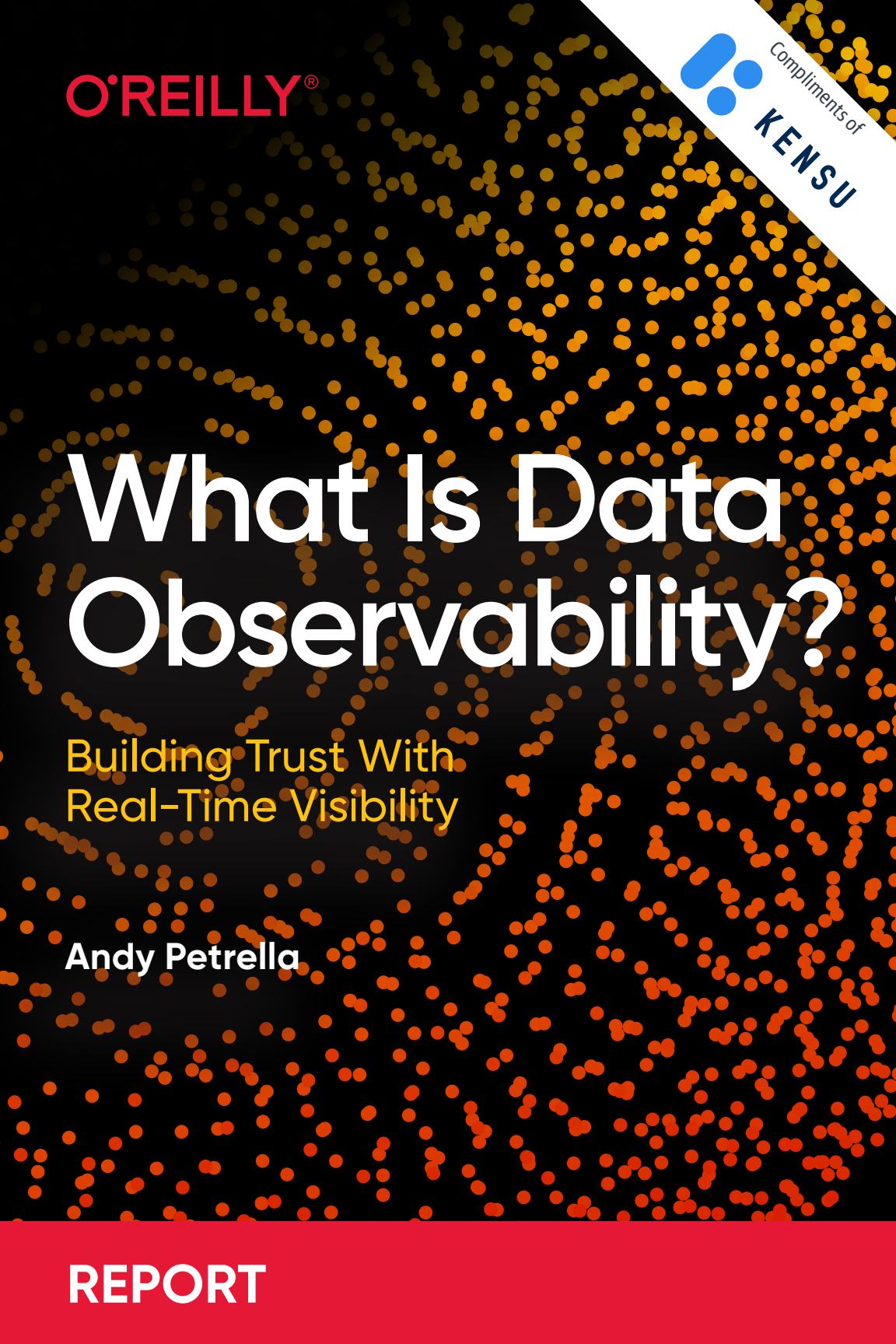
O'REILLY®
Compliments of
KENSU
What Is Data Observability?
Building Trust With Real-Time Visibility
Andy Petrella
REPORT

KENSU

Trust What You Deliver

Our low latency data observability solution alerts about data issues, prevents propagation, and highlights which applications are impacted.

Discover how our “Data Observability Driven Development” method is a paradigm shift for data teams that want to scale up their activities without adding complexity to their daily operations.

Learn More

www.kensu.io

What Is Data Observability?

Building Trust with Real-Time Visibility

Andy Petrella

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

What Is Data Observability?
by Andy Petrella

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: Jessica Haberman
Development Editor: Jeff Bleiel
Production Editor: Kate Galloway
Copyeditor: nSight, Inc.
Proofreader: Elizabeth Faerm
Interior Designer: David Futato
Cover Designer: Randy Comer
Illustrator: Kate Dullea

February 2022: First Edition

Revision History for the First Edition
2022-01-26: First Release

978-1-098-12096-2

[LSI]

Table of Contents

CHAPTER 1

It's Time to Rethink Data Management

In 2016, Microsoft deployed an AI chatbot named Tay on Twitter. Tay had to be shut down only 16 hours after it launched because a number of its nearly 100,000 tweets mimicked offensive and controversial language used by Twitter users that the bot was supposed to be learning from.

Peter Lee, corporate vice president of Microsoft Healthcare, posted the following apology: "We are deeply sorry for the unintended offensive and hurtful tweets from Tay, which do not represent who we are or what we stand for, nor how we designed Tay."[1]

This is an example of how badly a brand, even a leader in its industry, can be hit when data-based decisions go "freestyle." It also shows how a simple data issue, such as missing or incomplete data, can have significant repercussions.

With the tremendous growth in the volume of data that organizations are collecting, and the scale of its usage, data issues are occurring more often. By 2025, it's estimated that the global volume of data will expand to 180 zettabytes, more than double the volume of data in 2020. Fueling this dramatic growth is the fact that almost every function of every organization now generates data. It's also

1 Peter Lee, "Learning from Tay's Introduction," March 25, 2016, Microsoft Corporation, blog post, *https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction*.

being driven by the advancement of open source machine learning, which relies on large datasets for training models.

To manage this exploding data growth, data teams have risen in importance and size. Small data teams with few stakeholders have been replaced by large teams that must answer to executive pressure. Yet, despite the increased value and presence data has within organizations, little thought has been given as to how to monitor the quality of the data itself.

While IT and DevOps have numerous quality control measures (such as development practices like continuous integration and testing) to protect against application downtime, most organizations don't have similar measures in place to protect against data issues. But just as organizations depend on high levels of reliability from their applications, they also depend on the reliability of their data.

When data issues—such as partial, erroneous, missing, or inaccurate data—occur, the impact can rapidly multiply and escalate in complex data systems. These data incidents can have significant consequences and multiple negative repercussions on the business, including lack of trust and loss of revenue. A lack of control or visibility into the data quality can also produce faulty insights, which can lead the organization to make poor decisions that can result in lost revenue or a poor customer experience.

Resolving data incidents can be time-consuming and difficult. Identifying the root cause of this type of failure requires knowing where the data flowed from and, therefore, what applications were participating in creating value from that data. But early identification of data issues, which are unknown to the data owners in most cases, is inherently challenging in complex data systems. IT and data teams must become archeologists, trying to discover what's gone wrong. Yet each team is siloed and doesn't have complete visibility. The further downstream the issue, the harder it is to figure out what happened and the more information gets lost in translation. Patches may fix the issue, but if the root cause isn't identified, there's little confidence that the issue won't reoccur. This creates a lack of trust in the data and in the team responsible for shepherding that data to the data user. This, in turn, mutes the data's value.

The lesson? No matter how vast an organization's data assets are or how advanced the technologies are that support the management and analysis of data, they cannot be useful without reliable data

quality. Organizations must pay more attention to data collection, storage, and preparation practices before data analysis takes place. Indeed, the unmanageable size and quantity of data create significant challenges, but you can confront and tackle these issues if sound data management practices are in place.

The practices that will create visibility and add the monitoring capabilities to deal with such challenges are what composes data observability. We'll cover it in more detail in Chapter 2, but before that, let's review together the impact of the status quo in the long run.

Why the Lack of Confidence Can Jeopardize the Future of AI

AI is one of the fastest-growing and most popular data-driven technologies in use. Nine in ten businesses currently have ongoing investments in AI.[2] Moreover, 86% of companies say that AI will be a "mainstream technology" at their company by the end of this year.[3]

However, a contradictory "2020 Big Data and AI Executive Survey" also showed that there has been a 39.7% decline in organizations that say they are accelerating their AI investments.[4] It's beginning to look more likely that businesses will start to pull back on investments in AI if better returns aren't seen soon.

Because AI depends upon datasets to feed its models, the reliability of the data can directly impact the success of AI models and, more broadly, the innovation and progression of AI. The issue isn't that these AI models need *more* data to be successful but that they need *high-quality data*. Even then, a high-quality data model can fail simply because the world has changed too much for the model, which based its learnings on a previous state of the world and therefore can no longer work efficiently under the new conditions.

2 "NewVantage Partners Releases 2020 Big Data and AI Executive Survey," Business Wire, January 6, 2020, *https://www.businesswire.com/news/home/20200106005280/en/NewVantage-Partners-Releases-2020-Big-Data-and-AI-Executive-Survey*.

3 "AI Predictions 2021," PwC, October 2020, *https://www.pwc.com/us/en/tech-effect/ai-analytics/ai-predictions.html*.

4 "NewVantage Partners Releases 2020 Big Data and AI Executive Survey."

For instance, if a retailer's ecommerce store's target audience suddenly switches from teens to pregnant women because it started carrying several lines of maternity clothes, the AI model might not be capable of predicting the right recommendations for site visitors anymore because it is essentially a new population. So while the data itself might be correct, the real-world circumstances have changed.

Still, data quality is always a precondition for AI. Garbage in will produce garbage out if we don't pay attention to how the outputs are created by the application. A better approach is for the application to detect bad-quality data (garbage in) when it's being inputted into the application to avoid producing a poor-quality data output (garbage out).

According to Roger Magoulas and Steve Swoyer's "The State of Data Quality in 2020 Survey," over 60% of enterprises see their AI and machine learning projects fail due to too many data sources and inconsistent data.[5] Even small-scale errors in training data can lead to large-scale errors in the output. Incomplete, inconsistent, or missing data can drastically reduce prediction accuracy.

Think of the impact of prediction degradation on something like self-driving cars. A poor prediction, such as not accurately predicting the car's proximity to a human crossing at a crosswalk, could lead to a fatal accident. Something similar has already happened. A Tesla Model S, being operated in full self-driving (FSD) mode, missed a curve in the road, causing it to hit a tree and kill two people. Uber's testing of self-driving technology has also resulted in some grim outcomes. Most recently, an Uber self-driving car killed a pedestrian crossing the road. These types of prediction-failure incidents have created considerable skepticism that self-driving cars will ever be safe. In the case of Uber, it has led to the company halting the testing of the technology in Arizona, where the accident occurred.

If organizations don't implement better quality control mechanisms—and at a scale that can keep pace with the ingestion of massive volumes of data—the risk of failures within AI-driven applications will become too great for organizations to bear. CEOs will begin to say that while the promise of AI was great, it hasn't

5 Roger Magoulas and Steve Swoyer, "The State of Data Quality in 2020," O'Reilly Media, February 12, 2020.

lived up to expectations. They'll begin to believe there's too much at stake in deploying new products or services that rely on AI because they won't feel they can control the quality of the decisions. Companies can avoid the occurrence of this reaction by filling the data quality control gap with data observability.

For example, big data is eventually seeing its hype cycle era coming to an end (see Figure 1-1). Nearly three-quarters of organizations cite big data as an ongoing challenge, and only 37.8% say they've been able to create a data-driven organization. One positive aspect of this shift away from big data is that there's now less focus on collecting large volumes of data and more emphasis on the challenges of processing, analyzing, and interacting with massive amounts of data.

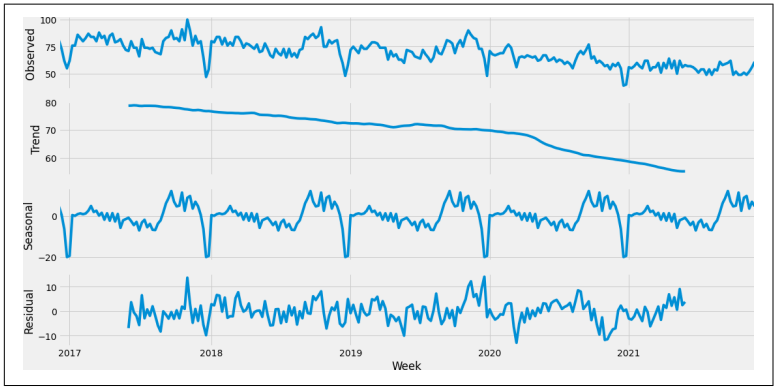

Figure 1-1. Timeseries analysis performed on Google Trends for searches of "big data" worldwide

We are at a pivotal point in the AI hype cycle. If AI is to avoid the fate of other AI hype cycles, its success must continue to impress. By impress, I mean that AI must continue to be simultaneously creative (new and innovative) and performant (efficient and trustable). For both conditions to occur, as I'll discuss in more detail later on, quality data assurance is a must.

Inadequate Data Management Is Inhibiting Data (Application) Innovation

Data systems are already complex. The data being used to power AI models and ML technologies is beyond the scope of human capabilities to process. This fact raises ethical and practical concerns when building new data applications.

While there are humorous instances of AI failure, such as an AI-powered camera mistakenly tracking a soccer player's bald head instead of the ball, the reality is there are many situations where AI could cause actual harm. There have been instances of AI applications that violate privacy laws, show inherent gender and racial discrimination, and promote disinformation. But AI's potential for malicious use doesn't stop there.

Some of the world's leading technology companies are already running up against this dilemma. An artificial intelligence engine created by Facebook, for example, was shuttered because the AI had made its own unique language that humans couldn't understand.

As these use cases highlight, if you can't control the quality of both the inputs and outputs, the application is also at risk of being uncontrollable. If this becomes the case (or even if this becomes the prevailing sentiment about AI-powered applications), many organizations and developers will feel it's necessary to limit themselves to less complex AI applications to reduce risk. However, this will also restrict their ability to be creative and innovative with the technology.

Another way data application innovation could become limited in the future is due to regulatory concerns. Data privacy regulations like General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA) must be followed. But at the same time, regulatory concerns can dampen experimentation and innovation. For instance, if a developer wants to use consumer data in an application but doesn't know exactly how they want to use it, access to that data may be restricted. The data owner won't want to share the data because they bear ultimate responsibility for that data. If the data is used in a way that doesn't comply with data regulations, they'll be held responsible.

We can already see examples of this type of situation occurring. For instance, when Facebook shared personal user data with Cambridge Analytics, it endured significant blame and backlash—perhaps even more so than Cambridge Analytics—for how Cambridge Analytics misused that data. Yet Facebook did not have prior knowledge that this was how the data would be used. Having seen what happened to Facebook, other data owners have become much more concerned about the implications of sharing their data with third parties. These concerns threaten to limit developers' ability to experiment with and innovate new data applications.

One workaround for data owners is to sample and anonymize the provided data to alleviate risk. But an anonymized sample with some of the data stripped out doesn't replicate the real world, leaving the analyst with little confidence about how that application will perform in production.

An analogy here is a tightrope walker. When the tightrope walker is first attempting to tightrope walk, they need the security of a net beneath them because the likelihood of falling is high. Data owners and developers also need the security of knowing that they have a safety net to help them identify data quality issues and remain in compliance with data regulations. Otherwise, they will limit the complexity of what they build in order to limit their own risk.

This fear of losing control of the data quality as complexity increases doesn't just limit data application innovation; it also limits AI and ML creativity and innovation because these are also data-driven applications.

Because there are significant negative repercussions possible from AI, including revenue loss, reputational damage, regulatory penalties, and a lack of public trust, organizations must proceed with caution. So how do you manage the complexity of data-driven AI systems? How do you ensure the quality of the data so you can mitigate risk? How do you build trust in that data so you can have confidence in the outputs of the AI algorithms?

To answer those questions, the fundamental challenge is how developers can get timely insight into a system's state, especially when complex cloud-native ecosystems built upon massive data volumes make it much more difficult to predict a system's behavior. To address this challenge, there's been a growing movement toward implementing observability to gain deeper contextual insight into

the correctness and performance of these complex data systems. With regards specifically to data management, data observability has emerged as a tool that can provide the visibility and safety net necessary to spur greater data-driven innovation.

In the next chapter, we'll dive deeper into what data observability means and how it can be applied to complex data ecosystems.

CHAPTER 2

What Is Data Observability—and Why Do We Need It?

In your organization, has someone ever looked at a report and said the numbers were wrong? Likely this has happened more than once. No matter how advanced your data analytics and modeling tools are, if the data you're ingesting, transforming, and flowing through your pipelines isn't correct, the results won't be reliable.

Even one new field added to a table by one team may cause another team that relies on that same data, but for a different use case, to have inconsistent data. If this issue isn't quickly identified, the downstream impact could result in compliance risks, poor decision making, or lost revenue. If it happens too often, it will also result in a severe loss of confidence by business users.

However, discovering these kinds of data failures is much more challenging than typical application or system failures. In the case of an application, when something isn't working, the symptoms are more evident. For instance, if an application crashes, freezes, or restarts without warning, you know you've got a problem. However, data issues are generally hard to notice since the data won't freeze, crash, restart, or send any other signal that there's a problem.

At the usage level, data also can't just be "fixed." You can only correct it by requesting the producer to publish the fixed data or by looking at the application(s) producing the data to determine what's gone wrong. For instance, if a column or row is missing when the data is ingested, you can't just fix the data. In this specific case, the data

isn't there at all, so fixing the data requires fixing the application upstream to ensure it ingests the entire dataset. In another case, the data may simply no longer be available and, therefore, the producer of that data needs to be made aware so that they can adapt their analysis to this new reality. Further, when you're performing an analysis of the data, certain assumptions are made that may no longer be true at a later point in time. Here's an example of what I mean.

When an analyst initially builds a model to analyze a retail organization's previous year's sales data, they can see that the organization collected data every month for the prior year. Thus, the model is trained based on a period of 12 months of data from the last year. However, while the data was complete when the data model was initially built, that may no longer be the case. Perhaps, when the analysis is deployed six months later, the assumption of having the 12 months of data (with the seasonality of four quarters) is no longer true because the system can no longer handle this amount of data. As a result, the previous year's data is removed from the analysis. Since Q4 tends to be some of the highest-grossing months for a retailer, this change can have a major impact on whether the insights generated from the data are accurate, especially when the assumption is that this data is present. However, because the analyst hasn't shared their assumptions with the producer and the producer doesn't have that level of visibility into the data models, they are not aware of this "implicit" constraint coming from the data usage. Hence, they also don't know that the output from the model (their analysis) is inaccurate.

This is where data observability comes into play. You need to be able to observe and be aware of these types of silent changes to the data so that you can fix them preemptively—before your CEO is coming to tell you that numbers look wrong.

Data observability is a solution that provides an organization with the ability to measure the health and usage of their data within their system, as well as health indicators of the overall system. By using automated logging and tracing information that allows an observer to interpret the health of datasets and pipelines, data observability enables data engineers and data teams to identify and be alerted about data quality issues across the overall system. This makes it much faster and easier to resolve data-induced issues at their root rather than just patching issues as they arise.

Critically, what makes a data observability solution unique from application observability is that the data must be logged and traced from within the data pipelines, where the data is created and activated for use. This allows you to measure the pertinence of the usage of your data within your entire system (all your applications and pipelines) as well as monitor health indicators of the overall system.

Why is this important? Because while your data pipelines may look fine—the data isn't using too much memory or taking up too much storage space—if the data outputs from those pipelines are providing garbage data, then the value of the data is worthless. On the other hand, if you only observe a table within your database, it may tell you what queries were performed, but you won't know how it's being used, or by whom. This matters because you won't know if the data being used is of value to the end user.

Thus, to ensure that the data analytics you're performing are accurate and valuable, you need to observe both the data and the pipelines at the same time. If you're unable to have this type of end-to-end visibility, you're likely to suffer what I call a *data incident* or a *data failure*. In the next section, we'll dig deeper into what this means and why they should be avoided at all costs.

Characteristics of a Data Incident

Data incidents result from a missing gap in data management that leads to catastrophic impacts on businesses. In an increasingly complex system of data pipelines, applications, and numerous inputs of large volumes of data, there has been a proliferation of such incidents recently. So what causes them?

The cause could be inaccurate, incomplete, or inconsistent data. Or it may be that data is not being refreshed at the right timing interval, so that when the business user needs to pull insights from the data, the data hasn't been updated yet. Or the data might be refreshed on time, but it's too old to be valid (i.e., the data may have just been refreshed, but if it's pulling the previous month's sales numbers and not the current month's numbers, it's incorrect).

Assumptions that were made earlier in a data experiment may not be true anymore. Perhaps the age parameters have changed in the data, so the previous assumption that all data will be from people aged 65 and older is now incorrect. Or the data could also be

out of compliance and therefore unusable. For instance, you can't use personally identifiable information (PII) from a health-care provider database to predict the next election results. This data may be accurate, but it's illegal to use health PII data in this manner.

Data issues can also occur when an application has been altered because it has received a request from a business user to change the way data is transformed. While this change may be helpful for one business user, the change may impact other users who have a different need for perceiving and using the data. So even if the data is published in the same format and at the same time, if some of the data structure has changed, the data may no longer be relevant for certain use cases. What's more, the impacted business users (those who didn't request any change to the data) also won't have the visibility to know that the structure of the data has changed. As a result, they may be producing incorrect analyses from the data.

In a large organization, even one minor change somewhere in the dataset or data pipeline can create a snowball effect, where the change upstream creates a bigger and bigger problem as it moves downstream, making the impact much greater. Inaccurate data can either be injected into the system at the time of collection from a third party, such as a customer who submits an incomplete form, or during the transformation process. For instance, a data engineer may have been asked either to remove incomplete fields (those with too many nulls) or to fill them with approximations. However, by making this change to the business rule, the data is now corrupted for another user with a different use case. Thus, the application produces incorrect insights, which leads to poor decision making from the business because the data injected was inaccurate.

As you can see, there are multiple steps in the data process, and at any step, the data can be corrupted. By corrupted, I mean merely that the quality of the data is not adequate for how it's going to be used. Data itself cannot be wrong or inaccurate, but its usage becomes inadequate for a particular use case. Thus, the earlier you are able to intercept a data adequacy issue, the less damage there will be and the higher the loss prevention.

When considering the explosive growth of AI and the complexity in these models and applications, incidents can be nearly impossible to resolve at the root cause without visibility into the data and data pipeline. But if issues aren't resolved adequately and continue to

reoccur, it will ultimately lead to a lack of confidence in the ability to create complex data-driven applications.

Knowing that data observability is necessary to avoid a drawdown in confidence in using large datasets and AI-driven applications, I'll now discuss how to achieve data observability.

Achieving Data Observability

We've discussed the necessity for data observability and the need to observe the data along its usage within the systems. But, in practice, how does data observability work?

Similar to DevOps observability, which relies on automated monitoring to identify leading indicators of service degradation or unauthorized activity, data observability also relies on automated logging and tracing of the data and data pipelines to evaluate data quality and identify issues.

Assessing Data Quality

There are many indicators of data quality[1] to consider. Here are some of the most usual ones:

Accuracy
: The data values should be reliably representative of the information (e.g., phone numbers should be real phone numbers).

Completeness
: The data should contain all the necessary and expected information (e.g., a complete address would contain the street address, city, state, and zip code).

Consistency
: Data should be recorded in the same manner across all systems (e.g., phone numbers should all be collected either with or without the country code, rather than a mix of some with country codes and some without).

1 "Data Quality," Wikipedia, last updated November 16, 2021, *https://en.wikipedia.org/wiki/Data_quality*.

Conformity
: Data should be collected in the same format (e.g., phone numbers should only have numbers, not letters, and email addresses should always have an @ sign and end in *.com*).

Integrity
: The data must be connected to other data across all relationships and not be considered as an orphan (e.g., if you have a phone number in your database, but it's not associated with a person, the data is invalid).

Timeliness
: Data must be up to date, and it must be ingested at the correct intervals (e.g., if you are analyzing weekly point-of-sale [POS] data, the data should be refreshed weekly).

Unique
: Data needs to be cleaned of duplicate entries (e.g., if you have two entries, one with the name Jane Smith and another with Jane G. Smith, and both correlate to the same individual, one record should be deleted).

Observing Data Quality Within Your Data Systems

Now that we've defined what constitutes data quality, the next step is to ensure that these quality indicators are all present within your datasets and systems. Using principles similar to software application observability and reliability, and applying it to data quality, these issues can be identified, resolved, and proactively prevented.

To generate this type of end-to-end data observability, it's necessary to log and trace the following information within your data pipelines and datasets:

Application
: You need to log the application that is using or creating the data. This activity is often wrongly considered part of logging and tracing the lineage. However, lineage activity shows the links between the data independently of the application, so you must also log and trace the application.

Users
: It is important to log who created, maintained, and ran the applications interacting with the data. This way, you can quickly

identify who can help with data issues and resolve the matter faster.

Lineage
: Logging lineage allows you to map the connection between applications and data sources (i.e., the data flow) so you know which applications are generating the data and who is accessing or using that data. By tracing the lineage when an application changes or its outputs become erroneous, you can identify what data and data users are impacted.

Distribution
: Distribution refers to the shape of the data. It can be the number of occurrences of categories or the descriptive stats (e.g., average, minimum, quantiles, etc.) of numerical values. It is important to know the distribution of values to match their relative adequacy to their usage. If data distribution varies too much, it's an indicator that the initial assumptions may be wrong, which in turn means the analysis of the data may be inaccurate.

Time-based metrics
: There are several forms of time-based metrics that you should log and trace. These include:

  Frequency
  : Logging the pace at which the data is updated allows you to determine whether it is being updated at the appropriate cadence.

  Freshness
  : Logging when data is created, updated, or deleted allows you to determine how "fresh" the data is. This will then allow you to identify whether it's up to date or if it needs to be refreshed.

  Time frame
  : Logging the length of time the data spans (e.g., the time frame could be five years, a year, or only a week of data) allows you to determine whether the data is complete.

Completeness
: The data must be complete to ensure the maximum information available is at play. There are two dimensions to consider:

Missing values
: Logging missing values indicates that no data value is stored for the variable in an observation. Knowing that you have a missing value is important because missing values can add ambiguity and result in an inaccurate analysis of the data.

Volume
: Logging the amount of data used or created by an application allows you to determine whether your data generation is meeting expected thresholds and determine the completeness of your data. For instance, if your data volume suddenly changes from 500 million rows to 5 million, you'll want to take a closer look to determine why.

Schema
: Schema changes occur when fields or tables are added, removed, or changed. Logging schema changes will help you with ensuring the accuracy, completeness, and consistency of your data.

In addition to monitoring these events, a robust and holistic approach to data observability also requires observing data usage within both the pipelines and the overall data systems. Thus, I'm emphasizing once again (because it's such a critical point!) that you want your applications to create logs for both data usage and the applications within the data pipelines or the overall systems.

Finally, when measuring data quality, you need to think about the differences between input and output (i.e., what inputs you need to get an accurate output). One way to approach this problem when you have massive datasets is to create metrics only around the data the application consumes. For instance, if you have a database with 10 billion entries, but your application only uses 1 million of those entries, then you need only log and trace what you care about within the 1 million entries. So, if the 10 billion entries include data from all countries globally in this instance, but your use case only relates to the 1 million entries from Russia, you should determine what your data quality key performance indicators (KPI) are specifically for those 1 million Russian entries. Those KPIs might be tracing whether you received too many missing addresses, whether you're getting the same number of fields in a table, whether you still have the same number of entries over time, etc.

Whatever your KPIs are, by setting metrics for your data quality and then logging events across all systems, you have the visibility to observe whether those data quality metrics are being met. And when they're not, through tracing, you can discover what's causing the error. This combination—metrics, logging, and tracing—provides true end-to-end visibility across all pipelines and data.

Now that you understand what is required to create data observability within your data pipelines and datasets, we'll move on to discussing the more practical issue of how to manage data issues that are within your area of responsibility and those outside it.

Using Circles of Influence to Reduce Circles of Concern

When we implement data observability and data quality, it's important to recognize the full complexity of the situation within an organization. When it comes to data management, there are often issues outside our control that impact the data quality for a particular use case. This can lead to high levels of frustration.

Acknowledging that we all have challenges outside our power to control, Stephen Covey developed a framework called the "circle of influence and control" in his book, *The 7 Habits of Highly Effective People* (Free Press). Here's how he defines each circle in the framework:

Circle of concern
: Challenges or issues that have a direct impact on you

Circle of influence
: Areas within your life where you can indirectly control or influence the outcome

Circle of control
: Areas within your life where you have direct control of the outcome

As Covey points out, the challenge is that there are a lot of issues within our circle of concern that we do not have direct control over. We cannot directly, as individuals, eliminate many of the concerns we have that impact our quality of life, such as the high cost of health care. However, there are ways we can indirectly influence these areas of concern. For instance, we can use our influence to eat

healthier and exercise in order to reduce the likelihood of needing to use health-care services. Or, if we are concerned about the high cost of gasoline, we can purchase an electric vehicle to avoid paying for gas altogether.

Let's take this circle of concern/influence/control framework and apply it to data management within the enterprise. We can see that in the world of data there are often circles of control and influence and circles of concern:

The circle of concern
: The area or scope where data issues can create problems, but you have no power to fix them

The circle of influence
: The area or scope of data issues within which you have the capability to influence a resolution of the issue, if not directly fix it

The circle of control
: The area or scope of data issues that you can directly fix or resolve

To give an example of how this might apply to a real-world scenario, let's look at a pharmaceutical company that ingests data from multiple external labs and uses that data to determine what molecules or compounds might be effective in developing a new drug. The pharmaceutical company has an application to help them with this process and specific data inputs needed to conduct the analysis. This application, which was built by the pharmaceutical company, is within its circle of control. The pharmaceutical company has complete power to make changes to the application. However, the data that comes from their external labs and is ingested by the company's application is outside of its circle of control. If that data is incomplete or inconsistent when the application ingests it, it will produce a faulty analysis. The problem is that this issue falls into the circle of concern—an area where the pharmaceutical company is impacted by the issue but has no direct way to control the outcome (e.g., identify and fix the data issue).

However, through data observability, the pharmaceutical company can still use its circle of influence to help resolve data issues within its circle of concern. By logging and observing the data quality within its own application, the pharmaceutical company can identify

the specific lab the data issue is coming from, and it can even further pinpoint what data is creating the issue. So, while the company can't fix the problem itself because it's outside the company's circle of control, it can use its circle of influence by sharing its findings with the affected lab to help speed a much faster resolution to the concern. Additionally, this level of observability allows the pharmaceutical company (the consumer) and the labs (the producers) to create mutually acceptable service level agreements (SLAs) that can validate from both the producer and consumer perspective that the data matches expectations and meets contractual requirements. This is creating a culture where teams are sharing concerns and combining their influence in order to face issues together, as represented in Figure 2-1.

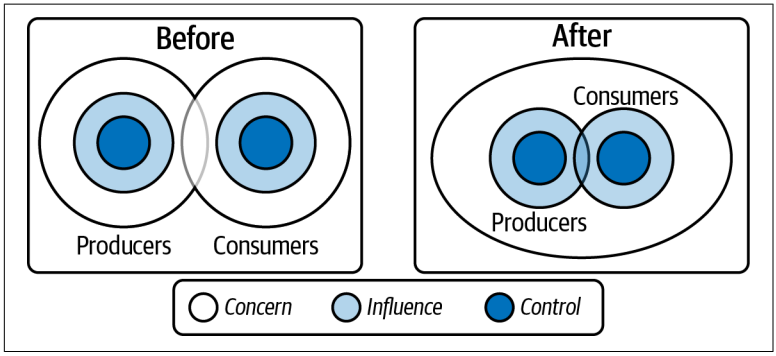

Figure 2-1. With data observability, circles of influence expand and enable stronger collaboration

Without data observability, not only would the data quality issue be outside the pharmaceutical company's control, but it would also be very difficult and time-consuming even to pinpoint which external input (lab) was creating the problem. And even once the company could identify the lab causing the issue, without observability into the data itself, it still would not have a clear understanding of the root cause. But, by being able to trace the issue within its own systems and datasets, the company provided a much more informative and detailed report of the issue to the lab. In turn, this improves the affected lab's ability to resolve the issue and speeds the time it takes them to do so. Thus, using circles of influence, you can observe and identify data quality issues, build greater trust in the entire system, and find and fix the root cause of a data issue more quickly.

Next, let's look at how data observability facilitates teams working together.

How Data Observability Influences Team Dynamics

Another area of complexity regarding data management within an organization is how roles and responsibilities focused on managing and using data are structured. Within any data-driven organization, there are several teams involved in the use of data. The exact structure and types of teams may differ in every organization (e.g., the analytics teams might be split across the data and business teams in some organizations), but there are typically four primary teams involved in managing, analyzing, and utilizing data. Each team has different responsibilities as well as different dependencies on other teams to ensure they can manage their responsibilities (Table 2-1).

Table 2-1. The teams

Type of team	Responsibilities	Dependencies
IT team/ production and operation team	Build and maintain the platform; manage security and compliance; oversee site reliability engineering (SRE).	Receive recommendations from other teams about what to monitor and what to consider as an error.
Data team	Build the data pipelines and manage the data from deployment to production.	Rely on IT to build the platform and set up the tools to monitor the pipeline and systems, including data observability tools.
Analytic/data science team	Build analytics and AI/ML models that can analyze and interpret data for the business team's use cases.	Rely on the data team to build the pipeline to generate the data it will be using in its models.
Business/ domain team	Sponsor use cases and use data analysis to make business decisions.	Rely on the other teams to ensure data and analyses are accurate.

While this structure allows each team to maintain its own quality controls and outcomes, this process also creates silos. Each team has limited visibility into the functions of the other teams and how the data or pipelines might need to be changed (or not) to fit those functions. This makes it difficult for the IT team to anticipate any changes and take preventative measures to avert any issues experienced by the data or analytic teams.

As a result, the data and analytic teams are alerted to potential issues either too late or not at all. Inevitably, this causes failures in their areas of responsibility, resulting in inaccurate insights being delivered to the business team. Without visibility, it's also very hard to find the root cause of an issue. Instead, the IT or data team might uncover a proximal cause, which they patch. But because the root cause hasn't been discovered, there's little confidence that the same issue won't occur again.

Another aspect influencing team dynamics within organizations is the widespread embrace of a data architecture built on cloud data lakes. The real-time availability and streaming this cloud-based architecture provides is intended to make managing large volumes of data from multiple sources easier and faster to ingest, transform, and serve up. However, this type of architecture often gives teams less control over the increasing volumes of data, resulting in less accountability, which leads to a backlog of unresolved data issues. Consequently, the business team begins to lose trust in IT and data teams. As a result (not surprisingly), a culture of shadow IT emerges as analytics and business teams seek to bring more confidence and control into the data analysis process by internalizing some IT resources within business departments instead of relying on the existing IT department.

What can be done to resolve this negative feedback loop that leaves no one within the organization fully accountable and responsible for resolving data issues? An emerging approach that is beginning to gain wider acceptance is the use of a data mesh.

Let's take a closer look.

Using a Data Mesh to Increase Team Accountability and Responsibility

As discussed earlier, the typical organizational team structure for managing data has led to accountability and responsibility issues—each team is siloed and not accountable for issues that arise upstream or downstream from its area of responsibility. The data mesh—an organizational construct that supports data-driven organizations by leveraging a distributed, domain-specific, self-serve design—helps alleviate these silos by increasing accountability and facilitating cross-collaboration communication.

Developed by Zhamak Dehghani, the director of emerging technologies at Thoughtworks, the data mesh is "a distributed data architecture, under centralized governance and standardization for interoperability, enabled by a shared and harmonized self-serve data infrastructure."[2]

While a data mesh doesn't introduce data quality standards, it provides standard guidelines to enable better management of data products, including their quality. These underlying standards (such as standardizing on governance, discoverability, formatting, etc.) help facilitate cross-domain collaboration, especially when data is important to more than one domain. As Dehghani explains in her book, a data mesh has an architecture similar to what is shown in Figure 2-2.

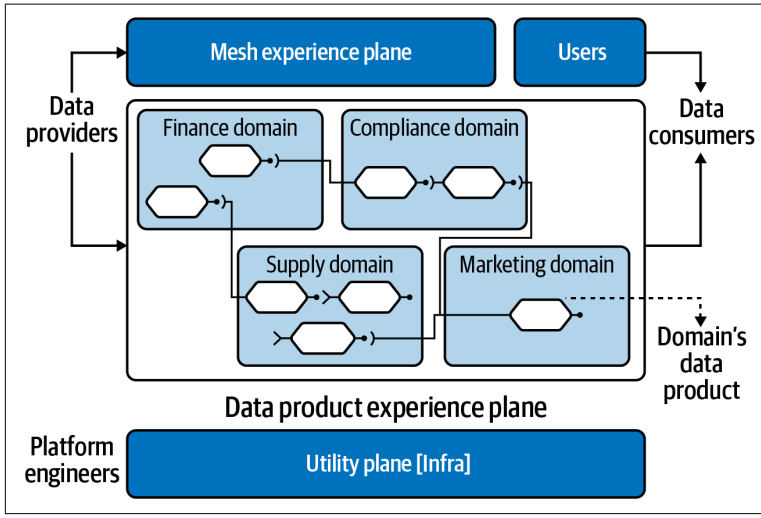

Figure 2-2. Data mesh planes supporting the domain's data products and their usage[3]

The data mesh's universal standard of data governance distributes data ownership among domain data owners responsible for providing their data as products. It also ensures a standardized and scalable

2 Zhamak Dehghani, "How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh," May 20, 2019, *https://martinfowler.com/articles/data-monolith-to-mesh.html*.

3 Adapted from an early version of Zhamak Dehghani's book, *Data Mesh* (O'Reilly, 2022).

way for individual domains to handle agreed-upon data quality, such as:

- Data discovery and versioning
- Data product schema
- Data governance
- Data standardization
- Data lineage
- Data monitoring and logging

As a result, teams can apply observability to the data to determine whether their data is fresh, complete, correctly distributed, and so on. In many ways, the data mesh is like the data platform version of microservices: each domain handles its own data pipelines with defined and agreed-upon SLAs that they guarantee to users.

Because data mesh principles primarily relate to accountability, transparency, and responsibility, there is shared visibility into the data and pipelines. Thus, it becomes much easier for both teams to discern their circles of influence. As a result, problems can be more easily identified, and IT and data teams can work together to fix issues upstream before they create downstream issues. And, with more time freed up, data teams can focus more time on data innovation.

Because of three factors—the rapid growth in the number of pipelines, the increase in the number of people managing the pipelines, and the introduction of complex AI/ML models—many of the data issues we've been discussing have compounded. Therefore, in the next section, we'll explore how increased transparency (or observability) can be applied to organizations to allow them to scale data quality management.

Applying DevOps Practices to Data

The concept of monitoring key business processes has existed for quite some time. In the IT and DevOps environment, due to the need for greater visibility, confidence, and speed in their applications and systems, those teams developed processes around their release cycles to avoid errors and dependency failures. These processes were initially manual, but as organizations scaled from a few

developers to thousands, it became necessary to automate these processes.

With the almost universal adoption of the cloud, IT introduced continuous integration/continuous deployment (CI/CD). Implementing CI/CD has allowed the IT and DevOps teams to automate the integration of code changes from multiple contributors into a single software project and then test and validate that the new code won't break the application. By adding automation to the process, IT and DevOps teams vastly improved their ability to deploy applications confidently and quickly. Automation has allowed them to:

- Reduce the overhead of manual coordination
- Reduce the amount of manual work
- Let developers experiment fearlessly
- Create automated tests
- Increase confidence in each release
- Find out sooner when something breaks
- Automate deployments
- Centralize monitoring

We're now seeing a similar recognition within organizations for visibility, confidence, and speed at the data level, not just the application and systems level. Just like businesses can't afford application downtime, they've also come to realize that they can no longer afford issues with the data because they're using it to make strategic decisions.

In the past, there hasn't been a way to automate the testing and validation of data in the same way there is in the IT and DevOps environment. Thus, a data issue can be very time-consuming and difficult to resolve because it requires a lot of manual coordination and work.

For instance, an ecommerce platform may be using complex machine learning models to provide personalized recommendations to site visitors. Data from many different business segments are piped into this model to train it—customer relationship management (CRM) data, POS data, third-party partner data, contact center data, and so forth. But if that data breaks at some point from when it's created to when it's fed into the model (perhaps some of the

demographic fields from the CRM, such as the age or gender, are incomplete), the training models will be incorrect. In this example, customers might start seeing recommendations that are completely misaligned with their interests, such as a 40-year-old male receiving a recommendation to purchase a dress designed for girls between the ages of 6 and 12.

Suppose the IT team of the ecommerce platform conducts a manual investigation to determine why its personalized recommendation engine is broken. The team may find that it's almost impossible to find the root cause, but it does discover a proximal cause. In this instance, the numbers of missing genders and ages in the dataset have increased. Therefore, other features of the person have taken precedence over age and gender, which leads to poor quality recommendations as these two variables are known to have a major influence on the quality of the model. The result is that the distribution of visitors' ages has varied to the point that the application can no longer manage it. This "garbage out" data is then consumed by the marketing automation tool that sends the recommendations to consumers, thus creating a garbage in, garbage out scenario.

However, simply patching the issue doesn't resolve the root cause, which may have possibly been due to an error occurring during the profile processing. This leaves the organization in a precarious situation. Either it must spend significant resources and time trying to uncover the root cause manually, or it can simply fix the proximal cause quickly but risk that the issue could reoccur and cause even more significant issues next time.

This type of issue can be intercepted to avoid the downstream cascade of incidents, but it requires constant checkups and visibility into your pipeline and datasets. A data observability solution is a system that aims at supporting DevOps practices in the context of data and provides the necessary visibility and validation of your data quality, and it does so in an automated fashion. This is similar to how IT and DevOps have deployed CI/CD to give them more visibility and confidence in their applications and systems while also speeding time to market and reducing downtime.

Having observability of your data and data pipelines is important across all data use cases but particularly for AI/ML, where there is also a high level of complexity in the algorithms. If IT and data teams don't have visibility into data and data pipelines, the

probability for data incidents increases significantly. At the same time, the ability to resolve these issues in a timely manner decreases dramatically.

In the next chapter, we'll explore more about how you can use data observability to make your organization data incident–resilient and strengthen AI innovation and data application creativity.

CHAPTER 3
Conclusion: The Benefits of Data Observability

A data observability platform helps IT and data teams detect and stop the propagation of data incidents by tracking and measuring data usage performance across systems, projects, and applications in real time. With a data observability platform, it's much easier to find and fix the root cause because you have:

- Better visibility into how data is being collected, copied, and modified by any application
- The ability to leverage lineage and historical data information to find the initial cause
- The ability to detect anomalies based on historical data information, which is particularly useful in AI/ML applications

In addition, a data observability platform can facilitate better communication across teams and organizations because it provides you with evidence-based information on your data management ecosystem.

You can also use a data observability solution preventatively to observe issues before they occur. You can do this by:

- Logging context and runtime information from within the application code
- Defining quality control rules and thresholds to anticipate data issues

- Validating rules before and continuously in production to generate notifications about specific data events and their context
- Reviewing or refactoring your delivery upon triggered rules

As organizations continue to scale their use of data, they must actively work to become data incident–resilient. Otherwise, data issues will snowball and so too will the negative repercussions to the organization. To achieve data resiliency and avoid data incidents, there are three key steps organizations must take:

1. *Increase the visibility and awareness of data concerns*

   While many IT and data teams may have heard of data observability, it is not currently a common best practice within most organizations. But it must become one. The awareness of data concerns should also go beyond the data engineer all the way to the CIO and even to the CEO (more on this soon).

2. *Contextually log, measure, and trace data usage*

   Logging, metrics, and tracing must also be automated at the systems and data levels, especially within the production environment. Because the data changes, validating code against data change must be part of the development and deployment phases.

3. *Use automation to continuously validate and test the quality of data in production*

   Automation is the only realistic answer at the scale at which today's data-driven organizations use or want to use data. Achieving automation requires also building in data control rules on indicators (or metrics) as part of the development process of every application. This is a new step in the application's life cycle compared to IT's continuous integration and testing process.

With these steps in place, it becomes possible to quickly identify in near real time where data issues are occurring, even proactively, and resolve them at the root issue. Data observability builds greater confidence in the data throughout the organization, allowing it to improve decision making and experiment and innovate more with data applications.

To realize the possibilities and opportunities of AI, organizations must first build a culture that understands the value and importance of proper data management and prioritizes ensuring the uptime of its data through continual data observability. However, only 24% of companies say they've built data-driven cultures.[1] Nearly half (40%) cite a lack of alignment within the organization as a barrier to being data-driven.[2] Moreover, around 1 in 10 businesses say they don't know where data ownership has sat or will sit in the future.[3] This will need to change if organizations want to support better data management and maintain a coherent and cohesive data strategy.

While Dun & Bradstreet reports that there's a growing recognition that responsibility for data should be a priority for the C-suite, there is still uncertainty around who in the leadership team owns data management. The most common answer is that data responsibility lies with the chief executive officer rather than the chief technology officer or chief information officer. In truth, the responsibility lies with all three executives. Cross-organization buy-in of the importance of maintaining data quality is necessary to put the appropriate technologies and processes in place to achieve good data management governance throughout the organization.

Additionally, to fully realize the benefits of AI, organizations will need to build their business practices around replacing a culture of secrecy with one of transparency. While achieving this ideal "data state" is still a distant dream state for many organizations, there are signs that executives see the value in building a stronger data-driven culture, especially to support AI initiatives. In one study, 81% of executives said they were optimistic about the outlook for data and AI within their firms.[4]

---

1 NewVantage Partners, *Big Data and AI Execution Survey 2021: The Journey to Becoming Data-Driven—A Progress Report on the State of Corporate Data Initiatives, 2021*, *https://c6abb8db-514c-4f5b-b5a1-fc710f1e464e.filesusr.com/ugd/e5361a_d59b4629443945a0b0661d494abb5233.pdf*.

2 Randy Bean and Thomas H. Davenport, "Companies Are Failing in Their Efforts to Become Data-Driven," *Harvard Business Review*, February 5, 2019, *https://hbr.org/2019/02/companies-are-failing-in-their-efforts-to-become-data-driven*.

3 Dun & Bradstreet, *The Past, Present, and Future of Data*, 2019, *https://www.dnb.com/content/dam/english/dnb-data-insight/DNB_Past_Present_and_Future_of_Data_Report.pdf*.

4 NewVantage Partners, *Big Data and AI Execution Survey 2021*.

There will be a positive ripple effect as more organizations begin to make the necessary culture shifts. This includes adding more accountability and ownership for the various teams that interact with the data and prioritizing the confidence of data usage through continual data observability, as DevOps and business applications have already done. We'll see data observability acting as the safety net and thus providing greater confidence in complex AI/ML models and applications. Hence, developers will also have more confidence that they can experiment safely with data applications—even highly complex ones. This, in turn, will boost data creativity and innovation, and keep the hype of AI alive.

About the Author

Andy Petrella is an entrepreneur with a mathematics and distributed data background.

In the data community, Andy is known as an early evangelist of Apache Spark and the creator of Spark Notebook. He has also been an O'Reilly author and trainer since 2015, covering topics such as distributed data science, data lineage essentials, data governance, and machine learning model monitoring.

Andy is also the founder and CEO of Kensu.io, a low latency data observability solution that comes with a specific method: Data Observability Driven Development.