FOR CONSUMER GOODS BRANDS

Build vs. Buy: Commerce AI Agents

How to decide what to build and what to buy, and why the data foundation is the part almost no brand should build

By Lasse Holmstedt, Chief Technology Officer, Alloy.ai

Alloy.ai

AT A GLANCE

Five key takeaways



01

Build versus buy isn't one decision. There are about five layers between a retailer portal and a replenishment order an agent can actually recommend, and you can reasonably build or buy each one.



02

The layer I'd almost never build is ingestion and normalization. It's a solved problem and a never-ending maintenance burden at the same time.



03

Getting a first version of a data integration working is much cheaper than it was two years ago. Getting it right, and keeping it right, is not.



04

An AI agent is only as trustworthy as the data it uses. Models work best against a harmonized data model where the numbers have been sanity-checked and trace back to source.



05

The more of your business an agent can see, the better its recommendations. Give it your ERP order data and the replenishment call gets sharper. But that only holds if the agent is built the right way: the recommendation is computed deterministically, the same result every time, never left to the LLM to figure out.

THE CATEGORY

What is a commerce AI agent?

Software that reads a brand's retail data and prepares an action, a replenishment order, a performance report, an adjusted forecast, for a person to review and approve. Its output is only as reliable as the data foundation it runs on.

START HERE

The short answer

If you have a real data engineering team and only two or three clean, well-documented sources, building can look like a realistic path. What we usually see is that a brand starts in e-commerce, where the APIs are easy to use, decides the problem is tractable, and then expands into brick-and-mortar retail. That's where it gets complicated.

Once you're pulling from twenty or fifty or a hundred retailers, distributors and 3PLs, I don't think you should build the ingestion and normalization layer yourself, and I'd say that even if I didn't work at Alloy. Acquiring the data isn't where the value is for a consumer goods brand. The value is in what you derive from clean data, and in the automated workflows you run on top of it.

Everything above that bottom layer is a judgment call. I'll walk through how I'd make it.

"Acquiring the data isn't where the value is for a consumer goods brand."

LASSE HOLMSTEDT, CTO, ALLOY.AI

ANATOMY OF AN AGENT

The layers underneath an agent

The agent that drafts a replenishment order or a performance report looks like the product. It's just the tip though. The bulk of the work is underneath it, and it stacks, each layer depending on the one below:

THE VISIBLE LAYER

5 **The agent**
Reads all of the above and prepares an action for a person to approve.

THE FOUNDATION UNDERNEATH

4 **Deterministic computation**
The math has to come out the same every time, and every number has to reconcile with the data underneath it.

3 **Derived measures**
Lost sales, weeks of supply, sales velocity, phantom inventory, forecasts. These are just some of the modeling problems you may need to solve in a repeatable way. Two reasonable people will define weeks of supply differently and both be defensible.

2 **Normalization**
Making the sources agree, across product identity, location, units of measure, and calendars. Easy with two or three retailers. Genuinely hard across a hundred.

1 **Ingestion**
Pulling data out of every retailer portal, distributor feed and 3PL system you sell through. Each has its own credentials, cadence, and format. Roughly 85% of them have no API to work with.

You can buy or build at any of these lines. Most conversations I have jump straight to layer 5 and skip 1 through 4, which is backwards, because 5 is the layer that's easiest to replace and hardest to trust.

Agentic execution frameworks have essentially commoditized in the last few months. You can get your own fleet of custom agents running in an afternoon, on Claude's Managed Agents or something similar. But a framework does nothing without clean data hooked into it. Layer 5 got cheap; 1 through 4 didn't. Building them yourself is a commitment for as long as the agents run.

THE REAL COST

Why the first version isn't the expensive part

Here's the thing I'd want to know if I were responsible for IT at a consumer goods brand. A model will write you a working retailer scraper this afternoon. Our team has done exactly that. It works at first, and then it doesn't, and the failures are the interesting part.

Two things break. Feeds change, constantly, the long-tail ones and the ones from the biggest retailers alike. Some of the failures are loud, others quiet. My favorite example, if you can call it that: a store stops reporting for a day. A naive data pipeline reads the silence as a zero. You've just invented an out-of-stock that never happened, and every number downstream inherits it. Nobody gets paged. Someone notices three weeks later when a figure looks off in a retailer meeting.

> **"You've just invented an out-of-stock that never happened, and every number downstream inherits it."**

I want to correct something I've heard framed too neatly: it's not that building got cheap and only maintenance stayed expensive. Building it properly is still expensive. Thorough coverage beyond the basic feeds is hard, and QA is harder, getting all the numbers to actually agree across sources takes real time from someone who knows what the numbers are supposed to look like. Maintenance is on top of that, not instead of it.

A few real numbers from our side, so this isn't abstract. We've updated ingestion for parts of one major US retailer 23 times in the last twelve months, almost all of it small portal changes on their end. In the same window, six retailers changed the format they deliver data in, each requiring its own update. A major e-commerce retailer changed its exports and we chased that through both the download and the ingestion side.

You can do this too. But a lot of build vs. buy isn't about what the team needs now. It's about what still works when the data is in the terabytes and the business is global. You'll want monitoring, an on-call rotation, and possibly even an offshore team so someone has eyes on it around the clock.

And a lot of what makes this work is knowledge that isn't written down anywhere a model could have learned it. One example: a major US retailer's store and e-commerce reports don't sum to the total the portal shows you. The fix is to pull them separately, ignore the total, and do the math yourself. As far as I know that isn't documented publicly. We learned it the way you'd expect. There are hundreds of these, and we're still finding them after ten years.

Industry benchmarks put software maintenance at well over half of a system's lifetime cost, often several times the original build. That's not an Alloy number. It's the shape of every long-lived system, and it matches what we see.

“A lot of what makes this work is knowledge that isn't written down anywhere a model could have learned it. There are hundreds of these, and we're still finding them after ten years.”

LASSE HOLMSTEDT, CTO, ALLOY.AI

CANDIDLY

What we've gotten wrong

I'd rather tell you this than have you find it. A whitepaper where the vendor got everything right isn't worth reading, and the failure modes are the actual content. If you build this yourself, these are the ones that will find you too.

2023 · Restated sales on empty locations

In 2023, a major US retailer started to report sales on random locations without associating them with a location number. Alloy ingested this data. The retailer promptly restated the sales, and Alloy ingested the restated data as well. However, our platform did not correctly delete the data on empty locations upon restatement: a genuine corner case, with the issue stemming from incorrect data at the source. We fixed this by adding additional validation at ingestion for empty locations and more robust data reloading.

2025 · Modeling e-commerce fulfillment

Customers kept asking to see how their e-commerce sales were actually being fulfilled. We had initially expressed e-commerce channels through the location dimension, which made it difficult to view fulfillment without custom attributes and filters. In 2025, we ran a multi-month engineering project to model this correctly, and migrated every Alloy customer onto this new model. This involved sweeping changes from the ETL layer to the query layer. As a result, Sales now breaks down by delivery channel natively without workarounds.

SCOPE

What we don't cover

We do not bring in every type of data you own.

The heart of Alloy is the data ingestion, normalized data model for products, locations and dates and the query engine, all of which power AI. Most real-world data fits our model well, from retailer data to inventory and ERP data. However, we do not bring in all data into Alloy today. If the data does not share a shape of product-locations cleanly, we do not bring it into our data model. Planogram data, raw materials (BOM), PII data about consumers are some of the categories of data Alloy does not handle today.

TRUST

How we keep the agents accurate

The numbers Alloy's agents show aren't generated by the language model. The model doesn't do arithmetic. It writes a query, a deterministic engine runs the math against validated data, and it hands back a result. So the number the agent acts on is the number on your dashboard, the same one, every time. A better autopilot doesn't help if the instruments are wrong. The instruments are your data. If the agent does decide to do math on its own, e.g. because you imported an existing sales report for additional context, the agent tells you the number did not exist in Alloy's data engine.

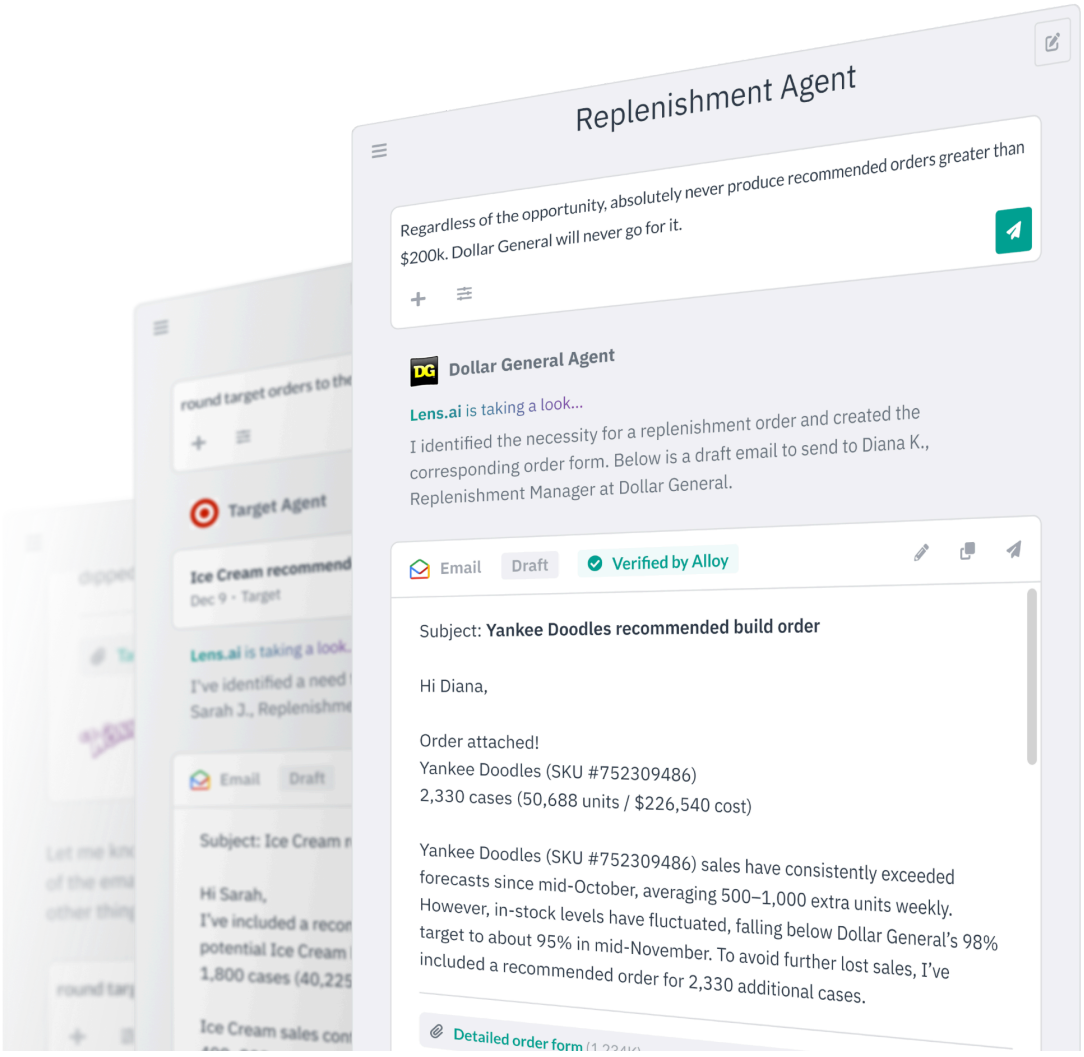
> **"A better autopilot doesn't help if the instruments are wrong. The instruments are your data."**

Models drift, and the governance has to cover that. A general model can change behavior between versions, and a prompt that worked last quarter can quietly regress. This is the part a do-it-yourself setup rarely handles, and honestly it's the part we spent the longest getting right. We re-run our regression evaluations on every model update and every prompt change, so the agents keep behaving the way you signed off on.

On security. Lens, our AI analyst, runs on leading foundation models, GPT, Claude, and Gemini. Customer data is never used to train them. Data is sent transiently to answer each question, under zero-data-retention terms where available. Conversation history lives in our own encrypted infrastructure, TLS 1.2 or better in transit, AES-256 or better at rest. Alloy is SOC 2 compliant, hosted on GCP, an isolated instance per customer, and the agents respect the role-based access your users already have. The details are at trust.alloy.ai, written for someone who's going to read them closely.

The model is the least defensible layer. The advantage lives above and below the model, in the validated data feeding it and the commerce expertise shaping it, earned across hundreds of brands and retailers. That's what a general model won't have and a single team can't easily build and maintain.

A bought foundation helps your engineering team focus elsewhere not just now, but going forward as well. We maintain the integrations, more than 500 today and counting, plus the normalization and the upkeep as retailers change. Typical implementation is about eight weeks with white-glove onboarding, and the data feeds your existing BI, ERP, and warehouse rather than replacing them. And on your side, standard setup takes essentially no engineering. The exceptions are narrow: bringing us custom data like an ERP feed, or configuring your own cloud warehouse or lake as a destination. Both of those need some work on your end. Everything else stays on us. It's no maintenance burden on your team.



The Replenishment AI Agent drafts the order; your team reviews and approves it

YOUR OPTIONS

The four paths

Build versus buy is a spectrum. It's really a question of how far up the stack you carry the build yourself, and the resources at your disposal.

Option	What you own	Right when	What you maintain
Build all of it	Everything, including your own agents	Narrow, API-only footprint, a strong data engineering team, and control matters more than cost	All of it, indefinitely
Buy the foundation, build your own agents	Your data in your own warehouse, plus the measures and the agents you build yourself on top	You have a strong data science team and specific modeling you want to own	The logic layer and your own agents
Buy the analytics, build your own agents	Your data in your own warehouse, plus your own agent layer	You want the modeling handled but want to own the agents as your differentiation	Your agent layer and its upkeep
Buy all of it, agents included	Your data in your own warehouse, and Alloy's agents running on top	The work is recurring, wide, or acting on its own, being wrong is expensive, and you want our agents rather than your own	A subscription

In any option that includes us, your data lands in your own warehouse and stays yours.

One thing comes up from time to time, so I want to be clear about it: Our agents run on our stack, top to bottom. You can build your own agents on our data, and plenty of teams do, that's the middle of the table. But you can't run ours on a foundation we don't maintain, and we don't sell them on their own. If you want our agents, you're buying the whole stack. That isn't packaging. It's the only way the numbers underneath them stay trustworthy.



THE FIFTH OPTION

There's a fifth option nobody writes on a slide: keep the spreadsheets. Keep the weekly report cycle you already run. It's the choice most companies start with, and usually the most expensive one, and it involves the most manual effort. On top of the continuous maintenance costs, the usual lack of data validation and normalization leaves a lot of room for errors. In my role, I want to look at how to maximize the impact of engineers in my team and manual work is not it.

To put a number on data visibility, IHL puts total inventory distortion, overstocks and out-of-stocks together, at roughly $1.2 trillion globally. Whatever this year's exact figure, maximizing sales is ultimately at the heart of supply chain visibility systems, homegrown and bought alike. How your company gets there the fastest is a decision that's best answered by you and your colleagues.

WORKSHEET

Three questions to ask yourself

Tick the answer that fits each workflow you're weighing.

01 Is the problem general, or specific to you?

Retail ingestion and normalization are general, solved the same way for everyone. The way you run your business is specific. I wouldn't vibe-code my own CRM, and you probably wouldn't build your own warehouse. Buy the general part. Put your engineers on the specific part.

- ☐ **Points to build:** specific to how you run your business
- ☐ **Points to buy:** general, solved the same way for everyone

02 How often does it run, and who sees it?

A one-off analysis with a person reading every line is a fine place to point a general model at raw data. A workflow that runs weekly, feeds a dozen people, or places an order is not.

- ☐ **Points to build:** a one-off analysis with a person reading every line
- ☐ **Points to buy:** runs weekly, feeds a dozen people, or places an order

03 What does being wrong cost?

If a wrong number gets caught by the analyst who asked for it, build lightly. If it flows into an order, a forecast, or a conversation with a retailer, “usually right” is a liability, and the layer beneath the agent should be somebody's full-time job.

- ☐ **Points to build:** a wrong number gets caught by the analyst who asked for it
- ☐ **Points to buy:** it flows into an order, a forecast, or a conversation with a retailer

If your answers point at building, build. Sometimes that's the right call, and I'd tell you so on a call.

For most consumer goods brands that isn't the shape of the problem, and the recommendation is almost never all-build. I run engineering at the company selling this, so my answer bends toward buy. Discount it as you see fit. To me though, the question is about leveraging your own resources in the most effective way, and building something that is going to need me to lock in our resources for years to come on a project is unlikely to be it. That's why buying the whole stack, the maintained data and the agents that run on it, just makes the most sense to me: it frees up your IT to do e.g. building workflows specific to your company.

The real exception is a brand with proprietary data science it wants to own. One engineering-first brand we work with runs everything through its own warehouse and builds its own marketing-mix and enrichment models on our clean data. That team isn't running our agents, it built its own on top. That's a legitimate path. It just isn't most brands, most of whom have far less they actually need to own.

On the agents themselves, I'll be straight about where the category actually is. They're new. Anyone selling a finished, autonomous future is selling, and building a good one is hard. The thing worth holding onto is smaller than the pitch: the agent is the visible layer, and it's only worth running if the data underneath it is maintained and worth trusting. Get that wrong and a better model doesn't save you, it just makes the wrong number sound more convincing.

There's a leadership version of this decision too. A lot of these projects start the same way: IT promised to build it, a year goes by, and the business still doesn't have it. Buying the foundation is how you avoid being that project. You deliver the capability in weeks instead of quarters, and you get your engineers' time back for the roadmap you were actually hired to lead.

IN SUMMARY

The bottom line

You own your data and where it lives. Our job is to keep it correct, and to run the agents on top of it so your team never inherits that maintenance either. For most brands the honest answer is to buy the whole stack, the foundation and the agents that already sit on it, and build only the parts that are genuinely specific to how you operate. Whatever runs on your numbers, hold it to one standard: deterministic, traceable, and versioned. If a vendor can't explain that mechanism in a paragraph, that's your answer.

Deterministic | **Traceable** | **Versioned**

For the layer beneath the agents, the data platform and why buying it beats building it, read the reasons to choose Alloy over building your own. To see the deterministic model and the agents in action, book a demo and ask for engineering to be on the call.

FAQ

Frequently asked questions

Can I just use ChatGPT or Claude on my own retail data?

For a one-off analysis where a person checks every output, yes, and I'd encourage it. For recurring work, or anything an agent acts on, a general model pointed at raw retail data will give you confident, well-written answers that are quietly wrong, because the hard part is the normalized data underneath, not the model.

Is it cheaper to build or buy commerce analytics and AI agents?

Getting a first version working is much cheaper than it used to be. Getting it right and keeping it right isn't. Industry benchmarks put software maintenance at well over half of a system's lifetime cost, and retail feeds change constantly, so the durable expense is upkeep rather than the initial build.

What is the hardest part of building retail data pipelines?

Greatly differing ways of exposing data, from APIs to portals, different authentication methods. After you get the data, normalizing dozens of retailers into one comparable view, and keeping every connection correct as portals, formats, and definitions change indefinitely. The failures are also quiet, which is worse than loud ones.

What is data normalization in retail?

The work of making data from different retailers agree, one calendar, one product identity, one definition of a unit, so numbers from multiple sources can be compared. Without it, retail data can't be trusted for decisions.

When does it make sense to build instead of buy?

When the problem is narrow and specific to your business, you have two or three clean sources, a person reviews every output, and being wrong is cheap and easily caught. One caveat worth looking a year or two ahead on: if you're only in e-commerce today but expect to expand into more retailers and channels, a build sized for right now becomes a cost you either write off or keep maintaining once the footprint grows. The wider your footprint, and the more you expect it to widen, the more buying the maintained foundation wins.

ABOUT THE AUTHOR Lasse Holmstedt is Chief Technology Officer at Alloy.ai, where he has been the primary architect of the data platform for about ten years and led the work to build the company's AI agents on top of it.

TRUSTED BY CONSUMER BRANDS

Crayola · BIC · ANKER · Valvoline · LIQUID I.V. · Melissa & Doug · grüns · BOSCH

Alloy.ai

Alloy.ai is the Commerce Intelligence System for consumer goods brands. Through 500+ integrations with retailers, e-commerce platforms, online marketplaces, social commerce channels, distributors, 3PLs, and ERPs, Alloy brings in the actual sales and inventory data commerce partners report, not estimates, and reconciles it with internal shipment data for one consistent view down to the SKU and location. On that foundation, Alloy delivers the analytics that brands run their business on, and its AI Agents turn what the data shows into replenishment orders and executive reports for humans to review and approve. Sales and Supply Chain teams at brands including Liquid I.V., Crayola, and Valvoline use Alloy to recover lost sales, protect margins, and keep products on the shelf.

Book a demo →

https://alloy.ai/book-a-demo

