting at the source). The lesson: not every red error in n8n's log is a workflow bug — check whether the thing a node connects to is actually reachable before asking what you misconfigured in the node.

FAILURE 02 · VANISHING FIELDS

An AI Agent node quietly ate its own input.

n8n's Agent and Chain LLM nodes replace `$json` with their own output by default — any field that arrived but wasn't part of what the agent generated simply disappears downstream. Two Set nodes needed the sender's `companydata` and `run_id`, attached upstream; after each agent ran, that data was gone. The fix was explicit re-attachment with `includeOtherFields: true`, re-pulling the originals rather than assuming they'd survive the trip. A genuine n8n gotcha with no warning in the UI.

FAILURE 03 · BROKEN LINEAGE

Split In Batches doesn't preserve paired-item lineage across its loop boundary.

A Set node several hops downstream threw `Cannot convert undefined or null to object` — unhelpful alone. The full error included `pairedItemNoInfo, nodeCause: "Loop Over Items"`: n8n admitting the loop had broken the paired-item chain that lets `$('NodeName').item.json` find its row. The fix was index-based lookups — `.all()[$runIndex]` — which sidestep paired-item tracking by relying on this branch running exactly once per loop pass. A workaround, not a first-choice pattern.

FAILURE 04 · UNHELPFUL VALIDATION

Airtable's table-creation API is stricter about capitalization than its own field names.

Creating the `sourced_at` field with `"timeZone": "UTC"` failed validation with a message that only said the value didn't match "one of the possible values," with no list of what those were. It wanted lowercase `"utc"`. A reminder that errors describing that something is wrong without describing what would be right cost more debugging time than they should.

FAILURE 05 · PHANTOM TIMEOUT

The MCP test-execution tool has an undocumented 5-minute ceiling.

Running the full five-agent pipeline through the test-execution tool repeatedly hit a hard stop around the 300-second mark — not a workflow error, but a limit belonging to the test harness itself, found by watching several runs die at nearly the same wall-clock time regardless of the lead. The workaround was polling the execution's real status directly rather than waiting on the single test call to return. A two-round Generator/Critic cycle across five agents comfortably exceeds five minutes for one lead — a property of the harness, not evidence the workflow is too slow.

FAILURE 06 · OPEN ITEM

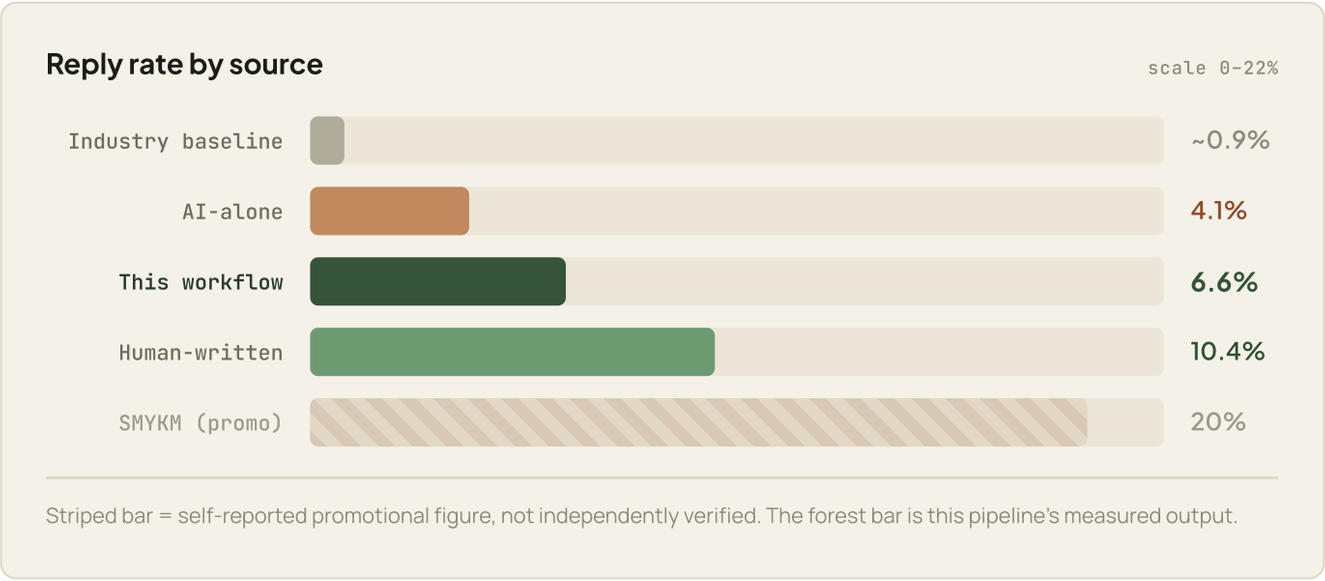
A referenced sourcing tool and the configured one don't quite match — still open.

The Research Agent's system prompt names one enrichment tool for follow-up lookups and a second as fallback, while the Lead node's actual configuration only has the fallback wired up outright. Nothing has broken yet, but it's exactly the kind of drift between what a prompt assumes and what a node is wired to that compounds silently over time — tracked, by name, as an open item rather than quietly corrected without mention.

The throughline across all six: almost none were the interesting, headline kind of failure. They were small, unglamorous, easy-to-miss inconsistencies between what a tool documents and what it actually does under a specific condition — and every one cost real debugging time precisely because the error message rarely said what was actually wrong. That's the part of building an agentic pipeline that doesn't show up in an architecture diagram, and it's the part that took the most hours.

Measurable Outcomes

Four numbers matter here, and they come from three very different kinds of source — which is exactly why they need to sit next to each other rather than be quoted in isolation.



SOURCE	REPLY	SPAM-FLAG	WHAT IT MEASURES
Industry baseline general cold email	~0.9%	n/a	Typical unpersonalized performance, per the research underlying this build
AI-alone outreach 12,000-email study	4.1%	7.8%	AI handling both research and writing, with no grounding discipline
Human-written same 12,000-email study	10.4%	2.9%	The benchmark this architecture is trying to close the gap toward
THIS BUILD **Groundwork** real test batch, 100+ emails	6.6%	4%	This pipeline's actual, measured output
SMYKM promotional McKenna / #samsales	20% 43% open	n/a	Published claim — promotional figures, explicitly *not* independently verified

That last row needs its caveat stated every time it's cited, not just once in a footnote: the 43% open / 20% reply figures come directly from McKenna's own marketing materials, not from an independently audited study, and the research underlying this project flagged them as such from the start. They're included because the user of this pipeline asked for them to be, and because they represent the aspirational ceiling the SMYKM method claims for itself — not because they're a verified benchmark this workflow should be measured against directly.

What the pipeline's own number actually says: 6.6% reply and 4% spam, on a real batch of just over 100 emails, sits meaningfully above the 4.1%/7.8% AI-alone baseline this project set out to beat, and closer to (though still short of) the 10.4%/2.9% human-written benchmark. That's a real result and a genuinely encouraging one. It's also a result from one test batch, not a controlled study with a control group, and the honest read is that a 100-email sample is enough to say "this is working, and working better than the generic-AI baseline" without being enough to say "this is a stable, precise long-run rate." The gap that's left — between 6.6% and 10.4%, and further still from a 20% promotional ceiling that isn't independently verified in the first place — is what further Research Agent tuning, more revision-loop data, and simply more volume would start to close in on. Grounded clarity means reporting the number that was actually measured, on the sample size it was actually measured on, and being precise about which comparisons are apples-to-apples and which aren't.

11 CLOSING

Reflections

A well-run SDR team doing genuine account research before every send is not a new idea. Sales organizations with the headcount for it have done exactly this for years: someone reads the prospect's LinkedIn, finds an actual hook, writes a real email, sends it. What's changed is not the method. It's who can afford to run it.

Five agents, three Airtable tables, a couple of cloud model APIs, and an evening or two of prompt engineering is a cost structure available to a two-person company, not just a funded sales org with a dozen SDRs and a Sales Ops team maintaining their tooling. That's the actual argument for AI in a workflow like this one — and it's not that it writes better than a person can, because on the numbers in Section 10, it still doesn't. The argument is that the discipline behind good outreach — actually researching before writing, actually checking a draft against what's true before it goes out — no longer requires a headcount most companies this size don't have. Bringing that discipline down to a cost and skill level a small team can run themselves is the whole point, and it only works if the discipline survives the automation instead of getting flattened by it.

That survival was never going to be automatic, and this project is really a record of the engineering it took to keep it intact. Separating research from writing, because a single model asked to do both will quietly cut the research short. Giving a second, differently-reasoning model the job of catching the first model's fabrications, because self-review doesn't reliably catch what it can't see about itself. Refusing to add a more impressive-sounding orchestrator layer once the evidence said it was the wrong shape for a fixed sequence, even though it would have made for a flashier diagram. And naming the one place — the self-counted revision cap — where the current build ships with a known, smaller reliability guarantee than the fully hardened version would have, rather than either hiding that gap or paying its cost before it was clear the simpler version wouldn't hold up in practice.

> The real challenge was never getting a model to sound like a person. It's building a harness disciplined enough that the model is never allowed to sound convincing about something that isn't true.

Every AI cold-email tool on the market already sounds like a person, convincingly enough to pass a first read. This one is built so it can only sound convincing about something it actually verified.

This workflow is built and validated end-to-end through Airtable persistence. Sending is the next deliberate step, wired in on the reviewer's own timeline rather than this build's. Appendix A carries the full data-model reference, and Appendix B the node-level configuration excerpts, for anyone building a similar pipeline of their own — nothing here is abridged for effect.

A — APPENDIX

Airtable Schema, Field by Field

Leads `pipeline-state / checkpoint`

One row per lead. The workflow reads `pipeline_status` *on start to skip already-completed leads.*

FIELD	TYPE	PURPOSE
`lead_id`	single line text (primary)	LinkedIn URL — stable identifier
`pipeline_status`	single select	Resume / checkpoint mechanism
`run_id`	single line text	Groups leads by batch run
`final_subject`	single line text	Finalized subject line
`final_body`	long text	Finalized email body
`critic_score_json`	long text	Full structured scores from the passing (or best) round
`error_log`	long text	Summary of which dimension(s) kept failing, on escalation
`retry_count`	number	Revision rounds actually used
`personalization_tier`	single select	From the Brief's `primary_angle.tier` — real signal vs. fallback
`research_gaps`	long text	Documented gaps, surfaced for human review
`last_updated_at / created_at`	date time	Audit timestamps

Lead_Research_Events `polymorphic audit trail`

One row per event emitted by any agent for any lead. Backbone of grounding verification — the record that

lets any claimed personal fact be traced back to where it came from.

FIELD	TYPE	PURPOSE
`event_id`	single line text (primary)	Row key
`lead_id`	single line text	Matches `Leads.lead_id`
`agent_name`	single select	Which agent emitted this event
`event_type`	single select	`research_finding` / `draft` / `critique`, etc.
`payload`	long text	The actual structured content — snippet + source, draft text, or critic JSON
`recorded_at`	date time	Timestamp

Lead_Profiles `durable identity record`

One row per sourced lead, captured directly from the lead-sourcing output right after the Lead node fires. Serves as a durable identity/firmographic record independent of any single run's pipeline status, kept in its own table so it persists regardless of what happens downstream.

FIELD	TYPE
`lead_id, first_name, last_name`	text
`email`	email
`linkedin_url, organization_linkedin_url`	url
`headline`	long text
`title, sanitized_address, organization_name, organization_primary_domain, run_id`	text
`organization_founded_year`	number
`sourced_at`	date time

B APPENDIX

Node Configuration Excerpts

Lead-sourcing node

Dynamic `customBody`, built from the intake form's Section B fields

```
{{ JSON.stringify({
  company_industry: ($json.company_industry || '').split(',').map(s =>
    s.trim()).filter(s => s.length > 0),
  company_locations: ($json.company_locations || '').split(',').map(s =>
    s.trim()).filter(s => s.length > 0),
  contact_email_status: ["verified"],
  contact_email_exclude_catch_all_domains: true,
  // …
}) }}
```

Set — Attach Company Data

The single point where sender company data is packaged, once, before the per-lead loop

```
{{ {
  companydata: {
    sender_company_name: $('On form submission').first().json.sender_company_name,
    icp_description: $('On form submission').first().json.icp_description,
    target_persona: $('On form submission').first().json.target_persona,
    offering_1line: $('On form submission').first().json.offering_1line,
    value_prop: $('On form submission').first().json.value_prop,
    common_objections: (…)
  },
  run_id: …
} }}
```

Set — Prep Synthesis / SMYKM Input

The fix for the paired-item lineage break described in Section 9

```
{{ {
  companydata: $('Set - Attach Company Data').first().json.companydata,
  run_id: $('Set - Attach Company Data').first().json.run_id,
  linkedin_url: $('Set - Attach Company Data').all()[$runIndex].json.linkedin_url
} }}
```

`includeOtherFields: true` on both of these nodes re-attaches everything else the upstream agent call had silently dropped (Section 9's second failure).

Format for Sending

The clean handoff contract

```
{{ {
  ready_to_send: $json.pipeline_status === 'ready_to_send',
  pipeline_status: $json.pipeline_status,
  subject: $json.final_subject || null,
  body: $json.final_body || null,
  personalization_tier: $json.personalization_tier || null,
  research_gaps: $json.research_gaps || null,
  lead_id: $json.lead_id,
  run_id: $json.run_id,
  critic_scores: $json.critic_score_json ? JSON.parse($json.critic_score_json) :
    null,
  review_notes: $json.error_log || null
} }}
```

Groundwork

Multi-agent AI for B2B cold email — grounded in real research, verified before it sends.

Written and diagrammed by Tesfa Michael

Applied AI Engineer — Agentic Systems & n8n Automation

CASE STUDY · 2026