CloverDX

Cracking the Build vs Buy Dilemma for Data Integration Software

How to choose which option is right for you

Facing the build vs buy dilemma

Most data management solutions start off as something that's grown organically – a 'shadow IT' process where people start individually improving and automating parts of a process because they just desperately need it and don't have the time (or the confidence) to turn it into a full IT project. This is the world of Excel macros or Python scripts that often end up being connected with other small parts into something bigger, then slightly bigger, then even bigger, you get the idea. Which is great – you're being efficient and automating processes – until it's not great any more and your solution doesn't scale, or stops working entirely.

Or maybe you have a purpose-built application that was designed and built to solve a specific problem. This can be an excellent way of addressing your exact needs, but over time it can bring frustration. It might be very time-consuming to maintain; require something that's no longer supported; or it doesn't keep up with newer standards (e.g. a desktop vs web application).

This is the point at which companies often look into buying a data management or data integration platform to replace an aging or convoluted home-grown system.

Buying and implementing a data management system is a different experience to buying a CRM or HR platform off the shelf - an enterprise system like that needs to be configured and customized to your environment, but a data integration platform is more like a development environment. It allows you to implement what you've been doing to date (i.e. connecting small, ad-hoc things), in a more controlled, efficient and sustainable way.

But you'll still need to be coding or customizing to build out your data flows, you'll still need developers or similarly technical people to implement and maintain your off-the-shelf system, and you'll have to adjust some of your existing processes to fit the platform.

In this guide we look at the specific challenges around deciding whether to build your data solution in-house or to buy a platform. And despite our bias as a data platform provider, there is no one right answer for everyone.

For smaller organizations or those with uncomplicated data needs (or giant corporations where money and resource are no object), hand-coding what you need can absolutely be the best option. You have complete control over what you build so can create something that 100% addresses your needs. And maybe your organization isn't anywhere near being able to justify a large expenditure on software.

But for many companies the decision isn't clear cut. There are advantages and disadvantages of each approach and we've tried to lay out what we've learned from our experience as data consultants, so you can make a more informed decision.

Comparing build, buy, and buy-and-extend

This guide looks at the different areas you should consider, based on the phases you'll encounter in your data project:

- Preparation
- Design and development
- Operational considerations
- Support and maintenance

We've also given real weight to a consideration that too many build vs buy conversations still treat as an afterthought: data sovereignty and control. For a growing number of organizations, it's now the first question a board or a compliance team asks.

The third path: buy-and-extend

This decision is often framed as a straight binary: build it yourself, or buy something off the shelf. In practice, pure builds and pure buys are both rare. Most data integration decisions land somewhere in between, and it's worth naming that middle path directly, because it's where most successful projects actually end up.

Buy-and-extend means choosing a platform that gives you a solid operational foundation, connectivity, orchestration, monitoring, deployment, security, and then extending it with your own logic, coding, and customization for the parts of your pipeline that are genuinely specific to your business.

This is a different proposition from buying a rigid, configuration-only tool. A data integration platform that combines full coding capability with visual pipeline design gives you the speed and governance of something bought, plus the flexibility to build exactly what you need where off-the-shelf logic won't cut it.

A simple way to decide where each piece of work belongs

A useful test for any given piece of functionality is to ask two questions: how differentiating is this to our business, and how much of a solved, commodity problem is it already?

- Build it (or build it on top of a platform) when it's a genuine competitive differentiator, or when it touches sensitive data that you need to control completely.
- Buy it outright when the need is a solved, well-understood problem that many other organizations share, like connecting to a common database or file format.
- Buy-and-extend when you want the operational heavy lifting handled for you, monitoring, scheduling, deployment, security, but need real flexibility for the logic that makes your data estate genuinely yours.

Running this test function by function, rather than trying to make one big build-or-buy decision for the whole project, tends to produce a much more honest answer than either extreme.

PRIMARY DECISION DRIVER

Data sovereignty and control

Data residency used to be treated as a footnote in build vs buy conversations. The regulatory and security landscape has moved fast: new and tightened frameworks for financial services, critical infrastructure, and public sector data, alongside a general hardening of expectations around who can see your data and where it physically sits. AI has sharpened this. The moment your organization starts sending data, prompts, or specifications to an external AI provider as part of a pipeline or a development workflow, sovereignty stops being a compliance checkbox and becomes an active, everyday question: where does this data go, who can see it, and under whose jurisdiction does it sit while it's being processed?

Build

Building in-house gives you sovereignty by default, in the sense that nothing leaves your infrastructure unless you explicitly send it somewhere. But that default comes with full responsibility attached.

You have to prove it, not just assume it. That means documenting where every environment lives, how every backup is stored and encrypted, who has access to what, and how you'd demonstrate all of this in an audit. None of that happens automatically just because the code is yours.

It also means every third-party library, AI coding tool, or cloud service your developers bring into the project needs the same scrutiny. A single team member connecting an AI assistant to a production codebase can quietly move your sovereignty posture without anyone deciding that it should.

Buy

This is the area where 'buy' has changed the most. On-premises is no longer a legacy fallback option, it's increasingly a deliberate choice made by organizations that have looked at cloud-only and sovereign-cloud alternatives and decided they need something else.

When evaluating a platform, look closely at deployment model. Can it run entirely inside your own infrastructure, your own cloud account, or a hybrid of both, with no requirement for your data to pass through the vendor's own servers at any point?

This matters just as much for AI features bundled into a platform. A vendor that processes your data on its own AI service is a different proposition to one where you connect your own AI provider or your own model (BYOM), and your data goes only where you've told it to go.

Ask this early...
Where does our data physically go, at every stage of this pipeline, including any AI-assisted steps, and can we prove that to an auditor without relying on a vendor's word for it? If a platform can't answer this clearly, it's worth pausing before you go further down the evaluation.

Preparation

PREPARATION

Capturing business requirements

Whether you're building something in-house orlooking at buying something off the shelf to solve your data problem, you're going to need to outline your business requirements: what is it you're trying to accomplish? It's crucial to have everyone in the business aligned on this point before you embark on your project. You then need to translate those business requirements, either into a specification for developers to build, or you need someone who can translate your business requirements in terms of the software you're buying.

Build

When you build a solution in-house, you need to specify the 'non-business' requirements too, such as how you'll monitor the platform, how to update it, and how to report on it, since you'll have to build or incorporate all of this yourself. Skipping these tends to cause significant trouble later.

With a DIY option, there's usually more overhead than you'd expect. Think of it like decorating a room. You can build your own house, or move into something already built. If you want something special that's not readily available, building your own gets you exactly what you want. But if what you really want is new furniture, you don't want to be laying foundations first.

Buy

You'll also need to consider the 'non-business' part when buying, but you can expect your platform to solve a considerable portion of it out of the box.

You do need to make sure your particular requirements are genuinely met. If you need certain connectivity or reporting capability, that needs to be part of your evaluation criteria. It's generally not something you have to build yourself.

PREPARATION

Hitting your deadlines

Your timeline for having a solution up and running might be driven by a business deadline, so you'll need to figure out what's achievable in the time available. This is an area where perceptions of build vs buy can be unrealistic.

Build

There's often an assumption that building means you can start immediately, since you already have a development team. That's not quite true. Developers are valuable and expensive, and your new project will need to displace something already in progress, or wait for capacity.

AI coding tools have shortened the first mile here. A prototype that once took weeks of developer time can now often be sketched out in days. But don't skip the research phase. Looking at what already exists helps you avoid reinventing features, badly, that a platform would have given you for free.

Buy

The buy route has its own timeline challenges. Approval and purchase processes can be long, especially in regulated industries where legal and security teams need to sign off. Your team will also need onboarding and training.

On the other hand, a vendor can speed up implementation. Most have consultants or partners dedicated to getting you running fast, and it's usually easier to hold an external vendor accountable to a timeframe than an internal team with competing priorities.

PREPARATION

Estimating budget

Part of your project preparation is estimating costs, and this can be tricky. It's hard to have a realistic grasp of how much work is involved until you actually start, so projects frequently get longer and more expensive as they go on.

Build

Even with a skilled in-house team, time and cost are often underestimated, especially once you factor in other business priorities competing for the same people. There's frequently a feeling that in-house development is free, since there's no invoice to sign off, but the real cost of a couple of months of a development team's time looks very different once you calculate it properly.

AI has lowered the cost of getting to a first working version. What it hasn't lowered is the total cost of ownership. Maintenance, security patching, monitoring, documentation, and governance don't get cheaper because the first draft was faster to produce, and a faster start can mean more of these pipelines exist for your team to carry long term. A cheaper prototype is not the same thing as a cheaper platform.

Buy

Estimating costs for buying a solution is often easier upfront. The harder part is total cost estimation, since you probably don't have people already skilled in the new tool, so it's worth asking your vendor to help you estimate this properly.

They'll have experience with similar projects and solid onboarding practices.

You could also consider outsourcing a skeleton or MVP of your project to a vendor. Given your opportunity cost, it might work out cheaper than it first appears.

PREPARATION

The hidden cost of AI-assisted building

There's a cost to building that's easy to miss, because the tools that create it are new enough that most organizations haven't caught up with what they change. It's worth understanding before you weigh up build vs buy, whichever direction you're leaning.

AI coding assistants have made it fast to spin up a working connector or pipeline, sometimes in an afternoon. That speed is real and it's valuable. But it changes what 'building' actually costs you, and the cost shows up somewhere that's easy to miss.

Every integration is not just code. It's a credential: an API key, a database login, an OAuth grant, often with broader permissions than strictly necessary, because narrow scopes take longer to configure and nobody had the time under deadline pressure. Build twenty connectors this way and you now own twenty standing credentials, spread across ad platforms, CRMs, databases, and warehouses, each one created by whoever happened to be on the team that week.

When a small team, or a single person, can generate a working pipeline in a day with an AI assistant, organizations end up with far more of these credentials, created faster, by more people, often outside the visibility of IT or security. This is shadow IT, but at a speed and scale that didn't really exist five years ago.

There's a simple tell for whether this has already happened in your organization: can anyone produce a complete list of every connection currently pulling company data, who owns it, and what permissions it holds? Most organizations that try this exercise find gaps.

There's a familiar warning in build vs buy conversations that documentation for a homegrown solution often lives in one person's head.

That risk hasn't gone away. It's actually sharper now, because the 'one person' might have used an AI assistant to generate a connector that nobody else reviewed, running with permissions nobody audited, moving data nobody is tracking.

Whichever path you take, the practical question is the same: does every connection your organization creates, AI-assisted or not, run through the same visibility, audit trail, and governance as everything else, or does speed mean it quietly runs around those controls instead?

Something worth checking...

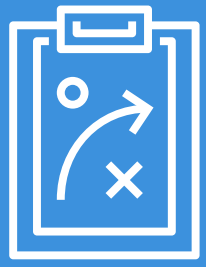
The more of your organization is building with AI assistance, the more this becomes a question of platform rather than policy. Policies get skipped under deadline pressure. An environment that makes the governed path the fastest path doesn't need to be skipped, because there's no faster shortcut to take instead.

Tips: How to buy software effectively

Write it down. Putting your specification down on paper not only helps you get aligned on what you really need and keeps you focused on your requirements, but you can send this document to vendors. It's easier than trying to remember all your points on multiple calls.

Use your vendor. Make your vendors work for you. Talk to them about your setup, they have likely come across similar scenarios before and can help suggest effective approaches.

Try the software (properly). Put effort into your trial to really see if it can work for you. Set up a certain goal, for example, implementing a data pipeline or connecting to a specific system, so you can compare vendors properly.



Consider a Proof of Concept (PoC). Pick a limited scope and ask the vendor to show you how it works. Most good vendors will want to help you because they want to prove that their platform can solve your use case.

Design and development phase

DESIGN AND DEVELOPMENT PHASE

Development and environment

Before you start designing and developing your solution, you need your development environment and processes ready, and a plan for how you'll build your data pipelines.

Build

If you're building the solution yourself, you'll need to consider who will eventually use it. Just the developers, or others too?

You most likely don't want to commit the development team to be constantly available for every change, so you'll need to define the interface and how much freedom users should have.

Buy

Buying a platform means you're getting an already thought-through interface and environment. There's a range of products on the market, from easy-to-use tools for business users to fully flexible environments for developers and data engineers.

There's no single right answer, but this is an area where your vendor should help. They're the experts on their own platform.

DESIGN AND DEVELOPMENT PHASE

Agile Design

Being able to quickly design, iterate, and deploy is a key part of an agile, responsive data culture, and it matters more every year.

Build

Building something in-house gives you the ultimate flexibility to design a solution to do exactly what you want it to do. And your development team don't have to learn a new skill, they get to work with familiar tools and languages. This can often mean happier teams, especially at the beginning – and can mean it's easier to staff those teams with experienced people.

But while working in the language you're used to can mean getting to an initial prototype faster, it can be at the expense of proper scoping of what the end result needs to be. Lack of scalability and fragmentation of pieces of your solution can mean you quickly run into problems.

Another downside can potentially be the effect of "closed source". Not necessarily that you don't have access to the code, but the notion of getting to it, understanding what the code does and sharing that understating among non-technical peers. If this is too difficult it might become prohibitive to collaboration and the code might become "closed" to just a few highly technical team members, with the rest of the organization potentially reluctant to engage and ask for changes and improvements.

Another downside to this type of solution is that documentation is often an afterthought. You can find that all the knowledge to maintain and update your code lives in one person's head. If that person leaves the business, you can be in trouble.

Coding from scratch also means you'll need a skilled development team both to build and evolve your solution – powerful but expensive. And working in code can mean that adding or replacing data sources can have an impact on your whole solution and require major reworks.

Buy

Buying a data platform means you're getting something specifically designed and optimized for data jobs, with dedicated features such as instant inspection of data jobs and any issues or errors.

The advantages of a dedicated data platform include potentially making it easier for less-technical staff to be productive, with data workflows that are easier to build, edit and maintain in a scalable way.

But unlike coding, you'll have to work within the confines of your chosen platform. This can be both a good thing and also potentially limiting, and is something you'll want to thoroughly research during your buying process. The flexibility vs ease-of-use trade off varies by platform, so it's important to decide which balance will suit your needs.

There are a lot of options on the scale of ease-of-use vs flexibility and power, and it's important to choose wisely. You'll need to consider your user base, and how much you want to enable business or less-technical staff to be involved.

If you want business users to have control over data processes, while the IT team still benefits from the power of coding, there are platform that will allow you to combine both, allowing for collaboration between the two teams without the danger of 'closed source'

One of the big advantages of buying is that you'll likely find some built-in integrations with databases, web services and APIs, as well as standardized data formats. It means your team doesn't need to reinvent standard components, and that these connectors will be maintained and optimized by the vendor.

A data management platform can also give you a more scalable base to build on. If for instance operations on your data are abstracted from any particular database, format or infrastructure, it means you can evolve your solution much more easily. Adding or mixing data doesn't require a re-working of your whole workflow.

DESIGN AND DEVELOPMENT PHASE

Testing and QA

Testing and QA is a vital part of any project, but it's often neglected. People think of testing the business process being implemented, but the testing process covers far more than that, and the build vs buy approaches differ.

Build

You'll need to test two different things: the platform or tooling you've built, and your actual data pipelines.

You'll also need to test that reporting, logging, and everything the support team relies on works as expected, and integrate with tools like Jenkins for builds and deployments.

Buy

One advantage of buying a platform is that a lot of testing has already been done for you, and typically better optimized with edge cases resolved. There's still work to do to test that your configuration works as expected for your setup.

If you opt for a platform with strong orchestration and automation, you can use that capability to introduce automated testing of your pipelines, often in combination with your existing unit-test tooling.

DESIGN AND DEVELOPMENT PHASE

AI in the development process: provider model vs. your own model

AI has become part of the development conversation. Both build and buy paths involve some form of AI assistance: generating code, suggesting mappings, drafting documentation, even planning a whole implementation from a specification document. This raises a question: whose AI is doing the work, and where does your data go while it works?

Two different models

Provider-hosted AI means you send your data, specifications, code, or schemas to a shared AI service run by whichever tool you're using. It's convenient, but you're trusting someone else's infrastructure, someone else's data retention policy, and often a model version you don't get to choose.

Bring your own key, or bring your own model, works differently. You choose which AI provider does the work, or you run a model entirely within your own infrastructure. You hold the contract, you hold the cost, and you keep control over exactly where your data's final destination is. Nobody else sits in the data path between you and the model you've chosen.

Why is this important?

- **Cost control.** You pay only for what you use, directly to the provider you've chosen, without an additional markup layered on top.
- **Flexibility.** You can pick the best model for a given job, or move to a different provider as the market shifts, without being locked into one vendor's roadmap.
- **Control.** Your data and prompts go only where you've decided they should go, which matters enormously if you're already thinking carefully about data sovereignty, as covered earlier in this guide.

When you're evaluating any platform, build or buy, that offers AI-assisted features, this is worth asking directly: does it run on a model and provider that you choose and control, or are you locked into whatever the vendor has picked on your behalf?

It's also worth asking whether the platform lets you connect your own coding tools and AI assistants through an open standard, rather than confining you to a single, built-in assistant. Open protocols for this, such as MCP (Model Context Protocol), are becoming a meaningful differentiator between platforms that give you a walled garden and platforms that give you a genuinely open foundation to build on.

Tips: Effective deployment

Take advantage of training. Most vendors will offerdifferent levels of training. Making sure that every user is trained properly will pay dividends in how quickly and efficiently you can get up and running, as well as how effective you will be going forward.

Ask your vendor for help with deployment. They canhelp you deployin a way that provides maximum security, stability and scalability of the solution.

Treat your relationship with your vendor as a long-term one. A good vendor won't want to just take your money and run, but they'll want to help you and grow with you. This means helping you to expand your use of the platform in a way that brings you benefits, but from their point of view also keeps you wanting to pay a subscription or buy more products.

Operational considerations

The implementation phase of a project often attracts all the focus and overshadows the, arguably more important, need for operational reliability. Ensuring your solution is maintainable, scalable, and secure is crucial to delivering business value. If you're not delivering accurate data on time to where it needs to go, that has an impact across the organization.

OPERATIONAL CONSIDERATIONS

Reliability and trust in production

When you get to production, this is where you really reap the benefits of the work you put in earlier to think about how you monitor your solution.

Build

In our experience, operation in production is often an afterthought with in-house builds. It's one thing to build something that works, and something else entirely to keep it running reliably, 24/7. It's often only after the first real failure that monitoring and alerting become a priority.

Third-party tools such as New Relic, Datadog, and Splunk can help with a lot of this, and it's far easier than trying to build monitoring from scratch.

Buy

Look for platforms that focus on the operational side of data pipelines just as much as the design side, if not more. Your investment is only valuable if it delivers and people can trust it.

The platform you buy should include strong monitoring, whether that's built into the solution itself or through integration with your existing standardized tools.

OPERATIONAL CONSIDERATIONS

Scalability and performance

It's rare that you deploy something and it stays the same forever. Data volumes grow, new use cases appear, or new functionality is needed.

Build

Proper scalability is something you should plan for from the start of a build, or deliberately decide not to if you're confident you'll stay small.

When the time comes to scale up, it often calls for a complete overhaul or rewrite, and scaling can be hard to implement and test without prior experience of clustering or parallel processing.

Buy

You also need to think about scalability when buying, even though implementing it isn't your responsibility. Think about how much data you'll need to process a few years from now, so you're not setting yourself up to repeat this whole process.

The advantage of buying is that scaling tends to be more predictable, often just a case of buying more licenses or capacity, though it's worth understanding how those costs evolve over time.

OPERATIONAL CONSIDERATIONS

Security

Security considerations vary considerably from organization to organization. Whether you're building or buying, you need to make sure your solution meets, and can keep up with, your security requirements.

Build

Your organization might be lucky and have excellent security infrastructure and a development team used to implementing things properly. But often that isn't the case, and additional capabilities like SSO or corporate directory integration need to be built out separately.

This now extends to AI usage too. If your developers are using AI coding assistants day to day, the same questions apply: what data is being sent to those tools, under what retention policy, and who has visibility of it?

Buy

It's easier to look for tools that already have what you need in terms of security integrations, such as SSO or LDAP, or that connect cleanly to your existing tools.

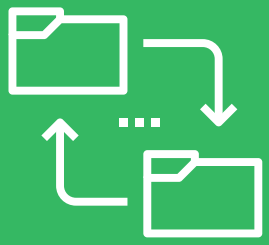
The same applies to any AI features bundled into the platform. Look for the same BYOK or BYOM control described earlier in this guide, so AI-assisted work is covered by the same security posture as everything else.

Tips: Effective production operation

Monitor everything. It can seem expensive, but can really pay off to monitor everything you can, such as the number of records that flow into and out of the system, how many jobs fail and how often, how long jobs take and more.

Involve your business users. Business users often know at a high level what kind of data is supposed to be flowing through the system, and if you provide them monitoring summaries they'll learn how to spot if something looks wrong.

Look for improvements. Generally speaking, business users love it when technical teams look for improvements to inefficiencies – especially if it means they get their reports faster or more accurately.



Have backups (and test them). We've seenseveral times the problems that arise if companies don't have backups. Or they had backups but didn't test them - so when the time came to restore something, they couldn't. So again it's something you should plan for from the beginning.

Solution support & maintenance

SOLUTION SUPPORT & MAINTENANCE

Support

As soon as you deploy something into production, you need to start supporting it, whether that's in-house support for something you built yourself, or working with an external company.

Build

If your solution is homegrown, your support will be too, and it usually falls to the original development team.

In our experience, this is something every developer dislikes doing, so it's worth planning and budgeting for accordingly.

Buy

When you buy software, support is often easier, in the sense that you can pay for the level you need. 24/7 support from a vendor can be cheaper than doing it yourself, since the cost is spread across their whole customer base.

You'll still need some in-house resource for tasks like user management and basic queries, but people for these roles are generally easier to find, and cheaper, than developers.

SOLUTION SUPPORT & MAINTENANCE

Platform Updates

Keeping your platform updated is something you should plan for and focus on, but it's not always top of mind.

Build

If you've built your own solution, updates are unlikely to happen most of the time, because they tend to become reactive: an issue gets reported, a security update is needed, or a new business requirement appears.

You're unlikely to be developing new features 'just because'.

Buy

If you buy, the product keeps being updated, as long as the vendor wants to stay in the market, so you get new functionality for free. Vendors are now shipping AI-assisted features at real pace, so keeping on top of new versions matters more than ever if you want to benefit from them.

Depending on your vendor, you may also be able to influence new development by suggesting features that would help you.

New requirements and fixes

The second kind of update is one that stems from new requirements from the business, as well as bug fixes.

Build

If you want to change something in a homegrown platform, you'll need to adopt a full development lifecycle for it, because any change might impact the platform in ways you haven't anticipated.

Buy

Changing something in a purchased platform can be as simple as updating business logic, adding a transformation, or modifying a data layout. The ability to make substantial changes without reconsidering your purchase depends heavily on how flexible your chosen platform is.

Tips: Effective support and maintenance

Keep track of the product. This is another place where you can get an advantage from keeping a close relationship with your vendor. Keeping on top of new features can help you decide when to upgrade, as well as enabling you to incorporate new functionality into your processes.

Use support. Working with vendor support teams is (or should be) more effective and efficient than struggling on your own or trying to create workarounds.

Update often. Lagging several releases behind not only means you miss out on new functionality, but makes it harder to do big jump updates.

Build vs buy – What does CloverDX have to offer?

CloverDX is built for the buy-and-extend path described earlier in this guide: a platform that gives you a production-grade operational foundation, with full coding capability underneath it for the parts of your data estate that need to be genuinely yours.

Design and implementation

- A design environment specifically built for data jobs, by a team with an extensive background in data projects for clients
- Full coding capability combined with visual representation of data pipelines giving the flexibility and familiarity of code, with the transparency and speed of a visual designer
- Ready-made and optimized integrations for reliable operation
- Easier team collaboration through reusable components, version control and self-documenting modules

Production operation

- Visual workflows for data jobs and related processes, all in one centralized place, for simpler management and troubleshooting
- Robust execution platform for reliable and optimized operation
- Built in monitoring and triggering mechanisms so you can automate workflows on a schedule or in response to an event
- Works with your tech stack. Deploy on-prem or in cloud, hybrid, on Linux, Mac or Windows, and in containers such as Docker.
- Designed to scale for performance and/or reliability, as well as handling increases in data volumes or sources – all at predictable costs (scales by CPU core and/or cluster node).

www.cloverdx.com

AI, sovereignty and control

- Self-hosted deployment, on-prem, in your own cloud, or hybrid. Your data never has to leave your environment.
- Bring your own key or bring your own model. Choose your AI provider, such as Anthropic, OpenAI, Azure OpenAI, or Google, or run local models, with no CloverDX markup on usage.
- An open MCP Server that lets you connect your own AI tools and coding assistants directly to your data platform, under your existing security controls, rather than confining you to a single built-in assistant.
- A full change history and rollback across every job, whether it was built by a person or with AI assistance, so nothing is a black box and nothing lives only in one person's head.

Support and maintenance

- We'll help you design and build your architecture for maximum efficiency, to avoid costly mistakes down the line.
- Tech support to help you overcome obstacles quickly.
- Extra capacity of experienced resource available if your project hits tight deadlines or you need to augment your team.
- Regular platform updates with improved functionality, security updates, and new features

www.cloverdx.com

Cracking the Build vs Buy

Dilemma for Data Integration Software

CloverDX

The data integration platform that puts you in control.

CloverDX is a data integration platform with AI-assisted pipeline development, AI-enabled transformations, and self-service tools for business users, deployed on-prem or in the cloud, on your infrastructure and your terms.

Build pipelines visually, in code, or with AI. Run them on autopilot or on demand. Engage more of your people in the work itself. Know exactly what happens at every step, and prove it.

And own the whole thing: the infrastructure it runs on, the data inside it, and the experience your people use

www.cloverdx.com