# THE DATA PLATFORM SURVIVAL GUIDE

Put data lakes, data warehouses, and data catalogs to work for humans



## INTRODUCTION

The soul of a data platform is to accelerate discovery in cross-functional teams. Creating a successful data platform requires attention to component technologies, user experience, best practices, and company culture. The Data Platform Survival Guide (DPSG) provides reusable patterns and methods that you can apply to deploy your platform faster, with fewer bugs, and with fewer missing features. DPSG is for all personas on a data-driven team: including leaders, architects, engineers, analysts, scientists, data scientists, IT, and data engineers. Until and unless your cross-functional stakeholders align their mental models of a successful data platform, progress will prove elusive.

We lead with an overview of cloud technologies for data platforms, follow with an executive summary of six insights, and then illuminate pitfalls and solutions to three common business problems:

- **Getting discoveries to market faster**
- **Preparing for IPO (and securing intellectual property)**
- **Improving data quality**

We conclude with a few sections on data collaboration and data governance as elements of a data-driven company culture. Why company culture? Because the artifact that we call "data" is a function of business culture, not merely a function of technology. If you wish to build a data-driven organization, you will need a culture where every stakeholder can create, download, document, search, and visualize datasets without coding—as well as the ability for the team to securely govern data over the months, years, and decades to come.

### Terms we'll use

- **Return on data** — (investment)/(profit_impact), the ratio of dollars invested into data infrastructure to investment impact on the company's bottom line.
- **Data provenance or lineage** expresses where data come from in terms of space, time, interest, and human relationships (STIR). STIR are the canonical dimensions of data context.

- **Data velocity** is the speed with which data are turned into actionable insights.
- **Data collaboration** is the process by which cross-functional teams share and access data to make business decisions.
- **Data superiority** is a state in which, all other things being equal, your data enables you to dominate the competition via higher-quality decisions and models.
- **Outer data** is data born beyond the perimeter of the data lake or data warehouse. Outer data include files in Box, attachments in email, files in network attached storage (NAS), and data in third-party services like Salesforce or Quip.
- **Dataset** — A heterogeneous collection of files needed to make or justify a business decision (e.g. images, spreadsheets, tables, documentation, models).
- **Data hub** — An emerging software category that makes data in lakes, catalogs, and warehouses accessible and actionable to cross-functional teams. Data hubs focus on integrating data sources, capturing lifecycle events, documentation, and lineage.
- **Data quality** — An informal measure of how complete, fresh, comprehensible, and trusted your datasets are.
- **Business user** — Any non-developer stakeholder in datasets and decisions. Examples include scientists, leadership, product, marketing, and finance team members. A "non-developer" is someone who does not wish to write code, or does not have the time to write code in order to interact with a dataset.
- **ETL (extract transform load)** — a broad family of tools and techniques for transforming data from one format or structure to another

## FOUR COMPONENTS OF DATA PLATFORMS

There are broadly four categories of products that enterprises evaluate as components for a data platform: data lakes, data warehouses, data catalogs, and document stores. These categories are more continuous than discrete. To take one example, Databricks can function as a data warehouse and a data lake, and has even advocated for a hybrid "lakehouse" architecture.

Table 1 Data platform technologies

| Category | Function | Schemas | Examples |
|---|---|---|---|
| **Data lake** | Store and analyze unstructured data | Schema-on-read (flexible, determined at query time) | S3, LakeFormation, Pachyderm, Databricks, Presto DB (AWS Athena) |
| **Data warehouse** | Store and analyze structured relational data from transactional systems | Schema-on-write (fixed, known in advance) | Microsoft SQL Server, Snowflake, Oracle DB |
| **Data catalog** | Discover data | Varies; traditionally schema-on-write | Collibra, Alation |
| **Document store** | Store data for business users | None | Box, Egnyte, Dropbox |

# EXECUTIVE SUMMARY OF INSIGHTS

## 01 Stop thinking in PowerPoint slides. Start automating reports.

Slide presentations are grossly inefficient for communicating and analyzing business findings. There are three reasons for the inefficiency of slides:

- **Slides are static**. Unlike a dashboard or web application, slides are not interactive. Without costly in-person meetings, slide decks do not self-explain, obscure data provenance, and do not afford for common sense-making operations like drill-downs, pivots, and summary statistics
- **Somebody wasted hours of their life creating the slide presentation**. The tragedy of slides is that the data they contain already live in files, lakes, and warehouses. These data are then painstakingly translated from their dynamic native formats (e.g. spreadsheets, notebooks, and graphing applications) into flat slides that are soon to be discarded.
- **Slides incentivize expensive hard-to-schedule meetings** since they are designed for, and depend upon, narration.

Efficient enterprises replace PowerPoint presentations with auto-generated, interactive reports that live on the web. (See **data velocity and reporting** for more).

## 02 Box and networked attached storage (NAS) are dead ends for data science

Increasingly, business and research decisions are guided by models and analyses from data science teams. But this guidance becomes impossible when there are no training sets to model! That is to say, modeling initiatives hit a wall when data is unlabelled, inaccessible, or untrusted.

> To give you a sense of how painful unlabeled data is, Cambridge's Broad Institute grew so tired of orphaned data that they implemented a policy to automatically delete unlabeled data after 30 days 😬.

While document stores like Box and Dropbox, and NASs like Dell Isilon, are convenient for non-technical users, they nevertheless evolve into silos of data that are nearly useless for modeling due to the following constraints:

- **Their data are typically unlabeled, unstructured, and undocumented**; having been hastily dumped there under deadline pressure
- In contrast to cloud storage solutions, **they are insufficiently structured for programmatic access** to data and metadata
- They are **segregated from compute resources**, leading to expensive and slow copying to cloud or on-premise compute resources—or, worse yet, no compute analysis whatsoever
- They are **too slow for latency-sensitive applications** like machine learning and high-throughput analysis

Given the above weaknesses of cloud document stores and NASs, teams can adopt one or more of the following procedures to improve their chances of turning documents into training data:

- **Automatically synchronize data** from document stores and NASs to the cloud with sync agents and ETL pipelines
- **Create lifecycle policies** and user interfaces that enforce dataset labeling and documentation
- **Provide dedicated user interfaces for business users** that offer users the ability to label, document, and upload data (while transparently syncing this data to blob storage, databases, and other cloud data stores)

## 03 Moving data to compute is slow, expensive, and unnecessary

In local and mobile computing we think of downloading files to the device. In cloud computing, which separates compute from storage, downloading data is exactly the wrong thing to do. Businesses that use cloud platforms should adopt the mindset of leaving data in place and *moving compute to data* which is more flexible, cheaper, and faster than the reverse. We say "more flexible" since by moving compute to data your team can change its approaches and opinions on compute without disrupting or changing the data layer. Teams that move compute to data can fearlessly mix and migrate across a wide variety of analysis engines, including EMR, Snowflake, K8s, PrestoDB, Spark, Pachyderm, and more. The flexibility of moving compute to data is radically more powerful than old-school systems like Hadoop and Oracle DB which force you to scale compute and storage in lockstep. Moving compute to data is cheaper and faster since in-data-center transfers are free and fast, whereas out-of-data-center transfers are slow and costly.

## 04 Electronic notebooks are the future—but require dataset storage

Electronic notebooks like Jupyter, Benchling, R Markdown, and IDBS are enterprise workhorses. Specialists use notebooks to organize wet science and data science alike. Notebooks are a mixture of documentation, small-scale primary data, and metadata. Nevertheless, **notebooks are not suited to storing large data** such as training sets, serialized models, and experimental results such as sequencing files. Therefore notebooks can only function as reproducible units of experimentation, auditing, and compliance if their code and markdown cells include hyperlinks to larger datasets in the data lake.

For the above reasons you should seek a cloud data storage solution that supports immutable links to large datasets.

## 05 Manage data like code to increase business velocity

Collaboration, debugging, and reproducibility are impractical without software version control. Every modern software team relies on version control systems like git, mercurial, and Perforce to provide a common frame of reference for project collaborators. Why then do so many so-called "data-driven" companies fail to version control their data?

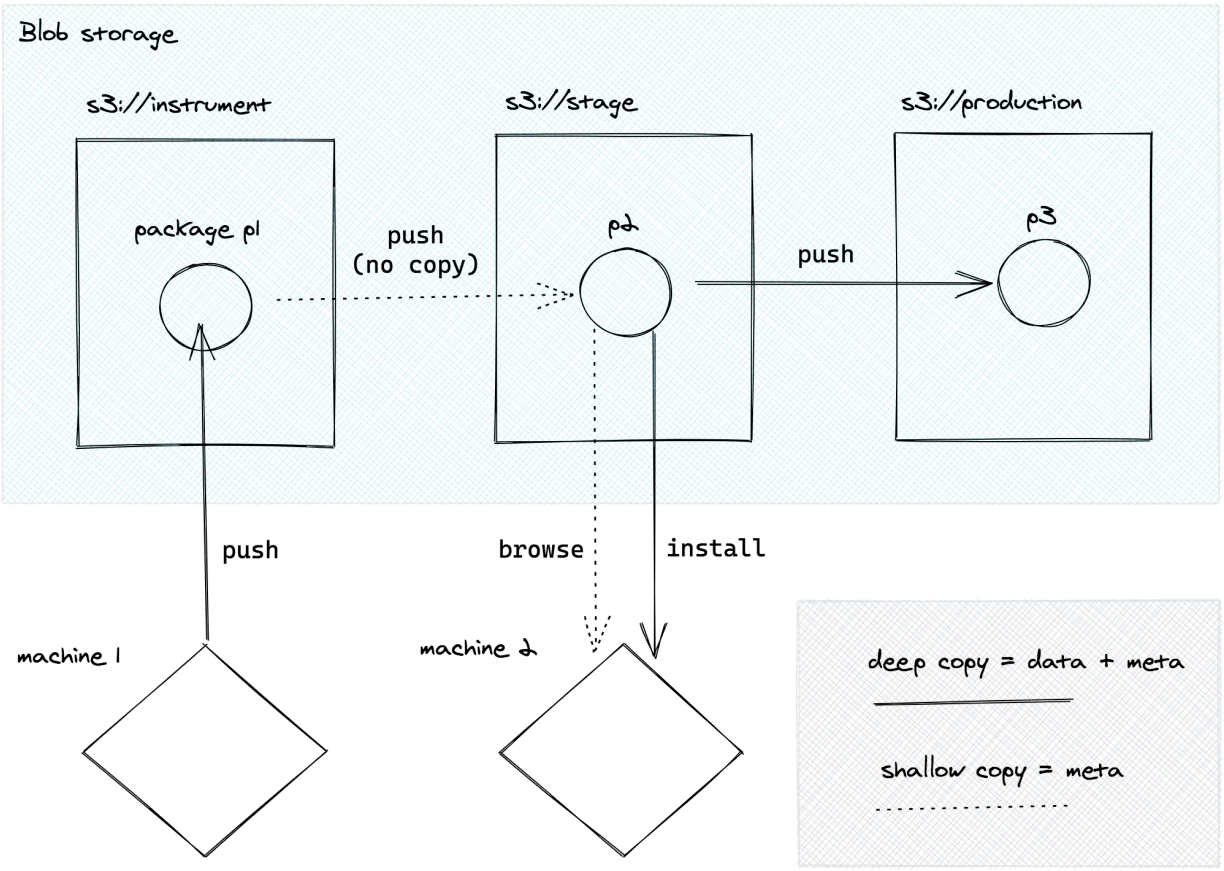
> “[Data management] is so bad it sometimes feels like stepping back in time to when we coded without source control.” —Pete Warden, Google

Data versioning is an emerging field, but we can infer its trajectory from decades of successful revision control in the world of code, starting with IBM's SCCS (1973) and evolving into systems like git and mercurial (2005). "Managing data like code" has four key features. Taken together these features enable cross-functional teams to discover, debug, and collaborate with greater velocity. The features of managing data like code are as follows:

- **Every revision of the dataset has an immutable cryptographic hash**. Said another way, dataset revisions have an unforgeable digital fingerprint that uniquely identifies a revision for all time. As a rule, unless required for compliance or security, revisions are retained.
- **Users can easily time-travel** across revisions for the purpose of compliance, auditing, double-checking past results, and forging new projects from pre-existing datasets.
- **The documentation is embedded** within the dataset and can be written or read by anyone who interacts with the dataset
- **Revisions have a lifecycle reflected in branches**. Branches are self-contained experiments that move from informal ideas to trusted foundations.

You can think of dataset branches as having at least three distinct lifecycle phases: raw, refined, and curated.

Fig 1. Data lifecycle is reflected in branches that map to blob storage buckets




## 06 You bought a data lake or warehouse. Now what?!

The first thing that customers find when they acquire a new data lake, data warehouse, or data catalog is that they immediately require another six to eighteen months of in-house development to make these technologies accessible and useful to cross-functional teams. Why? There are two reasons:

1. **Data warehouses and data lakes are built by developers and for developers**
2. **Data catalogs focus on tables to the exclusion of files**

As a result **non-developers and their files are excluded from your data platform on day one**. Teams are then forced to develop custom, in-house applications, dashboards, and web portals to bring business users and their data into the platform. Now you've got two problems: a new platform that half the company isn't using, and a daunting list of new projects needed to solve the first problem. In the next section we'll discuss how a data hub makes it possible to translate the capabilities of lakes, warehouses, and catalogs into higher data velocity and higher decision quality for cross-functional teams.

# WHAT'S MISSING FROM DATA LAKES, WAREHOUSES, CATALOGS, AND DOCUMENT STORES?

The value of data lakes, warehouses, and catalogs is well-established. Nevertheless due to their common evolutionary history, they share blindspots. Data platforms began with SQL (1970), expanded to Hadoop (2005), and are now transitioning away from Hadoop to cloud platforms that separate compute and storage (2010 to present). This evolution, while tested and proven to create business value, has failed to account for three essential and ineluctable features of modern data:

1. **Schemas—if they are even present—are often unknown and evolving.** A myth that began in the SQL era is that schemas should be known or planned up front. In reality data schemas happen. Files are collected ad hoc, by a variety of people, and slowly evolve into curated datasets where the schemas are partially known. We say "partially known" for two reasons. First, we must always leave room for evolutionary change. Moreover, many of your most critical files will never have a schema: images, text documents, videos, etc.

2. **More than half of your most precious data is outer data, born without schemas, outside the walls of your data lake and data warehouse.** Outer data include the files that emerge from web applications like SalesForce, instruments like sequencers, Desktop applications like Excel, and mobile applications like Gmail. As a rule, data warehouses, data lakes, and data catalogs neglect outer data. Data lakes and data warehouses were designed by engineers for engineers. Therefore business users find it difficult or impossible to access, add, and annotate data in lakes and warehouses. Traditional data catalogs, in fairness, offer some business-user-friendly search features, but nevertheless focus on *structured* data stores, like databases, that have been populated by developers. The effect of the developer-focused and structure-focused biases of lakes, catalogs, and warehouses is that they fail to capture outer data. As a result, your company is deprived of outer data, its meaning and its application.

3. **Document stores are a silo, disconnected from cloud computing, disconnected from developers.** Document stores, while convenient for business users, are a dead-end from the standpoint of data science and engineering. In an era where data science needs to analyze and model data produced by business users, this is an unacceptable tradeoff. The principal weaknesses of document stores are as follows:

    - The APIs are insufficient relative to the gold standard of cloud blob storage
    - There are no mechanisms to verify schemas
    - Support for versioning and heterogeneous collections is weak
    - Search and discovery do not extend into file contents
    - Data have to be moved and copied, at significant expense
    - Performance, particularly network throughput, of copying data to compute is substandard

There are three major consequences of the above blindspots both of which will delay the success of your data platform:

- **Data lakes, data catalogs, and data warehouses—out of the box—fail to serve business users** and therefore fail to capture any of the precious data that business users generate
- **Document stores become a curse to data science** as they constitute yet another pile of unlabeled, unstructured, mystery data that cannot be modeled
- **Moving data to compute is a costly mistake**. The industry migration away from Hadoop is proof that it is more efficient to separate compute and storage than it is to scale them in lockstep. Given that data volumes grow geometrically, the most efficient strategy is to leave data in place in the cloud, where you can bring a wide variety of in-data-center compute systems to the compute, such as Spark, EMR, PrestoDB, Kubernetes, Pachyderm, and more. In short, you should emphasize moving compute to data, as this is far more flexible, cheaper, and faster than the reverse. As a rule, cloud providers like Amazon Web Services do not charge for in-data-center transfers, so be sure to use compute in the same region as your storage.

## THE EMERGENCE OF THE DATA HUB

Given that data warehouses, lakes, and catalogs are designed to interact with structured data, they exhibit a top-down bias which asserts that "if it's not in the database, it doesn't exist." In light of the fact that the bulk of your business's data is neither born in data lakes nor born in data warehouses, but rather is born on personal machines and document stores in a variety of unstructured formats, the industry has begun to recognize the need for a data hub. A data hub interoperates with lakes, catalogs, warehouses, and document stores to provide a layer of reproducibility, discovery, and trust that is accessible to both developers and non-developers. A data hub combines the user-friendliness of a document store with the scale and performance of cloud blob storage, and the schema-awareness of data lakes and data warehouses. Note that data hubs—rather than replacing lakes, warehouses, catalogs, and document stores—interoperate with them by using links, federation, and foreign keys to reference data in other systems.

**You can think of a data hub as the connective tissue between data technologies and human beings. Human beings think, not in tables or schemas, but in collections of files that are known as datasets**

## BUSINESS PROBLEMS, PATTERNS, AND SOLUTIONS

### 01 You wish to bring discoveries to market faster

Data-driven discovery is the engine of growth for enterprises in Healthcare, Life Sciences, Finance, Retail, Transportation, Communications, Energy—virtually every industry vertical. Each of these industries relies on an iterative process of experimentation, analysis, and modeling to yield unique and profitable breakthroughs.

### What most companies do

Faced with the urgency of bringing discoveries to market, most companies do one or more of the following:

- Expand headcount in data architecture, data science, IT, and engineering
- Adopt cloud platforms like AWS, GCP, and Azure
- Acquire data warehousing solutions like Snowflake, Oracle, and Redshift
- Acquire data lake technologies like Lake Formation, Dremio, and Amazon S3
- Acquire a data catalog like Collibra

### What most companies fail to anticipate

The above activities lead to an inevitable and intimidating question: "Now what?!" You've ratcheted up your head count, cloud spend, and technology investments, but now there are miles of custom infrastructure and procedures to build to make your data infrastructure useful to the stakeholders and decision makers in your company.

Where did things go wrong? The are three common anti-patterns that drive the "now what" problem for data platforms:

1. Emphasizing technology over usability
2. Underemphasizing non-developer use cases (a.k.a. business use cases)
3. Neglecting to evangelize a company-wide belief in the value and responsibilities of data governance
4. Failing to account for the custom development and training—often months to years in duration—that will be required to drive adoption of the new platform across the organization

### What uniquely successful companies do

Companies that succeed at accelerating discoveries to market have a culture of data velocity. Data velocity begins not with technology but with the people in your organization. The following platform features support data velocity:

1. Automated reports—as soon data arrives from devices, instruments, and customers it instantly triggers the creation of web-based, interactive summaries and reports that can be reviewed and investigated by team members at any time—without asking a data engineer, without calling a meeting, and without bugging a subject-matter expert for context. In the absence of automated reports, weeks of employee time are lost making slide decks, calling meetings, reviewing results with executive teams, and waiting-waiting-waiting for the data already in your company's possession to be analyzed, reviewed, or annotated.
2. A culture of data collaboration. (See Questions to prepare for data collaboration for more.)

3. A practical, widespread initiative for data governance and data stewardship. (See "Data governance" sounds boring, but it's rad.)

**Example of automated reports:** In enterprise life sciences, automated screening begins when data from a Contract Research Organization (CRO) or in-house instrument is added to Amazon S3. Next, an S3 event initiates a pipeline in AWS Batch or AWS Lambda. The pipeline runs both proprietary and off-the-shelf analysis software. Finally, the pipeline emits an interactive web report that summarizes and analyzes the results of the sequencing run. Team members can now consume the report without waiting on anyone to perform an analysis or schedule a meeting.

## 02 You wish to prepare for an IPO (and safeguard intellectual property)

Preparing for an IPO is a stressful process that sparks vehement disagreements over data, metrics, and what they mean for the business. Why? Because the data are typically produced in silos, by different teams, with little to no shared understanding of what the numbers mean. The remedy for this fragmentation is a collaborative, company-wide process for refining data from raw to trusted (also covered in our sections on data collaboration and data quality).

The following concerns tend to leap to the foreground:

1. Security — so as to avoid a costly data breach
2. Reproducibility and defensibility of results — for auditing, compliance, and legal defense of our innovations
3. Releasing new products on-time (see also **Problem 1, getting discoveries to market faster**
4. Trust and alignment — so that key metrics are accurate, widely trusted, and widely agreed upon by individual contributors and the leaders who communicate these metrics ot the outside world

## What teams do and where it fails

Table 2. Stumbling points on the road to IPO

| Area | Action | Unforseen weakness |
|---|---|---|
| **Security** | Emphasize IAM and other role-based access controls (RBACs) | IAM and RBAC are essential but incomplete and often miss the point. Rigid user roles serve the needs of IT and ensure legal compliance, but tend to neglect the needs of everyday users.<br><br>As a result a *Shadow IT* of workarounds and privately purchased software is employed to bypass what users see as a byzantine data bureaucracy. |
| **Security (continued)** | Acquire network attached storage (NAS), such as Dell EMC Isilon | On-premise network-attached storage seems attractive in that it is on-premise. But NASs prove more expensive, less reliable, and less flexible than cloud blob storage. NASs require you to slowly copy data to an on-prem or cloud compute engine. In recent years, cloud storage performance has also begun to approach and in some cases exceed NAS performance. Therefore, choosing NAS over cloud storage results in a headache that most teams regret. |
| **Understandability & documentation** | Use wikis to document data collections and include links to long-term storage | Wikis like Confluence are subject to change, abandonment and, since they are not stored with the data, divergence from the indicated sources of truth.<br><br>Moreover, links to long-term storage in Box, S3, etc. are themselves highly unreliable and may break at any moment. |
| **Reproducibility** | Maintain numerous archival copies of data | While copies provide some security against deletion, they harm uniqueness and lead to doubts about which copy is the authoritative one. |
| **Metadata & lineage** | Purchase or create systems for tracking metadata, lineage, and provenance | Metadata and lineage are complex problems that require multiple man-years of software engineering to address. Moreover, storing metadata and data in separate systems leads to divergence and inaccuracy. |

| | | |
|---|---|---|
| **Enabling non-developers to store data (scientists, leadership, business users)** | Continue to store business and scientific data in commercial document stores like Box | Storage technologies like Box, while convenient, form silos that are poorly accessible to engineering and modeling teams, encourage data duplication, carry all the security risks of the multi-tenant public cloud, and divide your data governance efforts across multiple fronts (the document store, the data lake, etc.) |

## What uniquely successful companies do: manage data like code

The urgency of compliance and auditing uncovers a need for reproducibility of data across time, across machines, and across collaborators. At the core of data reproducibility is data versioning. Although companies invest heavily in source control—which enables collaboration, auditing, and security at scale—they fail to make equivalent investments in data version control.

In brief, a **successful data version control system** has the following characteristics. We discern these characteristics from careful observation of analogous code hubs, and from years of industry experience in writing and maintaining data versioning software in the open source:

1. **A bomb-proof core.** GitHub was founded on a stable, performant, well-adopted core technology, git. In the same way, successful data versioning systems use the most popular, scalable, and reliable storage primitive in the world, Amazon S3. Note that S3 is such a dominant technology that it is multi-cloud. Both Azure and Google Cloud support the S3 APIs.
2. **Immutable revisions**. Reliable systems are built from pure, idempotent functions that produce outputs that, for the same inputs, can never change. Immutability makes datasets easier to trust, revise, and reason about. Immutability further makes debugging and collaboration simpler by composing pipelines and outputs from well-understood, immutable building blocks in data and code. (See our technical blog post on Versioning data and models for rapid experimentation in machine learning for more.)
3. **Time Travel**. Compliance and auditing can be boiled down to the ability to prove that you did what you said you did when you said you did it. As for data, this implies the ability to reproduce, with perfect fidelity, the state of your datasets at any point in the past, also known as time travel. As a rule, time travel implies a cryptographic hash function, like SHA-256, that provides a unique digital fingerprint for each revision of the dataset. Time travel, combined with immutability, further provides protection against deletion and erroneous edits.
4. **Unified data, metadata, and documentation**. Disparate sources of data, metadata, and documentation are technical debts that lead to divergence, confusion, mystery datasets, and reduced trust in data and decisions. The solution is to introduce datasets as a first-class abstraction in your data platform. The dataset abstraction should be schema-aware but

structure and scale neutral, with the ability to handle millions of heterogeneous files of any type, of any size, and in any physical storage location—so that primary data, metadata, documentation, models, and visualizations can all be unified into a unified, monolithic revision that avoids redundant copies and divergence. Refer to Quilt's open-source efforts on "package manifests" for further details on datasets as a usable abstraction.

5. **A searchable data catalog of files and datasets**. In the opening section we discussed data catalogs as an element of data platforms and noted that most data catalogs have a "schema-first" bias: they tend to catalog tables in databases, missing files in document stores, missing files in NAS, and missing files on personal machines. Data hubs emphasize, on the other hand, a "files-first" approach that automatically indexes files that users place in document stores and data lakes. For instance, teams can use data hub search to find critical spreadsheets by their contents, critical instrument files by their instrument settings, and so on.
6. **A single source of truth for both developers and non-developers**. Decisions and product releases are intensely cross-functional activities that involve leadership, data science, finance, product management, data engineering, and more. In this respect, data has a much wider audience than code. Successful companies will therefore emphasize a single source of truth that is usable by both developers (through an API) and non-developers (through a UI). The finesse here is to enable programmatic access and schema enforcement without introducing rigid and complex user interfaces that harass non-developers to endure an information mugging that feels like filing taxes just to create a dataset.

## 03 You wish to improve data quality

We think of data quality in four dimensions:

1. **Completeness** — is this dataset everything we know about the subject?
2. **Trust** — is this dataset valid (true to the real world), verified (true to our business goals), and of known provenance?
3. **Freshness** — is this based on the latest knowledge we possess?
4. **Explainability** — what does this mean?

The completeness of your data is driven by the extent to which your data stewards are sufficiently incentivized and sufficiently cross-functional to discern what a dataset might be missing. (See our section on Data Governance for more on data stewards.) Completeness is therefore a function of the access and capability that you provide not just to your developer, but to your business users. Completeness requires that your data platform actively seeks to integrate the outer data that business users produce by ensuring that business users have sufficient permissions, knowledge, and tooling to label, document, and transition data across lifecycle stages.

The validity and veracity dimensions of trust arise from stewardship of data over a well-defined data lifecycle that tracks data from raw, to refined, to curated (see Manage data like code to improve

business velocity for details on data lifecycle). Once you establish a traceable data lifecycle, your platform automatically constructs the corresponding lineage. As data graduates across stages, data quality tests ensure that the data are both verified and valid. We recommend the following software for data quality tests:

- Structured data — Consider data profilers like Great Expectations
- Semi-structured data — Use JSON schemas
- Unstructured data — Use custom code, such as image processing libraries; consider systems like Stanford Deep Dive for extracting structure from text document

## PRINCIPLES OF DATA COLLABORATION

**Data collaboration leads to data superiority, a sustainable competitive advantage**. Data superiority means that, all other things being equal, the company with more trusted data will be the company that makes higher-quality decisions and wins in the market.

1. Enable all stakeholders to create, download, edit, and view datasets (either programmatically or with a graphical user interface)
2. Work with actionable, self-contained collections of data, known as datasets. Actionable datasets inform models, decisions, and product releases. An actionable dataset is self-contained in that anyone who interacts with the dataset can get the gist of its contents and intentions. Actionable datasets include primary data, visualizations, metadata, models, notebooks, and documentation so that team members can understand and act upon the data before them.
3. Seed a culture of documenting data with **a README that anyone, irrespective of technical skill, can write and publish to the web**. The emotional satisfaction and social reinforcement that team members get from creating something that their colleagues can consume and use creates a positive feedback loop. In the world of data collaboration, each dataset is documented with notes, metadata, and labels. Moreover, a cultural habit of data documentation frees teams from fragile *tribal knowledge*—facts and nuances that live in people's heads but are lost as old employees leave, new employees arrive, and human memories decay. Tribal knowledge is unwritten documentation that is essential for navigating the data forest—such as "a NaN in column 4 means the customer cancelled her subscription."
4. Define a *data lifecycle*. Data are born in swampy, ambiguous conditions and gradually mature into trusted, curated collections. A data lifecycle enables flexibility and creativity in the early phases, and conformity and quality in the later phases. At a minimum a data lifecycle has three phases: raw, refined, and curated. As data graduates from raw to refined to curated, it's trust, discoverability, and schema specificity increase.

5. Automatically run data quality checks to gate data lifecycle transitions. Data quality checks may verify file types, sizes, contents, schemas, labels, documentation, and model outputs.
6. Establish a single and shared source of truth (SSOT) that synchronizes and deduplicates data across a variety of sources (laptops, network attached storage, cloud storage, Box, etc.)

### Questions to get your team moving towards data collaboration

1. Who are the key cross-functional stakeholders in this project?
2. Do the stakeholders appreciate the importance and goals of sustainable data governance for the next one to ten years? (See the Appendix on **Data Governance**.)
3. How do the stakeholder groups vary in their knowledge of data storage, download, and analysis? Take special note of skill differences across groups. E.g. business users may store in technologies like Box and analyze with Excel, whereas data engineers may store Amazon S3 and analyze with Apache Spark.
4. Which data-driven project is high-impact yet simple enough for you to pilot your platform with, so that you can incrementally advance towards our data platform goals?
5. What is our data governance strategy for the next one to ten years? (Let's be honest, this is a scary question. But avoiding this question is a form of technical debt with a high rate of interest. If, like most companies, your data volumes are growing geometrically—while your headcount and machine count are growing linearly—you will necessarily find your teams overwhelmed with data in the near future.)

## "DATA GOVERNANCE" SOUNDS BORING, BUT IT'S RAD

A simplified model of data governance is to develop a plan to promote the following data qualities across your organization:

- **Accuracy** — Is this data correct?
- **Completeness** — Is this conclusion based on all of the data we have access too?
- **Consistency** — Do data science and research agree on these numbers?
- **Timeliness** — Is this the latest data?
- **Validity** — Do these data correspond to the real world?
- **Uniqueness** — How many needless duplicates of this data are floating around?

Can you think of a data-driven enterprise that's not interested in all of those dimensions?

You can educate your team on data governance by sharing with them the common goals (Table 1), and identifying the key data stewards in your organization. A data steward is someone who is responsible for data quality, completeness, and correctness of your data and metadata assets. The more data stewards you have—and the stronger your accountability standards—the smarter and more profitable your organization.

Table 1—Goals of data governance (adapted from **Data Governance, Wikipedia**)

1. Increase consistency and confidence in decision making
2. Improve data security; define and verify requirements for data distribution
3. Minimize or eliminate re-work
4. Maximize income generation potential of data
5. Optimize staff effectiveness
6. Enable better planning and decisions by supervisory staff
7. Establish process performance baselines to enable improvement efforts
8. Designate accountability for information quality (data stewardship)
9. Decrease the risk of regulatory fines
10. Designating accountability for information quality
11. Acknowledge and hold all gain

## RESOURCES TO LEARN MORE

- Quilt data's tutorials and webinars on data hubs (YouTube)
- Industry survey: Get to ROI—Eliminate data bottlenecks so that modeling can improve the bottom line
- Article: Principles of lazy data documentation and how to get your team onboard
- Technical article: Versioning data and models for rapid experimentation in machine learning
- Open Source Software: Quilt's immutable data packages for reproducibility, discoverability, and trust

## CONTACT

We welcome your questions and feedback on this guide. Email contact@quiltdata.io or find us on twitter.