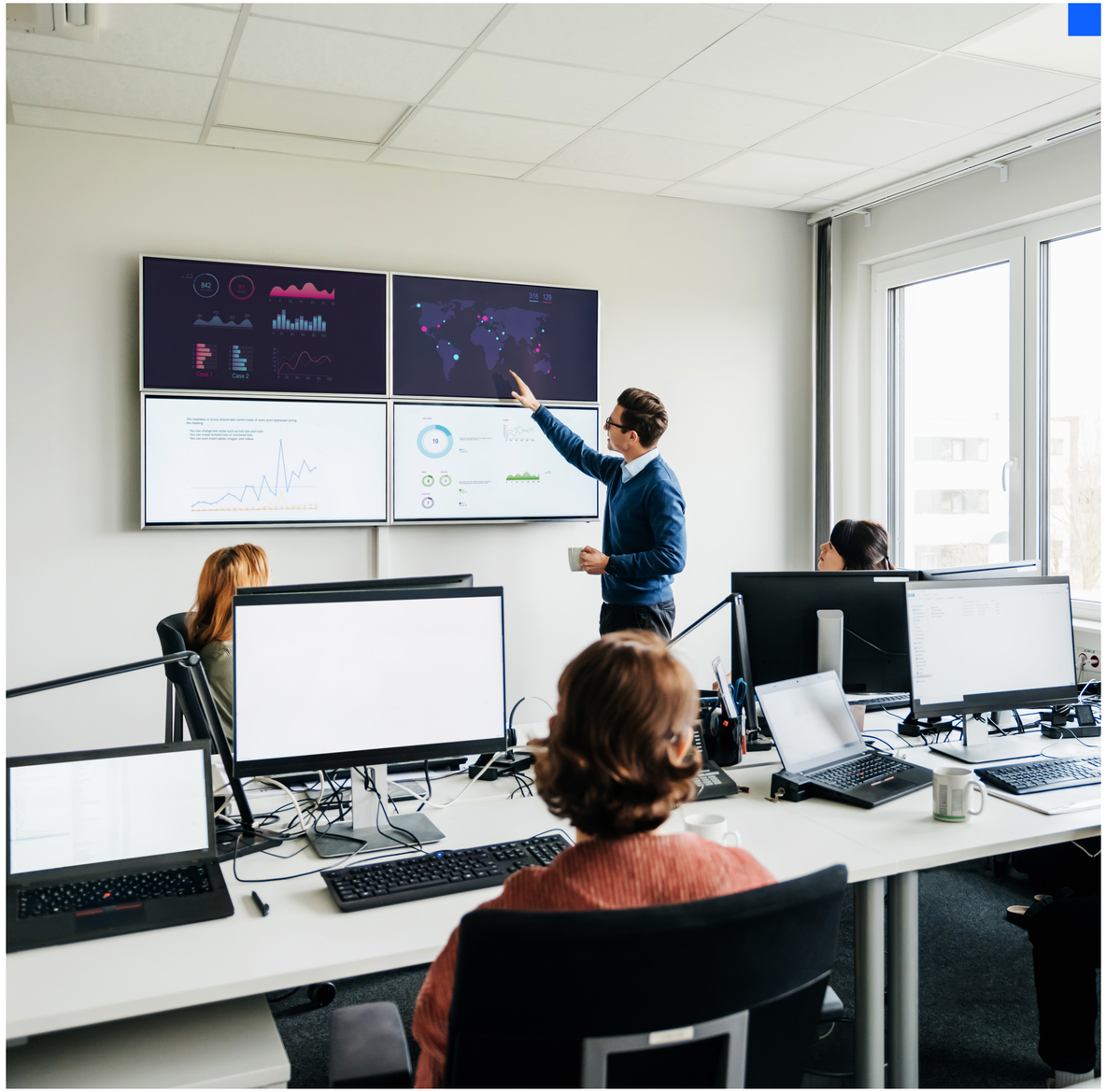
10 advanced data pipeline strategies for data engineers



Contents

01 →
Overview

02 →
Understand the precedent

03 →
Build incrementally

04 →
Document your goals as you go

05 →
Build to minimize cost

06 →
Identify the stakes
and tolerance

07 →
Organize in functional
workgroups

08 →
Implement data
pipeline monitoring
and observability tools

09 →
Use a decision tree
to combat tool sprawl

10 →
Build your pipeline
to control for all four
dimensions of data quality

11 →
Document things as
a byproduct of work

12 →
Conclusion

01

Overview

Schematics for an ideal data pipeline are nice but not always helpful.

The gap between theory and practice is vast, and it’s common for people to make suggestions online irrespective of realities such as, say, budget. Or without knowing that you rarely spin up data pipeline architecture entirely from scratch.

Far more common is to inherit a data pipeline mess. Or if you do get to build something from scratch, you’re working under serious constraints, with alerts such as:
– High null count detected!
– Data schema has changed!
– Pipeline duration anomaly!

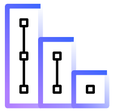
Save yourself a few headaches next time you have to untangle messy pipelines. IBM® Databand is a unified data observability platform that can help you:
– Deliver on-time data
– Ensure data completeness
– Maintain data accuracy
– Expedite remediation on data issues

01

Overview

Here are 10 strategies for how to build a data pipeline drawn from dozens of years of our own team's experiences.

1.
Understand the precedent

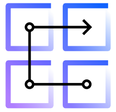


2.
Build incrementally

3.
Document your goals as you go

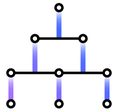
4.
Build to minimize cost

5.
Identify the stakes and tolerance



6.
Organize in functional work groups

7.
Implement data pipeline monitoring and observability tools



8.
Use a decision tree to combat tool sprawl

9.
Build your pipeline to control for all four dimensions of data quality

10.
Document things as a byproduct of work

02

Understand the precedent

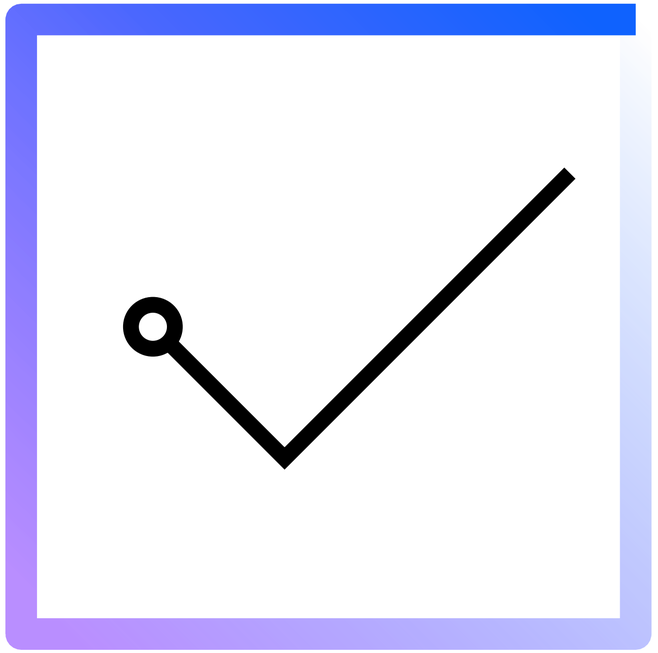
Before you do anything, spend time understanding what came before.

- Know the data models of the systems that preceded yours.
- Know the quirks of the systems you're pulling from and importing to.
- Know the expectations of the business users.

Call it a data audit and record your findings along with a list of outstanding questions.

For example, at a large retailer, the most exciting thing isn't the tool that works by itself but the one that works cooperatively with the existing architecture and helps you migrate off it. Often, such teams have 10,000 hours of work invested in some of their existing products.

If someone tries something new and it fails in a big way, they may lose their job. Given the option, most would rather not touch it. For them, compatibility is everything, so they must first understand the precedent.



03

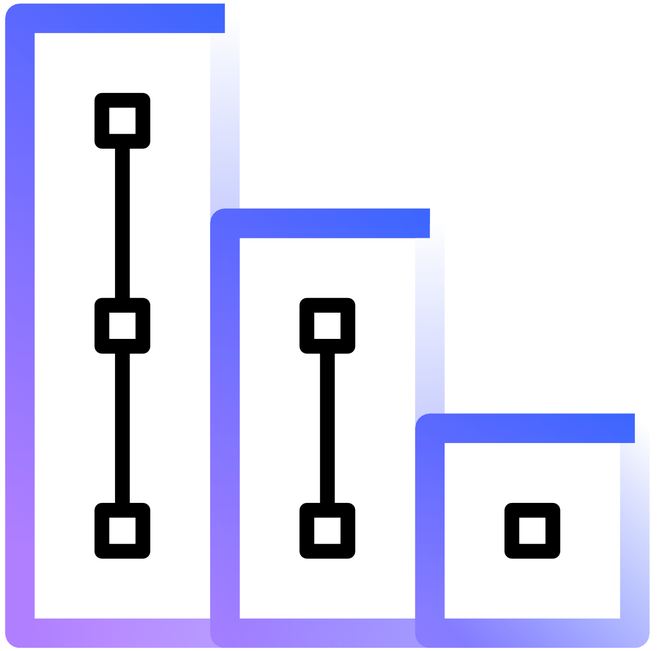
Build incrementally

Build pieces of your pipeline as you need them in a modular fashion that you can adjust.

The reason for this is that you won't know what you need until you build something that doesn't quite suit your purpose. It's one of the many paradoxes of data engineering. The requirements aren't clear until a business user asks for a time series that they only just now realized they need, but which is unsupportable.

Guarantee every incremental build is hitting your key performance indicators (KPIs).
As you build incrementally, you should be comparing each iteration's performance in a local environment against the pipeline already in production. Without the right tools, it's difficult to quickly compare pipeline performance across different environments.

IBM Databand makes it easier to:
- Centralize your pipeline metadata in one place
- Compare and visualize pipeline performance metrics
- Quickly ID the sections of your pipeline that are causing delays



04

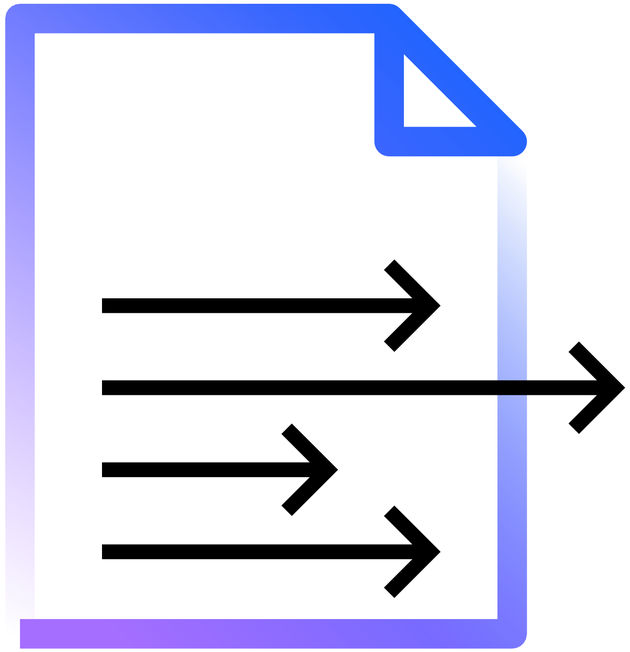
Document your goals as you go

Your goals will continue to evolve as you build.

Create a shared living document such as a Google Doc and revisit and update it.

Ask others who will be involved in the pipeline, upstream or downstream, to document their goals as well.

Everyone tends to presume what others are thinking. It’s only by documenting that you realize someone wants a metric that, say, includes personally identifiable information (PII), which is not allowed.



05

Build to minimize cost

Costs will always be higher than you expect.

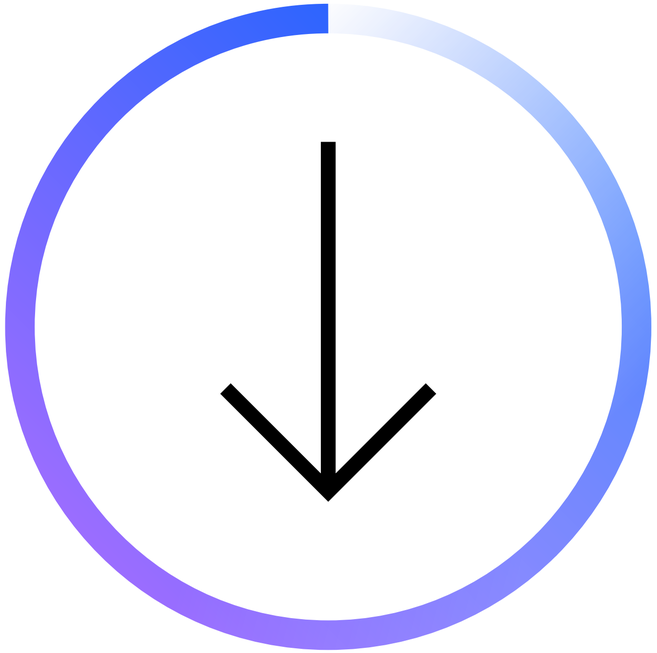
We've never heard an engineer say, "And to our great surprise, it cost half as much as we first thought." When planning spend, all the classic personal finance rules apply: Overestimate costs by 20%, don't spend what you don't yet have, avoid recurring costs and keep a budget.

If there are components that will have to grow exponentially and you can pull them off a paid platform and do it for free or at low cost, that may be the key to accomplishing twice as much with this pipeline, and building more.

Even as data lake providers launch features such as cost alerts and budgetary kill switches, the principle remains: Build to minimize cost from the very beginning.

Optimize your pipeline costs to score quick wins for your team. Businesspeople have a hard time understanding the nuances of data engineering. But they do understand costs. Reducing the cost of projects currently in production can win your team the favor you need to get your next project pitch approved.

Pinpoint exactly where inefficiencies in your tech stack are occurring and quickly ID the direct and root causes.



06

Identify the stakes and tolerance

High stakes and low-tolerance systems require careful planning. Consider a rocket going into space with human lives onboard. But in the data world, most decisions are reversible. That means it can often be cheaper in terms of your time and effort to simply try it and revert rather than agonizing for weeks while deciding.

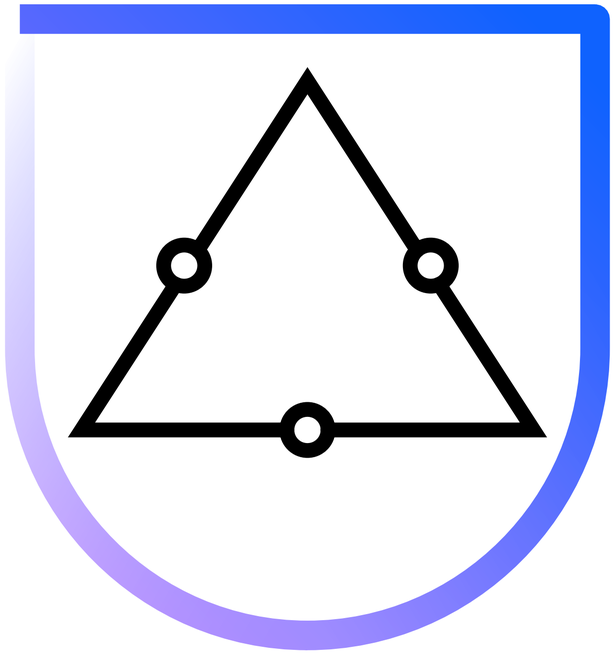
For an e-commerce company, the stakes might seem low at first. But after talking to business users, you might learn that the downstream effects of a data error could make millions of products appear available in a store when they're not, creating a web of errors and missed expectations you can't easily untangle.

Knowing the stakes and tolerance tells you how much "breaking" you can afford to do.

Mitigate the risk of failure in low-tolerance systems.
When you're dealing with high-stakes, low-tolerance pipelines, you need to know there's an issue before your data service level agreements (SLAs) are missed.

With IBM Databand, you can use out-of-the-box, customizable metrics tracking and machine learning–powered anomaly detection to get notified of issues as they crop up.

Receive alerts on the leading indicators of data pipeline issues for your critical assets so you can begin working on a resolution immediately.



07

Organize in functional workgroups

Create working groups that include an analyst, a data scientist, an engineer and possibly someone from the business side.

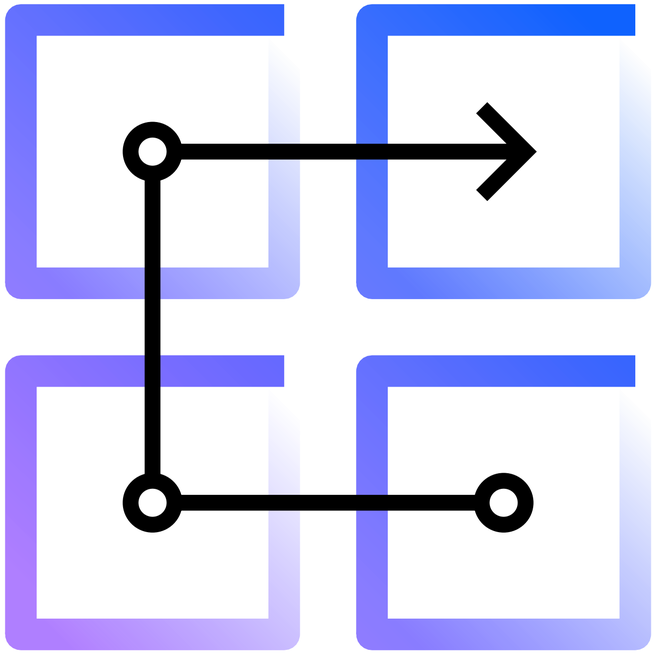
Have them focus on problems as a unit. It's far more effective. If they simply worked sequentially, tossing requirements over the fence to one another, everyone would eventually grow frustrated, there'd be a lot of inefficient "work about work," and things would take forever.

Functional groups tend to build better data pipelines that cost less. This approach also gives data engineers a seat at the table when decisions are being made so they can vet ideas at the outset.

If all they do is wait for notebooks from the data scientist, they'll often discover they don't work and either have to send them back or rewrite them themselves. Or they'll find that other teams continuously ask for columns that are derivable from other data but must be transformed.

"A constant challenge is ensuring my data engineers have a good contract with data scientists and know how to take products from them and smoothly integrate them into the system. Even with pods, it's not always smooth."

Data engineering team lead



Implement data pipeline monitoring and observability tools

Some tools help you keep costs low, and observability tools fall into that category.

They provide instrumentation to help you understand what's happening within your pipeline. Without highly specific answers to questions around why data pipelines fail, you can spend an inordinate amount of time diagnosing the proximal and root causes of pipeline issues.

Guarantee your SLAs
Identify when anomalous durations or failures will cause late data deliveries and missed SLAs.

Root cause analysis
When issues occur, drill into the source of errors or data corruptions across your pipelines.

Unified logging
Access execution details, success logs, error messages and runtime states from your data tasks, queries and functions.

Manage resources
See pipeline resource consumption levels and durations from underlying cloud or compute systems.

Trace lineage of issues
Track how data corruptions or execution issues cascade across your pipelines, from central teams to downstream consumers.

Data health metrics
Monitor data schemas, data distributions, completeness and custom metrics.

09

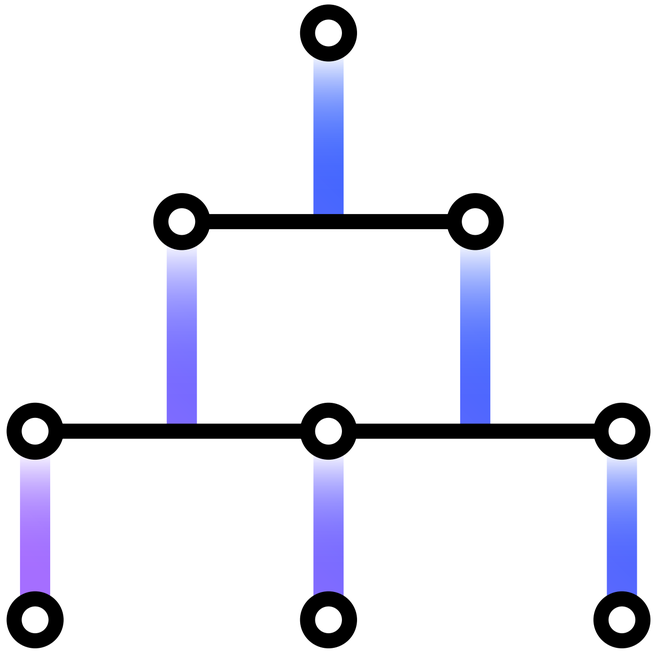
Use a decision tree to combat tool sprawl

Nobody wants yet another point-solution tool that requires maintenance.

Create a decision tree for your team to decide when it makes sense to add another tool versus adjust an existing one, or evaluate a platform that would consolidate several functions.

It’s good for data quality too.

The fewer moving pieces, the less there is to diagnose.

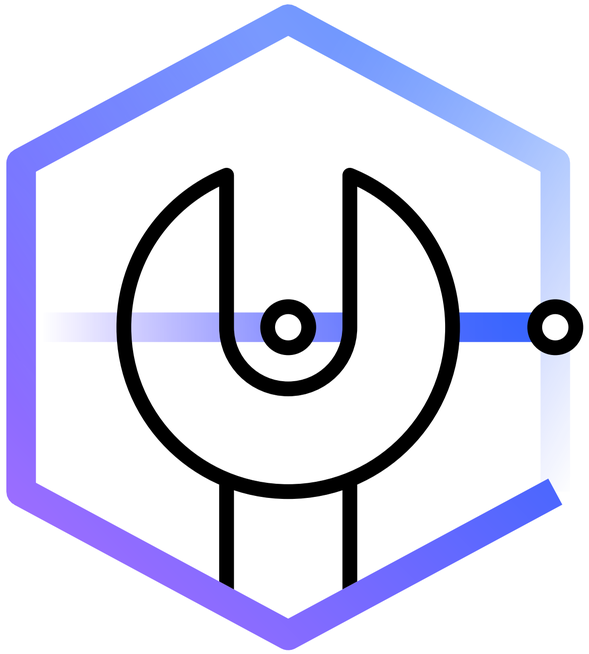


Build your pipeline to control for all four dimensions of data quality

Ensure your pipelines maintain the four dimensions of data quality that matter to engineers—fitness, lineage, governance and stability. These dimensions must exist in equilibrium, and you cannot maintain quality without addressing all four.

Increase the ROI of your data product. Unified visibility over your entire tech stack helps ensure your data quality KPIs are being met during every step of your data's journey.

Centralize your pipeline metadata to ensure the consistent delivery of accurate, fresh, complete data.



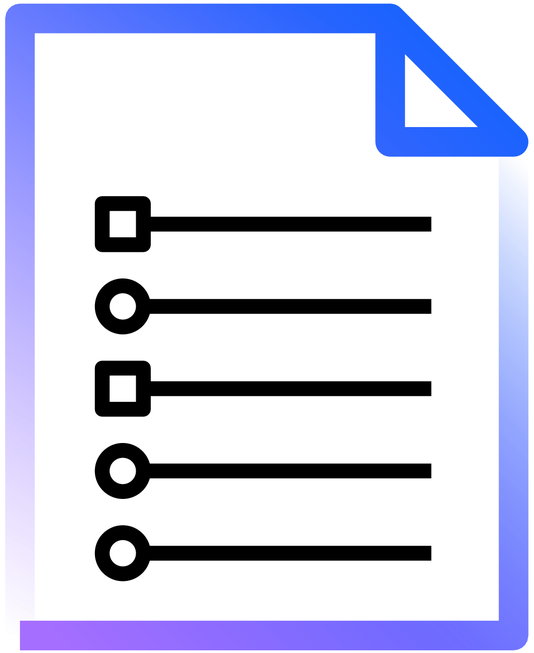
11

Document things as a byproduct of work

You should be in the habit of documenting what you do, and at the very least, keeping a log that your team can access. This is also known as “knowledge-centered service.”

The highest achievement for a data engineer is not being a hero that the entire company depends on but constructing a system so durable that it outlasts you.

Documentation should be intrinsic to your work.



Conclusion: Catch bad data before it gets through

Data engineers are the backbone of modern data teams. But for the average data engineer, it’s a challenge to make sure jobs are running successfully, data is meeting quality standards, and business stakeholders are satisfied. For companies that depend on accurate, on-time data flows, bad data poses a huge problem. IBM Databand helps data engineers scale their infrastructure alongside their organization while maintaining data health standards.

Make big data observability manageable.

Why IBM?
IBM Databand delivers trusted data to your business. Learn more about how IBM Databand can help your organization automatically observe dynamic data pipelines, promote data quality and reliability, and continuously monitor AI and machine learning reliability.

IBM Corporation
New Orchard Road
Armonk, NY 10504

Produced in the United States of America
January 2023