Automation and AI Are a Must for Data Observability

Chapter 9

unravel™

Why isn't our data observability as intelligent as our data stack? Ironically, our modern data analytics systems can take petabytes of data from hundreds of disparate sources and turn it into practical insights, yet the way we monitor and manage those systems—data observability—doesn't do the same. Data observability as we know it today remains fractured and incomplete, taking us only halfway to where we need to be. It's the weak link in our data operations.

Automation and AI are needed to bring data observability out of the Dark Ages. The whole purpose of data observability is to help us quickly understand what's going on—and why—to either reactively fix a problem or proactively prevent an issue from becoming a problem. But truthfully, how quickly can we optimize data applications/pipelines based on those signals? Does data observability tell us "why"?

It might point us in the right direction—not enough memory has been allocated for a particular job, there's a problem with the code, the size of our dataset has doubled—but remediation can still be a lot of manual effort requiring time and expertise.

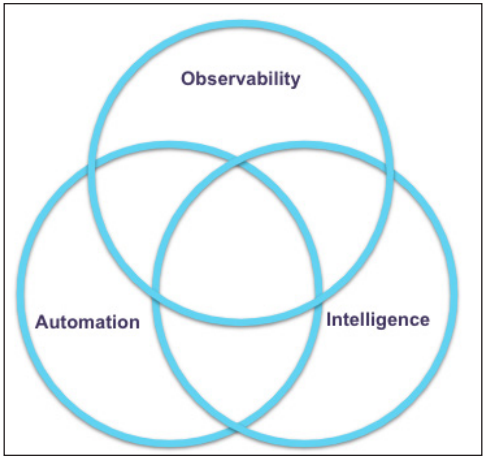

Figure 1. Data observability must be coupled with automation and intelligence (AI).

What we want to know is what we need to do next.

A more efficient approach is to automate the time-consuming manual detective work: continuous and automatic discovery of what's running in our data estate to govern costs, capturing and correlating workload profiles to choose the ideal instance type, mapping dependencies and lineage for data quality, assessing data layout and data skew for faster query performance, troubleshooting pipeline delays, and improving data processing efficiency.

AI enables us to "throw some math at the problem"—applying (hundreds of) machine learning and statistical algorithms to the observability data to identify and correlate patterns, detect anomalies, flag data issues, automate root cause analysis, and derive meaningful insights into emerging or existing problems.

Well-trained ML and highly sophisticated AI algorithms applied to a complete set of observability data can even go "*beyond observability*" to give us precise, prescriptive recommendations on what we should do to resolve the problem. For example, if there's one container in one part of one job improperly sized that is causing the entire pipeline to fail, AI can not only pinpoint the issue but tell us what the proper configuration settings would be. Or specify code changes to optimize a slow pipeline. Or optimize for cost by comparing the size, number, and resources needed to run the job vs. what is configured.

Automated AI-driven data observability cuts to the chase. Instead of relying on a handful of scarce and overwhelmed experts to tell everyone else what to do, the answers are served up to everyone automatically for an entire class of issues. This is enormously more efficient and the only way we can ever manage performance, quality, and cost at scale and speed within today's increasingly complex modern data stack.

But this approach requires a more comprehensive, unified, and holistic way to think about data observability.

Let's think differently about data observability

The term "data observability" has traditionally meant observing the data itself, but now must also include observing how the data is moving through applications/pipelines and how much it costs to do so. Briefly, data observability encompasses the dimension of quality, performance, and cost.

The term "data observability" has been around for only 4-5 years, originally coined as an umbrella term for harnessing all the information needed to understand the condition and health of data in our systems. And this is still very much a critical capability. We need visibility, context, and insights into what's going on with our data to deliver reliable results on time, every time. Nothing else really matters if the data is stale, incomplete, broken, or otherwise untrustworthy. Bad data in a good pipeline doesn't help anyone.

But neither does good data in a bad pipeline. We can have the freshest, most complete, highest-quality data in the world, but if it doesn't get where it needs to be on time—whether that's an analytics dashboard or an ML engine or another part of the application/pipeline further downstream—we've got a problem. We need to understand how the individual application/pipeline is running and how the entire data ecosystem runs end to end.

For many (if not most) problems, the issue is that there's a bad marriage between the data being processed and how it's being processed. Something somewhere changed that caused a failure or bottleneck, things aren't in sync, or the app isn't running the way it's supposed to. The dynamic nature of big data products/projects, their complex interdependencies with one another, with everything broken down into smaller parts processing in parallel, make it hard to understand what went wrong, where, and why. That's what sets off the blame game, with all the friction and finger-pointing among those who built the application and those tasked with running it (plus the frustration of people using the application's output).

These twin requirements—to deliver reliable results on time, every time—manifest themselves as SLAs revolving around performance and quality. That has been the data teams' mandate since day one.

But now there's a new kid in town: cost. As organizations increasingly move more data workloads to the cloud and their (uncontrolled) cloud data spending skyrockets and blows through budgets, the cost has become a co-equal consideration. It has become a first-class SLA, right alongside performance and quality. Good data in a good pipeline is no longer enough. Now we demand good data in a good pipeline at the most cost-effective price.

What are the 4 foundational areas of data observability?

Data observability cuts across four but interrelated contextual dimensions of data analytics. Historically, these different aspects have been treated as discrete domains, with different team members myopically focused on just one dimension or another.

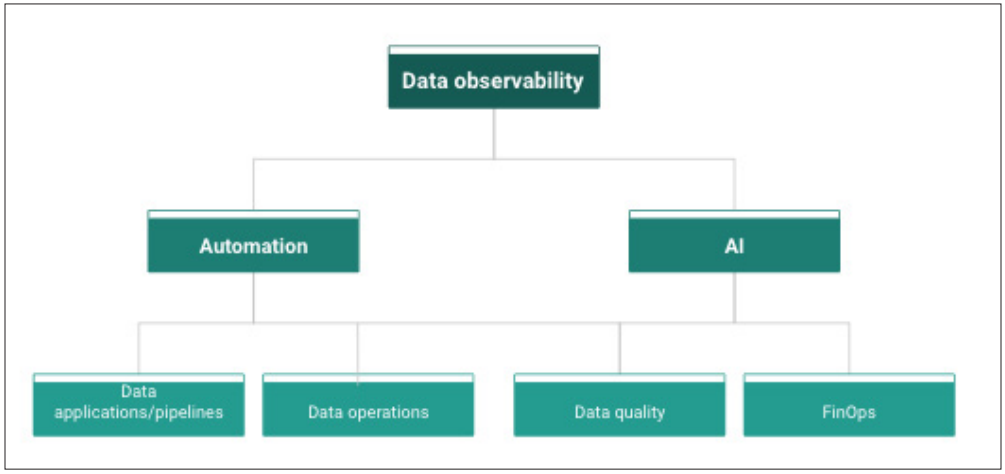

Figure 2. Data observability encompasses monitoring and managing data applications/pipelines, platforms, data quality, and FinOps.

- Data application/pipeline observability stitches together into a data-specific context millions of granular application and job-specific details that are "hidden in plain sight" but spread across disparate sources. This would be logs, metrics, traces, events, and other metadata from, say, Spark and Kafka and all the other dozens of components comprising the modern data stack, from the application down to infrastructure and everything between.

  This is basically the same observability that APM tools (Datadaog, Dynatrace, AppDynamics, New Relic) provide to software teams for web applications, but for data observability needs to be designed for the unique needs of data applications/pipelines.

- Operations observability enables data teams to understand how the entire platform, or "system of systems," is running end to end instead of pinpointing problems with one specific application. It provides a unified, holistic view of how everything is working together—both horizontally and vertically—within the Databricks, Snowflake, BigQuery, Amazon EMR, Dataproc, etc., ecosystems.

- Data quality observability looks at the health of the data making its way through the pipeline by assessing the integrity, accuracy, completeness, reliability, freshness, and consistency of data by capturing and correlating information related to data layout, schema, lineage, etc. It's observing the datasets themselves, running quality checks to ensure correct results, identifying where the data is coming from and where it's going, showing where it's used, who can access it, how it's being stored, etc.

- FinOps observability within the context of data applications/pipelines is all about using observability insights to enable engineering, business, and finance teams to work collaboratively to make data-driven decisions about their collective cloud spend. This covers everything from understanding with precision where the money is going and tracking budgets to identifying waste and inefficiencies to optimizing cloud operations for optimal ROI.

How data observability solves business problems

When considering why data observability is important—the value it delivers to the business—it's helpful to look at how observability has enabled software teams to deliver value to the business through the mature realization of DevOps.

Data teams today face many of the same generic problems that software teams faced 10+ years ago. The ever-increasing volume, velocity, and variety of work is overwhelming everyone on all fronts. There's too much work to be done and not enough hours in the day or enough expertise to go around. Troubleshooting a slow Spark or Kafka application or untangling the wires to track the lineage and dependencies of a data quality problem requires some advanced technical know-how. And there's no cavalry coming over the hill to help. The demand for skilled data professionals has always exceeded supply, and the greater complexity of the modern data stack is only exacerbating the chronic understaffing, talent shortage, and skills gaps. Data teams are spending too much time firefighting "by hand." A Gartner survey found that data teams spend only 22% of their time on innovation that delivers value. The rest is taken up by the monotonous toil of responding to trouble tickets, tracking down problems, and resolving them.

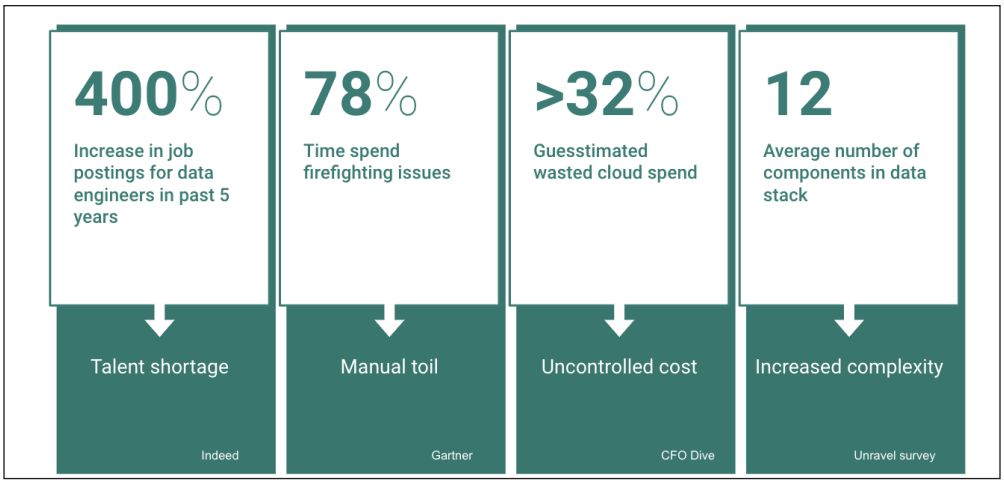

Figure 3. Business problems addressed by data observability

The combination of more work and not enough people creates a vicious cycle of complex issues landing in the lap of an already overburdened workforce without the time or expertise to deal with the problem quickly. We wind up with precious (and expensive) experts doing time-consuming break-fix work—the exact opposite of delivering strategic value. There's too much working in silos.

Data teams comprise multiple roles, and different team members use different tools for different tasks. Communication and collaboration among the team break down, and we're left operating in a Tower of Babel. Operational fragmentation and silos lead to friction and finger-pointing. Everything takes too long; trouble tickets pile up, SLAs get missed, productivity slows to a crawl.

And as more data workloads move to the cloud, the price tag gets higher and higher—seemingly with no end in sight. Modern platforms like Databricks, Snowflake, BigQuery, and Amazon EMR have greatly improved agility and speed-to-market, but it's become like the Wild West, where everybody is spinning up instances left and right but nobody has clear visibility into how efficiently they are running or control over their cost.

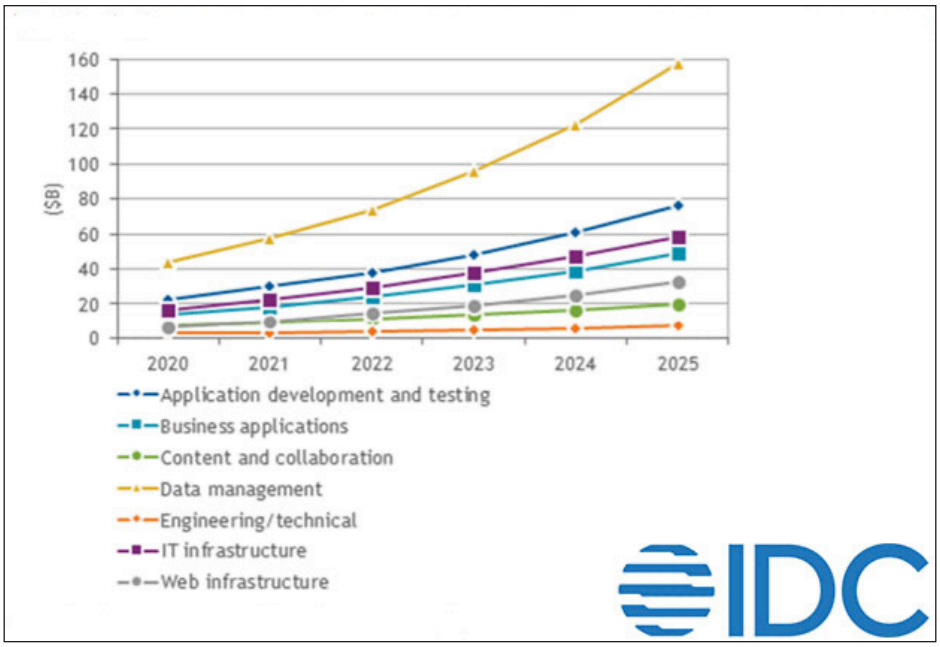

Figure 4. Data workloads are the fastest-growing cloud cost category and, if they're not already, will soon be an organization's #1 cloud expense.

The bottom line is that organizations aren't getting enough "bang for their buck" from their big data investments. The ROI from their people, processes, and technologies isn't where it should be. The goal of delivering reliable results on time, every time, in the most cost-effective manner falls woefully short.

Software teams have been able to overcome these challenges with their observability and application performance monitoring (APM) tools. Providing everyone—developers, operations, the business side—with holistic visibility into what's going on and analysis of why things happened fuels two foundational capabilities that are the bedrock of accelerating the DevOps life cycle. First, it enables a shift-left approach where individuals are empowered to self-service issues before they become problems. Second, for those issues that impact or need to involve other team members, it promotes a more collaborative methodology where everyone is looking at the same set of "facts" to tackle the task at hand.

As observability is the foundation upon which DevOps is built, data observability is the bedrock of DataOps.

The unique challenges for data observability

But while there may be generic similarities between DevOps observability and DataOps observability, data apps are a different animal from web apps. A common pitfall many organizations make is trying to "borrow" APM from their software teams to use for data observability. It seems that Datadog, Dynatrace, New Relic, AppDynamics, etc., do the same thing—but there are several reasons APM tools don't work for data observability.

First, data applications/pipelines are built on a different set of technologies, platforms, frameworks, and systems than web applications. Think Spark, Kafka, Airflow, dbt, Databricks, Amazon EMR, Dataproc, BigQuery, Snowflake.

Second, the nature of the work is different. Web applications are consistent products, and DevOps aims to accelerate the delivery of an improved version of that relatively consistent product. Data applications and pipelines start with fluid and ever-changing data and data sources, and the goal of DataOps is to accelerate the delivery of fluid and ever-changing data analytics requirements.

And there is the dimension of the data itself. In web apps, data is a static piece of information that traverses the service call chain, usually via a database API call. APM gives DevOps teams the information they care about—success/failure of the API call and how long it took—but that's about it. Data teams need to understand not only how the data flows horizontally from component to component across pipelines and vertically between data stack layers, but also the data's condition (or quality) as it flows. APM tools simply were never designed to comprehend this duality.

The different requirements for observing data applications/pipelines vs. observing software applications can be boiled down to three things:

- We need to capture very different types of telemetry data on a more granular level
- We need to visualize all this information in a context that makes sense to different data team members for their particular task at hand
- We need to do a completely different analysis to understand and fix issues

What data observability should look like

The dimensions of performance, quality, and cost are all interrelated, so it's important to view them holistically rather than as separate issues.

But make no mistake: moving to a shift-left and collaborative approach to holistically managing the performance, quality, and cost of data applications/pipelines is a seismic sea-change for data teams. We have to make it easy for every team member—and the easier it is, the greater our chances of success.

What is vitally important is that the information provided by data observability be consistent, accessible, and understandable across all data team functions: data scientists and analysts, data engineers, operations teams, data architects, data stewards, business stakeholders, C-suite executives, finance teams, etc.

What must be avoided is hopping between tools, jumping from screen to screen, manually cobbling together information from various discrete sources, with different tools giving different answers to the same problem. All the tools must integrate and play well together. Everybody needs to work from the same data while looking at things through their own lens and understand how everything fits together—a single pane of glass for a single source of truth.

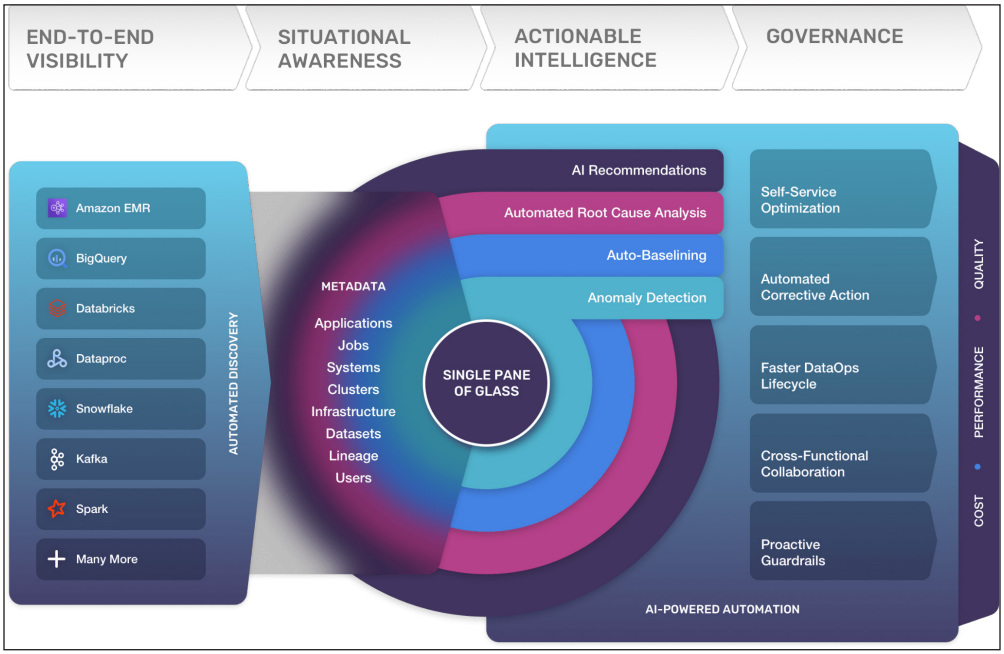

Figure 5. Automated AI-driven data observability

Automation and AI make this happen. First by providing end-to-end visibility and situational awareness, then by delivering actionable intelligence and enabling automated governance.

End-to-end visibility automatically stitches together telemetry data and metadata from the multitude of components, assets, technologies, and processes across the full data stack—both horizontally and vertically—to give a unified, in-depth understanding of the behavior, performance, cost, and health of data and data workflows.

This entails capturing information from engines, schedulers, services, and cloud providers about all data applications (pipelines, warehousing, ETL/ELT, machine learning models) and datasets (size and layout of tables, lineage, users, business units, computing resources, infrastructure), including code, configuration details, and metrics for each component in the stack.

Situational awareness puts all this aggregated information into a meaningful context for the task at hand. It's not enough simply to have a ton of granular details in dashboards, charts, and graphs, they need to be correlated, visualized, and analyzed so we can make sense of it all immediately. AI connects the dots among all the deep details about jobs, pipelines, clusters, data, infrastructure, and users into a single unified "mission control" view that shows at a glance all interrelationships (parallelism, resource contention, dependencies, lineage) and provides one-click drill-down into configuration, code, containers, resource usage, cost, data tables, quality checks, errors and logs, Gantt charts and DAGs, and more.

If we have all the details captured and correlated from everything we have running, we essentially have built a data model onto which AI/ML can be applied. But to get actionable intelligence with a high degree of confidence—i.e., to get answers, not just clues—we must have complete information about our data estate, with no blind spots. Otherwise, we're back to tackling performance, quality, and cost myopically as unrelated issues. And the observability solution needs algorithms and ML with a high degree of sophistication that have learned what the data applications are trying to do, how they behave, etc.

This actionable intelligence allows us to go from reactive to proactive—and get a level of automated governance via intelligent alerts or autonomous remediation action. Governance just converts AI-powered recommendations and insights into impact. The best way to reduce firefighting is to avoid having a fire to put out in the first place. Have the system apply the recommendations automatically. No human intervention needed. Policy-based governance rules could be as benign as sending an alert to a user (or send it up the chain of command) if some sort of data layout, performance, or cost threshold is violated—giving a heads-up that an SLA will get missed or a budget is in jeopardy of being exceeded—or as aggressive as automatically requesting a configuration change for a container with more memory or triggering preemptive corrective "circuit breaker" actions like shutting down a runaway job or rogue user.

Conclusion

Data observability is all about automation and AI. Using automation to capture and correlate millions of data points scattered across the highly complex conglomeration of technologies, platforms, frameworks, and cloud vendors provides a clear understanding of what is happening in our data estate.

Then using AI to figure out why something happened—and tell us what to do about it. And this needs to happen across the dimensions of performance, data quality, and cost. With the volume, velocity, and complexity of today's modern data stack, these aspects are increasingly interrelated—organizations can now address all three using automation and AI to realize the full potential of their data operations.