RTInsights Research

Build vs Buy:

The Data Leader's Handbook to Choosing a Data Observability Platform

Sponsored by Bigeye

Table of Contents

Chapter 1- Data Observability Introduction 4

What is Data Observability?

Why Data Observability?

How Does Data Observability Work

Where Should Data Observability be Used?

How is Data Observability different from Data Quality and Monitoring

Chapter 2 - Data Observability State of the Union 9

Data Observability Landscape

Observability Tools: Capabilities and Criteria

What it takes to build a Data Observability tool

Chapter 3- Data Observability Selection Criteria 13

To buy or build a data observability solution

Factors to Consider When Deciding to Build

Checklist for Building a Data Observability Solution

Understanding the Reality of Building In-House

When Building In-House Makes Sense

Challenges and Pitfalls of Building In-House

Path to Production and Maintenance

Factors to Consider When Deciding to Buy

When Buying Makes Sense

Common Pitfalls When Buying

Should You Build or Buy?

Chapter 4 - Build Vs Buy Decision Analysis 18

Metrics for Data Observability

Total Cost of Ownership (TCO)

Cost Estimates for Data Observability Solutions

Building In-House

Open-source solution

Vendor Solution

Final Thoughts 23

About 25

Preface

Imagine this: a database used for your organization's daily operations suddenly goes offline. Scheduled data ingestion halts, transformation processes fail to execute, dashboards go dark, and downstream analytics grind to a standstill. For enterprise data teams, the challenge is clear yet daunting—identifying the scope of the problem, determining root causes, and restoring operations across a sprawling data ecosystem. Even for teams with strong processes, addressing issues of this scale can quickly become a time-consuming and resource-intensive effort.

As data platforms grow in complexity, these types of disruptions are becoming more frequent and impactful. Data engineers are often stretched thin, battling to meet tight SLAs while juggling a growing list of requests from stakeholders. This firefighting approach not only increases operational risk but also contributes to team burnout, delayed insights, and lapses in data governance and quality.

This guide examines how data observability can help enterprises move from reactive to proactive data management. It provides a framework for evaluating whether to adopt a vendor solution or build an in-house tool, with a focus on understanding trade-offs in scalability, reliability, and operational efficiency. Choosing the right approach is critical for navigating the increasing complexity of today's disaggregated data stacks.

Data observability is a rapidly evolving discipline that goes beyond traditional data quality. It empowers teams to monitor the health of their data, pipelines, and architecture continuously, enabling them to identify and resolve issues faster. In a world where enterprises depend on data for strategic operations, revenue growth and AI, ensuring data availability and accuracy is no longer optional—it's essential.

To meet these demands, data teams must rethink their approach, treating data as a dynamic, ever-evolving resource. By shifting from static testing methods to strategies rooted in continual monitoring and observability, teams can enhance resilience, streamline workflows, and safeguard their organization's most critical assets. This guide will help you explore your options, weigh the pros and cons of building versus buying, and unlock the full potential of data observability for your enterprise.

What is Data Observability?

Enterprise data observability is the practice of understanding and monitoring the health and state of data across its entire life cycle. From ingestion to processing and final utilization, data observability provides a comprehensive view into how data is moving, where it is consumed, and whether it meets the quality standards required for effective use.

Data observability detects drifts & anomalies – data, schema, configuration, infrastructure usage - that are outside the purview of data quality and therefore represent "unknown unknowns".

Think about it as the diagnostic lens for your data ecosystem, enabling teams to answer questions like, "Is the data ready for downstream consumption - for AI, BI and advanced analytic applications - with high quality, reliability, trust and governance." It tracks lineage by mapping upstream and downstream artifacts, giving data teams a clear understanding of where the data came from, who interacted with it, changes made and ultimately by whom, how and when it is consumed.

Data observability primarily focuses on detecting data quality issues that might disrupt downstream applications. It analyzes data, data metrics, events and traces to identify critical issues- like application or transformation errors, volume changes, or unexpected shifts- across the entire data stack. Additionally, it leverages metadata, telemetry data, lineage data, data infrastructure resource logs and events to detect abnormalities, allowing teams to catch problems that could affect SLAs (Service Level Agreements) and SLOs (Service Level Objectives), before they happen.

Learn more about observability terminology in our **Glossary of Data Observability Terms**.

Why Data Observability?

Data downtime is inevitable, and the cost of compromised data quality increases with the complexity of the data environment. When issues arise, having a fast, reliable means of detection and resolution is essential. The proliferation of internal and third-party data sources only adds to this complexity, increasing the risk of data errors. Data observability empowers data teams to be proactive, catching potential failures in the data supply chain before they cascade downstream.

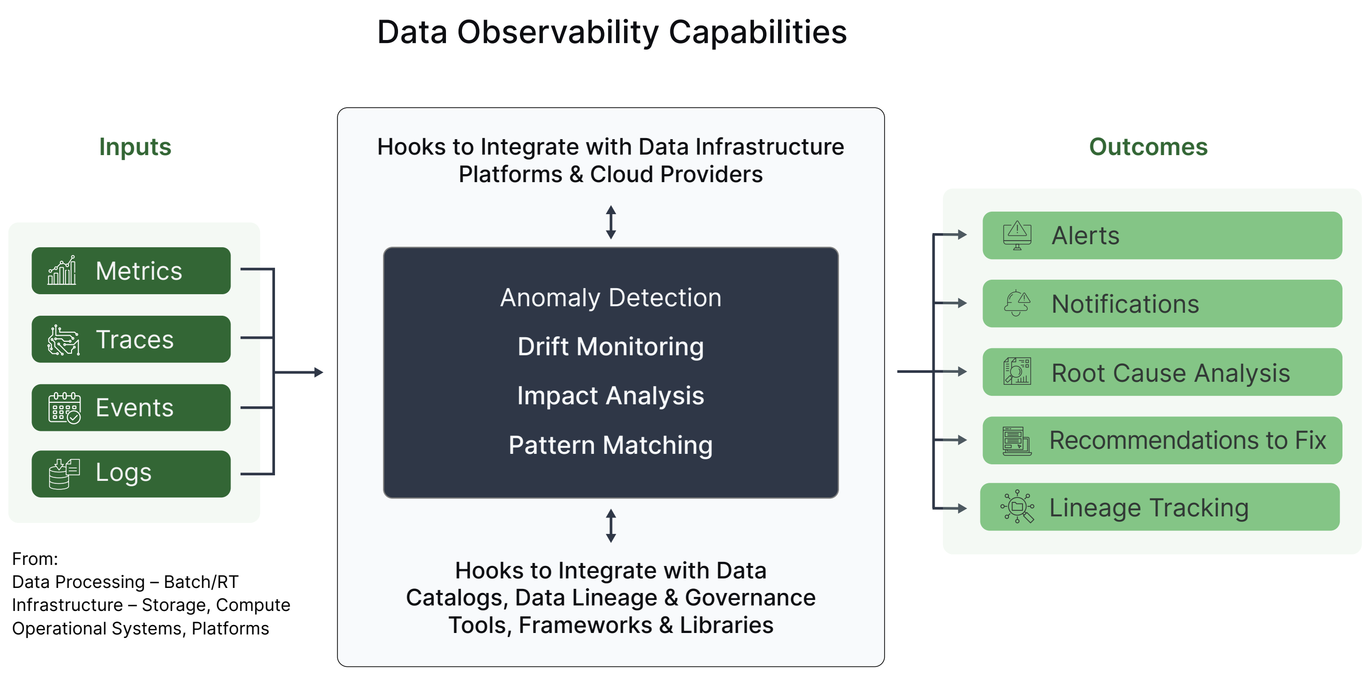
Beyond preventing data downtime, data observability builds trust and enhances decision-making by providing a reliable, real-time view of data health. This capability helps organizations address questions like, “Is this data production-ready?” or “When was this dataset last updated?” Advanced data observability solutions can detect data drift, identify root causes of issues, and even recommend or automate remediation steps, ensuring that data remains fit for AI models, regulatory compliance, and critical business decisions.

Data observability also reduces the mean time to detect and repair data issues, helping teams understand the quickest route to fixing a problem, mitigating downstream impacts and minimizing trust erosion. Data observability needs to be applied across all components of the enterprise data stack residing across the data centers (on-prem, cloud) to really justify its value and ROI. Without data observability, data teams are often blindsided to critical issues that can compromise enterprise data trust.

How Does Data Observability Work?

Data observability operates by collecting signals at every step of the data life cycle. These signals, sourced from components like logs, jobs, datasets, schemas, dashboards, and models, are fed into a data observability tool to detect issues at scale. The tool then alerts the relevant team of anomalies, outliers, or unexpected changes in data behavior.

Figure 1 shows how these data observability systems work, at a high level.



Data Observability Capabilities
Inputs
Metrics
Traces
Events
Logs
From:
Data Processing – Batch/RT
Infrastructure – Storage, Compute
Operational Systems, Platforms
Hooks to Integrate with Data Infrastructure Platforms & Cloud Providers
Anomaly Detection
Drift Monitoring
Impact Analysis
Pattern Matching
Hooks to Integrate with Data Catalogs, Data Lineage & Governance Tools, Frameworks & Libraries
Outcomes
Alerts
Notifications
Root Cause Analysis
Recommendations to Fix
Lineage Tracking

Figure 1: Data observability system

Where Should Data Observability be Used?

Look at the cross-sectional view of an end-to-end data pipeline (highly simplified).

DATA OBSERVABILITY IN AN END-TO-END DATA PIPELINE

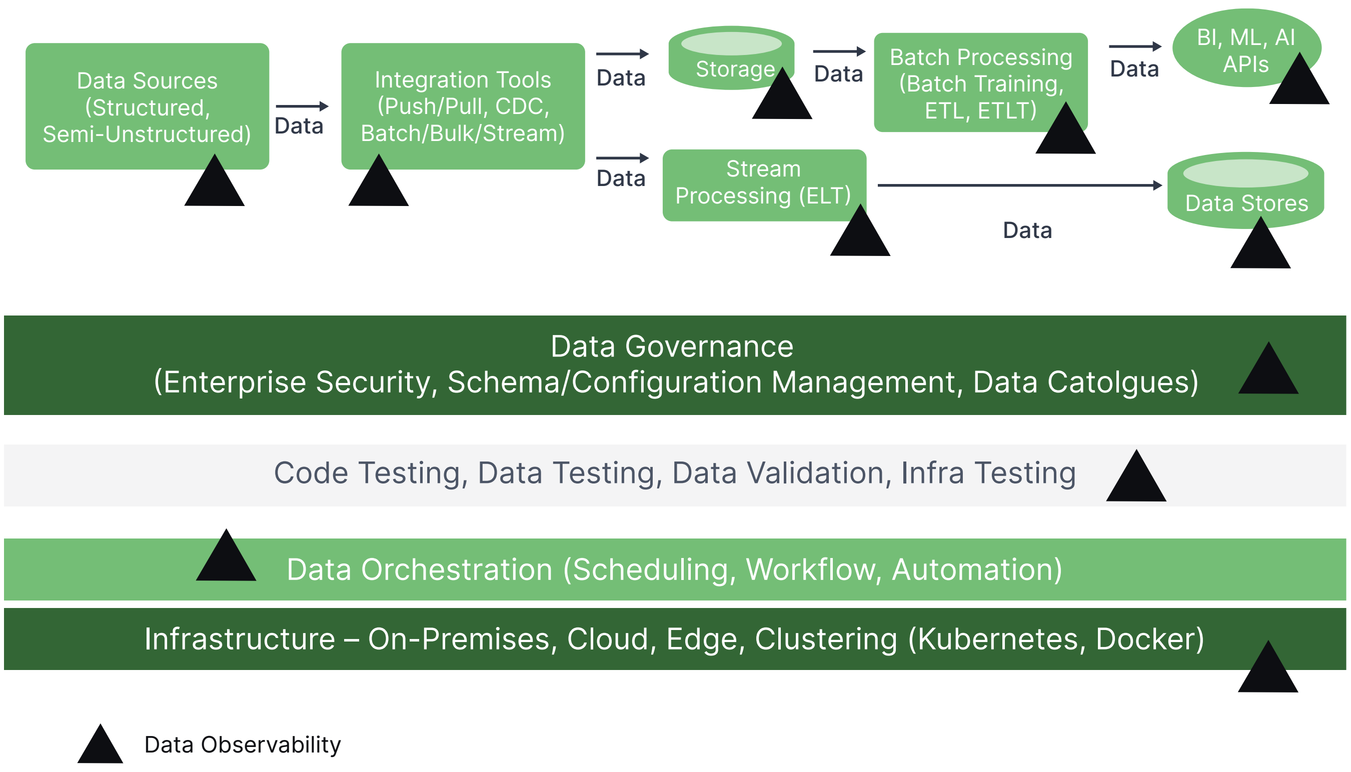

Data Sources (Structured, Semi-Unstructured)
Data
Integration Tools (Push/Pull, CDC, Batch/Bulk/Stream)
Data
Storage
Data
Batch Processing (Batch Training, ETL, ETLT)
Data
BI, ML, AI APIs
Data
Stream Processing (ELT)
Data
Data Stores
Data Governance (Enterprise Security, Schema/Configuration Management, Data Catolgues)
Code Testing, Data Testing, Data Validation, Infra Testing
Data Orchestration (Scheduling, Workflow, Automation)
Infrastructure – On-Premises, Cloud, Edge, Clustering (Kubernetes, Docker)
Data Observability

Figure 2: Data observability in a data platform

Each block could be running on different hardware, across different data centers (from on-premises to cloud and multi-cloud). Typically, most enterprises will have 10-1,000s of pipelines running on this kind of a platform. Each block can be a failure or choke point affecting downstream systems and violate SLOs, SLAs causing data downtime.

How does one go about debugging, troubleshooting issues to figure out what broke, why and where and how to fix it– all while minimizing the MTTR (Mean Time To Resolution)? These tasks are challenging enough for a single data pipeline; now imagine the scope and scale of the problem across thousands of interconnected pipelines.

This is where data observability becomes indispensable. It relieves data teams by proactively providing deep insights into the platform's inner workings, spanning every component of the data architecture. By analyzing data from logs, traces, events, metrics, lineage monitoring and more, it enables teams to identify potential issues related to data freshness, volume, or accuracy. Advanced capabilities like automated root cause analysis and issue resolution recommendations further reduce operational overhead and increase the speed with which data incidents can be resolved.

Think about data observability as a silent agent deployed on each of the blocks, that is continuously monitoring data and applying pre-built ML algorithms (which were developed offline – from historical data but could be updated online) to detect possible failure scenarios.

One of the key enablers of this proactive approach is advanced data lineage. At scale, understanding the flow of data across pipelines, systems, and transformations is critical to pinpointing the origin of a failure. With **lineage-enabled workflows**, data teams can map dependencies, trace issues back to their source, and understand downstream impacts, turning an otherwise complex troubleshooting process into a streamlined, efficient operation.

Incorporating advanced data lineage into data observability workflows isn't just a best practice—it's a requirement for scaling observability across enterprise environments. Without it, teams face blind spots that hinder their ability to quickly diagnose and remediate issues. With it, data leaders can ensure trust, reliability, and efficiency in their data operations.

How is Data Observability Different from Data Quality and Monitoring?

Data quality, data monitoring and data observability are distinct but complementary practices. Data quality focuses on measuring the overall cleanliness and reliability of data through processes and metrics. Data monitoring offers baseline insights into data by tracking specific metrics and alerting on thresholds, typically without deeper context into the causes of those issues.

In contrast, data observability extends this by continuously monitoring data health, applying machine learning models and providing real-time notifications for proactive issue resolution. While data quality and monitoring help prevent problems and flag basic issues, data observability enables teams to investigate and resolve issues across complex, multi-source environments, ensuring data integrity at scale.

Data quality and data observability are complementary. While data quality helps to prevent problems from occurring, data observability takes it further and helps to alert and detect data issues before the data is used to make critical business decisions. If an error does occur, data observability helps to identify and troubleshoot problems and reduce time to resolution. Data testing & data validation is a subset of data quality, and we will not refer to either of those terms in the eBook.

Once organizations understand the core value of data observability, and how it is different from traditional data quality approaches, they need to determine how to incorporate this into their data supply and consumption chain. There are two choices – either procure a third-party vendor tool or build an internal tool and integrate it to their data ecosystem. To make this decision effectively, it's essential to understand the data observability landscape, identify relevant selection criteria, and assess the resources required to build a data observability solution in-house. Additionally, it is important to figure out where the concerned organization is on their data maturity strategy and data quality maturity capabilities.

Data Observability Landscape

While data quality tools have been around for some time, recently there has been a rapid expansion of data observability tools in the market seeking to improve on existing data quality approaches. Some dataOps, data platform and data quality vendors have begun adding data observability functionalities. However, few vendors offer comprehensive coverage functionality that spans data flows, pipelines, infrastructure (cloud, on-prem, and hybrid), governance, enterprise integration and deployment options.

Data observability solutions generally fall into two categories: **standalone and embedded**. Embedded solutions are easier to implement but have limited functionality and integration options, and are not intended to provide an end to end visibility. For example, embedded solutions like those offered by a cloud data warehouse or a lakehouse are proprietary and work only within the confines of that one platform. Standalone tools, on the other hand offer a broader range of capabilities with better integrations and interoperability across the entire data ecosystem.

Standalone vendor offerings vary based on instrumentation architecture for data collection. At a fundamental level, all solutions deploy agents at the data sources that collect data or read logs and traces, store and analyze data, using it to train ML models to detect outliers, patterns, data quality and KPI metrics. These tools issue alerts and notifications based on rules, conditions, SLAs and SLOs. The collected data is used for data lineage, root cause analysis and recommended remediation.

1st Generation Tools: These tools, derived from APM (Application Performance Monitoring) solutions, offer basic data quality alerts and notifications for issues like accuracy and completeness.

2nd Generation Tools: Enhanced with ML algorithms, these tools provide root cause analysis and predictive capabilities, enabling them to detect potential failures before they occur. They extend beyond data quality to include schema drift detection and ML model observability.

3rd Generation Tools: These solutions provide comprehensive data observability automation, encompassing data quality, anomaly detection, and continuous monitoring across data ecosystems. Some vendors in this category provide a combination of monitoring plus lineage enabled data observability for better data incident detection, prioritization, triage, and resolution.

Observability Tools: Capabilities and Criteria

The following image (Figure 3) outlines the essential capabilities of data observability tools across several dimensions. Not all organizations need every capability offered by these tools. Enterprises should carefully consider which features are most relevant to their specific data infrastructure and operational needs, especially if they are considering building a solution in-house.

Key Considerations:

Scalability: Determine the scalability requirements of the generation you need. Modern third party data observability tools offer significant scalability to handle large, complex data ecosystems, while a 1st-generation tool may not have the same scalability.

Interoperability: Consider whether the tool will need to integrate with diverse systems and workflows. 3rd-generation data observability tools can provide deeper, more seamless integration across multiple platforms and architectures including legacy, cloud and hybrid environments.

Ease of Implementation: Assess the complexity of deployment issues. 3rd generation tools are better equipped for automation and integrate with CI/CD and dataops practices making them easier to integrate with existing data ecosystems.

Security and Compliance: Managing data security with user authentication and authorization at different levels of granularity is difficult to implement in-house. With constantly changing compliance and regulatory requirements and certifications, it can be challenging for inhouse data teams to continually comply and update the inhouse solutions. This is a critical aspect enterprises should consider carefully when implementing a data observability solution.

Maintenance Requirements: New data observability tools provide automated updates, are secure and require low maintenance, compared to previous generation tools that may be cumbersome to update, patch and secure.

DATA OBSERVABILITY TOOL SELECTION CRITERIA

OPERATIONS

- Monitor
- Deploy
- Recovering Data Pipeline Processes
- Alerting
- Notifications
- Integration with other tools / frameworks

AUTOMATION

- Automated Reports
- Automated Remediations

PERFORMANCE SCALABILITY

- Data Collection
- Data Processing

INTEGRATION

- API
- Data Sources
- BI Tools
- Data Processing Tools
- Data Catalogs
- Incident Management

DATA QUALITY

- Schema Drift Detection
- Data Corruption Detection
- Accuracy
- Completeness
- Consistency
- Conformity
- Integrity
- Timeliness
- Uniqueness

GOVERNANCE

- Regulation and Compliance Violation
- Subtopic 6
- RBAC / ABAC
- Lineage / Traceability
- Data Privacy Issues
- Data Retention Policies

DATA PIPELINE HEALTH

- Root Cause Analysis

USER EXPERIENCE

- Self Serve
- Documentation
- Customization

OTHERS

- FinOps
- Licensing

Figure 3: Data observability selection criteria

What it Takes to Build a Data Observability Tool

- **Data Collection Agents:** Instrument data sources to capture operational data, whether on-premises or in cloud-native systems.
- **Data Connectors:** Link data sources to agents for seamless data collection.
- **Data Storage:** Stores data collected by agents, used for training predictive, recommendation, and remediation algorithms.
- **Data Processing and ML Modules:** Power algorithms for outlier and anomaly detection, predictive analytics, and more.
- **Data Quality Modules:** Implement proprietary algorithms to assess metrics like completeness, freshness, and accuracy.
- **Schema Modules:** Detect schema drift and validate incoming data for consistency and accuracy.
- **Lineage Analysis:** Track data flows, traceability, and data provenance within the data supply chain.
- **Privacy Preservation:** Incorporate anonymization or tokenization for sensitive data, such as PII.
- **Enterprise Integration:** Integrate with dashboards, scorecards, BI tools, workflow tools (e.g., Slack, Microsoft Teams), and orchestration tools (e.g., Jira, GitHub). This also includes enterprise authentication, authorization, RBAC, compliance management, and data cataloging tools.
- **CLI, API, or SDK:** Offer integration points for customizing the tool's functionality.
- **Single-Pane-of-Glass Interface:** Provide an end-to-end visibility dashboard to manage data assets and processes seamlessly.

To wrap up, choosing the right data observability tool—especially if building one in-house—depends on understanding your organization's specific data goals, infrastructure, and resource availability. Whether opting for foundational data quality checks or aiming for a comprehensive, end-to-end observability solution, each generation offers distinct benefits tailored to different operational scales and complexity levels.

In the next chapter, we'll dive deeper into factors that will help determine if buying an off-the-shelf solution is the better route.

The previous chapter discussed the key capabilities and core components of data observability solutions. Now that you understand the landscape and technical underpinnings, the next critical question is whether it makes sense to use a data observability solution from a third-party vendor or develop one in-house.

To Buy or Build a Data Observability Solution

At this stage, your organization likely understands what data observability is, why it is needed and the value it can add to your data ecosystem. Clarifying the vision and goals you have for your organization's data strategy is essential to making the right decision, as it will help answer solution design questions and ensure alignment with your organization's long-term strategy. Learning more about what approaches have worked for others and learning from industry best practices can also provide helpful insights as you weigh the options.

Factors to Consider When Deciding to Build

If your organization is leaning toward building an in-house solution, here are a few key considerations to assess before embarking on this journey. Building a data observability solution is rarely a task for a small team—it requires dedicated resources like: data engineering, data governance, data architecture, a clear understanding of the scope of the data challenges you are seeking to solve, and a strong grasp of how to design and implement a data monitoring and data observability solution.

Checklist for Building a Data Observability Solution

- ✓ **Why are we considering building?** Identify the primary drivers, such as cost, autonomy, or the specific features unavailable with current vendors. Consider if the core problem is unique enough to justify building a new solution.
- ✓ **What are our specific requirements?** If the needs are straightforward—like capturing a few metrics and conducting basic outlier analysis—an in-house solution may work. But if your requirements are complex and beyond the basic tool requirements, building could be costly, time consuming and lead to scope creep.
- ✓ **Is the ROI worth it?** Assess the expected return on investment to ensure that the time, cost, and effort will result in long-term benefits. Key criteria to understand include internal personnel requirements, solution maintenance requirements, infrastructure expenditures, and overall time to value.
- ✓ **Will we have the agility and control needed?** Consider how much flexibility and independence your team requires in managing the observability solution.
- ✓ **Do we have the required expertise?** Building a robust solution internally will require a team with technical expertise, domain knowledge, and functional skills in data observability.
- ✓ **Are we prepared for maintenance?** A homegrown solution requires ongoing support, updates, and troubleshooting. Consider the resources required for these tasks.

Understanding the Reality of Building In-House

Building a custom data observability solution often appears simpler at the outset but can grow complex quickly. Unless your organization has resources similar to large tech firms like FAANG/MAMAA, the result may be closer to an MVP (Minimum Viable Product) than a full product—one that might be limited in scalability, usability, and extensibility. If your requirements are simple, building a solution may be viable; otherwise, this route can introduce significant limitations. Though larger enterprises may have the resources to build a home grown solution the bigger challenge is of time to market and time to value lost in building a reliable solution.

Most organizations pursuing an in-house build do so for end-to-end product ownership and vendor independence. However, to succeed, data teams need to answer several critical questions:

- What's the core problem we're solving? Is there an existing vendor solution that can solve our problem?
- Will our internal solution be better, faster, and more reliable than vendor solutions?
- Is there a clear plan for data collection, storage, and processing along with observability automation?
- What anomaly detection and data quality solutions are needed?
- Do we have the maturity level and skilled staff required? Building observability tools often involves developing system-level agents, which requires specialized knowledge.
- How much time will we need, and does this align with our strategic goals and budget?
- How will we ensure the quality and accuracy of a homegrown solution compared to battle-tested vendor options?
- How will we manage continual maintenance and upgrades including new features, security considerations and additional capabilities?

When Building In-House Makes Sense

Building a data observability solution in-house may make sense in situations where:

- The in-house solution offers a critical competitive differentiator.
- High agility and control are needed to integrate with legacy applications or a wide array of systems.
- The organization's strategy is to avoid vendor lock-in.
- The solution is core to the company business strategy and there is a definitive strategy to maintain it.
- There is a strong need for the organization to provide a customizable self service interface for usage by the non-technical users.

Challenges and Pitfalls of Building In-House

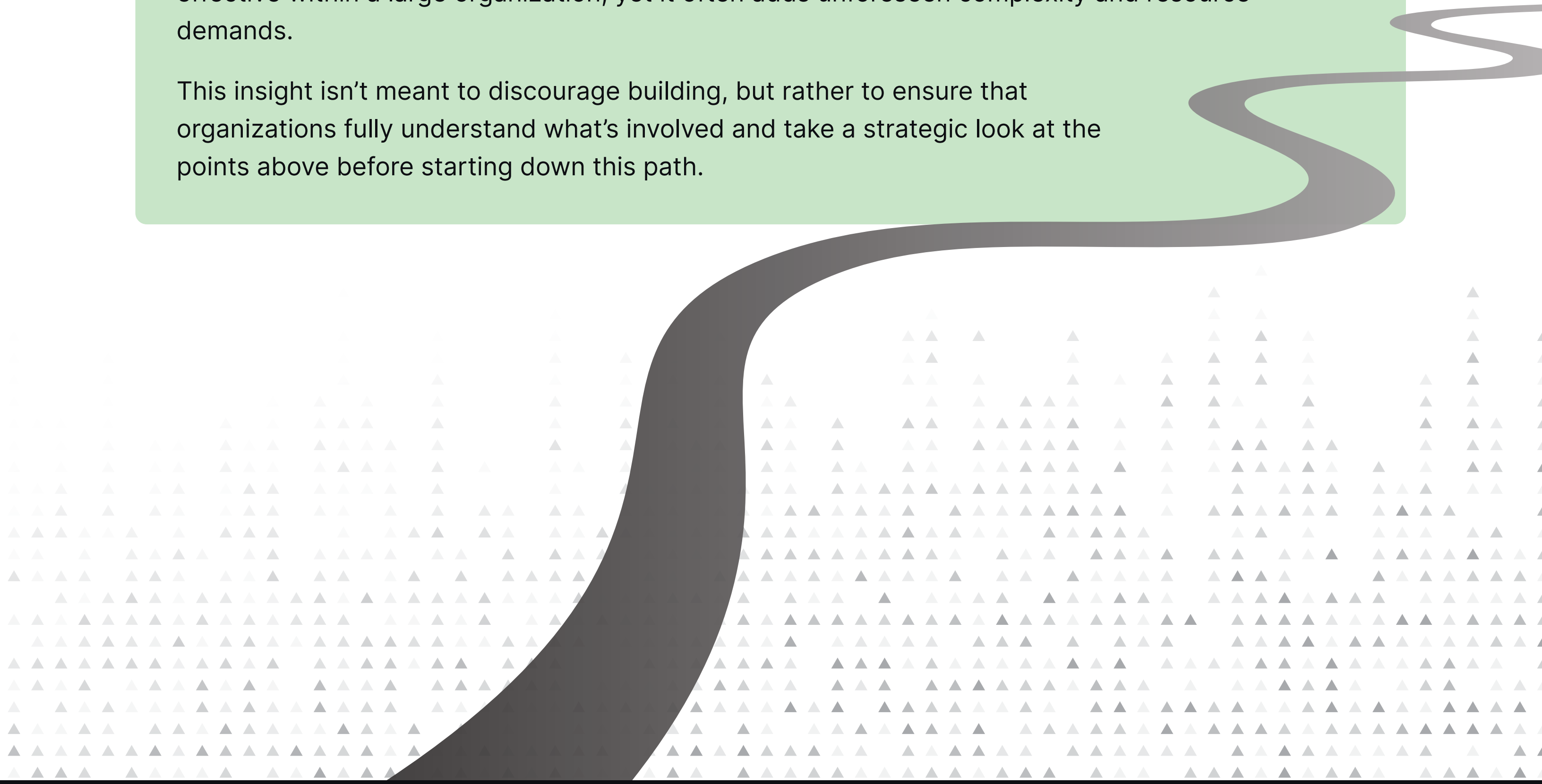
However, building an in-house solution can encounter issues such as:

- **Extended timelines and cost overruns.** Projects often start with manageable goals but quickly become late and over budget.
- **Missed opportunities and delayed ROI**. Time spent building data observability capabilities could have been used on core business applications, delaying time-to-value.
- **Limited functionality and integration.** Vendor solutions often come with out-of-the-box connectors and greater interoperability than a homegrown system can achieve.
- **Risk of subpar quality and accuracy.** Without deep expertise, homegrown solutions may miss important data signals and lack the refinement of vendor tools.
- **Difficulty meeting changing needs.** New demands and requirements from users or departments can lead to scope creep, challenging in-house teams.
- **Personnel risks.** Key developers may leave, resulting in knowledge gaps and instability in maintaining the system.

Path to Production and Maintenance

A critical, often-overlooked aspect of building in-house is the long-term maintenance required. “Day 2” scenarios involve continuous deployment, updates, and integration as new components enter the data pipeline or new use cases arise. Maintenance is essential for a solution to remain effective within a large organization, yet it often adds unforeseen complexity and resource demands.

This insight isn’t meant to discourage building, but rather to ensure that organizations fully understand what’s involved and take a strategic look at the points above before starting down this path.



Factors to Consider When Deciding to Buy

The data observability vendor market is fast-maturing and highly competitive. Some vendors have adapted to meet enterprise observability needs, while others are repurposing products to match market demand. When considering a vendor solution, here are critical factors to keep in mind:

1. Shortlist relevant vendors and create a POC plan with success and rejection criteria.
2. Understand the data collected, saved, and where it's stored for ML training.
3. Review privacy and security considerations around saved data.
4. Assess the monitoring architecture and deployment methodology. Large enterprises may have deployment restrictions.
5. Check for latency and schema drift detection capabilities. Ensure these align with your organization's requirements.
6. Confirm integration compatibility. Does the tool work seamlessly with other frameworks and tools, and provides platform extensibility with APIs and SDKs?
7. Clarify licensing costs and factors like usage-based or compute-based fees.

When Buying Makes Sense

- ✓ Buying a data observability solution may be ideal if:
- ✓ Vendors have already solved common customer challenges with mature, field-tested solutions.
- ✓ Vendors can provide industrial-strength anomaly detection.
- ✓ Packaged applications align well with your use case(s).
- ✓ Speed to value and reduced time-to-market are priorities.
- ✓ Fit with current enterprise data architecture and integration with critical data sources and tools.
- ✓ Ability to meet enterprise data security requirements.
- ✓ The need for anomaly detection and data observability is urgent.

Common Pitfalls When Buying

Buying can also have its drawbacks:

- X **Overbuying.** Acquiring a solution with too many features can lead to unnecessary complexity.
- X **Underbuying.** The tool may not meet 100% of the requirements, leaving teams to fill in gaps.
- X **Integration challenges.** Not every vendor solution will integrate seamlessly with existing tools, picking a solution that works with your existing data stack will be an important consideration.
- X **Slow adoption.** If there's extensive security or compliance review, procurement can be delayed, leading to a slower time-to-value.
- X **Siloing teams.** Departments can become more siloed as teams use different vendors or even different approaches to data management entirely.

Should You Build or Buy?

Unless an organization's requirements are highly specific or limited, building a data observability solution can quickly fall short of expectations. A basic custom solution may work initially, but as data volume and pipeline complexity grow, homegrown capabilities may not scale or adapt as needed. This isn't to discourage building; rather, it encourages organizations to thoroughly consider buy options and evaluate the economic and strategic trade-offs before making a final decision. A clear-eyed look at both build and buy options will ensure a solution that aligns with your organization's goals, resources, and long-term data strategy.

The decision to build or buy is one that organizations need to make quickly yet carefully to ensure that data issues don't prevent key stakeholders from accessing insights for critical business decisions impacting trust.

Data observability solutions can be integrated into a data platform in three primary ways:

1. Build an in-house data observability tool
2. Use an open-source data observability tool and integrate it
3. Buy an off-the-shelf data observability tool

This chapter focuses on estimating the costs involved in each of these options.

Metrics for Data Observability

Imagine you're preparing for a major presentation, only to find that an unexpected data issue has skewed the numbers. Unfortunately, this scenario is all too common, and delays in catching data issues can erode trust quickly. This is where tracking metrics like Mean Time to Discover (MTTD) and Mean Time to Resolve (MTTR) come into play.

Mean Time to Discover (MTTD): This measures how long it takes to detect a data issue from the moment it occurs. Many organizations struggle with long MTTDs, leading to data quality issues slipping by unnoticed and affecting decision-making. By establishing MTTD baselines, teams can monitor and aim to reduce the gap between when data issues happen and when they are discovered.

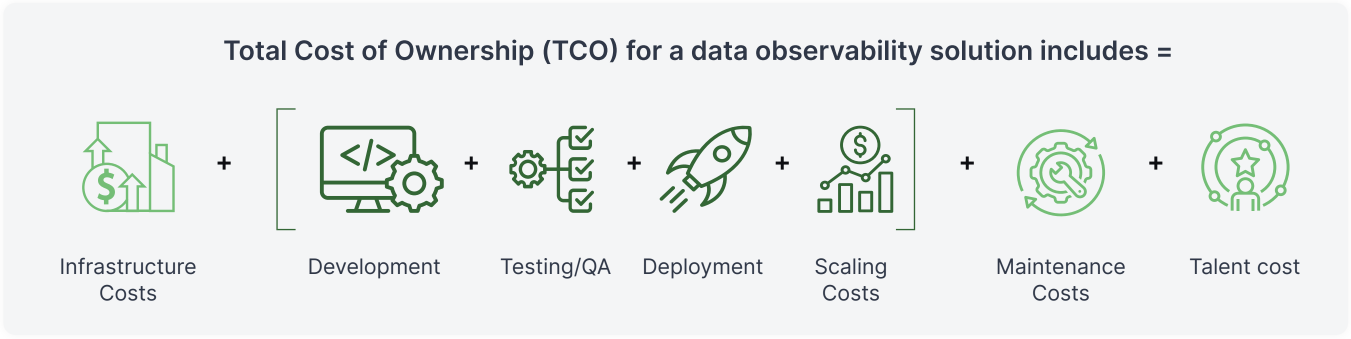
Mean Time to Resolution (MTTR): This metric tracks how quickly teams can fix data issues after they've been identified. Short MTTRs are essential for minimizing disruptions to business operations and maintaining confidence in data reliability. In large organizations with hundreds of interconnected data processes and pipelines, achieving shorter MTTRs is challenging but crucial to avoid repeated firefighting of the sale issues.

When MTTD and MTTR go unchecked, data teams can end up spending excessive time fixing issues rather than innovating, resulting in lost opportunity costs. These costs are more than just time- it impacts data teams' ability to focus on building new data products that drive value. Additionally, the 'time value of data' decays as delays in resolving issues increase. The longer an issue is unresolved, the more difficult it becomes to restore data to its original, reliable state, often requiring extensive reprocessing, backfilling and reconciliation. Decisions made on inaccurate data may also need to be revisited and revalidated, eroding confidence in data.

Reducing MTTD and MTTR is no small task, especially for organizations with large, complex data systems where hundreds of interdependent processes run simultaneously. Building and maintaining baseline metrics for incidents is key to tracking improvements over time and creating a path towards more reliable data resolution practices.

Total Cost of Ownership (TCO)

Total Cost of Ownership (TCO) includes all costs associated with setup, building, deploying, maintaining (including troubleshooting and bug fixes), scaling, and operational expenses throughout the data observability tool's life cycle. While it doesn't cover intangible benefits like opportunity cost or time to market, understanding these costs is crucial for organizations deciding to build or buy.



The goal of any data observability solution is to reduce MTTD, MTTR, and TCO. Figure 4 provides a high-level comparison of the pros and cons for each approach across several important criteria.

Figure 4 - In-house Platform vs Open Source Solution vs vendor comparative analysis

	In-house	OSS	Vendor
Time to Value	LONG - Take Months – generally delays Time to Market	SHORT - Take weeks – delay Time to Market – depending on how much of a good fit it is	SHORTEST - Started in days, Faster implementation Faster benefits
Cost	HIGH - Significant upfront costs and investment and continuous operational and maintenance cost	MEDIUM - Upfront costs in evaluation and continuous operational and maintenance cost	LOW - Lower TCO with more predictable pricing
Interoperability	DIFFICULT - Responsible for integrations can lead to bottlenecks and delays and cost overruns.	DIFFICULT - Depending on integration capabilities – can be a bottleneck for integrations especially edge cases	EASY - Mostly API / connector oriented can integrate much more easily with third party or other in-house systems
Maintenance	HIGH - Requires internal teams to continually maintain, upgrade, fix bugs and security patches and updates	MEDIUM - Will require internal teams for updates to new releases & bug fixes and security patches. Will need developers to understand code deeply	LOW - Battle tested across customers – results in higher reliability, better extensibility and scalability
Vendor Lock-in	NO	YES, to the specific OSS	YES, but can be designed with an abstraction level to minimize tight coupling
User Experience	LOW - Can result in sub-par UX	LOW - Very basic UX capabilities in most cases	HIGH - More intuitive and seamless UX
Training	HIGH - Requires time to train and create resources for internal teams, both technical and non-technical users.	HIGH - Long cycle time to install, setup and configure and effectively integrate with existing data ecosystems.	LOW to MEDIUM - Often, well established training, tutorials and guides are made available by the vendors for self-guided learning.
Customization	HIGH - In-house tools are not usually built with high customizability in mind.	MEDIUM - Can require significant effort to customize the tool beyond a certain level. Often need code modifications and changes.	LOW - Vendor solutions thrive on being customizable and configurable, as they are built with those capabilities in mind.
Risks	HIGH - There is significant risk of moving goal posts and changing market landscape. Especially related to data governance and security capabilities.	MEDIUM - Risks are related to the organizations in house capabilities to work with an external codebase which is completely new and unknown. Bug fixing and security updates can pose significant risk.	LOW - Vendor solutions are battle tested across numerous customer deployments. They have dedicated teams to solve high priority bugs and security patches.

Cost Estimates for Data Observability Solutions

Below is a framework for estimating costs associated with each approach to data observability. Fixed costs in infrastructure, data storage, and data processing (such as data cleaning and feature engineering) apply across the board.

Building In-House

Chapter 2 covers the components and steps required to build an in-house data observability solution. The table below outlines initial build costs, which vary based on the scope and capabilities an organization plans to include in their solution.

Role	Role / Responsibilities	Number	Costs	Time (Weeks)
Data Engineer	Data Ingestion Data Collection Data Processing Define, implement Use case & domain specific DQ rules Integration with data sources and data frameworks in the data platform Integration with altering and notification services	2-3+	HIGH	12-15
Data Scientist	Feature Engineering Anomaly detection algorithm development	1-2	MEDIUM	4-6
DataOps / DevOps / MLOps	CI/CD Infrastructure provisioning Infrastructure management Scalability Performance management	1	MEDIUM	2
Data QA Engineer	Testing QA rules Testing integration with other services/ data source Testing Anomaly detection Testing Scalability, Performance	1-2	MEDIUM	6-8

Maintenance costs for an in-house built solution will always be higher than the other options.

Open-Source Solutions

There aren't many OSS for data observability. A single open-source data observability tool usually does not have all the features required to enable complete visibility in the data systems. The available tools are mostly libraries and frameworks with APIs and need to be integrated into the data platform to enable observability. Some tools work with logs and metric collection, while others with events and traces. It may be the case that organizations have to leverage a few of these tools in order to get complete visibility.

The unknown factor with OSS is – how easy it is to integrate with the organization's data platform, the quality and availability of out of the box algorithms for outlier and anomaly detection and data quality rules.

The empirical estimates below assume that the OSS solution has already been selected and relevant POC has been done.

Role	Role / Responsibilities	Number	Costs	Time (Weeks)
Data Engineer	Integrating OSS into the Data Platform Tuning / Modifying / Extending Data Quality rules / algorithms	1-2	MEDIUM	2-4
Data Scientist	Tuning / Modifying Anomaly Detection Algorithm	1	MEDIUM-LOW	1-2
DataOps / DevOps / MLOps	- CI/CD - Infrastructure provisioning - Infrastructure management - Scalability - Performance management	1-2	MEDIUM	1
Data QA Engineer	- Testing QA rules - Testing integration with other services/ data source - Testing Anomaly detection - Testing Scalability, Performance	1-2	MEDIUM	2-3

Vendor Solution

The empirical estimates below assume that the vendor solution has already been selected and relevant POC has been done.

Role	Role / Responsibilities	Number	Costs	Time (Weeks)
Data Engineer	Integrating vendor solution in the Data Platform Tuning / Modifying / Extending Data Quality rules / algorithms	1-2	LOW	1-2
Data Scientist	Tuning / Modifying Anomaly Detection Algorithm	1	LOW	1
DataOps / DevOps / MLOps	- CI/CD with vendor tools and scripts - Deploying agents - Infrastructure management - Setting up permissions etc.	1	LOW	1
Data QA Engineer	Not needed			

Final Thoughts: The decision to build or buy any data stack component depends on the unique needs and limitations of an organization. Building a solution is a significant investment and understanding time to value and opportunity cost is essential before taking this path. If the value of building and associated costs are unclear, this path may not be worth pursuing.

In general, we recommend evaluating commercial solutions first, followed by open-source, and lastly in-house development. Commercial solutions often offer lower time to market and reduced risk. However, if customization and domain specific compliance are crucial, and the organization has the maturity and resources to build in-house, a custom solution may be the best option.

Make the Right Call for Your Data Observability Strategy

Deciding whether to build or buy a data observability solution is a pivotal moment for your data team. To guide your next steps, explore these essential resources:

- **2025 Ultimate Guide to Data Observability:** A comprehensive breakdown of data observability's impact, benefits, and best practices.
  Download now
- **RFI Template for Data Observability Platforms:** Ensure you're asking the right questions when evaluating observability vendors.
  Download now
- **The Data Observability Team Deployment Guide:** Practical rollout suggestions tailored to your team structure, whether embedded, centralized, or hybrid.
  Download now

Ready to take the next step towards data confidence?

Request a **personalized demo** to see how Bigeye can help you implement enterprise-grade data observability—no matter the size or complexity of your data environment.

About

Bigeye

Bigeye is enterprise-grade data observability for modern and legacy data stacks. Bigeye brings together data observability, end-to-end lineage, and scalability and security to give enterprise data teams unmatched insight into the reliability of data powering their business—no matter if it's on-prem, in the cloud, or hybrid.

RTInsights Research

RTInsights Research is the definitive voice in the evolving landscape of real-time insights and its transformative business impact of better decision making. We stand as a cornerstone for CxOs, operations leaders, and forward-thinking data and IT professionals, providing them with on-the-pulse industry trend analysis of emerging technologies that are leading towards business disruption.

The **RTInsights Research** Team has deep expertise across the real-time ecosystem, and how leading companies are implementing emerging technologies to create value. Our team has published reports and delivered webinars that inform business and technology strategy in areas such as Edge computing and IoT, Real-Time and Advanced Analytics, and the deployment of AI and Generative AI in enterprises. In addition, the RTInsights Research team has deep expertise on the supporting infrastructure that enables real-time insights, including Cloud and Hybrid Cloud Data Management, Machine Learning, DataOps and AIOps.

RTInsights Research provides business and technology leaders with actionable insights, helping to drive growth through emerging technologies.