MadeTech

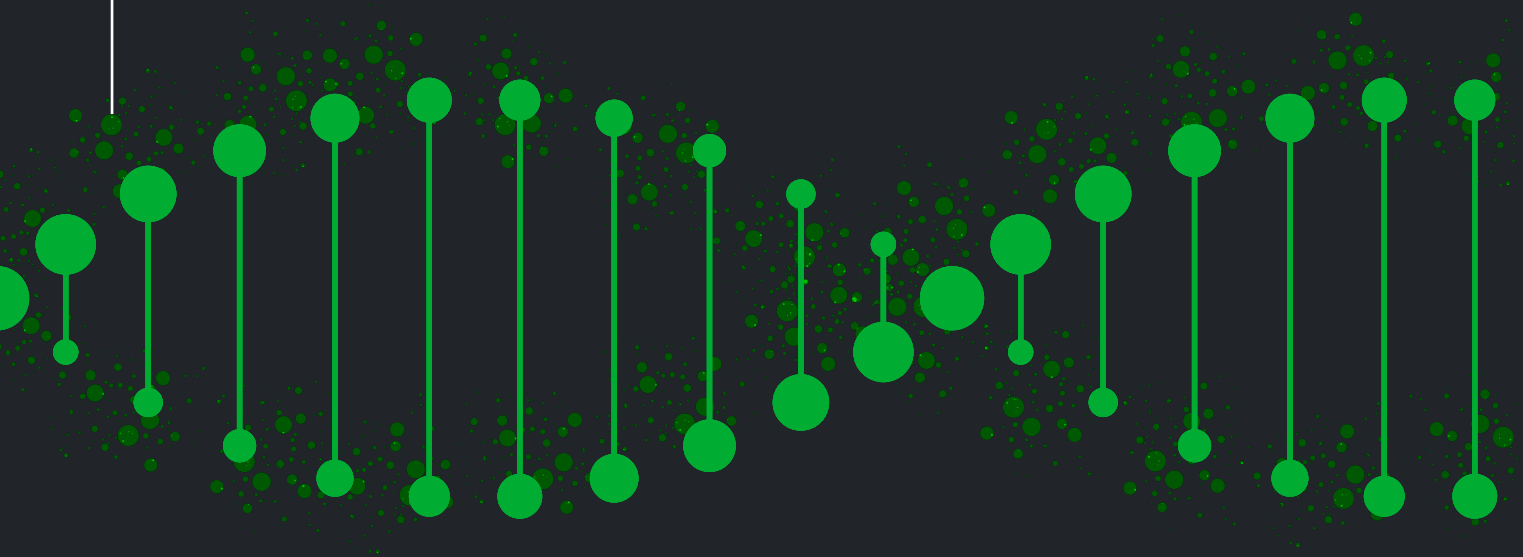
Building future-ready data platforms

When it comes to designing a modern data platform, one thing is certain: change is inevitable. The tools you use today may not be the ones you rely on tomorrow – and that's okay. The real measure of a successful platform isn't how perfect it is on day one, but how easily it can adapt and evolve over time.

Whether you're starting from scratch or modernising an existing stack, the goal is the same. Build something flexible enough to scale, smart enough to avoid costly mistakes and open enough to change direction when you need it to.

This guide shares practical insights on how to do just that. From selecting the right foundations to applying DevOps thinking. If you're planning your next data platform move, start here.

A quick look at what's inside this guide



Contents

Building future-ready data platforms

Design for change

When planning your data platform, it's easy to think that the latest technology (with all the bells and whistles) is the answer to everything. But it's important to accept one truth early on. Most of the technology you choose will eventually need to be replaced. This doesn't mean you've made the wrong decisions. It just means your platform should be designed to embrace change from the start.

Evolving architecture and emerging design principles are key. They allow you to build with the expectation of future transformation. Designing for change also means choosing tools and technologies that don't tie you to a single provider.

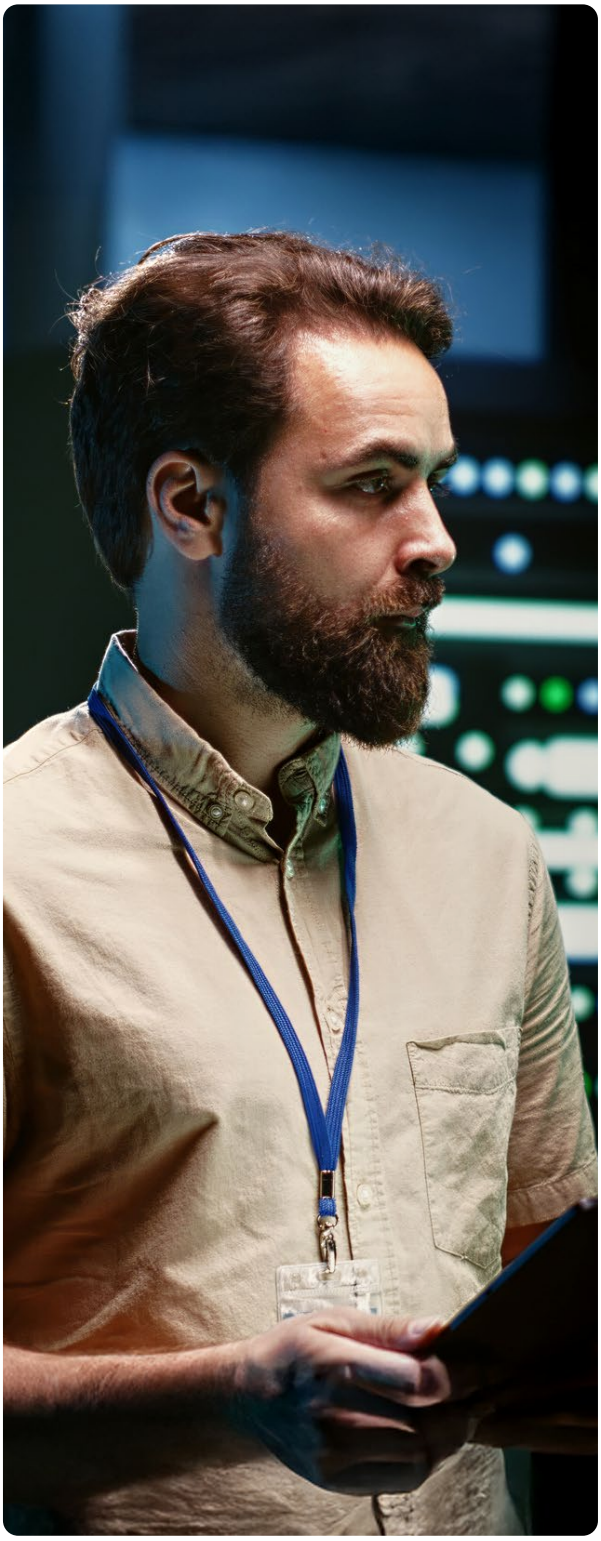
Start simple, scale when ready

Choosing the right technology isn't easy. It needs to reflect where your organisation is now and where you're heading.

In our experience, over 50% of the technologies you implement will likely be replaced in 2-3 years as your data maturity evolves and your use cases change.

Instead of sticking with familiar tools, assess both your technical and data maturity, as well as the complexity of your use cases. The sweet spot where these factors intersect is where the ideal technology choice lies.

For example, your ingestion mechanisms might vary based on complexity. If you're just copying CSV files, a hand-crafted Spark script might suffice. But if you're pulling data from systems like Salesforce or SAP, it's often better to invest in pre-built connectors rather than writing your own. The same principle applies across all services: start simple, address the immediate need and scale complexity as your platform and team mature.



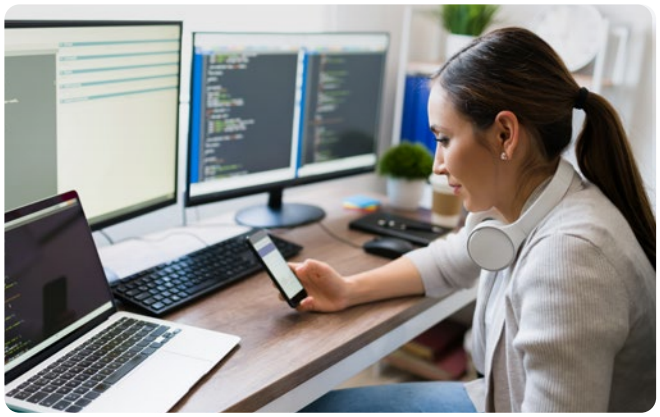
Reducing the risk of vendor lock-in

One of the biggest risks when choosing technology is relying so heavily on one provider that switching later becomes difficult or extremely expensive.

Wherever possible, opt for open-source tools and technologies, specifically ones that have managed instances in the main cloud provider platforms. Apache Spark is a great example. Multiple engines support its code execution, so you're not tied to a specific cloud or compute provider. Similarly, tools like Airflow offer open-source and managed options across the 3 major cloud platforms, allowing you to balance flexibility with convenience.

The goal is to minimise vendor lock-in while maintaining agility. Yes, open-source technologies can bring their own form of dependency, but they provide far more freedom than proprietary solutions.

So far we've looked at how starting simple, aligning technology with your needs and avoiding vendor lock-in sets the stage for a platform that can evolve over time. But this is just the start. Building for the future means prioritising flexibility at every decision point.



Avoid the gold-plated trap

It's easy to get carried away with the advanced, feature-packed technology solutions that promise to do it all. But if you go all out and invest everything in one perfect solution upfront, you may end up with a gold-plated platform – one that's overly complex, expensive and possibly redundant in 2 years time.

Instead, focus on your use case and the cultural maturity of your organisation. You can spend a lot of money on any service, but it's critical to consider where that investment is actually needed. In many cases, a simpler, 'good enough' approach will work.

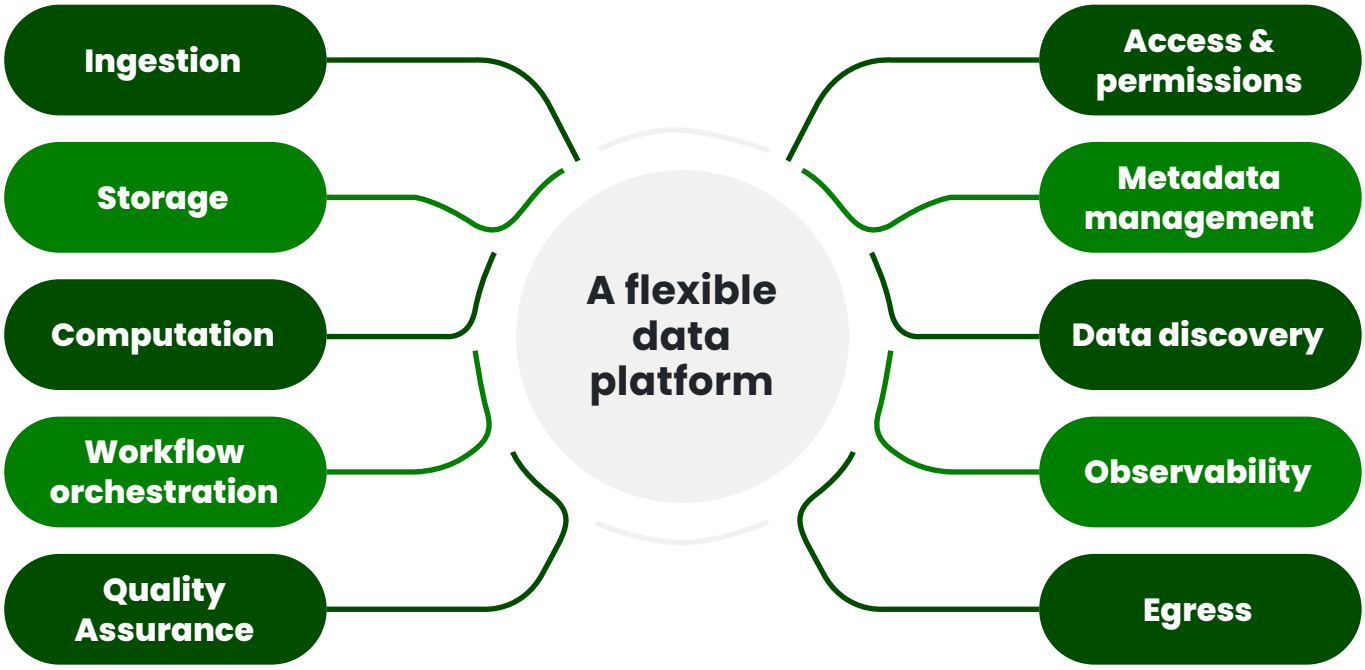
Take one public sector client, for instance. Their primary focus was on cataloguing, discovery and metadata contracts. Recognising that their work revolved around sharing data with other organisations, they prioritised building a robust data catalogue and metadata management system. This meant that their analysts could easily find and use the data they ingested. They designed everything else – ingestion, storage, and transformation – to meet the smallest possible functional, secure and safe standards while keeping costs in check.

By aligning their technology investments to these key priorities, they avoided over-engineering their platform. They delivered exactly what their use case demanded – nothing more, nothing less.

It's an important reminder to design a platform that prioritises what's really important to your organisation.

The 10 key factors to focus on when designing a future-proof data platform

A future-ready data platform needs more than just the right tech. Every decision – how you ingest, store, process and manage data – should prioritise flexibility. A rigid platform will slow you down, while a well-designed one will allow you to adapt to changing requirements with minimal disruption.



By designing your platform with flexibility in mind, you can avoid over-engineering. The result will be a platform that remains responsive to your future needs.

But having a flexible tech stack is just one piece of the puzzle. To truly future-proof your data platform, you also need the right operational practices to keep it efficient and adaptable.

Here are the 10 elements to consider when designing a future-proof data platform:

1. Ingestion

Define the principles for pulling data into your platform in a way that reduces dependency on specific systems. Loose coupling means the platform will be able to adapt to evolving data sources. Assigning clear ownership will create accountability and predictability when exchanging data. How will your pipelines scale to handle both current and future data volumes?

2. Storage

The design of your data storage should accommodate change with minimal disruption. Go for modular solutions that allow you to adjust the format, schema or location of data without rewriting your entire platform. Use partitioning and naming conventions to simplify upgrades, cut costs and speed up retrieval.

3. Computation and transformation

Data transformations should be built on reusable, well-documented functions that rely on standardised domain language. Idempotent transformation patterns – where the same inputs always yield the same outputs – help with consistency and reliability. This approach means that as your platform evolves, new transformations can integrate seamlessly without breaking existing processes.

4. Workflow orchestration

Flexibility in triggering and linking transformation jobs is going to be key when the goalposts inevitably move. Choose tools and patterns that support dynamic scheduling and can allow workflows to scale or reconfigure as requirements change.

5. Quality Assurance

Introduce robust methods for testing and maintaining data quality as it flows through the platform. As data volumes grow or new sources are added, continuous monitoring and anomaly detection will allow you to catch and resolve issues early.

6. Access and permissions

Prioritise security and efficiency. Define user roles, audit trails and data handling to stay compliant while keeping onboarding flexible for new teams and partners.

7. Metadata management

Data about your data plays a pivotal role in discovery and management. One way we address this is through data contracts. These aren't legal agreements but rather encapsulations of all relevant metadata: schemas, usage rights, sharing policies, ownership and more. When co-published with data, contracts streamline cataloguing and discovery. This makes it easier for teams to find and use what they need, as well as adapt to changing requirements.

8. Data discovery

A platform is only as useful as its discoverability. Build tools and processes that help your users easily find, understand and use the available data. Make sure that your discovery mechanisms can scale to match your datasets.

9. Observability

Observability is critical for understanding how your platform operates. When something fails (and it will), you need to pinpoint where and why it failed, assess the impact, and re-run processes where necessary.

This is where tools like lineage tracking and telemetry come into play, helping teams stay on top of system health.

10. Egress

Prioritise modularity when it comes to your design. Whether data flows into visualisation tools, operational systems, or external APIs, make sure that your egress mechanisms are scalable and adaptable to new business needs.

Borrowing best practice from DevOps to build your data platform

A flexible tech stack is a great foundation for a future-ready data platform, but it's not enough on its own. Even the best-designed platform can become a bottleneck without the right governance, security and operational processes.

This is where borrowing from DevOps can help. DevOps practices have transformed how teams build and deploy software – offering the same benefits to data platforms.

By integrating automation, continuous deployment and secure-by-design principles, you make sure that your data platform remains adaptable and scalable. Let's take a look at some key DevOps strategies that can help you get there.

CI/CD pipelines improve data platform agility

Continuous integration and deployment (CI/CD) pipelines are the backbone of agility. They're essential for managing platform updates, releasing changes incrementally and reducing downstream impacts. For data platforms, this means ensuring updates don't introduce data pollution or create breaking changes in pipelines. Automating these processes creates consistency and allows for faster iteration without sacrificing quality.

Manage data environments with Infrastructure-as-Code (IaC)

Data platforms often require multiple environments (such as development, staging or production), but handling data across them comes with challenges. Personal data should never be used outside production. Instead, use synthetic data to replicate scenarios safely. Infrastructure-as-Code (IaC) ensures that your environments remain consistent, reducing the risk of configuration drift or security gaps.

Runbooks and cookbooks: simple guides for smoother operations

Runbooks provide documented solutions to common issues. They give you knowledge at your fingertips. Tie these directly to alerts so support teams can respond quickly and effectively. Cookbooks – repositories of best practices – serve as a go-to guide for recurring patterns or efficient approaches. They can be stored as code snippets, wikis, or documentation.

Use FinOps to keep costs under control

Data platforms can become expensive quickly, especially when mistakes happen. We've seen cases where something as simple as a webpage auto-refreshing overnight racked up thousands of pounds in cloud costs. By monitoring spending closely and adopting a defensive design approach, you can prevent runaway costs and ensure resources are used efficiently.

Make your data platform more resilient

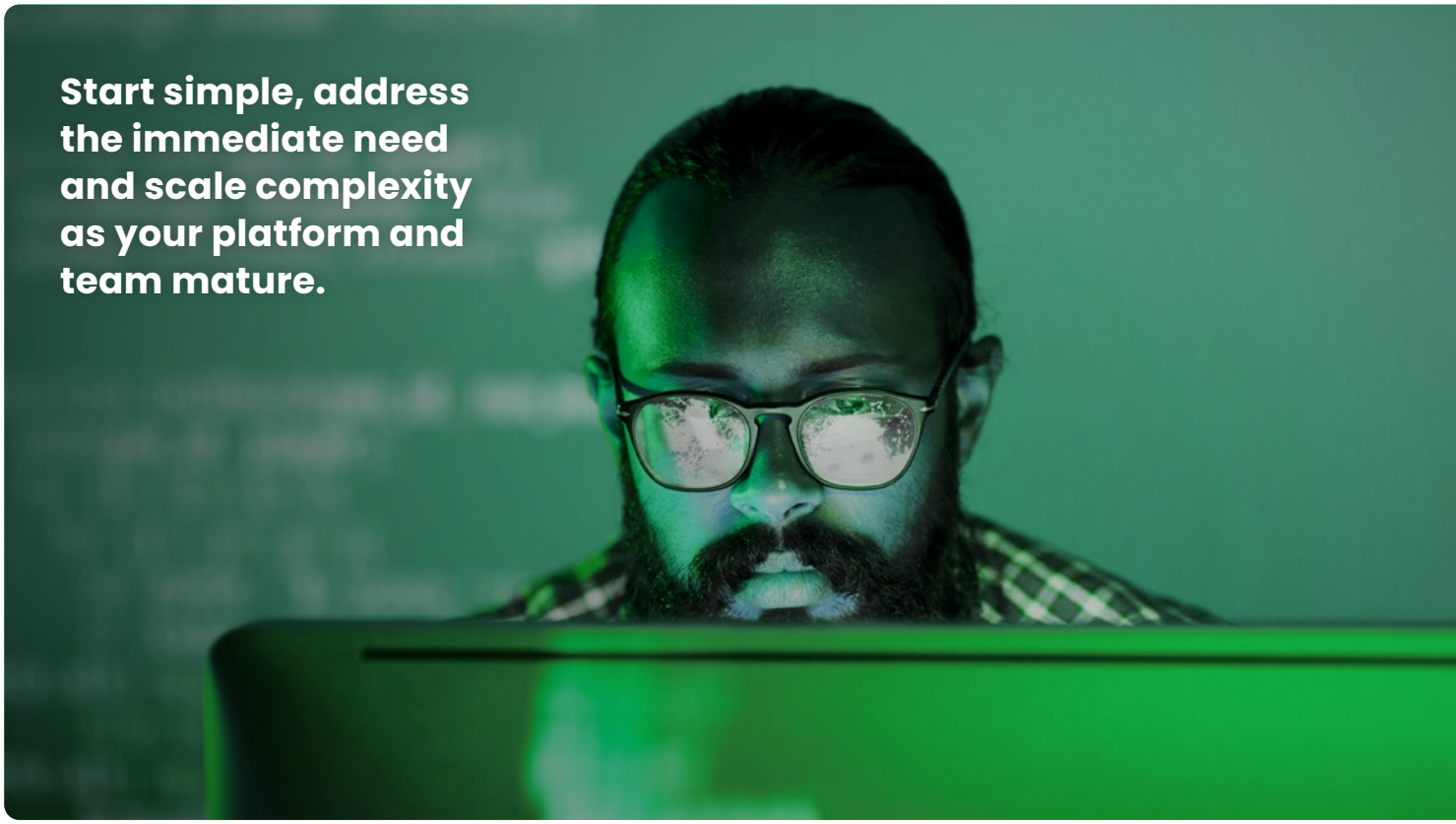
Defensive design, or designing with a 'what could go wrong' mindset ensures your platform can handle unexpected failures gracefully. Incorporate health checks, redundancy and fallback mechanisms. Use telemetry data and observability tools to proactively detect and address issues before they escalate.

Build your data platform 'secure by design'

Security should never be an afterthought. It's a foundation. Apply the same 'secure by design' and zero-trust principles used in DevSecOps. Enforce role-based access control, implement audit trails and conduct regular penetration testing. This will keep your platform resilient against threats.

Optimise maintenance with observability and cost control

Proactively design your platform with maintenance in mind. Observability tools, telemetry data and well-structured runbooks all help support teams identify and resolve issues efficiently. FinOps practices make sure the platform remains cost-effective, while documentation like cookbooks reduces friction for future development. By weaving DevOps principles into your build, you set the foundation for a platform that's not just functional but also secure and ready for change.



In summary

The best data platforms aren't perfect – they're built to evolve

In conclusion, it's not about getting everything perfect upfront. No matter how well you plan, your platform will evolve. Technology changes, use cases shift and new challenges arise. By designing with flexibility in mind, you make it easier to replace or upgrade components without disrupting the entire system. Building a successful future-ready data platform isn't about chasing the latest tech, it's about designing for continuous evolution. Your platform needs to be flexible, adaptable and cost-effective, ready to embrace change rather than resist it.

If you're looking for support in building a data platform fit for the future, visit our data and AI webpages or speak to one of our data and AI experts.

Data and AI webpages
www.madetech.com/services/data-and-ai

Speak to one of our data and AI experts
www.madetech.com/contact

> **A future-ready data platform needs more than just the right tech. Every decision – how you ingest, store, process and manage data – should prioritise flexibility.**
>
> – Jim Stamp, Head of Technology

Jim Stamp is an experienced tech leader dedicated to using technology for positive change. As Head of Technology at Made Tech, he drives the mission to help public sector organisations use digital tools and data to improve public services. Jim makes sure these services are more efficient and user-focused.

With a background in research and software and data engineering, Jim has extensive expertise leading teams through complex projects. He shapes Made Tech's tech strategy, fosters innovation, and oversees the delivery of impactful projects.

Jim champions open source tech, data-driven approaches, and a culture of learning and collaboration. His leadership prioritises both technical quality and real societal benefits.

Outside Made Tech, Jim mentors future tech leaders and supports discussions on ethics and sustainability in technology, reinforcing his commitment to innovation and community growth.

About Made Tech

At Made Tech, we use technology to improve society – for everyone.

Made Tech uses technology to improve society for everyone. We work with public sector organisations to create modern, user-friendly digital services that make a difference. Whether it's helping the NHS improve patient access to mental health services or supporting the UK government to update outdated systems, our goal is to make technology work for the people.

We focus on building secure, data-driven solutions that are easy to use and quick to implement. By using agile methods, we help our clients move fast and stay flexible, ensuring they can adapt to the future without being held back.

At Made Tech, it's not just about delivering technology. It's about creating lasting change. We want to help public services improve lives, save time, and meet the evolving needs of today's society and beyond.

Find out more at www.madetech.com.

Made Tech